High-Order Inference, Ranking, and Regularization Path for Structured SVM

A dissertation submitted to
UNIVERSITÉ PARIS-SACLAY
DOCTORAL SCHOOL STIC
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

to obtain the title of
DOCTOR OF SCIENCES
in
APPLIED MATHEMATICS AND COMPUTER SCIENCES

presented by
PUNEET KUMAR DOKANIA
prepared at
CentraleSupélec and INRIA

jury

Advisors : Prof. M. Pawan KUMAR - Oxford University, UK
Prof. Nikos PARAGIOS - École CentraleSupélec, France
Reviewers : Prof. Carsten ROTHER - TU Dresden, Germany
Prof. Stephen GOULD - ANU, Australia
President : Prof. Bill TRIGGS - LJK, Grenoble, France
Examiners : Prof. Christoph LAMPERT - IST Austria
Prof. Nikos KOMODAKIS - Ecole des Ponts ParisTech, France

NNT : 2016SACLC044

THÈSE DE DOCTORAT
DE
L'UNIVERSITÉ PARIS-SACLAY
PRÉPARÉE À
CENTRALESUPELEC

ÉCOLE DOCTORALE N°580
Sciences et Technologies de l'Information et de la Communication
Spécialité de doctorat : Mathématiques & Informatique

Par

M. Puneet Kumar Dokania

High-Order Inference, Ranking, and Regularization Path for Structured SVM

Thèse présentée et soutenue à Châtenay-Malabry, le 30 Mai 2016 :

Composition du Jury :

M Bill Triggs, Directeur de Recherche, *Université Grenoble Alpes (LJK) France*, Président du Jury
M Carsten Rother, Professeur, *TU Dresden (CVLD) Germany*, Rapporteur
M Stephen Gould, Professeur, *ANU Australia*, Rapporteur
M Christoph Lampert, Professeur, *IST Austria*, Examinateur
M Nikos Komodakis, Professeur, *Ecole des Ponts ParisTech France*, Examinateur
M M. Pawan Kumar, Professeur, *Oxford University (OVAL) UK,* Directeur de thèse
M Nikos Paragios, Professeur, *CentraleSupélec (CVN) / INRIA France*, Co-directeur de thèse

---

Titre: Inférence d'ordre supérieur, classement, et chemins de régularisation pour les machines à vecteurs de support structurées

Mots-clés: Inférence, Classement, SVM structurées, Chemin de régularisation, Recalage, Imagerie médicale, Vision numérique, Apprentissage statistique

Résumé: Cette thèse expose de nouvelles méthodes pour l'application de la prédiction structurée en vision numérique et en imagerie médicale. Nos nouvelles contributions suivent quatre axes majeurs. Premièrement, nous introduisons une nouvelle famille de potentiels d'ordre élevé qui encourage la parcimonie des étiquettes, et démontrons sa pertinence via l'introduction d'un algorithme précis de type graph-cuts pour la minimisation de l'énergie associée. Deuxièmement, nous montrons comment la formulation en SVM de la précision moyenne peut être étendue pour incorporer de l'information d'ordre supérieur dans les problèmes de classement. Troisièmement, nous proposons un nouvel algorithme de chemin de régularisation pour les SVM structurés. Enfin, nous montrons comment le cadre de l'apprentissage semi-supervisé des SVM à variables latentes peut être employé pour apprendre les paramètres d'un problème complexe de recalage déformable.

Plus en détail, la première partie de cette thèse étudie le problème d'inférence d'ordre supérieur. En particulier, nous présentons une nouvelle famille de problèmes de minimisation d'énergie discrète, que nous nommons *étiquetage parcimonieux*. C'est une extension naturelle aux potentiels d'ordre élevé des problèmes connus d'étiquetage de métriques. Similairement à l'étiquetage de métriques, les potentiels unaires de l'étiquetage parcimonieux sont arbitraires. Cependant, les potentiels des cliques sont définis à l'aide de la notion récente de *diversité* [18], définie sur l'ensemble des étiquettes uniques assignées aux variables aléatoires de la clique. Intuitivement, les diversités favorisent la parcimonie en diminuant le potentiel des ensembles avec un nombre moins élevés d'étiquettes. Nous proposons par ailleurs une généralisation du modèle P^n-Potts [66], que nous nommons modèle P^n-Potts hiérarchique. Nous montrons comment l'étiquetage parcimonieux peut être représenté comme un mélange de modèles P^n-Potts hiérarchiques. Enfin, nous proposons un algorithme parallélisable à proposition de mouvements avec de fortes bornes multiplicatives pour l'optimisation du modèle P^n-Potts hiérarchique et l'étiquetage parcimonieux. Nous démontrons l'efficacité de l'étiquetage parcimonieux dans les tâches de débruitage d'images et de mise en correspondance stéréo.

La seconde partie de cette thèse explore le problème de classement en utilisant de l'information d'ordre élevé. En l'occurrence, nous introduisons deux cadres différents pour l'incorporation d'information d'ordre élevé dans le problème de classement. Le premier modèle, que nous nommons SVM binaire d'ordre supérieur (HOB-SVM), s'inspire des SVM standards. Pour un ensemble d'exemples donné, ce modèle optimise une borne supérieure convexe sur l'erreur 0-1 pondérée. Le vecteur de caractéristiques joint de HOB-SVM dépend non seulement des caractéristiques des exemples individuels, mais également des caractéristiques de sous-ensembles d'exemples. Ceci nous permet d'incorporer de l'information d'ordre supérieur. La difficulté d'utilisation de HOB-SVM réside dans le fait qu'un seul score correspond à l'étiquetage de tout le jeu de données, alors que nous nécessitons des scores pour chaque exemple pour déterminer le classement. Pour résoudre cette difficulté, nous proposons de classer les exemples en utilisant la différence entre la max-marginales de l'affectation d'un exemple à la classe associée et la max-marginale de son affectation à la classe complémentaire. Nous démontrons empiriquement que la différence de max-marginales traduit un classement pertinent. À l'instar d'un SVM, le principal désavantage de HOB-SVM est que le modèle optimise une fonction d'erreur de substitution, et non la métrique de précision moyenne (AP). Afin d'apporter une solution à ce problème, nous proposons un second modèle, appelé AP-SVM d'ordre supérieur (HOAP-SVM). Ce modèle s'inspire d'AP-SVM [137] et de notre premier modèle, HOB-SVM. À l'instar d'AP-SVM, HOAP-SVM prend en entrée une collection d'exemples, et produit un classement de ces exemples, sa fonction d'erreur étant l'erreur AP. Cependant, au contraire d'AP-SVM, le score d'un classement est égal à la moyenne pondérée des de la différence des max-marginales des exemples individuels. Comme les max-marginales capturent l'information d'ordre supérieur, et les fonctions d'erreur dépendent de l'AP, HOAP-SVM apporte une solution aux deux limitations susmentionnées des classificateurs traditionnels tels qu'SVM. Le principal désavantage de HOAP-SVM réside en le fait que l'estimation de ses paramètres nécessite la résolution d'un programme de différences de fonctions convexes [55]. Nous montrons comment un optimum local de l'apprentissage d'un HOAP-SVM peut être déterminé efficacement grâce à la procédure concave-convexe [138]. En utilisant des jeux de données standards, nous montrons empiriquement que HOAP-SVM dépasse les modèles de référence en utilisant efficacement l'information d'ordre supérieur tout en optimisant la fonction d'erreur appropriée.

Dans la troisième partie de la thèse, nous proposons un nouvel algorithme, SSVM-RP, pour obtenir un chemin de régularisation ε-optimal pour les SVM structurés. Par définition, un chemin de régularisation est l'ensemble des solutions pour toutes les valeurs possibles du paramètre de régularisation dans l'espace des paramètres [29]. Cela nous permet d'obtenir le meilleur modèle en explorant efficacement l'espace du paramètre de régularisation. Nous proposons également des variantes intuitives de l'algorithme Frank-Wolfe de descente de coordonnées par blocs (BCFW) pour l'optimisation accélérée de l'algorithme SSVM-RP. De surcroît, nous proposons une approche systématique d'optimisation des SSVM avec des contraintes additionnelles de boîte en utilisant BCFW et ses variantes. Ces contraintes additionnelles sont utiles dans de nombreux problèmes importants pour résoudre exactement le problème d'inférence. Par exemple, pour un problème à deux étiquettes dont la sortie possède une structure de graphe cyclique, si les potentiels binaires sont sous-modulaires, l'algorithme graph-cuts [74] permet de résoudre exactement le problème d'inférence. Pour imposer la sous-modularité des potentiels binaires, des contraintes additionnelles, tels que la positivité/négativité, sont employées dans la fonction objective du SSVM. Enfin, nous proposons un algorithme de chemin de régularisation pour SSVM avec des contraintes additionnelles de positivité/negativité.

Dans la quatrième et dernière partie de la thèse (Appendice A), nous proposons un nouvel algorithme discriminatif semi-supervisé pour apprendre des métriques de recalage spécifiques au contexte comme une combinaison linéaire des métriques conventionnelles. Nous adoptons un cadre de modèle graphique populaire [48] pour formuler le recalage déformable comme un problème d'inférence discrète. Ceci implique des termes unaires de vérité terrain et des termes de régularité. Le terme unaire est une mesure de similarité ou métrique spécifique à l'application, tels que l'information mutuelle, la corrélation croisée normalisée, la somme des différences absolues, et les coefficients d'ondelettes discrets. Selon l'application, les métriques traditionnelles sont seulement partiellement sensibles aux propriétés anatomiques des tissus. L'apprentissage de métriques est une alternative cherchant à déterminer une mise en correspondance entre un volume source et un volume cible dans la tâche de recalage. Dans ce travail, nous cherchons à déterminer des métriques spécifiques à l'anatomie et aux tissus, par agrégation linéaire de métriques connues. Nous proposons un algorithme d'apprentissage semi-supervisé pour estimer ces paramètres conditionnellement aux classes sémantiques des données, en utilisant un jeu de données faiblement annoté. La fonction objective de notre formulation se trouve être un cas spécial d'un programme de différence de fonctions convexes. Nous utilisons l'algorithme connu de différence de fonctions concave-convexe pour obtenir le minimum ou un point critique du problème d'optimisation. Afin d'estimer la vérité terrain inconnue des vecteurs de déformations, que nous traitons comme des variables latentes, nous utilisons une inférence « fidèle à la segmentation » munie d'une fonction de coût. Nous démontrons l'efficacité de notre approche sur trois jeux de données particulièrement difficiles dans le domaine de l'imagerie médicale, variables en terme de structures anatomiques et de modalités d'imagerie.

université
PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies
de l'information
et de la communication (STIC)

Title: High-Order Inference, Ranking, and Regularization Path for Structured SVM

Keywords: Inference, Ranking, Structured SVM, Regularization path, Registration, Medical imaging, Computer Vision, Machine learning

Abstract: This thesis develops novel methods to enable the use of structured prediction in computer vision and medical imaging. Specifically, our contributions are four fold. First, we propose a new family of high-order potentials that encourage parsimony in the labeling, and enable its use by designing an accurate graph cut based algorithm to minimize the corresponding energy function. Second, we show how the average precision SVM formulation can be extended to incorporate high-order information for ranking. Third, we propose a novel regularization path algorithm for structured SVM. Fourth, we show how the weakly supervised framework of latent SVM can be employed to learn the parameters for the challenging deformable registration problem.

In more detail, the first part of the thesis investigates the high-order inference problem. Specifically, we present a novel family of discrete energy minimization problems, which we call *parsimonious labeling*. It is a natural generalization of the well known metric labeling method for high-order potentials. Similar to metric labeling, the unary potentials of parsimonious labeling are arbitrary. However, the clique potentials are defined using the recently proposed notion of *diversity* [18], defined over the set of unique labels assigned to the random variables in the clique. Intuitively, diversity enforces parsimony by assigning lower potential to sets with fewer labels. In addition to this, we propose a generalization of the P^n-Potts model [66], which we call Hierarchical P^n-Potts. We show how parsimonious labeling can be represented as a mixture of hierarchical P^n-Potts models. In the end, we propose parallelizable move making algorithms with very strong multiplicative bounds for the optimization of hierarchical P^n-Potts models and parsimonious labeling. We show the efficacy of parsimonious labeling in image denoising and stereo matching tasks.

Second part of the thesis investigates the ranking problem while using high-order information. Specifically, we introduce two alternate frameworks to incorporate high-order information for ranking tasks. The first framework, which we call high-order binary SVM (HOB-SVM), takes its inspiration from the standard SVM. For a given set of samples, it optimizes a convex upper bound on weighted 0-1 loss. The joint feature vector of HOB-SVM depends not only on the feature vectors of the individual samples, but also on the feature vectors of subsets of samples. It allows us to incorporate high-order information. The difficulty with employing HOB-SVM is that it provides a single score for the entire labeling of a dataset, whereas we need scores corresponding to each sample in order to find the ranking. To address this difficulty, we propose to rank the samples using the difference between the max-marginal cost for assigning a sample to the relevant class and the max-marginal cost for assigning it to the non-relevant class. Empirically, we show that the difference of

max-marginal costs provides an accurate ranking. The main disadvantage of HOB-SVM is that, similarly to SVM, it optimizes a surrogate loss function instead of the average precision (AP) based loss. In order to alleviate this problem we propose a second framework, which we call high-order AP-SVM (HOAP-SVM). It takes its inspiration from AP-SVM [137] and HOB-SVM (our first framework). Similarly to AP-SVM, the input of HOAP-SVM is a set of samples, its output is a ranking of the samples, and its loss function is the AP loss. However, unlike AP-SVM, the score of a ranking is equal to the weighted sum of the difference of max-marginal costs of the individual samples. Since the max-marginal costs capture high-order information, and the loss function depends on the AP, HOAP-SVM addresses both of the aforementioned deficiencies of traditional classifiers such as SVM. The main disadvantage of HOAP-SVM is that estimating its parameters requires solving a difference-of-convex program [55]. We show how a local optimum of the HOAP-SVM cost can be computed efficiently by the concave-convex procedure [138]. Using standard datasets, we empirically demonstrate that HOAP-SVM outperforms the baselines by effectively utilizing high-order information while optimizing the correct loss function.

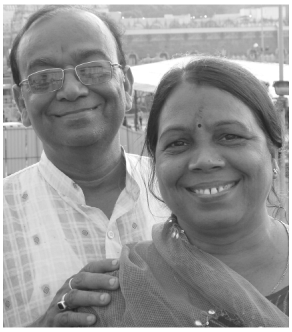
In the third part of the thesis, we propose a new algorithm (SSVM-RP) to obtain the ε-optimal regularization path of structured SVM. By definition, the regularization path is the set of solutions for all possible values of the regularization parameter in the parameter space [29]. It allows us to obtain the best model by efficiently searching the entire regularization parameter space. We also propose intuitive variants of the Block-Coordinate Frank-Wolfe algorithm for faster optimization of the SSVM-RP algorithm. In addition to this, we propose a principled approach to optimize the SSVM with additional box constraints using BCFW and its variants. These additional constraints are useful in many important problems in order to solve the inference problem exactly. For example, if there are two labels and the output structure forms a graph with loops, then if pairwise potentials are submodular, graph cuts [74] can be used to solve the inference problem exactly. In order to ensure that the pairwise potentials are submodular, additional constraints (for example, positivity/negativity) are normally used in the objective function of the SSVM. In the end, we propose regularization path algorithm for SSVM with additional positivity/negativity constraints.

In the fourth and the last part of the thesis (Appendix A), we propose a novel weakly supervised discriminative algorithm for learning context specific registration metrics as a linear combination of conventional metrics. We adopt a popular graphical model framework [48] to cast deformable registration as a discrete inference problem. It involves data terms and smoothness terms. The data term is an application specific similarity measure or metric, such as mutual information, normalized cross correlation, sum of absolute difference, or discrete wavelet coefficients. Conventional metrics can cope only partially – depending on the clinical context – with tissue anatomical properties. Metric learning is an alternative that seeks to determine a mapping between the source and the target volumes in the registration task. In this work we seek to determine anatomy/tissue specific metrics as a context-specific aggregation/linear combination of known metrics. We propose a weakly supervised learning algorithm for estimating these parameters conditionally to the data semantic classes, using a weak training dataset. The objective function of

our formulation turns out to be a special kind of non-convex program, known as the difference of convex program. We use the well known concave convex procedure to obtain the minima or saddle points of the optimization problem. In order to estimate the unknown ground truth deformation vectors, which we treat as latent variables, we use ‘segmentation consistent’ inference endowed with a loss function. We show the efficacy of our approach on three highly challenging datasets in the field of medical imaging, which vary in terms of anatomical structures and image modalities.

Dedication

I would like to dedicate this thesis to my loving parents
(Shri. Bimal Dokania and Smt. Meena Devi)



Acknowledgements

First of all I would like to thank my supervisors, Prof. M. Pawan Kumar and Prof. Nikos Paragios, for their consistent guidance, encouragement, and their faith in me. I owe my entire research career to both of them. Pawan transformed me from an inexperienced student to one who can courageously and independently work on research problems. His hard work, choice of problems, determination, and discipline has always inspired me. I would like to thank Nikos for his encouragement and support throughout my PhD. Along with his support in pursuing my research, I would also like to thank him for his presence as a guardian. I always had this confidence that he would be the rescuer whenever needed. I find myself extremely lucky to have such supervisors.

I would also like to thank my reviewers Prof. Rother and Prof. Gould for spending their valuable time for reviewing my thesis and providing useful comments and suggestions. My special thanks to my committee president Prof. Triggs who took all the pain to correct few chapters of my thesis and send corrections by post. His inputs greatly improved the presentation of first two chapters of the thesis. I would also like to thank Prof. Lampert and Prof. Komodakis for their role as an examiner. It was wonderful and insightful to have them all as my thesis committee members.

I would like to thank my funding agencies: French Government (Ministère de l'Education nationale, de l'Enseignement supérieur et de la Recherche), ERC Grant number 259112, MOBOT Grant number 600796, and Pôle de Compétitivité Medicen/ADOC Grant number 111012185. My special thanks to Prof. Iasonas Kokkinos for providing me funds to extend my PhD.

I am grateful to all my lab mates (present and past) or I must say my trustworthy friends for providing their selfless support and their presence whenever needed. Specially, I would like to thank Enzo for countless things he did for me, Siddhartha for listening to me (silently), Evgenios for being a nice and funny office mate, Eugene for plenty of technical discussions, Wacha and Aline for helping me out (they know what I mean), Rafael for teaching me many programming tricks, and Maxim for translating my thesis abstract into French. I am also grateful to few of my friends in India who always supported me. My special thanks to Natalia (CentraleSupélec), Carine (CentraleSupélec), Alexandra (INRIA), and Sandra

(Science-accueil) for helping me out with infinitely many French administrative formalities. I am afraid that without their help I would have spent most of my time dealing with paper works. I would also like to thank Dr. Simon Lacoste-Julien for offering me to work with him for almost 10 weeks and introducing me to very exciting and challenging problems. I would also like to thank my collaborators: Pritish, Enzo, Jean-Baptiste, Anton, and Pierre Yves.

I would like to thank Sriram Venkatapathy who told me about Pawan and a PhD opening under him during a dinner gathering in Grenoble in the year 2012. I find myself very fortunate to have met Sriram that day.

My thanks go to my school teachers in Kundahit Middle School (Jharkhand), Delhi (Sarvodaya Vidyalaya, Pitampura; and Pratibha Vikas Vidyalaya, Shalimar Bagh), and my Professors in Delhi College of Engineering and ENSIMAG Grenoble. Every school and college I went, I learned. Everything I learned has great impact on what I am today and what I'll be tomorrow, and for that I thank them. Special thanks to my uncle (*Late Shri. Bhagwandas Dokania*) who encouraged me and financially supported my education in 2004 that allowed me to prepare for my Engineering entrance examinations in India.

Above all, I would like to thank my first teachers, my *Maa* and *Papa*, for always believing in me. I would deeply like to thank my family for their unconditional love and for their faith in me even when I was doing something quite unconventional. I would also like to thank my younger sister *Neha* and my elder sister *Rashmi didi* for taking care of our parents in my absence when I was busy struggling with my personal life and doing my PhD in France.

Table of contents

List of figures **xvii**

List of tables **xxv**

1 Introduction **1**

1.1 High-Order Inference 4

1.2 Learning to Rank 6

1.3 Regularization Path 8

1.4 Thesis Outline 10

1.5 List of publications 11

2 Review of Structured SVM and related Inference Algorithms **13**

2.1 Motivation 13

2.2 Structured Output SVM 16

2.2.1 Prediction for the SSVM 18

2.2.2 Learning parameters for the SSVM 19

2.3 The Inference Problem 20

2.3.1 Graph cuts for submodular pairwise potential 22

2.3.2 The $\alpha-$expansion algorithm for metric labeling 28

2.3.3 α-expansion for the P^n Potts model 32

2.4 Learning Algorithms for the SSVM 34

2.4.1 Convex upperbound of empirical loss 35

2.4.2 Lagrange dual of SSVM 37

2.4.3 Optimization 39

2.5 Weakly Supervised SSVM 47

2.5.1 Latent SSVM 49

2.5.2 Difference of convex upper bounds of the empirical loss 49

2.5.3 Optimization of LSSVM using CCCP algorithm 50

3 Parsimonious Labeling 53
3.1 Introduction 53
3.2 Related Work 55
3.3 Preliminaries 56
3.3.1 The labeling problem 56
3.3.2 Multiplicative Bound 57
3.4 Parsimonious Labeling 61
3.5 The Hierarchical Move Making Algorithm 64
3.5.1 The Hierarchical Move Making Algorithm for the Hierarchical P^n Potts Model 65
3.5.2 The Move Making Algorithm for the Parsimonious Labeling 72
3.6 Experiments 75
3.6.1 Synthetic Data 75
3.6.2 Real Data 76
3.7 Discussion 81

4 Learning to Rank using High-Order Information 83
4.1 Introduction 83
4.2 Preliminaries 86
4.2.1 Structured Output SVM 86
4.2.2 AP-SVM 87
4.3 High-Order Binary SVM (HOB-SVM) 89
4.4 High-Order Average Precision SVM (HOAP-SVM) 92
4.5 Experiments 100
4.6 Discussion 103

5 Regularization Path for SSVM using BCFW and its variants 105
5.1 Introduction 105
5.2 Related Work 107
5.3 Structured SVM 108
5.3.1 Objective Function 108
5.3.2 Optimization of SSVM using FW and BCFW 110
5.4 Variants of BCFW Algorithm 111
5.5 SSVM with box constraints (SSVM-B) 113
5.5.1 Objective Function 114
5.5.2 Optimization of SSVM-B 116
5.6 Regularization path for SSVM 121

5.6.1 Finding the breakpoints 122
5.6.2 Initialization 125
5.7 Regularization path for SSVM with positivity constraints 127
5.7.1 Finding the breakpoints 127
5.7.2 Initialization 127
5.8 Experiments and Analysis 129
5.9 Discussion 134

6 Discussion 139
6.1 Contributions of the Thesis 139
6.2 Future Work 142

Appendix A Deformable Registration through Learning of Context-Specific Metric Aggregation 145
A.1 Introduction 145
A.2 Related Work 147
A.3 The Deformable Registration Problem 148
A.4 Learning the Parameters 150
A.4.1 Preliminaries 150
A.4.2 The Objective Function 152
A.4.3 The Learning Algorithm 153
A.4.4 Prediction 155
A.5 Experiments and Results 155
A.6 Discussions and Conclusions 159

Appendix B Optimization 163
B.1 Brief Introduction to Lagrangian Theory 163
B.1.1 Lagrangian 163
B.1.2 Lagrange dual objective 164
B.1.3 Lagrange dual problem 164
B.1.4 Strong duality and complementary slackness 165
B.1.5 Karush-Kuhn-Tucker (KKT) conditions 165
B.2 SSVM as quadratic program 166
B.3 Frank-Wolfe Algorithm 167

References 171

List of figures

1.1 Image denoising and inpainting. 1

1.2 An example of action classification problem for the action class ‘jumping’. The ‘input’ is a set of bounding boxes. The desired ‘ouput’ is the sorted list in which all the bounding boxes with jumping action are ranked ahead of all the bounding boxes with non jumping action. Figure 1.2b shows one of the many possible outputs. 2

1.3 An illustration of the value of high-order interactions in solving segmentation problems. *Image source [67].* 6

1.4 Differences between AP and accuracy. The relevant class here is ‘jumping’. The scores shown are obtained from the classifier. A positive score implies a relevant class (‘jumping’) and a negative score implies an irrelevant class (‘non jumping’). The bounding boxes are sorted based on the scores to obtain the ranking. An ideal ranking would have all the positive (or relevant) examples ahead of all the negative (irrelevant) ones. Notice that the accuracy is nothing but the fraction of misclassifications. 8

1.5 The example of high-order information in case of action classification. Notice that people in the same image often perform the same action. This can be one of the hypotheses that encode meaningful high-order interactions. . . . 9

2.1 An example of handwritten word recognition. The input is an image of a handwritten word $\mathbf{x} = \{x_1, x_2, x_3, x_4\}$, where each x_i is the bounding box around each character of the word. The task is to predict the correct word, which in this case is ‘ROSE’. 14

2.2 *Solution 1:*Each bounding box is classified separately. The set of classes (or labels) is $\{a,\cdots,z,A,\cdots,Z\}$. The final solution is the collection of the predictions corresponding to each bounding box. The number of classifiers required is $|\mathscr{L}|(=52)$, therefore, the solution is practically very efficient. However, it does not take account of the interdependencies between the output variables. 14

2.3 *Solution 2:*Each *word* is classified separately. The set of classes (or labels) is $\{aaaa,\cdots,zzzz,AAAA,\cdots,ZZZZ\}$. Note that the number of prediction functions required is $(52)^4 \approx 7$ Million. Therefore, this solution is practically infeasible even for moderate length words. 15

2.4 Few examples of the different possible interactions between the output variables. Variables in the edges with same colors form a clique. For example, in case of Figure 2.4c, set of cliques is: $\mathscr{C} = \{c_1, c_2\}$, where $c_1 = \{y_1, y_2, y_3\}$ and $c_2 = \{y_3, y_4\}$. Many other types of interactions are also possible. 17

2.5 The directed graph shown in Figure 2.5a is an example of an s-t Graph. All of the arc weights are assumed to be positive. In Figure 2.5b, the cost of Cut-1 is $(w_1 + w_6 + w_4)$, V_s is $\{s, v_2\}$, V_t is $\{t, v_1\}$, and the labeling is $\mathbf{y} = \{1,0\}$. Similarly, the cost of Cut-2 is $(w_1 + w_3)$, V_s is $\{s\}$, V_t is $\{t, v_1, v_2\}$, and the labeling is $\mathbf{y} = \{1,1\}$. 24

2.6 The *s-t* graph construction for arbitrary unary potentials. Figure 2.6b shows the *s-t* graph when the unary potential for label 1 is higher than that of label 0. Similarly, Figure 2.6b shows the *s-t* graph for the opposite case. Recall that a node in the set V_s and V_t is assigned the labels 0 and 1, respectively. . 26

2.7 Graph construction for a given pairwise submodular energy for the case when $A \geq C$ and $D \geq C$. Figures 2.7a and 2.7b are the graph constructions for $\theta_1(0) = A - C$ and $\theta_2(1) = D - C$ respectively. Figure 2.7c is the graph construction for $\theta_{12}(0,1) = B + C - A - D$. Figure 2.7d uses the additivity theorem to construct the final graph. 27

2.8 An example of the move space for $\alpha-$expansion with only four nodes in the graph. Figure 2.8a shows the initial labeling from where the expansion move will take place. Figure 2.8b shows the possible labelings when α is the ‘Green’ label. Let the numbering of the nodes be in the clockwise manner starting from the top left corner. Then different binary vectors for all possible labelings are as follows: $\mathbf{t}_a = \{0,0,0,0\}$, $\mathbf{t}_b = \{1,0,0,0\}$, $\mathbf{t}_c = \{0,0,1,0\}$, and $\mathbf{t}_d = \{1,0,0,1\}$. 30

2.9 An example of the costs assigned by the P^n-Potts model. Let γ_r, γ_b, and γ_g be the costs for the colors 'red', 'blue', and 'green', respectively. $\gamma_{max} > \max(\gamma_r, \gamma_g, \gamma_b)$. Then the costs for the cases (a), (b), and (c) are all the same, which is γ_{max}. The cost for the case (d) is γ_g. 32

2.10 Graph construction for the P^n Potts model. M_s and M_t are two auxiliary nodes. The weights are $w_d = \gamma + k$ and $w_e = \gamma_\alpha + k$, where, $k = \gamma_{max} - \gamma_\alpha - \gamma$. Figure reproduced from [66]. 33

2.11 Pictorial representation of one iteration of the CCCP algorithm [138]. Figure 2.11a shows the given difference of convex functions, which can be written as a sum of convex and concave functions. Figure 2.11b shows the linear upper bound for the concave function $(-v(\mathbf{w}))$ at a given point, which results in an affine function $\bar{v}(\mathbf{w})$. Finally, Figure 2.11c shows the resulting convex function as a sum of convex $u(\mathbf{w})$ and affine $\bar{v}(\mathbf{w})$ ones. 50

3.1 *An example of* r-HST *for* $r = 2$*. The cluster associated with root p contains all the labels. As we go down, the cluster splits into subclusters and finally we get the singletons, the leaf nodes (labels). The root is at depth of 1* $(\tau = 1)$ *and leaf nodes at* $\tau = 3$*. The metric defined over the* r-HST *is denoted as* $d^t(.,.)$*, the shortest path between the inputs. For example,* $d^t(l_1, l_3) = 18$ *and* $d^t(l_1, l_2) = 6$*. The diameter diversity for the subset of labels at cluster p is* $\max_{\{l_i,l_j\} \in \{l_1,l_2,l_3,l_4\}} d^t(l_i, l_j) = 18$*. Similarly, the diameter diversity at* p_2 *and* p_3 *are 6 and 0, respectively.* 64

3.2 An example of solving the labeling problems at leaf nodes. Consider a clique with three nodes which we intend to label. Figure 3.2a shows the subprobelms at each leaf node. Notice that each leaf node contains only one label, therefore, we have only one choice for the labeling. Figure 3.2b shows the corresponding trivial labelings (notice the colors). 65

3.3 *An example of solving the labeling problem at non-leaf node* (p) *by combining the solutions of its child nodes* $\{p_1, p_2\}$*, given clique c and the labelings that it has obtained at the child nodes. The labeling fusion instance shown in this figure is the top two levels of the* r-HST *in the Figure 3.1. Note that the labelings shown at nodes* p_1 *and* p_2 *are an assumption. It could get different labelings as well but the algorithm for the fusion remains the same. The diameter diversity of the labeling of clique c at node* p_1 *is 0 as it contains only one unique label* l_1*. The diameter diversity of the labeling at* p_2 *is* $d^t(l_3, l_4) = 6$ *and at p is* $\max_{\{l_i,l_j\} \in \{l_1,l_3,l_4\}} d^t(l_i, l_j) = 18$*. Please notice the colors.* 66

3.4 *Synthetic (Blue: Our, Red: Co-occ [87]). The x-axis of all the figures is the weight associated with the cliques* (w_c). *Figures (a) and (b) are the plots when the hierarchical* P^n *Potts model is known. Figures (c) and (d) are the plots when a diversity (diameter diversity over truncated linear metric) is given as the clique potentials which is then approximated using the mixture of hierarchical* P^n *Potts model. Notice that in both the cases our method outperforms the baseline [87] both in terms of energy and time. Also, for very high value of* $w_c = 100$, *both the methods converges to the same labeling. This is expected as a very high value of* w_c *enforces rigid smoothness by assigning everything to the same label.* 76

3.5 *Stereo Matching Results. Figures (a) and (d) are the ground truth disparity for the 'tsukuba' and 'teddy' respectively. Our method significantly outperforms the baseline Co-ooc [87] in both the cases in terms of energy. Our results are visually more appealing also. Figures (b) and (e) clearly shows the influence of 'parsimonious labeling' as the regions are smooth and the discontinuity is preserved. Recall that we use super-pixels obtained using the mean-shift as the cliques.* 78

3.6 *Comparison of all the methods for the stereo matching of 'teddy'. We used the optimal setting of the parameters proposed in the well known Middlebury webpage and [118]. The above results are obtained using* $\sigma = 10^2$ *for the Co-occ and our method. Clearly, our method gives much smooth results while keeping the underlying shape intact. This is because of the cliques and the corresponding potentials (diversities) used. The diversities enforces smoothness over the cliques while* σ *controls this smoothness in order to avoid over smooth results.* 78

3.7 *Comparison of all the methods for the stereo matching of 'tsukuba'. We used the optimal setting of the parameters proposed in the well known Middlebury webpage and [118]. The above results are obtained using* $\sigma = 10^2$ *for the Co-occ and our method. We can see that the disparity obtained using our method is closest to the ground truth compared to all other methods. In our method, the background is uniform (under the table also), the camera shape is closest to the ground truth camera, and the face disparity is also closest to the ground truth compared to other methods.* 79

3.8 *Effect of* σ *in the* parsimonious labeling. *All the parameters are same except for the* σ. *Note that as we increase the* σ, *the* w_c *increases, which in turn results in over smoothing.* 79

3.9 *Effect of clique size (superpixels). The top row shows the cliques (superpixels) used and the bottom row shows the stereo matching using these cliques. As we go from left to right, the minimum number of pixels that a superpixel must contain increases. All the other parameters are the same. In order to increase the weight* w_c*, we use high value of* σ*, which is* $\sigma = 10^5$ *in all the above cases.* 79

3.10 *Image inpainting results. Figures (a) and (d) are the input images of 'penguin' and 'house' with added noise and obscured regions. Our method, (b) and (e), significantly outperforms the baseline [87] in both the cases in terms of energy. Visually, our method gives much more appealing results. We use super-pixels obtained using the mean-shift as the cliques.* 80

3.11 *Comparison of all the methods for the image inpainting and denoising problem of the 'penguin'. Notice that our method recovers the hand of the penguin very smoothly. In other methods, except Co-oc, the ground is over-smooth while our method recovers the ground quite well compared to others.* 80

3.12 *Comparison of all the methods for the image inpainting and denoising problem of the 'house'.* 81

4.1 *Top 8 samples ranked by all the four methods for 'reading' action class. First row –* SVM*, Second row –* AP-SVM*, Third row –* HOB-SVM*, and Fourth row –* HOAP-SVM*. Note that, the first false positive is ranked 2nd in case of* SVM *(first row) and 3rd in case of* AP-SVM *(second row), this shows the importance of optimizing the* AP *loss. On the other hand, in case of* HOB-SVM *(third row), the first false positive is ranked 4th and the 'similar samples' (2nd and 3rd) are assigned similar scores, this illustrates the importance of using high-order information. Furthermore,* HOAP-SVM *(fourth row) has the best* AP *among all the four methods, this shows the importance of using high-order information and optimizing the correct loss. Note that, in case of* HOAP-SVM*, the 4th and 5th ranked samples are false positives (underlying action is close to reading) and they both belong to the same image (our similarity criterion). This indicates that high-order information sometimes may lead to poor test* AP *in case of confusing classes (such as 'playinginstrument' vs 'usingcomputer') by assigning all the connected samples to the wrong label. Same effect can be seen in* HOB-SVM *for 7th and 8th ranked samples.* . . . 103

5.1 In each figure, x-axis is the λ and y-axis is the number of passes. In case of regularization path algorithms (starting with RP), x-axis represents all the breakpoints for the entire path, and y-axis represents the number of passes required to reach $\kappa_1\varepsilon$ optimal solution at each breakpoint (warm start with ε optimal solution). In other two algorithms (no regularization path), x-axis represents 20 values of λ evenly spaced in the range of $[10^{-4},\ 10^{3}]$, and y-axis represents the number of passes to reach ε optimal solution (initialization with $\mathbf{w} = 0$). 135

5.2 Figure 5.2a shows the ‘legend’ for the purpose of clarity. In all other figures, x-axis is the λ and y-axis is the duality gap. In case of regularization path algorithms (starting with RP), x-axis represents all the breakpoints for the entire path. In other two algorithms (no regularization path), x-axis represents 20 values of λ evenly spaced in the range of $[10^{-4},\ 10^{3}]$. Recall that in case of regularization path algorithms the solution we seek must be $\kappa_1\varepsilon$ optimal. 136

5.3 In all figures, x-axis is the λ and y-axis is the test loss. In all the experiments, the test losses are obtained for models corresponding to 20 different values of λ equally spaced in the range of $[10^{-4},\ 10^{3}]$. 137

A.1 The top row represents the sample slices from three different volumes of the RT Parotids dataset. The middle row represents the sample slices of the RT Abdominal dataset, and the last row represents the sample slices from the IBSR dataset. 156

A.2 Overlapping of the segmentation masks in different views for one registration case from **RT Abdominal** (first and second rows) and **RT Parotids** (third and fourth rows) datasets. The first column corresponds to the overlapping before registration between the source (in blue) and target (in red) segmentation masks of the different anatomical structures. From second to sixth column, we observe the overlapping between the warped source (in green) and the target (in red) segmentation masks, for the multiweight algorithm (MW) and the single metric algorithm using SAD, MI, NCC and DWT). We observe that MW gives a better fit between the deformed and ground truth structures than the rest of the single similarity measures, which are over segmenting most of the structures showing a poorer registration performance. 157

A.3 Qualitative results for one slice of one registration case from **IBSR dataset**. Since showing overlapped structures in the same image is too ambiguous given that the segmentation masks almost cover the complete image, we are showing the intensity difference between the two volumes. The first column shows the difference of the original volumes before registration. From second to sixth column we observe the difference between the warped source and the target images, for the MW and the single metric algorithm using SAD, MI, NCC or DWT). According to the scale in the bottom part of the image, extreme values (which mean high differences between the images) correspond to blue and red colors, while green indicates no difference in terms of intensity. Note how most of the big differences observed in the first column (before registration) are reduced by the MW algorithm, while some of them (specially in the peripheral area of the head) remain when using single metrics. 158

A.4 Results for the RT parotids dataset for the single-metric registration (SAD, MI, NCC, DWT) and the multi-metric registration (MW). The weights for the multi-metric registration are learned using the framework proposed in this work. ‘Parotl’ and ‘Parotr’ are the left and the right organs. The red square is the mean and the red bar is the median. It is evident from the results that using the learned linear combination of the metrics outperforms the single-metric based deformable registration. 159

A.5 Results for the RT abdominal dataset for the single-metric registration (SAD, MI, NCC, DWT) and the multi-metric registration (MW). The weights for the multi-metric registration are learned using the framework proposed in this work. ‘Bladder’, ‘Sigmoid’, and ‘Rectum’ are the three organs on the dataset. The red square is the mean and the red bar is the median. It is evident from the results that using the learned linear combination of the metrics outperforms the single-metric based deformable registration. 160

A.6 Results for the IBSR dataset for the single-metric registration (SAD, MI, NCC, DWT) and the multi-metric registration (MW). The weights for the multi-metric registration are learned using the framework proposed in this work. ‘CSF’, ‘Grey Mater’, and ‘White Mater’ are different structures in the brain. The red square is the mean and the red bar is the median. It is evident from the results that using the learned linear combination of the metrics outperforms the single-metric based deformable registration. 161

B.1 The blue curve shows the convex function and the black polygon shows the convex compact domain. At a given $\mathbf{q}_k$, the Figure B.1a shows the linearization and the minimization of the linearized function over the domain to obtain the atom $\mathbf{s}_k$. Figure B.1b shows the new point $\mathbf{q}_{k+1}$ obtained using the linear combination of $\mathbf{q}_k$ and $\mathbf{s}_k$. The green line shows the linearization at the new point $\mathbf{q}_{k+1}$, and $\mathbf{s}_{k+1}$ is the new atom obtained as a result of the minimization of this function. 168

B.2 Visualization of the *linearization duality gap*. The blue curve is the convex function f. The red line is the linearization of f at $\mathbf{q}_k$, denoted as f_l. The atom $\mathbf{s}$ is obtained as the result of the minization of f_l over the domain. The duality gap is $g(\mathbf{q}_k) = f(\mathbf{q}_k) - f_l(\mathbf{s})$. 169

List of tables

4.1 *The AP over five folds for the best setting of the hyperparameters obtained using the cross-validation. Our frameworks outperforms* SVM *and* AP-SVM *in all the 10 action classes. Note that* HOAP-SVM *is initialized with* HOB-SVM. 101

4.2 *The AP of all the four methods. The training is performed over the entire 'trainval' dataset of PASCAL VOC 2011 using the best hyperparameters obtained during 5-fold cross-validation. The testing is performed on the 'test' dataset and evaluated on the PASCAL VOC server. Note that* HOAP-SVM *is initialized using* HOB-SVM. 101

5.1 No regularization path: Initialization with $\mathbf{w} = 0$. We used 20 different values of λ equally spaced in the range of $[10^{-4},\ 10^{3}]$ and four variants of the BCFW algorithm. Total passes represents the total number of passes through the entire dataset using all the 20 values of λ. Min test loss represents the minimum value of the test loss obtained by using all the 20 trained models. Clearly, the gap based sampling method (BCFW-STD-G) requires less number of passes to achieve the same generalization. 131

5.2 Regularization path: κ_1 vs total passes for four different variants of the BCFW algorithm.The total passes represents the total number of passes through the entire dataset to obtain the complete regularization path. 131

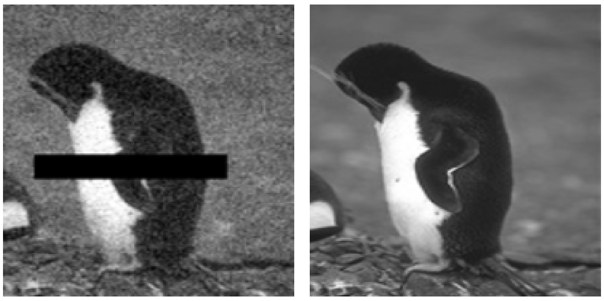
5.3 Regularization path: κ_1 vs number of regularization parameters (λ) for four different variants of the BCFW algorithm. The number of regularization parameters is basically the number of kinks that divide the complete regularization space into segments. 131

5.4 Regularization path: κ_1 vs minimum test loss. For a given regularization path, the test losses are obtained for models corresponding to 20 different values of λ equally spaced in the range of $[10^{-4},\ 10^{3}]$ (same as in case of no regularization path experiments). Notice that the models are trained using the regularization path algorithm. For each path, the minimum test loss is the minimum over the 20 different values. 132

Chapter 1

Introduction

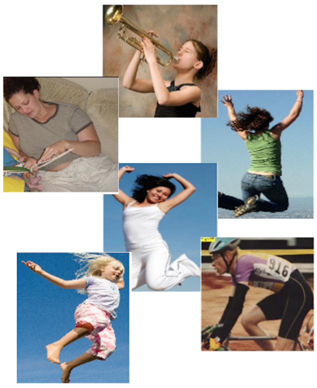
The main focus of this thesis is to develop *high-order inference* and *machine learning* algorithms to process visual and medical data in order to extract useful information. We begin with two challenging tasks in order to develop intuition about the need for inference and learning algorithms. The first task that we consider is *image denoising and inpainting* as shown in the Figure 1.1. Given an image with added noise and obscured regions (regions with missing pixels) as shown in Figure 1.1a, the problem is to automatically denoise the image and fill the obscured regions such that it is consistent with the surrounding. Another task under consideration is 'learning to rank'. We give an example of the learning to rank task in the context of action classification as shown in Figure 1.2. Given a set of bounding boxes (refer Figure 1.2a), the task is to find an ordering of the bounding boxes based on their relevance with respect to a given action class such as 'jumping'.



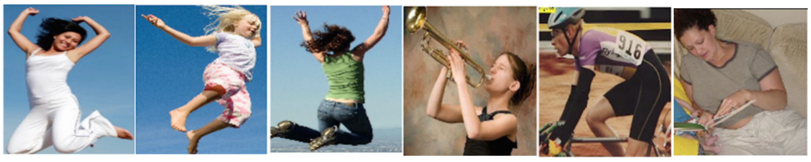
(a) Given Input. (b) Expected Output.

Figure 1.1: Image denoising and inpainting.

Let us first try to understand the complexity of the first task. Assume that the input image is gray scale of resolution $m \times n$. Therefore, each pixel in the image can be assigned any integer value in the range $[0, 255]$. In the simplest possible case, we assume that each pixel is independent of any other pixel in the image. Therefore, the denoising task can be solved



(a) Given Input.



(b) Expected output (a sorted list) based on the relevance to the action class 'jumping'.

Figure 1.2: An example of action classification problem for the action class 'jumping'. The 'input' is a set of bounding boxes. The desired 'ouput' is the sorted list in which all the bounding boxes with jumping action are ranked ahead of all the bounding boxes with non jumping action. Figure 1.2b shows one of the many possible outputs.

by processing each pixel independently. For example, thresholding based on some criterion such as average intensity value of the image pixels. This approach is computationally highly efficient as it requires only $N = m \times n$ operations. However, it is of little practical value for the following two reasons: (i) individually processing each pixel will not lead to a contextually meaningful output; and (ii) local information is not sufficient to solve the inpainting problem. Therefore, in order to get a meaningful solution, we must take into account the interdependencies between the pixels. Since each pixel can be assigned 256 possible values (also called labels), considering the interdependencies leads to a problem with $(256)^N$ possible combinations (or solutions), a much harder problem to solve. Finding the best solution out of these exponentially many possible combinations is known as the *inference* problem. In the case of vision related problems, inference involves thousands or sometimes millions of interdependent variables, which makes it very hard to solve. Normally, possible solutions are evaluated using a mathematical *energy function* that assigns a score to each possible solution. The solution corresponding to the minimum score is considered to be the optimal solution. In general, energy minimization problem is NP-HARD. However, careful modeling of the situation allows us to solve the problem in polynomial time using specialized and highly efficient algorithms. Some successful examples of such inference algorithms are graph cuts [74], belief propagation [105], and α-expansion [128]. A detailed discussion of these algorithms is given in chapter 2. They are very useful in many tasks such as semantic segmentation, pose estimation, stereo reconstruction, image inpainting, image registration, and many more. In this thesis, we develop a new family of high-order inference problems which we call *parsimonious labeling*, and propose very efficient algorithms for their optimization along with strong theoretical guarantees.

Now consider the 'learning to rank' task (Figure 1.2). Given a set of bounding boxes, the task is to rank them based on their relevance with respect to an action class such as 'jumping'. The quality of the ranking is evaluated based on a user defined metric such as average precision (a standard measure of the quality of a ranking). Let us assume that we are given a parametric energy function (similar to the previous example) which assigns a score to each bounding box. Given such scores, ranking reduces to sorting the bounding boxes based on their scores. Hence, the quality of the ranking depends entirely on the scores assigned to the bounding boxes by the energy function, which in turn depend on the parameters of the energy function. So in order to obtain scores that lead to high average precision, the parameters of the energy function must be chosen carefully. Normally, the energy function contains thousands of parameters and hand tuning these parameters is practically infeasible. In order to circumvent this problem, the standard approach is to learn the parameters values using sophisticated machine learning techniques[1]. In this thesis, we develop new machine learning methods for solving tasks that produce rankings as their final outcome. On top of this, we develop a new algorithm for the regularization path for structured SVM, that allows us to obtain the best possible mapping function in tasks that can be modeled using the well known structured SVM framework.

In the following sections, we briefly discuss the importance, complexity, and challenges of inference, machine learning (for ranking), and regularization path algorithms, then provide an outline of the thesis followed by the list of publications where the work in this thesis has previously appeared.

---

[1] Machine learning is the field of study that allows computers to learn from experiences. It evolved with time as a highly useful and effective blend of different fields such as computational statistics, mathematical optimization, and graph theory. More precisely, machine learning is the task of estimating a mapping function of the objects that we want to predict. In general supervised setting, a training data for which the prediction is known is used to learn this mapping. Few examples of highly successful applications of machine learning are search engines, character/voice recognition, spam filtering, news clustering, recommender systems, object classification, automatic language translation, and many more.

1.1 High-Order Inference

As discussed earlier, inference problem can involve thousands or millions of interdependent variables with solution spaces that are exponentially large in this number. To encode the semantics of the problem, we define a mathematical function known as the energy function that quantifies the quality of each possible solution. The solution with the lowest energy is the optimal choice. Mathematically, consider a random field defined over a set of random variables $\mathbf{y} = \{y_1, \cdots, y_N\}$, each of which can take a value from a discrete label set $\mathscr{L} = \{l_1, \cdots, l_H\}$. Furthermore, let $\mathscr{C}$ denote the set of maximal cliques that characterize the interactions between these variables. Each clique consists of a set of random variables that are connected to each other in the lattice. For example, in case of image denoising problem, the set of random variables are the pixels in the image, $N = m \times n$, and the label set is $\mathscr{L} = \{0, \cdots, 255\}$. To assess the quality of each output (or labeling) $\mathbf{y}$ we define an energy function as:

$$E(\mathbf{y}) = \sum_{i \in V} \theta_i(y_i) + \sum_{c \in \mathscr{C}} \theta_c(\mathbf{y}_c). \tag{1.1}$$

where $\theta_i(y_i)$ is the unary potential for assigning a label y_i to the i-th variable, and $\theta_c(\mathbf{y}_c)$ is the clique potential for assigning the labels $\mathbf{y}_c$ to the variables in the clique c. Once we have such a function, we need a method to evaluate each possible assignment in order to find the best one. The exhaustive search is practically infeasible. For example, in case of image denoising, if $m = n = 100$, then there are $(256)^{10,000}$ possible outcomes. One approach to solve this task is to use inference algorithms where we need to model the problem in a restricted yet useful setting to which a computationally efficient algorithm can be devised to find the optimal assignment among the exponentially many. Broadly speaking, there are two major steps to solving such inference problems: (i) modeling – selecting a suitable but tractable energy function (discussed in detail in chapter 2); (ii) optimization – finding the solution corresponding to the minimum energy (inference algorithm). The two steps goes hand-in-hand. Normally, we model the problem in such a manner that an algorithm can be devised to optimize it in polynomial time. Modeling defines restrictions on the potentials, interactions among the random variables (pixels in case of images), and optimality conditions for the inference algorithm. There are several situations in which careful modeling allows us to optimize the problem very efficiently. Let us have a quick look into some of these modeling strategies and their corresponding optimization algorithms.

- *Restricting the potentials:* In case the clique potentials are restricted to be submodular distance functions over the labels [1, 47], an optimal solution of the inference problem can be computed in polynomial time by solving an equivalent Graph cut problem [112]. A special case of this with two labels is well addressed in the seminal work [74]. Of course, restricting the potentials limits our modeling capabilities. However, submodular potential functions provide smoothness which is a desired property for many important tasks such as binary image restoration [14, 15, 51, 57], foreground background segmentation [13], medical image segmentation [11, 12], stereo depth recovery [14, 15, 56], and many other tasks.

- *Restricting the structure:* If the interactions between the random variables are restricted to form a tree (graph with no cycles), then the well known belief propagation method [105] can be used to obtain the *globally optimal* solution of the inference problem in polynomial time. Tree structures can still encode rich interactions useful to many vision tasks such as ‘pose estimation’ [134] and ‘object detection’ [34].

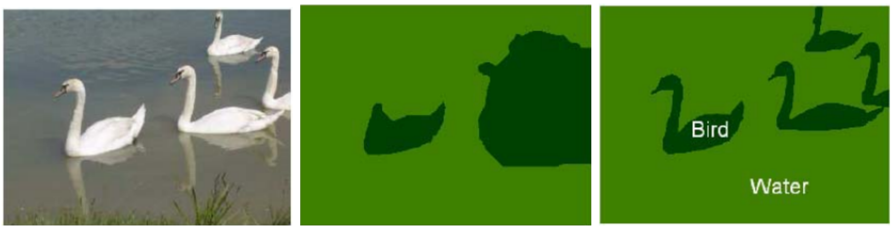
- *Compromising on optimality:* Sometimes it is not possible to obtain the globally optimal solution to the problem. In such situations, algorithms are designed to obtain a good local minimum with theoretical guarantees (multiplicative bounds). One example of such algorithm is the well known $\alpha-$expansion [15, 128] which obtains good local minima for problems in which the clique potential is a metric distance function over the labels.

The above mentioned algorithms are known to work well for pairwise interactions. However, higher order cliques (ones with more than two random variables) can model richer and more meaningful context, and have proven to be very useful in many vision tasks. one example representing the advantages of higher order interactions is shown in the Figure 1.3. However as the size of the cliques increases, the complexity of inference increases. Modeling and optimizing higher order energy functions is a highly challenging task. Technical restrictions on the clique potentials allows us to devise efficient algorithms for many high order problems. For example, the P^n-Potts model [66], label-cost based potential functions [28], and Co-occurrence statistics based potential functions [87]. Below we highlight two main challenges for devising inference problems over graphs with high order cliques.

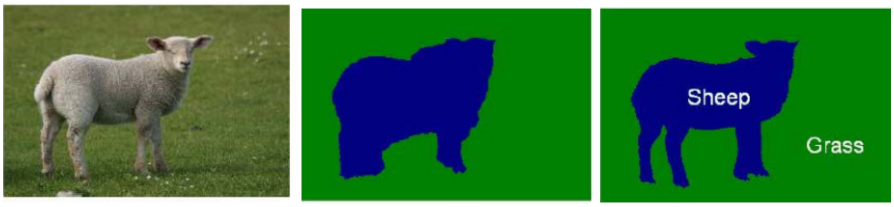
Challenges.

- *Defining new family of higher order clique potentials:* The known families of higher order clique potentials are either too restrictive with limited modeling capabilities or their optimization algorithms are inefficient and do not provide good local minima with theoretical guarantees. Therefore, it is challenging to define new sets of clique potentials with higher modeling capabilities that are useful for vision tasks and that support efficient optimization algorithms. For example, *parsimony* is a desirable property for many vision tasks. Parsimony refers to using fewer labels, which in turn provides smoothness. On one hand, the P^n-Potts model provides parsimony but it is too rigid and sometimes gives over-smooth results. On the other hand, Co-occurrence statistic based potential functions [87] are much more expressive but their optimization algorithm does not provide any theoretical guarantees.

- *Proposing efficient algorithms with theoretical guarantees:* As discussed earlier, for a given family of high order clique potential functions, it is a challenging task to provide efficient algorithms for the optimization. For example, the SoSPD [40] allows us to use arbitrary clique potentials, but its optimization algorithm is practically inefficient and is not advisable to use beyond cliques of size ten. Similarly, Co-occurrence statistic based potential functions [87] allow us to use clique potentials that encode *parsimony* and their optimization is efficient, allowing larger cliques (beyond clique size of 1200). However, the optimization algorithm of [87] does not provide any theoretical guarantees, thus, the local minima obtained may or may not be useful.



(a) Input (b) Pairwise (c) High Order



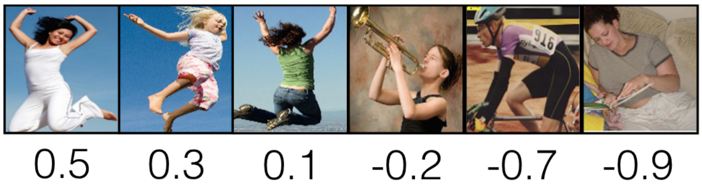
(d) Input (e) Pairwise (f) High Order

Figure 1.3: An illustration of the value of high-order interactions in solving segmentation problems. *Image source [67].*

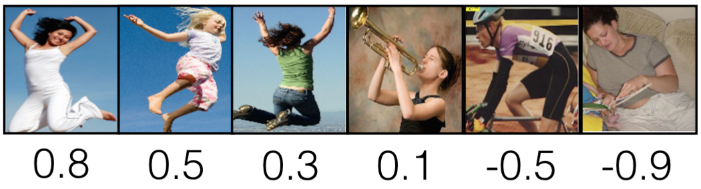
1.2 Learning to Rank

Many computer vision tasks require the development of automatic methods that sort or rank a given set of visual samples according to their relevance to a query. For example, consider the problem of action classification (or more precisely action ranking). The input is a set of samples corresponding to bounding boxes of persons, and an action such as ‘jumping’. The desired output is a ranking where a sample representing a jumping person is ranked higher than a sample representing a person performing a different action. An example of such problem is shown in the Figure 1.2. Other related problems include image classification (sorting images according to their relevance to a user query) and object detection (sorting all the windows in a set of images according to their relevance to an object category). Once the ranking is obtained, it can be evaluated using average precision (AP), which is a widely used measure of the quality of a ranking. The most common approach of solving this problem is to train a binary classifier (often a support vector machine (SVM)). SVM optimizes an upper bound on an accuracy based loss function to learn a parameter vector. At test time, the learned parameter vector is used to assign scores (confidence of being relevant) to each sample. The samples are then sorted based on these scores to get the final ranking. However, this approach suffers from following two drawbacks. First, average precision and accuracy are not the same measures. Figure 1.4 shows some examples where AP and accuracy are different. It is clear from these examples that a classifier that provides high accuracy may not give high average precision. In another words, optimizing accuracy may result in suboptimal average precision. Therefore, in a situation where the desired result is a ranked list, it is better to optimize an average precision based loss function (a measure of the quality of the ranking). Unfortunately, unlike accuracy based loss function, as used in SVM, the average precision is non decomposable and thus hard to optimize. AP-SVM [137] poses the problem as a special case of structured SVM and provides efficient algorithm for optimizing an upper bound on average precision based loss function. The second drawback is that SVM can use only first order information. Whereas there is a great deal of high order information in many vision related tasks (and other tasks as well). For example, in action classification, people in the image are often performing same action (see Figure 1.5 for a few examples). In object detection, objects of the same category tend to have similar aspect ratios. In pose estimation, people in the same scene tend to have similar poses (e.g. sitting down to watch a movie). In document retrieval, documents containing the same or similar words are more likely belong to the same class. Therefore, we need to use richer machine learning frameworks such as structured SVM, that are capable of encoding the high order interactions.

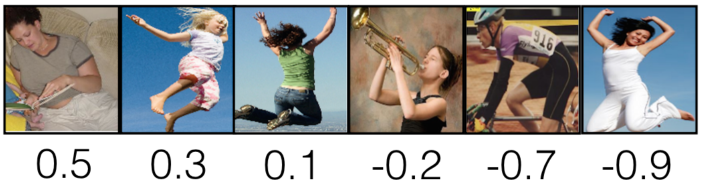
Challenge. The challenge lies in developing a framework capable of optimizing average precision while using high order information. On the one hand, a special case of the structured SVM (AP-SVM) can optimize average precision based loss functions, but it does not allow us to use high order information. On the other hand, when structured SVM is used to encode high



(a) AP/Acc = 1.0/1.0



(b) AP/Acc = 1.0/0.83



(c) AP/Acc = 0.55/0.66

Figure 1.4: Differences between AP and accuracy. The relevant class here is 'jumping'. The scores shown are obtained from the classifier. A positive score implies a relevant class ('jumping') and a negative score implies an irrelevant class ('non jumping'). The bounding boxes are sorted based on the scores to obtain the ranking. An ideal ranking would have all the positive (or relevant) examples ahead of all the negative (irrelevant) ones. Notice that the accuracy is nothing but the fraction of misclassifications.

order information, it only allows us to optimize accuracy based (or similar but decomposible) loss functions.

1.3 Regularization Path

In the previous section we talked about Structured SVM type frameworks for learning tasks. The objective function of an SSVM is parametric, depends on a particular regularization parameter λ (also referred as C, inversely proportional to λ), which controls the trade-off between the model complexity and an upper bound on the empirical risk (detailed in chapter 2). The value of the regularization parameter has a significant impact on the performance and the generalization of the learning method. Thus, we must choose it very carefully in order to obtain the best model. Finding an appropriate value for the regularization parameter often requires us to tune it, but lack of knowledge about the structure of the regularization problem compels us to cross validate it over the entire parameter space, which

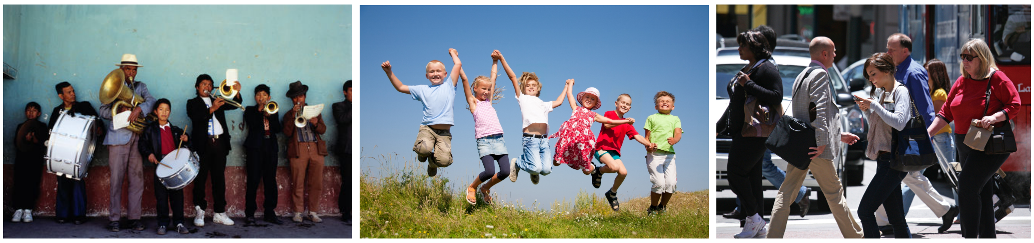


Figure 1.5: The example of high-order information in case of action classification. Notice that people in the same image often perform the same action. This can be one of the hypotheses that encode meaningful high-order interactions.

is practically infeasible owing to its computational cost. To circumvent this, the standard approach is to resort to a sub optimal solution by cross validating a small set of regularization parameter values on a given training dataset. Doing this is tedious and quickly becomes infeasible as we increase the number of regularization values tested. Therefore, we need an efficient algorithm to obtain the entire regularization path of SSVM. By definition the regularization path is the set of solutions for all possible values of the regularization parameter in the parameter space [29]. Obtaining the regularization path for any parametric model implies exploring the whole regularization parameter space in a highly efficient manner to provide the optimal learned model for any given value of $\lambda \in [0, \infty]$. This allows us to efficiently obtain the best possible model. The key idea behind the algorithm is to break the regularization parameter space into segments such that learning an ε-optimal model for any value of λ in a given segment guarantees that the same learned model is ε-optimal for all values of λ in the segment.

Challenges.

- *Finding the segments (or the breakpoints):* As mentioned earlier, the idea behind regularization path is to break the regularization parameter space $\lambda \in [0, \infty]$ into segments. This can be achieved by finding breakpoints or kinks such that if the learned model is ε-optimal for any $\lambda \in [\lambda_k, \lambda_{k-1})$, then it is ε-optimal for all $\lambda \in [\lambda_k, \lambda_{k-1})$. One of the challenge is to find these breakpoints in an efficient manner. The other challenge is to reduce the needed number of breakpoints.

- *Efficiently optimizing at each breakpoint:* Each breakpoint defines a new segment. Therefore, at each breakpoint we need to update (optimize) the learned model so that it is ε-optimal for the given segment. In order to make this practically feasible, we must devise an algorithm that can be warm-started using the solution of the previous segment for faster convergence.

1.4 Thesis Outline

Chapter 2 is a review of structured SVM and related inference algorithms. Specifically, we discuss the SSVM algorithm with examples, various inference algorithms such as graph cuts, α-expansion, metric labeling, and P^n-Potts model, and different optimization algorithms for SSVM. Finally, we talk about latent SSVM and the corresponding concave-convex procedure for the problems related to the weakly supervised settings.

In chapter 3, we describe a new family of inference problems that we call *parsimonious labeling*, which handle all the challenges discussed in section 1.1. Parsimonious labeling can be seen as a high order extension of the famous metric labeling problem. We also propose the hierarchical P^n-Potts model and show how parsimonious labeling can be modeled as a mixture of hierarchical P^n-Potts models. Finally, we propose an efficient algorithm with strong theoretical guarantees for the optimization of parsimonious labelings and show its efficacy on the challenging tasks of stereo matching and image in-painting.

In chapter 4, we propose a new learning framework, High-Order Average Precision SVM (HOAP-SVM), capable of optimizing average precision while using high-order information (see the challenges discussed in section 1.2). We show the efficacy of HOAP-SVM on the task of action classification.

In chapter 5, we present a new algorithm (SSVM-RP) to obtain ε-optimal regularization paths for SSVM. We propose intuitive variants of the Block-Coordinate Frank-Wolfe algorithm for the faster optimization of SSVM-RP. In addition to this, we propose a principled algorithm for optimizing SSVM with additional box constraints. Finally, we propose a regularization path algorithm for SSVM with additional positivity/negativity constraints. *All the research presented in this chapter were conducted under the supervision of Dr. Simon Lacoste-Julien at the SIERRA Team of INRIA (Paris) during my visit from* 15^{th} *June 2015 to* 15^{th} *September 2015.*

Finally, in appendix A, we present a novel weakly supervised algorithm for learning context specific metric aggregations for the challenging task of 3D-3D deformable registration. We demonstrate the efficacy of our framework using three challenging datasets related to medical imaging.

1.5 List of publications

Published in International Conferences and Journals

1. *Learning-Based Approach for Online Lane Change Intention Prediction*; **Puneet Kumar**, M. Perrollaz, S. Lefevre, C. Laugier; In IEEE Intelligent Vehicle Symposium (IV) 2013.
2. *Discriminative parameter estimation for random walks segmentation*; P. Y. Baudin, D. Goodman, **P. K. Dokania** , N. Azzabou, P. G. Carlier, N. Paragios, M. Pawan Kumar; In MICCAI 2013.
3. *Learning to Rank using High-Order Information*; **P. K. Dokania**, A. Behl, C. V. Jawahar, M. P. Kumar; In ECCV 2014.
4. *Parsimonious Labeling*; **P. K. Dokania**, M. P. Kumar; In ICCV 2015.
5. *Partial Linearization based Optimization for Multi-class SVM*; P. Mohapatra, **P. K. Dokania**, C. V. Jawahar, M. P. Kumar; In ECCV 2016.
6. *Minding the Gaps for Block Frank-Wolfe Optimization of Structured SVMs*; A. Osokin, JB Alayrac, I. Lukasewitz, **P. K. Dokania**, S. Lacoste-Julien; In ICML 2016.
7. *Rounding-based Moves for Semi-Metric Labeling*; M. P. Kumar, **P. K. Dokania**; In JMLR 2016.

Under Submission

1. *Deformable Registration through Learning of Context-Specific Metric Aggregation*; E. Ferrante, **P. K. Dokania**[2], N. Paragios.

---

[2] Joint first author with E. Ferrante.

Chapter 2

Review of Structured SVM and related Inference Algorithms

2.1 Motivation

Structured output prediction [88, 122, 126] is one of the key problems in the machine learning and computer vision community. It deals with learning a function f that maps the input space $\mathscr{X}$ (patterns or vectors) to a complex output space $\mathscr{Y}$ (graphs, trees, strings, or sequences). In other words, structured prediction is the problem of learning prediction functions that take into account the interdependencies among the output variables. In order to give better insight into the problem of structured prediction, we consider the task of recognizing handwritten words [63]. An example is shown in Figure 2.1. The problem is to predict the word given bounding boxes around letters. Below we discuss two possible solutions to this problem (from a discriminative classification point of view) and progressively build intuition to understand the complexity and importance of structured prediction. Before discussing the solutions, we define some notation for the purpose of clarity. We denote the number of letters in the word as p. In our example, $p = 4$. The input, which in this case is a collection of segmented bounding boxes in an image, is represented as $\mathbf{x} = \{x_1, x_2, x_3, x_4\}$, where x_i denotes the i-th box. Similarly, we define the output, which is a word, as a collection of letters $\mathbf{y} = \{y_1, y_2, y_3, y_4\} \in \mathscr{Y}$. Here each output variable can be assigned any letter from the alphabet. In other words, $y_i \in \mathscr{L} = \{a, \cdots, z, A, \cdots, Z\}$ for all $i \in \{1, 2, 3, 4\}$. Thus, the final output space is $\mathscr{Y} = \mathscr{L}^p$, where $p = 4$. For example, for the input image in the Figure 2.1, the ground truth output is 'ROSE', therefore, $y_1 = R$, $y_2 = O$, $y_3 = S$, and $y_4 = E$. Using this notation we discuss two possible solutions.

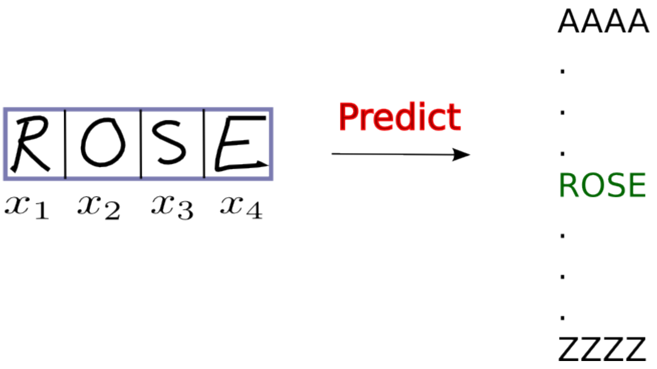


Figure 2.1: An example of handwritten word recognition. The input is an image of a handwritten word $\mathbf{x} = \{x_1, x_2, x_3, x_4\}$, where each x_i is the bounding box around each character of the word. The task is to predict the correct word, which in this case is 'ROSE'.

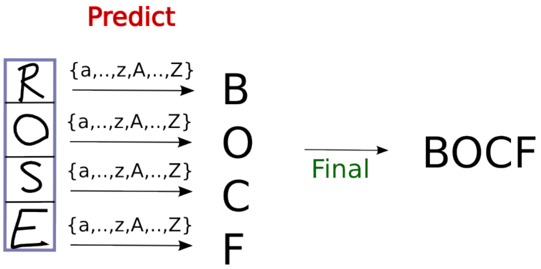


Figure 2.2: *Solution 1:*Each bounding box is classified separately. The set of classes (or labels) is $\{a, \cdots, z, A, \cdots, Z\}$. The final solution is the collection of the predictions corresponding to each bounding box. The number of classifiers required is $|\mathscr{L}|(=52)$, therefore, the solution is practically very efficient. However, it does not take account of the interdependencies between the output variables.

- **Solution 1:** One possible way to solve the above problem is to learn different prediction functions f_i for each letter $i \in \mathscr{Y} = \{a, \cdots, z, A, \cdots, Z\}$. This requires us to learn $|\mathscr{L}|(=52)$ different prediction functions. Given these functions and the input image $\mathbf{x}$, the prediction task reduces to independently classifying each letter (refer Figure 2.2). Thus, the predicted word $\hat{\mathbf{y}} = \{\hat{y}_1, \hat{y}_2, \hat{y}_3, \hat{y}_4\}$ is the collection of the different predicted letters where each predicted letter is $\hat{y}_j = \text{argmax}_{i \in \mathscr{L}} f_i(x_j), \forall j$. The method is very simple and computationally efficient. However, in effect, it reduces the search space from $\mathscr{L}^p$ to $p\mathscr{L}$. In other words, it does not take into account the interdependencies between the different letter predictions (the variables in the output space). For example, in English language, the likelihood of the letter 'F' occurring next to 'C' is very low while the likelihood of the letter 'S' occurring next to 'O' is high. Considering the interdependencies between the letters of words seems very promising to improve the

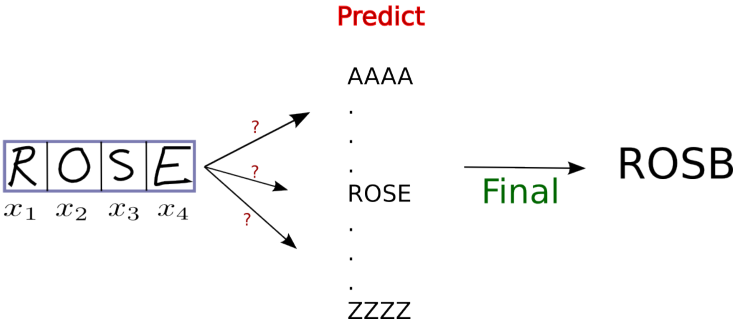


Figure 2.3: *Solution 2:*Each *word* is classified separately. The set of classes (or labels) is $\{aaaa, \cdots, zzzz, AAAA, \cdots, ZZZZ\}$. Note that the number of prediction functions required is $(52)^4 \approx 7$ Million. Therefore, this solution is practically infeasible even for moderate length words.

prediction accuracy. However, it may lead to a computationally expensive task, as discussed in the next approach.

- **Solution 2:** In order to explicitly take the interdependencies between the output variables into account, we can learn a prediction function f_w for each possible word $w \in \mathscr{W}$, where $\mathscr{W} = \{aaaa, baaa, \cdots, ZZZZ\}$ denotes the set of all possible words of length $p = 4$. Given all these prediction functions, the task reduces to finding the function that best describes the input $\mathbf{x}$ (refer to Figure 2.3). In other words, the predicted word is $\hat{\mathbf{y}} = \text{argmax}_{w \in \mathscr{W}} f_w(\mathbf{x})$. At first glance this method seem reasonable. However, a closer look reveals that the size of the set $\mathscr{W}$ is $|\mathscr{L}|^p$, which in this case is $(52)^p$ (total number of words of length p that can be formed using 52 letters). Another difficulty is to get a dataset for each word in order to train the corresponding classifier. Therefore, this approach leaves us with exponential numbers of prediction functions and the need for huge datasets to learn them all. Thus, it quickly becomes infeasible even for moderate length words. For example, $p = 4$ requires us to test almost 7 million different functions for prediction.

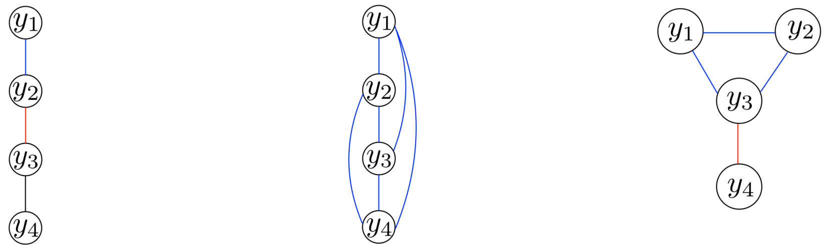
Clearly, solution 1 is computationally feasible but can not incorporate the interactions between the output variables, while solution 2 can incorporate interactions but it becomes computationally infeasible for even a small number of output variables. In this chapter, we discuss a third solution, Structured SVM (SSVM) [126], which is a compromise between the two aforementioned ones.

2.2 Structured Output SVM

A structured output SVM (SSVM), parametrized by $\mathbf{w}$, provides a linear *prediction* rule to obtain a structured output $\mathbf{y} \in \mathscr{Y}$ (strings, graphs, trees, or sequences) for a given input $\mathbf{x} \in \mathscr{X}$ (patterns or vectors). In order to *learn* the parameters $\mathbf{w}$, SSVM minimizes an objective function that models the trade-off between the model complexity and the empirical risk. This section gives an overview of these two tasks of prediction and parameter learning for SSVM. The latter sections discuss them in detail.

In order to understand the prediction and learning tasks, we require a mapping function $\Phi(\mathbf{x},\mathbf{y}) : \mathscr{X} \times \mathscr{Y} \rightarrow \mathbb{R}^d$, known as the *joint feature map*. In some works it is also referred as the *combined feature representation* or the *joint feature vector*. The joint feature map maps the input-output pair into a new d dimensional discriminative space that encodes the relationships between the output variables. More precisely, given an input $\mathbf{x}$, we get a d dimensional feature vector for each $\mathbf{y} \in \mathscr{Y}$. The joint feature map can also be seen as a *virtual machine* for feature generation where the discrimination between different inputs is encoded based on the interactions between the output variables. Given the mapping function, the prediction and learning tasks are performed in the new feature space. However, as will be seen shortly, we seek a special type of joint feature map that is restricted (discussed shortly) in order to allow us to perform the prediction and learning in a computationally efficient manner. Of course, by restricting the joint feature map we lose some modeling capabilities (discussed in detail in section 2.3). However, the restrictions still allow us to use SSVM practically for several vision related tasks.

Let us now discuss the restrictions on the joint feature map. Recall that each output $\mathbf{y}$ is a structured object consisting of the output variables. For SSVM, the types of interactions foreseen between the output variables need to be known a priori. Let us represent the interactions between the output variables by the graph $G = (V,\mathscr{C})$, where $V = \{1,\cdots,N\}$ is the set of nodes and $\mathscr{C}$ is the set of maximal cliques in the structured output object $\mathbf{y}$. In the case of the above handwritten word recognition example, $N = p = 4$. Using a 'Markovian' style argument [7], the joint feature map is assumed to decompose over the nodes and the cliques of the output structure represented by the graph G. Thus, $\Phi_i(x_i,y_i) \in \mathbb{R}^{d_i}$ and $\Phi_c(\mathbf{x}_c,\mathbf{y}_c) \in \mathbb{R}^{d_c}$ denotes the joint feature map corresponding to the node i and the clique c, respectively. The final joint feature map $\Phi(\mathbf{x},\mathbf{y})$ is formed using the node feature maps Φ_i and the clique feature maps Φ_c. In this work we will assume that Φ is a concatenation of Φ_i and Φ_c for all $i \in V$ and $c \in \mathscr{C}$, so $d = \sum_{i\in V} d_i + \sum_{c\in\mathscr{C}} d_c$. However, it is also possible to define the joint feature map in other ways. In order to further clarify the construction of the joint feature map, consider the following example.



(a) Interaction 1 (b) Interaction 2 (c) Interaction 3

Figure 2.4: Few examples of the different possible interactions between the output variables. Variables in the edges with same colors form a clique. For example, in case of Figure 2.4c, set of cliques is: $\mathscr{C} = \{c_1, c_2\}$, where $c_1 = \{y_1, y_2, y_3\}$ and $c_2 = \{y_3, y_4\}$. Many other types of interactions are also possible.

An example joint feature map. We now give an example of a joint feature map for the handwritten word recognition problem discussed in the section 2.1. Figure 2.4 shows three different types of possible interactions between the output variables. Many others are also possible. However, the interactions have to be known a priori, and should be decided based on the task under consideration. In our example, we focus on the case shown in Figure 2.4c but the same concepts can be applied to get the joint feature map for any given interactions. Recall that each output variable y_i is associated with the input bounding box x_i corresponding to a letter in the word. Let $\phi : x_i \to \mathbb{R}^h$ be a function that maps the given input x_i corresponding to the i-th bounding box (or node) into an h dimensional feature vector. A few examples of well known feature vectors are GIST [101], HOG [25], and BOW [26], but normally a task specific feature vector is chosen. Each output variable can be assigned to any of the following $|\mathscr{L}|$ classes (or labels) $y_i \in \mathscr{L} = \{a, \cdots, z, A, \cdots, Z\}$. For clarity we denote each class by its corresponding index in the label set. Therefore, without loss of generality, we can represent the classes as $y_i \in \mathscr{L} = \{1, \cdots, 26, 27, \cdots, 52\}$. Let $\mathbf{e}_i$ be the i-th canonical basis element of $|\mathscr{L}|$ dimensional Euclidean space. In another words, the vector $\mathbf{e}_i$ has i-th entry of one and all of the remaining entries are zeros. Under this setting, one way of defining the joint feature map corresponding to the nodes is as follows:

$$\Phi_i(x_i, y_i) = \mathbf{e}_{y_i} \otimes \phi(x_i) \in \mathbb{R}^{d_i}, d_i = h|\mathscr{L}|, \forall i \in V, \forall y_i \in \mathscr{L}. \tag{2.1}$$

where, $\otimes$ is the Kronecker product. As an example, if $\mathbf{e} = (0, 1)$ and $\phi(x) = (7, 4, 9)$, then $\mathbf{e} \otimes \phi(x) = (0, 0, 0, 7, 4, 9)$. Notice that, for a particular x_i, $\Phi_i(x_i, y_i)$ for all $y_i \in \mathscr{L}$ are perpendicular to each other. Thus, this particular form of the $\Phi_i(x_i, y_i)$ is discriminative with respect to the classes. One can come up with other criteria depending on the task. We now proceed to design a discriminative joint feature map corresponding to the cliques. We again take an intuitive route and define the joint feature map as follows:

$$\Phi_c(\mathbf{x}_c,\mathbf{y}_c) = \begin{cases} \mathbf{0}_{h\times 1} & \text{if } y_i = l \in \mathscr{L}, \forall i \in c, \\ var(\{\phi(x_i)\}_{\forall i \in c}) \in \mathbb{R}^h & \text{otherwise.} \end{cases} \tag{2.2}$$

where, $var(\{\phi(x_i)\}_{\forall i \in c})$ is the variance of the set of feature vectors corresponding the nodes (or bounding boxes) in the clique c. Equation (2.2) states that $\Phi_c(\mathbf{x}_c,\mathbf{y}_c)$ is a zero vector if all of the nodes in the clique are assigned the same label, otherwise, it is the variance of the feature vectors. Using the variance captures the spread of the node's features. If all of them are similar, the variance and thus the entries of $\Phi_c(\mathbf{x}_c,\mathbf{y}_c)$, will be small. Depending on requirements, we can enforce either smoothness or diverse labels by modifying the joint feature map parameters. The final joint feature map $\Phi(\mathbf{x},\mathbf{y}) \in \mathbb{R}^d$ is the concatenation of the node and the clique joint feature maps:

$$\begin{aligned} \Phi(\mathbf{x},\mathbf{y}) = [&\Phi_1(x_1,y_1);\Phi_2(x_2,y_2);\Phi_3(x_3,y_3);\Phi_4(x_4,y_4); \\ &\Phi_{c1}(\mathbf{x}_{c1},\mathbf{y}_{c1});\Phi_{c2}(\mathbf{x}_{c2},\mathbf{y}_{c2})] \in \mathbb{R}^d. \end{aligned} \tag{2.3}$$

where, $d = (4h|\mathscr{L}| + 2h) = 210h$.

Now that we have a notion of the design and properties of joint feature maps, we discuss the prediction and parameter learning of SSVM with a given joint feature map.

2.2.1 Prediction for the SSVM

The SSVM prediction for a given input $\mathbf{x}$ is obtained by maximizing a linear function parameterized by $\mathbf{w}$ on the joint feature map of $\mathbf{x}$. So, the predicted output $\hat{\mathbf{y}}$ is obtained as follows:

$$\hat{\mathbf{y}} = f_{\mathbf{w}}(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y}\in\mathscr{Y}} \mathbf{w}^\top \Phi(\mathbf{x},\mathbf{y}). \tag{2.4}$$

Note that the output space is exponentially large, $\mathscr{Y} \in |\mathscr{L}|^N$. For example, for our handwritten word recognition example, $|\mathscr{L}| = 52$ and $N = p = 4$. So, the number of possible outputs is $(52)^4$. Below we will see how the decomposability of the joint feature map Φ helps us to efficiently predict an output. Recall that $\Phi_i(x_i,y_i) \in \mathbb{R}^{d_i}, \forall i \in V$ and $\Phi_c(\mathbf{x}_c,\mathbf{y}_c) \in \mathbb{R}^{d_c}, \forall c \in \mathscr{C}$ represent the joint feature maps decomposed over the nodes V and the cliques $\mathscr{C}$ of the graph G that encodes the interdependencies between the output variables. Associated with each decomposed joint feature map we have parameter vectors $\mathbf{w}_i \in \mathbb{R}^{d_i}$ and $\mathbf{w}_c \in \mathbb{R}^{d_c}$. Therefore under this setting, the SSVM prediction problem can be written as:

$$\begin{aligned}\hat{\mathbf{y}} = \underset{\mathbf{y}\in\mathscr{Y}}{\operatorname{argmax}}\, \mathbf{w}^\top \Phi(\mathbf{x},\mathbf{y}) &= \underset{\mathbf{y}\in\mathscr{Y}}{\operatorname{argmax}} \Big(\sum_i \mathbf{w}_i^\top \Phi_i(x_i,y_i) + \sum_c \mathbf{w}_c^\top \Phi_c(\mathbf{x}_c,\mathbf{y}_c) \Big) \\ &= \underset{\mathbf{y}\in\mathscr{Y}}{\operatorname{argmax}} \Big(\sum_i \bar{\theta}_i(x_i,y_i) + \sum_c \bar{\theta}_c(\mathbf{x}_c,\mathbf{y}_c) \Big). \end{aligned} \tag{2.5}$$

where, $\bar{\theta}_i(x_i,y_i) = \mathbf{w}_i^\top \Phi_i(x_i,y_i)$ and $\bar{\theta}_c(\mathbf{x}_c,\mathbf{y}_c) = \mathbf{w}_c^\top \Phi_c(\mathbf{x}_c,\mathbf{y}_c)$ are known as the potential functions. The prediction problem in equation (2.5) is a combinatorial optimization problem popularly known as the *inference problem*. The inference problem is NP-HARD in general [74]. However, restricting the graph structure or the potentials (or both) leads to some specific forms of the inference problem that can be solved very efficiently in polynomial time. For example, restricting the graph structure to a tree allows us to use the well known belief propagation algorithm [105] to solve the inference problem in polynomial time. Similarly, restricting the potentials to be pairwise and submodular allows us to use Graph cuts [74] to solve the inference problem in polynomial time. A detailed discussion of different types of restrictions on the potentials, the graph structure, and their corresponding inference algorithms is given in the section 2.3. For now let us assume that the SSVM prediction is the solution of the inference problem.

2.2.2 Learning parameters for the SSVM

In the previous sections we talked about designing the joint feature map $\Phi(.,.)$ and saw how a decomposable joint feature map allows us to solve the prediction problem efficiently in polynomial time by modeling it as an inference problem. However, we assumed that the parameter vector $\mathbf{w}$ was known. We now discuss learning the parameter vector $\mathbf{w}$ from a given training dataset.

Let $\rho(\mathbf{x},\mathbf{y})$ be the unknown joint probability distribution of the input and output pairs $\mathbf{x} \in \mathscr{X}$ and $\mathbf{y} \in \mathscr{Y}$. Furthermore, let $\Delta(\mathbf{y},\bar{\mathbf{y}}) \in \mathbb{R}_{\geq 0}$ be a loss function (also known as the 'cost' function) between any two outputs $\mathbf{y}$ and $\bar{\mathbf{y}}$. We assume that $\Delta(\mathbf{y},\mathbf{y}) = 0$. It quantitatively measures the difference between the two given arguments. The goal is to learn a prediction function $f_\mathbf{w} : \mathbf{x} \rightarrow \hat{\mathbf{y}}$ such that the prediction $\hat{\mathbf{y}}$ is as close as possible to the ground truth label $\mathbf{y}$ for any input $\mathbf{x}$. In another words, the prediction function should minimize the expected risk (or expected prediction loss) over the entire joint distribution $\rho(\mathbf{x},\mathbf{y})$, where the expected risk for any given parameter $\mathbf{w}$ is defined as follows:

$$R(\mathbf{w}) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\rho(\mathbf{x},\mathbf{y})} \Delta(\mathbf{y}, f_\mathbf{w}(\mathbf{x})) = \int_{\mathscr{X}\times\mathscr{Y}} \Delta(\mathbf{y}, f_\mathbf{w}(\mathbf{x}))\, d\rho(\mathbf{x},\mathbf{y}) \tag{2.6}$$

Note that the distribution $\rho(\mathbf{x},\mathbf{y})$ is unknown. So there is no way to minimize the true expected risk $R(\mathbf{w})$. However, we can approximate the expected risk by the empirical risk over the training set. Given a sufficiently large training dataset of n input-output pairs, $\mathscr{D} = \{(\mathbf{x}_i,\mathbf{y}_i)\}_{i=\{1,\cdots,n\}}$, drawn *i.i.d.* from the distribution ρ, the average empirical risk is defined as [127]:

$$R_{emp}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\Delta(\mathbf{y}_i, f_{\mathbf{w}}(\mathbf{x}_i)) \tag{2.7}$$

Notice that the empirical risk can be calculated without the knowledge of the distribution ρ. Therefore, it is robust to the lack of information about the data distribution. There is a classic theory generalizing the law of large numbers that tells us the conditions under which approximating expected risk by the empirical risk is useful for learning the parameters $\mathbf{w}$ [127]. Finally, the objective function that minimizes the approximate empirical risk to learn the parameters can be written as:

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} R_{emp}(\mathbf{w}) = \operatorname*{argmin}_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n}\Delta(\mathbf{y}_i, f_{\mathbf{w}}(\mathbf{x}_i)) \tag{2.8}$$

In section 2.4, we talk about the difficulty of directly optimizing the above objective function and discuss the possible solutions.

2.3 The Inference Problem

As shown in the equation (2.5), given the SSVM parameters $\mathbf{w}$ and an input $\mathbf{x}$, the prediction can be obtained by solving an inference problem. In this section, we discuss different types of inference algorithms, focusing on those that will be used throughout the rest of the thesis. Hereafter, we will interchangeably use the terms ‘prediction’ and ‘inference’. Before moving ahead, let us first define a function $E(\mathbf{y};\mathbf{x})$, known as the *energy function*:

$$E(\mathbf{y};\mathbf{x}) = \sum_i \theta(y_i;x_i) + \sum_c \theta(\mathbf{y}_c;\mathbf{x}_c). \tag{2.9}$$

where, $\theta(y_i;x_i) = -\bar{\theta}_i(x_i,y_i), \forall i \in V$ and $\theta(\mathbf{y}_c;\mathbf{x}_c) = -\bar{\theta}_c(\mathbf{x}_c,\mathbf{y}_c), \forall c \in \mathscr{C}$. Therefore, the *inference* problem (2.5) can be equivalently written as follows:

$$\hat{\mathbf{y}} = \operatorname*{argmin}_{\mathbf{y}\in\mathscr{Y}} E(\mathbf{y};\mathbf{x}) = \operatorname*{argmin}_{\mathbf{y}\in\mathscr{Y}} \Big(\sum_i \theta(y_i;x_i) + \sum_c \theta(\mathbf{y}_c;\mathbf{x}_c)\Big). \tag{2.10}$$

Each $\mathbf{y} \in \mathscr{Y}$ is also known as a *labeling*. In this context, the *inference problem* is also known as the *labeling problem* where the objective is to find the labeling that corresponds to the minimum energy with respect to the energy function $E(\mathbf{y};\mathbf{x})$. Note that if $y_i \in \mathscr{L}$, $|\mathscr{Y}| = |\mathscr{L}|^N$, which is exponential in the size of the label set or the number of classes. This indicates that the inference problem may be computationally challenging. Indeed, in general settings, the inference problem is NP-HARD [74]. However, for some restricted classes of potential functions and restricted graph structures, approximate or exact inferences can be obtained in polynomial time [15, 74, 105, 128]. Before delving into the details of this, we briefly discuss the effects of the necessary compromises or restrictions on graph structure and potentials, and of any approximate minimization on the applicability of the SSVM.

Restricting the structure. If the interactions between output variables are restricted to form a tree (a graph with no cycles), then the well known belief propagation algorithm [105] can be used to obtain the *globally optimal* solution of the prediction problem (2.10) in polynomial time. Of course, restricting the structure limits our modeling capabilities as we can not explore all of the possible interactions between the output variables. However, tree structure can still encode quite rich interactions that are useful for many vision tasks such as 'pose estimation' [134] and 'object detection' [34].

Restricting the potentials. If the cliques are restricted to size 2 (edges only) and the clique potentials are restricted to be submodular distance functions over the labels [1, 47], an optimal solution of the prediction problem (2.10) can computed in polynomial time by solving an equivalent Graph cut problem [112]. A special case of this with two labels is well addressed in the seminal work [74]. Again restricting the potentials limits our modeling capabilities. However, submodular potential functions are proven to be very useful in many computer vision tasks. They can encourage smoothness, which is a desirable property for tasks such as binary image restoration [14, 15, 51, 57], foreground background segmentation [13], medical imaging [11, 12], stereo and motion [14, 15, 56], and many others. So even with such restrictions on potentials SSVM is still useful for many important vision tasks.

Compromising on optimality. If the cliques are of size 2 (edges only) and the clique potential function is a metric distance function over the labels $\mathscr{L}$, then the well known $\alpha-$expansion algorithm [15, 128] can be used to obtain a locally optimal solution of the prediction problem (2.10) in polynomial time. Such restrictions on the potentials are very useful for tasks such as image denoising and stereo matching. However, compromising on global optimality may lead to poor results in some cases.

We now discuss several inference algorithms in detail. In order to separate pairwise interactions (maximum clique size two) from the higher-order interactions (maximal cliques of size more than two), we define the pairwise energy function as follows (for the purpose of clarity):

$$E(\mathbf{y};\mathbf{x}) = \sum_{i} \theta_i(y_i) + \sum_{ij\in\mathscr{E}} \theta_{ij}(y_i,y_j). \tag{2.11}$$

where, $\mathscr{E}$ is the set of edges, $\theta_i(y_i) := \theta(y_i;x_i)$, and $\theta_{ij}(y_i,y_j) := \theta_{ij}(\{y_i,y_j\};\{x_i,x_j\})$. Therefore, $\theta_i(0)$ denotes the potential $\theta_i(0;x_i)$ when the i-th node is assigned the label 0. Similarly, for any edge $ij \in \mathscr{E}$, $\theta_{ij}(0,1)$ denotes the pairwise potential $\theta_{ij}(\{0,1\};\{x_i,x_j\})$ between nodes i and j when the node i is assigned the label 0 and the node j is assigned the label 1. In the following section, these short hand notations will be used to avoid clutter.

2.3.1 Graph cuts for submodular pairwise potential

Graph cut is a combinatorial optimization algorithm that can be used to solve the inference problem exactly (find globally minimum energy) if the potential function is a pairwise submodular distance function defined over the labels (discussed in section (2.3.1.1)) [74, 112]. Graph cuts have proven to be a very useful tool for optimizing energy functions that enforce piecewise smoothness [74]. They have been used widely in many computer vision tasks such as binary image restoration [14, 15, 51, 57], foreground background segmentation [13], scene reconstruction [73], medical imaging [11, 12], stereo and motion [14, 15, 56, 72], and many others.

In this section, we discuss graph cuts in the context of computer vision applications. However, the same concepts are valid for other fields as well. We focus on the following aspects — (i) submodular pairwise energy functions, (ii) the s-t cut, (iii) the relationship between s-t MINCUT and the *inference* problem, and (iv) graph constructions for solving the inference problem.

2.3.1.1 Binary submodular pairwise energy functions

Even though Graph cuts can be used to optimize pairwise submodular distance functions defined over multivalued labels [112], in this thesis we mainly focus on the special case of binary labels [74]. Before formally defining the binary submodular pairwise energy functions in the context of Graph cuts, let us first have a brief look at submodular set functions and their connections with sumodular pseudo-boolean functions.

Submodular set functions. Submodular set functions [1, 47] play an important role in disparate domains such as computer vision [74], operations research [54], machine learning and graphical models [64, 79, 81]. They are discrete analogues of convex functions that exist on vector spaces. Mathematically, for a given set $S = \{s_1, \cdots, s_n\}$, a set function $f : 2^S \to \mathbb{R}$ is submodular if and only if the following property is satisfied:

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \forall A, B \subset S. \tag{2.12}$$

Note that the domain of f is the subsets of the set S. Several functions that are commonly used in real world applications satisfy the above inequality either directly or in some restricted settings. Examples include, 'cut functions', 'entropy', 'mutual information', and the 'rank function' [1, 80].

Submodular pseudo-boolean functions. Every set function can be written as a Psuedo-Boolean Function (PBF) $f_b : \{0,1\}^n \to \mathbb{R}$, where f_b is a function of binary vectors. For example, if $S = \{s_1, s_2, s_3\}, A = \{s_1, s_2\}$, and $B = \{s_3\}$, then the binary vectors corresponding to the subsets A and B are $b_A = (1,1,0)$ and $b_B = (0,0,1)$. For pseudo-boolean functions the submodularity property (2.12) can be equivalently written as:

$$f_b(b_A) + f_b(b_B) \geq f_b(b_A \vee b_B) + f_b(b_A \wedge b_B), \forall b_A, b_B \in \{0,1\}^n. \tag{2.13}$$

where, $\wedge$ and $\vee$ are componentwise logical AND and OR operations. In the above mentioned example, the inequality (2.13) reduces to $f_b(1,1,0) + f_b(0,0,1) \geq f_b(0,0,0) + f_b(1,1,1)$. If the set size is fixed to two $(n = 2)$, all of the inequalities in (2.13) are satisfied trivially except for the following one:

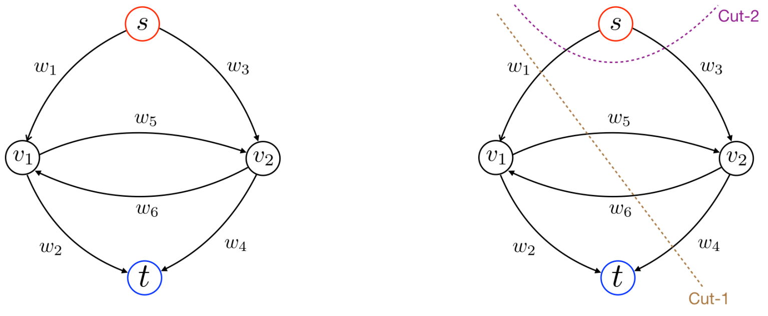
$$f_b(1,0) + f_b(0,1) \geq f_b(0,0) + f_b(1,1). \tag{2.14}$$

Therefore, any pseudo-boolean function f_b of binary vectors of length 2 (pairwise) is submodular if and only if the inequality (2.14) is satisfied.

The energy function. The energy function (2.11) is binary submodular if the label set contains two labels $\mathscr{L} \in \{0,1\}$, the maximum clique size is two, and the clique potential functions are the pseudo-boolean functions $\theta_{ij} : \{0,1\}^2 \to \mathbb{R}, \forall \{i,j\} \in \mathscr{E}$ satisfying the following inequality (submodularity):

$$\theta_{ij}(1,0) + \theta_{ij}(0,1) \geq \theta_{ij}(0,0) + \theta_{ij}(1,1), \forall \{i,j\} \in \mathscr{E}. \tag{2.15}$$

Notice that (2.15) enforces label smoothness by decreasing the energy if the nodes sharing an edge are assigned the same label. This type of potential function is of great importance in many tasks such as binary image restoration [14, 15, 51, 57], foreground background segmentation [13], medical imaging [11, 12], stereo and motion [14, 15, 56], and many others.



(a) s-t Graph (b) Example of cuts

Figure 2.5: The directed graph shown in Figure 2.5a is an example of an s-t Graph. All of the arc weights are assumed to be positive. In Figure 2.5b, the cost of Cut-1 is $(w_1 + w_6 + w_4)$, V_s is $\{s, v_2\}$, V_t is $\{t, v_1\}$, and the labeling is $\mathbf{y} = \{1, 0\}$. Similarly, the cost of Cut-2 is $(w_1 + w_3)$, V_s is $\{s\}$, V_t is $\{t, v_1, v_2\}$, and the labeling is $\mathbf{y} = \{1, 1\}$.

2.3.1.2 The s-t Graph and the s-t MINCUT problem

An *s-t* graph is a non-negatively weighted directed graph $G_d = (V \cup \{s, t\}, \mathscr{A}_d, w)$. Here V denotes the set of vertices and $\mathscr{A}_d$ denotes the set of arcs. The graph G_d contains two special vertices s and t, called ‘source’ and ‘sink’ respectively. These are called special vertices because the ‘source’ can have only outgoing arcs (no incoming arcs) and the ‘sink’ can have only incoming arcs. All of other vertices can have both incoming and outgoing arcs. The function $w : \mathscr{A}_d \to \mathbb{R}^+$ gives the weight of a given arc. An example of such a graph is shown in Figure 2.5a.

Under this setting, an *s-t cut* (also referred as the ‘cut’) is a partitioning of the graph vertices into two disjoint subsets V_s and V_t such that $s \in V_s$, $t \in V_t$, and $V \cup \{s, t\} = V_s \cup V_t$. The cost of a cut is the sum of the weights of the arcs that go from the subset V_s to the subset V_t. Figure 2.5b shows some examples of cuts and their corresponding costs. Formally, the cost of the cut partitioning the vertices into subsets V_s and V_t is:

$$C(V_s, V_t) = \sum_{v_i \in V_s, v_j \in V_t} w(v_i, v_j). \tag{2.16}$$

In this context, the s-t MINCUT problem is to find the *s-t* cut that incurs the minimum cost out of all possible *s-t* cuts. According to the Ford-Fulkerson theorem [41], the s-t MINCUT problem can be solved by finding the maximum flow from the source to the sink. Thus, solving the *s-t* MINCUT problem is equivalent to solving the 'max-flow' problem, which is a classical combinatorial optimization problem. However this algorithm converges to the global minimum if and only if the edge weights of the directed graph are positive. In what follows, we show how an inference problem with binary pairwise submodular clique potentials is related to an *s-t* MINCUT problem.

2.3.1.3 The s-t MINCUT and the inference problem

We begin with an example to show the relationship between an *s-t* cut and a labeling $\mathbf{y} \in \mathscr{Y}$. We use the simplest possible case of the interaction between two connected nodes. The Figure 2.5a shows the *s-t* graph, where vertices v_1 and v_2 represents the random variables associated with the inference problem. Let us assign the labels (or classes) 0 and 1 to the vertices s and t respectively. Therefore, for a given cut, all of the vertices that belong to the set V_s are assigned the label 0 and all of the vertices that belong to the set V_t are assigned the label 1. For example, the Cut-1 in the Figure 2.5b assigns the label 1 to the node v_1 and the label 0 to the node v_2. Furthermore, let us define the weights (assumed to be positive for now) of the arcs as follows:

$$w_1 = \theta_1(1) + \theta_{12}(1,1) \tag{2.17}$$

$$w_2 = \theta_1(0) \tag{2.18}$$

$$w_3 = \theta_2(1) \tag{2.19}$$

$$w_4 = \theta_2(0) + \theta_{12}(0,0) \tag{2.20}$$

$$w_5 = \theta_{12}(0,1) \tag{2.21}$$

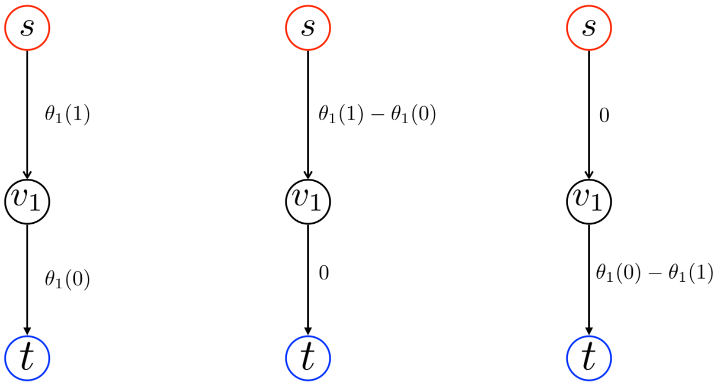
$$w_6 = \theta_{12}(1,0) - \theta_{12}(1,1) - \theta_{12}(0,0) \tag{2.22}$$

where, $\theta_i(.)$ and $\theta_{ij}(.,.)$ represents the unary and pairwise potentials. Now, for the cut (Cut-1) shown in the Figure 2.5b, the cost is $C = w_1 + w_6 + w_4 = \theta_1(1) + \theta_{12}(1,0) + \theta_2(0)$ (recall that the cost of the cut is the sum of the weights of the edges going from V_s to V_t, thus ignoring the edge weight w_5). The labeling corresponding to this cut is $y_1 = 1$ and $y_2 = 0$, which has energy $E(\mathbf{y};\mathbf{x}) = \theta_1(1) + \theta_{12}(1,0) + \theta_2(0) = C$. Thus, the cost of the s-t cut is exactly the same as the energy corresponding to the labeling obtained using the cut. This turns out to be true for any cut.

This example shows the relationship between the cost of an *s*-*t* cut and the energy corresponding to the labeling obtained. Hence, the inference problem, which is to find the labeling corresponding to the minimum energy, is exactly the same as finding the labeling corresponding to the minimum cost cut, i.e. solving the *s*-*t* MINCUT problem. However, as previously stated, the *s*-*t* MINCUT problem can be solved efficiently using 'max-flow' if and only if the arcs of the *s*-*t* graph have positive weights. In what follows we show how to construct such a graph with positive arc weights when the energy function is binary pairwise submodular.

2.3.1.4 The s-t graph construction

In this section we discuss the construction of an *s*-*t* graph for a given pairwise submodular energy function. We first consider construction for the unary potentials, then we move to the construction for pairwise submodular potentials, then finally we merge these two graphs (using the additivity theorem [74]) to build the final *s*-*t* graph. We also show how the submodularity condition ensures that we can always construct an *s*-*t* graph with positive arc weights.

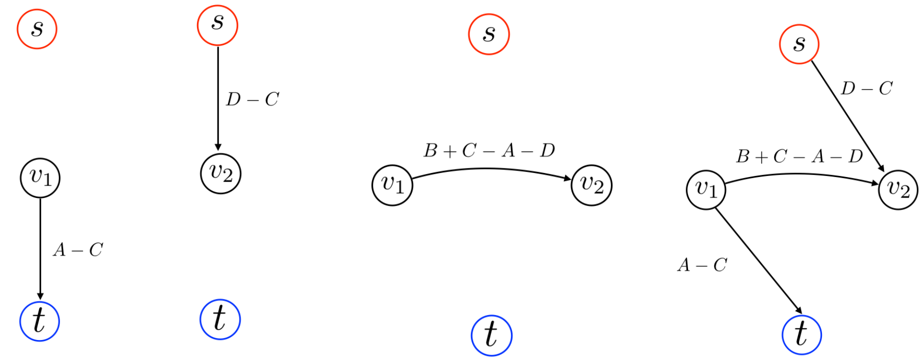


(a) Initial (b) $\theta_1(1) \geq \theta_1(0)$ (c) $\theta_1(1) \leq \theta_1(0)$

Figure 2.6: The *s*-*t* graph construction for arbitrary unary potentials. Figure 2.6b shows the *s*-*t* graph when the unary potential for label 1 is higher than that of label 0. Similarly, Figure 2.6b shows the *s*-*t* graph for the opposite case. Recall that a node in the set V_s and V_t is assigned the labels 0 and 1, respectively.

Arbitrary unary potentials. The graph construction for any given unary potential is shown in Figure 2.6. The initial graph shown in Figure 2.6a may or may not have positive edge weights. If $\theta_1(1) \geq \theta_1(0)$, Figure 2.6b shows a modified graph that has all of its edge weights non negative. Hence, it is an *s*-*t* graph. The idea is to modify the weights such that all of the weights of the graph are non negative and the labeling corresponding to the s-t MINCUT remains the same. In this case, the labeling corresponding to the *s*-*t* MINCUT (obtained by cutting the zero weight edge $v_1 - t$) is $y_1 = 0$. This is correct since $\theta_1(1) \geq \theta_1(0)$. However, since we have created a new graph, the relationship between the cost of the *s*-*t* MINCUT (which is 0 in this case) and the minimum energy (which is $\theta_1(0)$) follows a new rule: $E(\mathbf{y};\mathbf{x}) = C(V_s,V_t) + k$, where, $C(V_s,V_t)$ is the cost of the cut and k is a constant. In the case $\theta_1(1) \geq \theta_1(0)$, $k = \theta_1(0)$, otherwise, $k = \theta_1(1)$. Below we present simple rules, based on the above discussion, to create an *s*-*t* graph for arbitrary unary potentials.

- **Rules for the *s*-*t* graph construction for arbitrary unary potentials:** If $\theta_i(1) > \theta_i(0)$, then create an edge $s - v_i$ with weight $\theta_i(1) - \theta_i(0)$ and $k = \theta_i(0)$. Otherwise, create an edge $v_i - t$ with weight $\theta_i(0) - \theta_i(1)$ and $k = \theta_i(1)$.



(a) Unary (b) Unary (c) Pairwise (d) Final

Figure 2.7: Graph construction for a given pairwise submodular energy for the case when $A \geq C$ and $D \geq C$. Figures 2.7a and 2.7b are the graph constructions for $\theta_1(0) = A - C$ and $\theta_2(1) = D - C$ respectively. Figure 2.7c is the graph construction for $\theta_{12}(0,1) = B + C - A - D$. Figure 2.7d uses the additivity theorem to construct the final graph.

Pairwise submodular potentials. Recall that the edge weights of the directed graph for the graph cuts must be non-negative. For example, consider the following pairwise potentials:

$\theta_{12}(0,0) = 3$	$\theta_{12}(0,1) = 10$
$\theta_{12}(1,0) = -2$	$\theta_{12}(1,1) = -1$

If we construct the s-t graph with the edge weights as mentioned in the section 2.3.1.3 (assuming all the unary potentials to be zero), we get $w_1 = -1$ (please refer to the equation (2.17)). However, we can decompose the above pairwise potential as follows:

$$\begin{array}{|c|c|}\hline A & B \\ \hline C & D \\ \hline\end{array} = \begin{array}{|c|c|}\hline A-C & A-C \\ \hline 0 & 0 \\ \hline\end{array} + \begin{array}{|c|c|}\hline 0 & D-C \\ \hline 0 & D-C \\ \hline\end{array} + \begin{array}{|c|c|}\hline 0 & B+C-A-D \\ \hline 0 & 0 \\ \hline\end{array} + \mathrm{C}$$

where $\theta_{12}(0,0) = A$, $\theta_{12}(0,1) = B$, $\theta_{12}(1,0) = C$, and $\theta_{12}(1,1) = D$. Note that in the above decomposition, all of the entries in the decomposed matrices are non-negative since $A > C$, $D > C$, and $B+C-A-D \geq 0$ (the given potential is submodular, thus, $\theta_{12}(0,1) + \theta_{12}(1,0) \geq \theta_{12}(0,0) + \theta_{12}(1,1)$). Therefore, constructing *s-t* graphs for these decomposed potentials will always have positive edge weights. The potentials in the first table are independent of node v_2. Therefore, they can be treated as a unary potential $\theta_1(0)$. Similarly, the potentials in the second table can be treated as $\theta_2(1)$. The potential in the third table is $\theta_{12}(0,1)$. The graph construction for these potentials is shown in the Figure 2.7. Note that the above decomposition does not change the energy of the labeling. For example, $\theta_{12}(1,1) = 0+1+0-2 = -1$. However, it allows us to create a directed graph with positive edge weights. Similar decompositions are always possible if the given potential is submodular because $B+C-A-D \geq 0$ is always satisfied. Again, the energy and the cost of the cut are different but follow the relationship: $E(\mathbf{y};\mathbf{x}) = C(V_s,V_t) + k$. In the above discussed case, $k = C$. Below we present rules that create s-t graph for any pairwise submodular potentials.

- **Rules for the pairwise potential:**

  1. create an edge $v_i - v_j$ with weight $(B+C-A-D)$. See Figure 2.7c.
  2. if $A-C \geq 0$ create an edge $v_i - t$ with weight $A-C$. Otherwise, create an edge $s - v_i$ with weight $C-A$. See Figure 2.7a.
  3. if $C-D \geq 0$ create an edge $v_j - t$ with weight $C-D$. Otherwise, create an edge $s - v_j$ with weight $D-C$. See Figure 2.7b.
  4. Update k: This falls into the following three cases — (i) if $A > C$, $k = \min(C,D)$; (ii) if $A < C$ and $C < D$, $k = A$; and (iii) if $A < C$ and $C > D$, $k = A+D-C$.

The rules above are for a given pairwise clique. However, the additivity theorem [74] allows us to construct an *s-t* graph representing for the whole input graph by combining the *s-t* graphs from each unary and pairwise potential. Roughly speaking, if an arc (v_i, v_j) has weights $w_k(v_i,v_j)$ in the k-th *s-t* graph, then it will have the weight $\sum_k w_k(v_i,v_j)$ in the final *s-t* graph. An example is shown the Figure 2.7d.

2.3.2 The $\alpha-$expansion algorithm for metric labeling

In section 2.3.1 we discussed graph cuts where labels was binary and the clique potentials were pairwise submodular. In this section, we talk about a more general inference algorithm called α-expansion [15, 128] which approximately solves the inference problem when the number of labels is more than two. If the clique potentials are pairwise metric distance functions defined over the labels (also known as the *metric labeling* problem), the α-expansion algorithm provides strong theoretical guarantees. In the following section, we first talk about the metric labeling problem and then provide details on α-expansion and how it can be used to efficiently optimize metric labeling problems.

2.3.2.1 Metric labeling problem

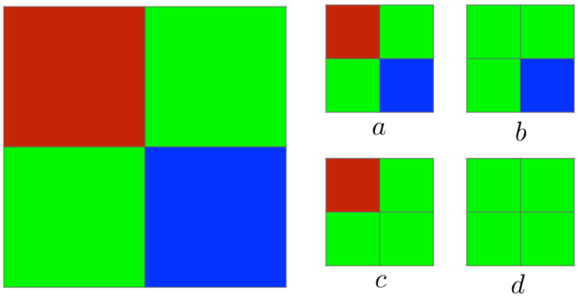
Metric labeling is an important special case of labeling or inference problems, which has been extensively studied in computer vision [15, 84] and theoretical computer science [20, 65]. In metric labeling, the unary potentials remain arbitrary while the clique potentials are decomposable as the sum of pairwise potentials specified by user-defined metric distance functions defined over the label space. Before formally defining the energy function for metric labeling, we first define a metric distance function.

Metric distance function. A function $\theta_{ij}(y_i,y_j) : \mathscr{L} \times \mathscr{L} \to \mathbb{R}_+$ is a metric distance function if and only if the following four conditions are satisfied:

- **Non Negativity:** $\theta_{ij}(y_i,y_j) \geq 0, \forall y_i, y_j \in \mathscr{L}$.
- **Identity of indiscernibles:** $\theta_{ij}(y_i,y_j) = 0$ if and only if $y_i = y_j$.
- **Symmetry:** $\theta_{ij}(y_i,y_j) = \theta_{ij}(y_j,y_i), \forall y_i, y_j \in \mathscr{L}$.
- **Triangle inequality:** $\theta_{ij}(y_i,y_j) + \theta_{ij}(y_j,y_k) \geq \theta_{ij}(y_i,y_k), \forall y_i, y_j, y_k \in \mathscr{L}$.

A few examples of metric distance functions are — (1) Truncated Linear: $\theta(y_i,y_j) = \min(M, |y_i - y_j|), M \geq 0$; (2)Uniform Metric: a truncated linear metric with $M = 1$; (3) Shortest Path: if $\{y_i\}_{i=\{1,\cdots,N\}}$ represents the set nodes in a weighted graph then $\theta(y_i,y_j) = d(y_i,y_j)$ is a metric if $d(y_i,y_j)$ is the length of the shortest path between nodes y_i and y_j; and (4) Minkowski Metric: $\theta(y_i,y_j) = l_p(|y_i - y_j|), p \geq 1$, where $l_p(.)$ is the $p-$norm.

The energy function. The pairwise energy function (2.11) defines a metric labeling problem if the pairwise clique potential functions for all the edges $\theta_{ij}(.,.), \forall\{i,j\} \in \mathscr{E}$ are metric distance functions defined over the label set $\mathscr{L}$. Metric labeling has been used to formulate several problems in low-level and high-level computer vision tasks. For example, for image denoising and stereo matching the truncated linear metric shows very promising results [118]. Similarly, for pose estimation the Minkowski Metric has been used extensively [35].



(a) Initial Labeling (b) α = ‘Green’

Figure 2.8: An example of the move space for $\alpha-$expansion with only four nodes in the graph. Figure 2.8a shows the initial labeling from where the expansion move will take place. Figure 2.8b shows the possible labelings when α is the ‘Green’ label. Let the numbering of the nodes be in the clockwise manner starting from the top left corner. Then different binary vectors for all possible labelings are as follows: $\mathbf{t}_a = \{0,0,0,0\}$, $\mathbf{t}_b = \{1,0,0,0\}$, $\mathbf{t}_c = \{0,0,1,0\}$, and $\mathbf{t}_d = \{1,0,0,1\}$.

2.3.2.2 The $\alpha-$expansion algorithm

Let us now talk about the general form of the well known α-expansion algorithm and see how it can be used for the metric labeling problem. α-expansion [15, 128] is a general framework designed to approximately solve the multilabel inference problem (see Algorithm 1). Intuitively, for a given $\alpha \in \mathscr{L}$, step 2 of the algorithm finds the best labeling within the expansion move space. An expansion move space consists of the set of labelings produced when the nodes can either retain their current label or switch to a given label $\alpha \in \mathscr{L}$ (see Figure 2.8 for a visualization). The α-expansion algorithm cycles through the $\alpha \in \mathscr{L}$ in some predefined order repeatedly minimizing the energy within the given α-expansion move space until the energy of the solution can not be decreased further. This produces a local minima of the global cost function. Notice that the expansion move spaces can be very large. They contain 2^P possible labelings, where $P \leq N$ is the number of nodes whose current label is not α.

Mathematically, an α-expansion move can be represented using a vector of binary variables $\mathbf{t} = \{t_i, \forall i \in V\}$, where $t_i = 0$ implies that the i-th node retained its original label and $t_i = 1$ implies that it moved to the new label alpha (see Figure 2.8 for an example). Given the current labeling $\mathbf{y}$ and the binary vector $\mathbf{t}$, the new labeling can be obtained using the following transformation function $T_\alpha(\mathbf{y},\mathbf{t})$:

$$T_\alpha(y_i,t_i) = \begin{cases} y_i, & \text{if } t_i = 0 \\ \alpha, & \text{if } t_i = 1 \end{cases} \tag{2.23}$$

It remains to see how to obtain the optimal binary vector $\hat{\mathbf{t}}$ which is the move corresponding

Algorithm 1 The α-expansion algorithm.

```
input  Initial labeling $\mathbf{y}$; Label set in some predefined order $\mathscr{L}$.
 1: Set success:=0
 2: for each label $\alpha \in \mathscr{L}$ do
 3:     Find optimal expansion move
            $\hat{\mathbf{t}} = \operatorname*{argmin}_{\mathbf{t} \in \{0,1\}^N} E^{\alpha}(\mathbf{t})$        (2.24)
 4:     if $E(T_{\alpha}(\mathbf{y}, \hat{\mathbf{t}})) < E(\mathbf{y})$ then
 5:         $\mathbf{y} = T_{\alpha}(\mathbf{y}, \hat{\mathbf{t}})$
 6:         success:=1
 7:     end if
 8: end for
 9: if success:=1 then
10:     go to 1.
11: end if
12: return $\mathbf{y}$.
```

to the minimum energy for a given α. Recall that each node has two options, either to retain its current label y_i, represented as $t_i = 0$, or, to move to a new label α, represented as $t_i = 1$. Therefore, the task of obtaining the optimal binary vector $\hat{\mathbf{t}}$ can be posed as the following optimization problem:

$$\hat{\mathbf{t}} = \operatorname*{argmin}_{\mathbf{t} \in \{0,1\}^N} E^{\alpha}(\mathbf{t}) \tag{2.25}$$

where, $E^{\alpha} = \sum_i \theta_i^{\alpha}(t_i) + \sum_{ij} \theta_{ij}^{\alpha}(t_i, t_j)$, is a new energy function with the following unary and the pairwise potentials:

$$\begin{array}{|c|} \hline \theta_i^{\alpha}(0) \\ \hline \theta_i^{\alpha}(1) \\ \hline \end{array} = \begin{array}{|c|} \hline \theta_i(y_i) \\ \hline \theta_i(\alpha) \\ \hline \end{array} \quad \Bigg| \quad \begin{array}{|c|c|} \hline \theta_{ij}^{\alpha}(0,0) & \theta_{ij}^{\alpha}(0,1) \\ \hline \theta_{ij}^{\alpha}(1,0) & \theta_{ij}^{\alpha}(1,1) \\ \hline \end{array} = \begin{array}{|c|c|} \hline \theta_{ij}(y_i, y_j) & \theta_{ij}(y_i, \alpha) \\ \hline \theta_{ij}(\alpha, y_j) & \theta_{ij}(\alpha, \alpha) \\ \hline \end{array}$$

Note that α-expansion can only be used in practice if the optimization problem (equation (2.25)), which amounts to finding the best expansion move, can be solved efficiently in polynomial time.

For metric labeling the new pairwise potentials $\theta_{ij}^{\alpha}(.,.)$ are pairwise submodular functions (as will be proven shortly). Therefore, the expansion move optimization problem can be solved globally and efficiently using Graph cuts [74] (as discussed in section 2.3.1). The graph for the new potentials $\theta_i^{\alpha}(.)$ and $\theta_{ij}^{\alpha}(.,.)$ can be easily constructed using the rules

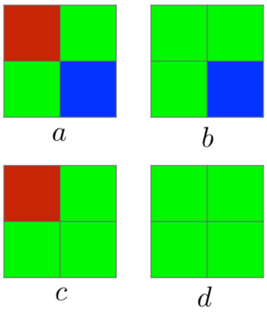


Figure 2.9: An example of the costs assigned by the P^n-Potts model. Let γ_r, γ_b, and γ_g be the costs for the colors 'red', 'blue', and 'green', respectively. $\gamma_{max} > \max(\gamma_r, \gamma_g, \gamma_b)$. Then the costs for the cases (a), (b), and (c) are all the same, which is γ_{max}. The cost for the case (d) is γ_g.

discussed in section 2.3.1.4. It remains to prove that the new pairwise potentials $\theta^{\alpha}_{ij}(.,.)$ are submodular for all $\{i,j\} \in \mathscr{E}$. Since $\theta_{ij}(.,.)$ is a metric distance function defined over the labels, using the triangular inequality and the fact that $\theta_{ij}(\alpha,\alpha) = 0, \forall i,j \in \mathscr{E}$, we get:

$$\theta_{ij}(y_i,\alpha) + \theta_{ij}(\alpha,y_j) \geq \theta_{ij}(y_i,y_j) + \theta_{ij}(\alpha,\alpha), \forall y_i,y_j \in \mathscr{L}, \forall i,j \in \mathscr{E}. \tag{2.26}$$

Inequality (2.26) is sufficient to prove that $\theta^{\alpha}_{ij}(.,.)$ satisfy the submodularity condition $\theta^{\alpha}_{ij}(0,1) + \theta^{\alpha}_{ij}(1,0) \geq \theta^{\alpha}_{ij}(0,0) + \theta^{\alpha}_{ij}(1,1), \forall i,j \in \mathscr{E}$. Hence, graph cuts can be used to obtain the globally optimal α-expansion move.

2.3.3 α-expansion for the P^n Potts model

In sections 2.3.1 and 2.3.2 we talked about Graph cuts for binary pairwise submodular energy functions and the α-expansion algorithm for the metric labeling problem. In both cases the maximal clique size is assumed to be two, which encodes only pairwise interactions. However higher-order interactions have been shown to be very useful in many vision tasks including object segmentation [28, 30, 67, 87, 121, 129], disparity estimation [62, 131], and image restoration [89]. There are several known energy functions (or models) that allows us to encode higher-order interactions for which efficient inference algorithms is still available, such as the P^n-Potts model [66], the Robust P^n-Potts model [67], the Label cost based model [28], and the Co-occurrence statistics based model [87]. In this section we discuss the P^n-Potts model which will play a key role in chapter 3.

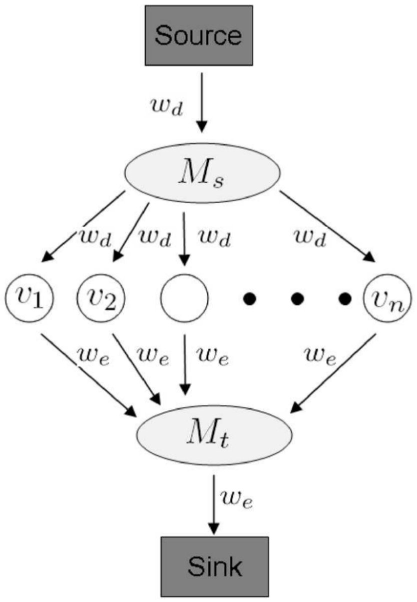


Figure 2.10: Graph construction for the P^n Potts model. M_s and M_t are two auxiliary nodes. The weights are $w_d = \gamma + k$ and $w_e = \gamma_\alpha + k$, where, $k = \gamma_{max} - \gamma_\alpha - \gamma$. Figure reproduced from [66].

2.3.3.1 P^n-Potts model

An important special case of the labeling problem with higher-order cliques is the P^n-Potts model [66]. This generalizes the well known Potts model [106] to higher-order energy functions. As opposed to the previous two models, submodular pairwise binary labels and metric labeling, the cliques in the P^n-Potts model can be of arbitrary sizes. The energy function defined in equation (2.9) represents a P^n Potts model if the clique potential functions are defined as ($\theta_c(\mathbf{y}_c) := \theta_c(\mathbf{y}_c; \mathbf{x}_c)$):

$$\theta_c(\mathbf{y}_c) \propto \begin{cases} \gamma^k, & \text{if } y_i = l_k, \forall i \in c, \\ \gamma^{max}, & \text{otherwise}, \end{cases} \tag{2.27}$$

where γ^k is the cost of assigning all nodes of the clique to label $l_k \in \mathscr{L}$, and $\gamma^{max} > \gamma_k, \forall l_k \in \mathscr{L}$. Intuitively, the P^n-Potts model enforces rigid label consistency by assigning a larger cost γ^{max} whenever there is more than one label in the given clique. Figure 2.9 shows an example of the costs assigned by P^n-Potts for four different labelings.

2.3.3.2 Optimization algorithm for the P^n-Potts model

We now discuss α-expansion for the optimization of energy functions defined using the P^n-Potts model [66]. As described in section 2.3.2, our goal is to find the optimal move $\hat{\mathbf{t}}$ that minimizes the cost shown in equation (2.25). As we are dealing with higher-order cliques, the energy function E^α has a more general form: $E^\alpha = \sum_i \theta_i^\alpha(t_i) + \sum_{c\mathscr{C}} \in \theta_c^\alpha(\mathbf{t}_c)$, where, $\mathbf{t}_c$ is the binary move vector for all of the nodes in the clique c. Recall that, $t_i = 0$ implies that the i^{th} node retains its original label, and $t_i = 1$ implies that it moves to the new label alpha. Therefore, the transformation function $T_\alpha(y_i, t_i)$ is the same as defined in equation (2.23). Using above definitions and the fact that the clique potentials have the P^n Potts form (2.27), the potentials after the move $\mathbf{t}_c$ can be computed as follows:

$$\theta_c(T_\alpha(\mathbf{y}_c, \mathbf{t}_c)) = \begin{cases} \gamma, & \text{if } t_i = 0, \forall i \in c, \\ \gamma_\alpha, & \text{if } t_i = 1, \forall i \in c, \\ \gamma^{max}, & \text{otherwise,} \end{cases} \tag{2.28}$$

where,

$$\gamma = \begin{cases} \gamma_l, & \text{if } y_i = l \in \mathscr{L}, \forall i \in c, \\ \gamma^{max}, & \text{otherwise,} \end{cases} \tag{2.29}$$

The optimal move $\hat{\mathbf{t}}$ for the above potential can be obtained by solving the S-T MINCUT problem in the graph shown in Figure 2.10. Figure 2.10 shows the graph construction for a particular clique c, but additivity theorem [74] allows us to construct the global graph by addition.

2.4 Learning Algorithms for the SSVM

In section 2.3, we talked about different inference algorithms useful for the SSVM prediction task. We assumed that the SSVM parameters $\mathbf{w}$ were already known. In this section, we talk about learning these parameters in a supervised fashion when a labelled training set is given. Broadly speaking, learning parameters for a task involves two major steps – (1) designing an objective function with desirable properties; and (2) optimizing the objective function. The design of the objective function is highly dependent on our capacity to optimize it: functions that encodes all of the desired properties may not yield to efficient optimization. This constraint limit us a few specific types of objective functions. As will be seen in section 2.4.1, for SSVM we upper bound the empirical loss in such a way that the final objective function becomes convex, and thus can be efficiently optimized using many known algorithms. In what follows we discuss about the design and optimization of the SSVM objective function in detail.

The objective function of SSVM can be derived using several interrelated approaches such as – (i) a probabilistic approach (minimizing the posterior expected loss), (ii) a geometric approach based on margin maximization (hyperplanes), (iii) a linear feasibility problem based approach, and (iv) a regularized risk minimization based approach (minimizing an upper bound on the empirical loss). In this chapter we focus on the risk minimization point of view, as briefly discussed in section 2.2.2, because this is simple and intuitive. We also talk about the geometric view point whenever it is useful for better understanding of the objective function. Finally, we talk about different algorithms for the optimization of the objective function. All of the notation is as described in section 2.2.

2.4.1 Convex upperbound of empirical loss

As discussed in section 2.2.2, given a sufficiently large dataset of n input-output pairs, $\mathscr{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=\{1,\cdots,n\}}$ drawn *i.i.d.* from an unknown distribution ρ, the optimized parameter vector $\mathbf{w}^*$ can be obtained by minimizing the following approximate empirical risk [127]:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} R_{emp}(\mathbf{w}) = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \Delta(\mathbf{y}_i, f_{\mathbf{w}}(\mathbf{x}_i)) \tag{2.30}$$

The loss function $\Delta(.,.)$ is normally piece wise linear and non-convex with respect to $\mathbf{w}$, so it is non-differentiable at many hinge points and its gradient is zero in many flat regions. Thus, direct optimization of the above function is prone to troublesome local minima. In order to alleviate this problem we optimize a convex upper bound for the loss function. For a given $\mathbf{w}$ and the predicted labeling $\hat{\mathbf{y}}_i = \operatorname{argmax}_{\bar{\mathbf{y}}_i \in \mathscr{Y}_i} \mathbf{w}^\top \Phi(\mathbf{x}_i, \bar{\mathbf{y}}_i)$, the loss can be upper bounded as follows:

$$\begin{aligned}
\Delta(\mathbf{y}_i, \hat{\mathbf{y}}_i) &= \Delta(\mathbf{y}_i, \hat{\mathbf{y}}_i) + \mathbf{w}^\top \Phi(\mathbf{x}_i, \hat{\mathbf{y}}_i) - \mathbf{w}^\top \Phi(\mathbf{x}_i, \hat{\mathbf{y}}_i) && (2.31)\\
&\leq \Delta(\mathbf{y}_i, \hat{\mathbf{y}}_i) + \mathbf{w}^\top \Phi(\mathbf{x}, \hat{\mathbf{y}}_i) - \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{y}_i) && (2.32)\\
&\leq \max_{\bar{\mathbf{y}}_i \in \mathscr{Y}_i} (\Delta(\mathbf{y}_i, \bar{\mathbf{y}}_i) + \mathbf{w}^\top \Phi(\mathbf{x}_i, \bar{\mathbf{y}}_i) - \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{y}_i)) && (2.33)\\
&= \underbrace{\max_{\bar{\mathbf{y}}_i \in \mathscr{Y}_i} \underbrace{(\Delta(\mathbf{y}_i, \bar{\mathbf{y}}_i) - \mathbf{w}^\top \Psi(\mathbf{x}_i, \bar{\mathbf{y}}_i))}_{H_i(\bar{\mathbf{y}}_i;\mathbf{w})}}_{\tilde{H}_i(\mathbf{w})} && (2.34)
\end{aligned}$$

where $\Psi(\mathbf{x}_i, \bar{\mathbf{y}}_i) = \Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \bar{\mathbf{y}}_i)$ and $\tilde{H}_i(\mathbf{w})$ is the hinge loss. Expression (2.32) follows from the fact that the predicted labeling $\hat{\mathbf{y}}_i$ is the argmax over the label set $\mathscr{Y}_i$ which already contains the ground truth labeling $\mathbf{y}_i$, so $\mathbf{w}^\top \Phi(\mathbf{x}_i, \hat{\mathbf{y}}_i) \geq \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{y}_i)$. Notice that the upper bounded loss $\tilde{H}_i(\mathbf{w})$ is convex in $\mathbf{w}$ because it is a maximum over affine functions in $\mathbf{w}$. Minimizing this upper bound on the loss over the entire dataset with the L_2 regularization leads to the following objective function:

$$\min_{\mathbf{w}} \quad \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{n} \sum_i \max_{\bar{\mathbf{y}}_i \in \mathscr{Y}_i} (\Delta(\mathbf{y}_i, \bar{\mathbf{y}}_i) - \mathbf{w}^\top \Psi(\mathbf{x}_i, \bar{\mathbf{y}}_i)). \tag{2.35}$$

Broadly speaking, the regularization term $\|\mathbf{w}\|^2$ provides two benefits – (1) it makes the objective function smoother; and (2) it reduces overfitting. This objective function can be written in constrained form as follows:

$$\min_{\mathbf{w}, \xi} \quad \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{n} \sum_{i=1}^{n} \xi_i \tag{2.36}$$

$$\text{s.t.} \quad \xi_i \geq \Delta(\mathbf{y}_i, \bar{\mathbf{y}}_i) - \mathbf{w}^\top \Psi(\mathbf{x}_i, \bar{\mathbf{y}}_i), \quad \forall i, \forall \bar{\mathbf{y}}_i \in \mathscr{Y}_i. \tag{2.37}$$

The constraints (2.37) ensure that $\xi_i \geq \max_{\bar{\mathbf{y}}_i \in \mathscr{Y}_i} (\Delta(\mathbf{y}_i, \bar{\mathbf{y}}_i) - \mathbf{w}^\top \Psi(\mathbf{x}_i, \bar{\mathbf{y}}_i))$ and the objective function minimizes ξ_i under them. Therefore, the constrained and the unconstrained objective functions of SSVM serves the same purpose of minimizing the upper bound on the loss. The unconstrained objective function is non-smooth owing to the 'max' over affine functions, whereas although constrained objective function is smooth, it has exponentially many constraints. As will be seen shortly, the different characteristics of the constrained and unconstrained forms allow us to use different algorithms to optimize them.

Intuitive Interpretation. We have seen that the unconstrained objective function for SSVM directly optimizes an upper bound on the empirical loss. We now develop an intuitive interpretation of constrained form for the SSVM objective function. The function can be rewritten as:

$$\min_{\mathbf{w}, \xi} \quad \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{n} \sum_{i=1}^{n} \xi_i \tag{2.38}$$

$$\text{s.t.} \quad \mathbf{w}^\top \Psi(\mathbf{x}_i, \bar{\mathbf{y}}_i) \geq \Delta(\mathbf{y}_i, \bar{\mathbf{y}}_i) - \xi_i, \quad \forall i, \forall \bar{\mathbf{y}}_i \in \mathscr{Y}_i. \tag{2.39}$$

or

$$\text{s.t.} \quad \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{y}_i) \geq \mathbf{w}^\top \Phi(\mathbf{x}_i, \bar{\mathbf{y}}_i) + \Delta(\mathbf{y}_i, \bar{\mathbf{y}}_i) - \xi_i, \quad \forall i, \forall \bar{\mathbf{y}}_i \in \mathscr{Y}_i. \tag{2.40}$$

For a given $\mathbf{w}$, let us denote $s(\mathbf{x}_i, \bar{\mathbf{y}}_i; \mathbf{w}) = \mathbf{w}^\top \Phi(\mathbf{x}_i, \bar{\mathbf{y}}_i)$ as the score of assigning labeling $\bar{\mathbf{y}}_i$ to the input $\mathbf{x}_i$. Therefore, the constraints (2.40) force us to learn a $\mathbf{w}$ such that the score of the ground truth labeling is higher than the score for any other labeling, with a margin proportional to the loss function. This make sense because our prediction is based on a maximization over $\mathcal{Y}_i$ and we would like our classifier to award the maximum score to the correct labeling. These constraints, however, may lead to overfitting of the training dataset if the regularization hyperparameter λ is not chosen properly. In order to avoid this, one way is to set λ by cross-validating it on the training set.

2.4.2 Lagrange dual of SSVM

For the sake of completeness we have given a brief introduction to the Lagrange theory [6, 10] in Appendix B.1. The constrained primal problem of SSVM (2.36) can be written in the standard form of equation (B.1) as follows:

$$\min_{\mathbf{w}, \xi} \quad \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{n} \sum_{i=1}^{n} \xi_i \tag{2.41}$$

$$\text{s.t.} \quad \Delta(\mathbf{y}_i, \mathbf{y}) - \mathbf{w}^\top \Psi(\mathbf{x}_i, \mathbf{y}) - \xi_i \leq 0, \quad \forall i, \forall \mathbf{y} \in \mathcal{Y}_i. \tag{2.42}$$

There are no equality constraints. Note that we are using the notation $\mathbf{y}$ instead of $\bar{\mathbf{y}}_i$ in order to avoid clutter. The primal variables of the problem are $\mathbf{w}$ and ξ_i and Lagrangian is:

$$L(\mathbf{w}, \xi, \alpha) = \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{n} \sum_{i=1}^{n} \xi_i + \sum_{i \in [n], \mathbf{y} \in \mathcal{Y}_i} \frac{1}{n} \alpha_i(\mathbf{y}) \left(-\xi_i - \mathbf{w}^\top \Psi(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}_i, \mathbf{y}) \right) \tag{2.43}$$

where $\alpha_i(\mathbf{y})$ is the dual variable associated with the constraint for the i-th input when the labeling $\mathbf{y} \in \mathcal{Y}_i$ is assigned to it. We have scaled the dual variables $\alpha_i(\mathbf{y})$ by $\frac{1}{n}$. This does not change any optimality conditions and it leads to a cleaner formulation. Let $\alpha_i \in \mathbb{R}^{|\mathcal{Y}_i|}$ be the dual variable vector corresponding to all of the constraints associated with the i-th sample. Then the final dual variable vector for all of the samples can be written as $\alpha = (\alpha_1, \cdots, \alpha_n) \in \mathbb{R}^{|\mathcal{Y}_1|} \times \cdots \mathbb{R}^{|\mathcal{Y}_n|} \in \mathbb{R}^m$, where $m = \sum_{i=1}^{n} |\mathcal{Y}_i|$ is the total number of constraints, which is exponentially large. Using the *first order* KKT condition (see Appendix B.1.5 for all four KKT conditions), which specify the optimal values of the primal variables of the Lagrangian, we obtain:

$$\nabla_{\mathbf{w}} L(\mathbf{w}, \xi, \alpha) = 0 \implies \mathbf{w} = \sum_{i \in [n], \mathbf{y} \in \mathcal{Y}_i} \frac{1}{n\lambda} \alpha_i(\mathbf{y}) \Psi(\mathbf{x}_i, \mathbf{y}), \tag{2.44}$$

and

$$\nabla_{\xi_i} L(\mathbf{w}, \xi, \alpha) = 0 \implies \sum_{\mathbf{y} \in \mathscr{Y}_i} \alpha_i(\mathbf{y}) = 1. \tag{2.45}$$

Plugging these optimality conditions into the SSVM Lagrangian (equation (2.43)) we obtain the following Lagrange dual problem (a concave maximization):

$$\max_{\alpha} \quad g(\alpha) := -\frac{\lambda}{2} \left\| \sum_{i \in [n], \mathbf{y} \in \mathscr{Y}_i} \frac{1}{n\lambda} \alpha_i(\mathbf{y}) \Psi(\mathbf{x}_i, \mathbf{y}) \right\|^2 + \sum_{i \in [n], \mathbf{y} \in \mathscr{Y}_i} \frac{1}{n} \alpha_i(\mathbf{y}) \Delta(\mathbf{y}_i, \mathbf{y}) \tag{2.46}$$

$$s.t. \quad \sum_{\mathbf{y} \in \mathscr{Y}_i} \alpha_i(\mathbf{y}) = 1, \forall i \in [n], \tag{2.47}$$

$$\alpha_i(\mathbf{y}) \geq 0, \forall i \in [n], \forall \mathbf{y} \in \mathscr{Y}_i. \tag{2.48}$$

The additional positivity constraints (2.48) ensure that the dual objective is always a lower bound on the primal objective (see Theorem 5 in Appendix B.1.2). The dual problem can be equivalently written as a convex minimization:

$$\min_{\alpha} \quad f(\alpha) := \frac{\lambda}{2} \|A\alpha\|^2 - \mathbf{b}^\top \alpha \tag{2.49}$$

$$s.t. \quad \sum_{\mathbf{y} \in \mathscr{Y}_i} \alpha_i(\mathbf{y}) = 1, \forall i \in [n], \tag{2.50}$$

$$\alpha_i(\mathbf{y}) \geq 0, \forall i \in [n], \forall \mathbf{y} \in \mathscr{Y}_i. \tag{2.51}$$

where, $f(\alpha) = -g(\alpha)$, $A \in \mathbb{R}^{d \times m}$ is a matrix with column entries as $\frac{\Psi(\mathbf{x}_i, \mathbf{y})}{\lambda n}$, and $\mathbf{b} \in \mathbb{R}^m$ is a vector with elements as $\frac{\Delta(\mathbf{y}_i, \mathbf{y})}{n}$, as shown below:

$$A = \frac{1}{\lambda n} \begin{pmatrix} & | & \\ \cdots & \Psi(\mathbf{x}_i, \mathbf{y}) & \cdots \\ & | & \end{pmatrix} \in \mathbb{R}^{d \times m}, \quad \mathbf{b} = \frac{1}{n} \begin{pmatrix} \vdots \\ \Delta(\mathbf{y}_i, \mathbf{y}) \\ \vdots \end{pmatrix} \in \mathbb{R}^m. \tag{2.52}$$

Notice that, from the KKT conditions

$$\mathbf{w} = \sum_{i \in [n], \mathbf{y} \in \mathscr{Y}_i} \frac{1}{n\lambda} \alpha_i(\mathbf{y}) \Psi(\mathbf{x}_i, \mathbf{y}) = A\alpha. \tag{2.53}$$

The above equation gives the relationship between the primal and the dual variables which will be useful when we discuss about the Block-Coordinate Frank-Wolfe algorithm for the optimization of the dual of the SSVM.

2.4.3 Optimization

For SSVM, the optimization of the primal problem (2.36) or its dual (2.49) both lead to the same solution. This is because the objective function is convex and strong duality holds. However as we have discussed, both the primal and the dual objectives contain exponentially many constraints so neither can be optimized directly using off-the-shelf quadratic program solvers. In this section we discuss three algorithms that can handle these issues: (1) Subgradient descent; (2) Cutting plane; and (3) Block-Coordinate Frank-Wolfe (BCFW). The subgradient descent algorithm works on the unconstrained primal objective function of SSVM (2.35). The cutting plane algorithm can be used with either the constrained primal or the dual objective functions. Lastly, the BCFW algorithm can only be used on the dual formulation of the SSVM but as will be seen shortly all of its updates are associated with primal variables.

Before delving into the details of these algorithms, let us first talk about the 'max-oracle' problem, which is a useful ingredient of all these methods.

The max-oracle problem. The 'max-oracle' problem at a given $\mathbf{w}$ is defined as:

$$\mathbf{y}_i^* = \underset{\mathbf{y}\in\mathscr{Y}_i}{\operatorname{argmax}}(\underbrace{\Delta(\mathbf{y}_i,\mathbf{y}) - \mathbf{w}^\top\Psi(\mathbf{x}_i,\mathbf{y})}_{H_i(\mathbf{y},\mathbf{w})}) \quad : \textit{`max-oracle' or `loss augmented inference'} \tag{2.54}$$

This finds the labeling that maximizes the augmented loss $H_i(\mathbf{y},\mathbf{w})$ (cf. equation (2.33)) for the i-th sample. The problem can be rewritten as:

$$\begin{aligned}
\mathbf{y}_i^* &= \underset{\mathbf{y}\in\mathscr{Y}_i}{\operatorname{argmax}}(\Delta(\mathbf{y}_i,\mathbf{y}) - \mathbf{w}^\top\Psi(\mathbf{x}_i,\mathbf{y}) \\
&= \underset{\mathbf{y}\in\mathscr{Y}_i}{\operatorname{argmax}}(\Delta(\mathbf{y}_i,\mathbf{y}) - \mathbf{w}^\top\Phi(\mathbf{x}_i,\mathbf{y}_i) + \mathbf{w}^\top\Phi(\mathbf{x}_i,\mathbf{y})) \\
&= \underset{\mathbf{y}\in\mathscr{Y}_i}{\operatorname{argmax}}(\Delta(\mathbf{y}_i,\mathbf{y}) + \mathbf{w}^\top\Phi(\mathbf{x}_i,\mathbf{y}))
\end{aligned} \tag{2.55}$$

The equality (2.55) comes from the fact that the term $\mathbf{w}^\top\Phi(\mathbf{x}_i,\mathbf{y}_i)$ is constant. Notice that the extra loss term $\Delta(.,.)$ is the only difference between the 'max-oracle' problem (2.55) and the prediction or the inference problem (2.4). Because of this 'max-oracle' is also known as the *loss augmented inference* problem. If the loss function is decomposable over the nodes of the output structure, the unary potentials of the standard inference problem can be trivially modified so that solving the above problem is equivalent to solving an inference problem. It can thus be optimized using the algorithms discussed in section 2.3. Few

examples of decomposable loss functions are – (1) the zero-one loss (2) the Hamming loss; and (3) the area under curve loss. These loss functions are very useful in many applications related to computer vision and medical imaging. However, even for some non-decomposable loss functions the 'max-oracle' problem can be solved efficiently in polynomial time. One example is the average precision based loss [137].
In what follows, we discuss the optimization algorithms in detail.

2.4.3.1 Subgradient descent algorithm

Subgradient descent is the simplest and easiest to implement algorithm for optimizing the SSVM [23, 108, 113, 114]. It uses the unconstrained SSVM objective function (2.35). As already discussed, this is a 'max' over an exponential number of affine functions, so it is non-smooth with many hinges. As the gradient is not defined at the hinge points we need to use a suitable 'subgradient' such as:

$$\nabla_{\mathbf{w}} f(\mathbf{w}) = \lambda \mathbf{w} - \frac{1}{n}\sum_{i}^{n} \Psi(\mathbf{x}_i, \mathbf{y}_i^*) \tag{2.56}$$

where,

$$\mathbf{y}_i^* = \operatorname*{argmax}_{\mathbf{y} \in \mathscr{Y}_i} (\Delta(\mathbf{y}_i, \mathbf{y}) - \mathbf{w}^\top \Psi(\mathbf{x}_i, \mathbf{y})) \tag{2.57}$$

This is exactly the 'max-oracle' problem (2.54) so it can be solved efficiently using the methods discussed at the beginning of this section. The resulting subgradient descent algorithm for SSVM is shown in Algorithm 2. The learning rate η can be any convergent series. Note that for each update of $\mathbf{w}$ we need to compute the subgradient which requires us to solve the 'max-oracle' problem for all of the samples. This makes the algorithm computationally inefficient. However, the gradient can be approximated using just few samples, known as online (sometimes batch) subgradient descent. Doing this decreases the cost of each step although the total number of steps needed for the convergence increases.

2.4.3.2 The cutting plane algorithm

The cutting plane algorithm [61] can be used to optimize either the primal or dual constrained form of the SSVM objective both having exponentially many constraints. The idea behind the cutting plane method is rather simple. Instead of solving the problem with all of the constraints active at once, the algorithm iteratively adds constraints that are violated (known as the *most violated constraint*) to a constraint set and re-optimizes the cost over this small

Algorithm 2 Subgradient descent algorithm for SSVM.

```
input $\mathscr{D}$, $\mathbf{w}_0$, $\lambda$, the tolerance $\varepsilon$.
t = 0, $\mathbf{w}_t = \mathbf{w}_0$, the learning rate $\eta = 1$.
repeat
    Compute $\mathbf{y}_i^*$ at $\mathbf{w}_t$ for each sample by solving the 'max-oracle' problem defined in equation (2.54).
    Compute the subgradient $\nabla_{\mathbf{w}_t} f(\mathbf{w})$ at $\mathbf{w}_t$ using equation (2.56).
    Update $\mathbf{w}_{t+1}$
        $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla_{\mathbf{w}_t} f(\mathbf{w})$        (2.58)
    $t = t + 1$ and $\eta = \frac{2}{t+1}$
until $f(\mathbf{w}_t) - f(\mathbf{w}_{t+1}) \leq \varepsilon$.
```

but growing subset of the constraints. In particular, the process can repeatedly add the most violated constraint and re-optimize until no violated constraint can found to add to the constraint set. In spite of the fact that the original problem contains exponentially many constraints, the cutting plane algorithm guarantees convergence when a polynomial number of violated constraints have been added to the working set.

Notice that the objective function (2.38) has only one slack variable ξ_i for each sample, shared across all labelings from the output set $\mathscr{Y}_i$. Therefore it has n slacks in total, which makes the problem size dependent on the size of the dataset. However, we can convert this to a '1-slack' formulation with a number of variables independent of the size of the dataset, as follows [61]:

$$\min_{\mathbf{w},\xi} \quad \frac{\lambda}{2}\|\mathbf{w}\|^2 + \xi \tag{2.59}$$

$$\text{s.t.} \quad \mathbf{w}^\top \frac{1}{n}\sum_{i=1}^{n} \Psi(\mathbf{x}_i, \bar{\mathbf{y}}_i) \geq \frac{1}{n}\sum_{i=1}^{n} \Delta(\mathbf{y}_i, \bar{\mathbf{y}}_i) - \xi, \quad \forall(\bar{\mathbf{y}}_1, \cdots, \bar{\mathbf{y}}_n) \in \mathscr{Y}^n. \tag{2.60}$$

where $\mathscr{Y}^n = \mathscr{Y}_1 \times \cdots \times \mathscr{Y}_n$. While the n-slack optimization problem (2.38) has $\sum_i |\mathscr{Y}_i|$ constraints, the 1-slack optimization problem (2.59) has $|\mathscr{Y}^n| = \prod_i |\mathscr{Y}_i|$ constraints, far more than the n-slack formulation. However, the 1-slack formulation has only one slack variable shared across all $|\mathscr{Y}^n|$ constraints. In order to find the *most violated constraint* for a given $\mathbf{w}$ we need to solve the following problem:

$$(\hat{\mathbf{y}}_1, \cdots, \hat{\mathbf{y}}_n) = \operatorname*{argmax}_{\forall(\bar{\mathbf{y}}_1, \cdots, \bar{\mathbf{y}}_n) \in \mathscr{Y}^n} \quad \xi := \frac{1}{n}\sum_{i=1}^{n} \Delta(\mathbf{y}_i, \bar{\mathbf{y}}_i) - \mathbf{w}^\top \frac{1}{n}\sum_{i=1}^{n} \Psi(\mathbf{x}_i, \bar{\mathbf{y}}_i). \tag{2.61}$$

Algorithm 3 Cutting plane algorithm for 1-slack SSVM.

```
input $\mathscr{D}$, $\lambda$, tolerance $\varepsilon$.
t = 0, $\mathscr{W} \leftarrow \emptyset$.
repeat
    Optimize the convex problem:
        $$(\mathbf{w},\xi) \leftarrow \underset{\mathbf{w},\xi}{\operatorname{argmin}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \xi \qquad (2.63)$$
        $$\text{s.t.} \quad \mathbf{w}^\top \frac{1}{n}\sum_{i=1}^{n}\Psi(\mathbf{x}_i,\hat{\mathbf{y}}_i) \geq \frac{1}{n}\sum_{i=1}^{n}\Delta(\mathbf{y}_i,\hat{\mathbf{y}}_i) - \xi, \quad \forall(\hat{\mathbf{y}}_1,\cdots,\hat{\mathbf{y}}_n) \in \mathscr{W}.$$
    Find the most violated constraint for each sample:
    for $i = 1,\cdots,n$ do
        get $\hat{\mathbf{y}}_i$ by solving problem (2.62).
    end for
    Update the constraint set (or working set): $\mathscr{W} \leftarrow \mathscr{W} \cup (\hat{\mathbf{y}}_1,\cdots,\hat{\mathbf{y}}_n)$.
    Compute the slack corresponding to the new most violated constraints (for the convergence criteria):
        $$\hat{\xi} = \frac{1}{n}\sum_{i=1}^{n}\Delta(\mathbf{y}_i,\hat{\mathbf{y}}_i) - \mathbf{w}^\top\frac{1}{n}\sum_{i=1}^{n}\Psi(\mathbf{x}_i,\hat{\mathbf{y}}_i). \qquad (2.64)$$
until $\hat{\xi} \leq \xi + \varepsilon$.
return $(\mathbf{w},\xi)$
```

This finds the set labelings, jointly, such that the slack variable (which is the upper bound on the empirical loss) is maximized for a given $\mathbf{w}$. In general, the joint maximization of this problem is very difficult. However, the problem is linear in $\mathbf{y}_i$ so it decomposes over $\mathbf{y}_i$. Hence, the joint maximization can be achieved by maximizing independently over each $\mathbf{y}_i$ as follows:

$$\hat{\mathbf{y}}_i \leftarrow \underset{\forall\bar{\mathbf{y}}_i\in\mathscr{Y}_i}{\operatorname{argmax}} \Big(\Delta(\mathbf{y}_i,\bar{\mathbf{y}}_i) - \mathbf{w}^\top\Psi(\mathbf{x}_i,\bar{\mathbf{y}}_i)\Big). \quad : \textit{`most violated constraint'} \qquad (2.62)$$

Notice that this is again the 'max-oracle' problem (2.54) so it can be solved efficiently using the inference algorithms discussed in section 2.3. The complete cutting plane algorithm for optimizing the 1-slack formulation of SSVM is shown in the Algorithm 3. Upon convergence, the algorithm guarantees that the learned $\mathbf{w}$ has a duality gap of at most $\frac{\varepsilon}{\lambda}$.

2.4.3.3 Frank-Wolfe and Block-Coordinate Frank-Wolfe algorithm for SSVM

An overview of the Frank-Wolfe (FW) algorithm [43, 59, 60] is given in Appendix B.3 (see Algorithm 19 there). To summarize, the FW algorithm is optimizes a continuously differentiable convex function over a convex and compact domain. In the context of the optimization of SSVM, the FW approach can be divided into following four steps: (1) solving the linearization problem; (2) convex combination to update the variables; (3) obtaining the optimal step size; and (4) computing the duality gap. Recall that in the FW algorithm, the linearization duality gap (or Fenchel duality gap) is obtained as a by-product without additional computation. In the case of SSVM, the linearization duality gap and the Lagrange duality gap turn out to be exactly the same, and thus can be used as the convergence criterion. The FW and BCFW algorithms for optimizing SSVM [86] are given in Algorithm 4 and Algorithm 5, respectively. Recall that the dual objective function of SSVM (2.49) is:

$$\begin{aligned} \min_{\alpha} \quad & f(\alpha) := \frac{\lambda}{2} \|A\alpha\|^2 - \mathbf{b}^\top \alpha \\ s.t. \quad & \sum_{\mathbf{y} \in \mathscr{Y}_i} \alpha_i(\mathbf{y}) = 1, \forall i \in [n], \\ & \alpha_i(\mathbf{y}) \geq 0, \forall i \in [n], \forall \mathbf{y} \in \mathscr{Y}_i. \end{aligned} \tag{2.65}$$

The function $f(\alpha)$ is convex and smooth as it is the negative of the dual function which is concave and smooth (does not contain any ‘max’ terms). For each sample the constraints $\sum_{\mathbf{y} \in \mathscr{Y}_i} \alpha_i(\mathbf{y}) = 1$ and $\alpha_i(\mathbf{y}) \geq 0, \forall \mathbf{y} \in \mathscr{Y}_i$, together form a probability simplex $\mathscr{M}_i$, which is known to be convex and compact. Since the dual variable vector α is the concatenation of α_i for all $i \in [n]$, the combined constraints form a Cartesian product of n probability simplices denoted as $\mathscr{M} = \mathscr{M}_1 \times \cdots \times \mathscr{M}_n$, which is a compact and convex polyhedral. Hence, the FW algorithm can be applied to the dual formulation of SSVM. However, for the primal formulation of SSVM (equation (2.36)), there is no guarantee of the compactness of the domain formed by the constraints, so the FW algorithm can not be used.

Briefly, the BCFW algorithm is an online and faster version of FW algorithm. As we have seen in the dual form of SSVM that each sample forms a simplex. These simplices are independent of one another. Based on this insight, the BCFW algorithm stores $\mathbf{w}_i^{th}$ equal contribution (2.53) to the overall parameter vector $\mathbf{w}$ and the loss ℓ_i for each i-th sample and updates them individually. The final parameter vector $\mathbf{w}$ and the loss ℓ are approximately updated using $\mathbf{w}_i$ and loss l_i respectively. Thus, requires only one ‘max-oracle’ call for each update of the parameter vector. This is a much cheaper operation than the parameter updates of the non-block FW algorithm, which requires ‘max-oracle’ calls for all n samples in the dataset.

In what follows, we describe the above mentioned four steps of the FW and BCFW algorithms for dual SSVM optimization in detail.

Solving the linearization problem. The linearization problem at a given α is defined as (refer to equation (B.22) in Appendix for details):

$$\mathbf{s} = \underset{\mathbf{p}\in\mathscr{M}}{\operatorname{argmin}} \langle \mathbf{p}, \nabla_{\alpha} f(\alpha)\rangle \tag{2.66}$$

where,

$$\nabla_{\alpha} f(\alpha) = \lambda A^{\top} A\alpha - \mathbf{b} \tag{2.67}$$

Using the relation $\mathbf{w} = A\alpha$ (from the KKT condition), we obtain $\nabla f(\alpha) = \lambda A^{\top}\mathbf{w} - \mathbf{b}$. Recall that, $A \in \mathbb{R}^{d\times m}$ is a matrix with column entries $\frac{\Psi(\mathbf{x}_i,\mathbf{y})}{\lambda n}$, and $\mathbf{b} \in \mathbb{R}^m$ is a vector with elements $\frac{\Delta(\mathbf{y}_i,\mathbf{y})}{n}$. Therefore,

$$\nabla f(\alpha) := \lambda A^{\top}\mathbf{w} - \mathbf{b} = \frac{1}{n}\begin{pmatrix} \vdots \\ \\ \underbrace{\mathbf{w}^{\top}\Psi(\mathbf{x}_i,\mathbf{y}) - \Delta(\mathbf{y}_i,\mathbf{y})}_{-H_i(\mathbf{y},\mathbf{w})} \\ \\ \vdots \end{pmatrix} \in \mathbb{R}^m. \tag{2.68}$$

where, H_i is the augmented loss. As (2.66) is equivalent to solving a linear program with respect to $\mathbf{p}$ over the polyhedron $\mathscr{M}$, it is guaranteed that at least one optimal solution lies on a vertex of the polyhedron. Notice that the problem $\min_{\mathbf{p}\in\mathscr{M}}\langle\nabla f(\alpha),\mathbf{p}\rangle$ splits linearly over $\mathbf{p}_i \in \mathscr{M}_i$, so it can be solved separately over each simplex $\mathbf{p}_i$ leading to the decomposition $\sum_i \min_{\mathbf{p}_i\in\mathscr{M}_i}\langle\nabla f_i(\alpha),\mathbf{p}_i\rangle$. Here, the vector $\nabla f_i(\alpha) \in \mathbb{R}^{|\mathscr{Y}_i|}$ contains the entries corresponding to the i-th sample (or block) in the vector $\nabla f(\alpha) \in \mathbb{R}^m$:

$$\nabla f_i(\alpha) = -\frac{1}{n}\begin{pmatrix} H_i(\mathbf{y}_1,\mathbf{w}) \\ \vdots \\ H_i(\mathbf{y}_{|\mathscr{Y}_i|},\mathbf{w}) \end{pmatrix} \in \mathbb{R}^{|\mathscr{Y}_i|}. \tag{2.69}$$

Therefore, the minimization over the product of simplices decomposes into separate minimizations over the simplices, which can be written as:

$$\mathbf{s}_i = \underset{\mathbf{p}_i \in \mathscr{M}_i}{\operatorname{argmin}} \langle \nabla f_i(\alpha_i), \mathbf{p}_i \rangle. \tag{2.70}$$

As we only need to search over the corners, the problem further reduces to finding the corner corresponding to the labeling $\mathbf{y}$ for which $-H_i(\mathbf{y}, \mathbf{w})$ attains its minimum value. This is precisely the 'max-oracle' problem $\mathbf{y}_i^* = \operatorname{argmax}_{\mathbf{y} \in \mathscr{Y}_i} H_i(\mathbf{y}, \mathbf{w})$ discussed in section 2.4.3. *So, the* FW *linearization step is exactly the same as solving the 'max-oracle' problem for each sample.* For BCFW, instead of solving max-oracle for all of the samples, it is solved only for one block in order to update the corresponding weight contribution, as will be seen shortly.

Convex combination. Let γ be the optimal step size, α^k be the current solution, and $\mathbf{s}$ be the corner obtained by solving the linearization problem. In the case of FW, the dual variables and the parameter vector can be updated as follows:

$$\begin{aligned} \alpha^{k+1} &= \gamma \mathbf{s} + (1-\gamma)\alpha^k \\ A\alpha^{k+1} &= \gamma A\mathbf{s} + (1-\gamma)A\alpha^k \\ \mathbf{w}^{k+1} &= \gamma \mathbf{w}^s + (1-\gamma)\mathbf{w}^k \end{aligned} \tag{2.71}$$

Notice that instead of updating the very high dimensional dual variable vector, we can directly update the primal variables. Thus, even though the algorithm works in the dual, it suffices to dealing with the low dimensional primal variables. Similarly, for FW the loss can be updated as follows:

$$\begin{aligned} \ell^{k+1} =& \mathbf{b}^\top \alpha^{k+1} = \mathbf{b}^\top (\gamma \mathbf{s} + (1-\gamma)\alpha^k) \\ =& \gamma \ell^s + (1-\gamma)\ell^k \end{aligned} \tag{2.72}$$

where $\ell^s = \frac{1}{n}\sum_i^n \Delta(\mathbf{y}_i^*, \mathbf{y}_i)$. For BCFW, as we update only one block the variables corresponding to the other blocks remain unchanged. Hence, $(\alpha - \mathbf{s}) = (\alpha_{[i]} - \mathbf{s}_{[i]})$, where, $\mathbf{a}_{[i]}$ denotes the vector with zero everywhere except for the i-th block. Therefore, the dual update can be written as:

$$\begin{aligned} \alpha^{k+1} &= \alpha^k + \gamma(\mathbf{s}_{[i]} - \alpha^k_{[i]}), \\ A\alpha^{k+1} &= A\alpha^k + \gamma(A\mathbf{s}_{[i]} - A\alpha^k_{[i]}), \\ \mathbf{w}^{k+1} &= \mathbf{w}^k + \gamma(\mathbf{w}_i^s - \mathbf{w}_i^k). \end{aligned} \tag{2.73}$$

Again, instead of working with the high dimensional dual variables, we can work only with the low dimensional primal ones. Since $\alpha_{[i]}^{k+1} = \alpha_{[i]}^{k} + \gamma(\mathbf{s}_{[i]} - \alpha_{[i]}^{k})$, using $\mathbf{w}_i^{k+1} = \mathbf{w}_i^k + \gamma(\mathbf{w}_i^s - \mathbf{w}_i^k)$ in the above equation we obtain

$$\mathbf{w}^{k+1} = \mathbf{w}^k + \mathbf{w}_i^{k+1} - \mathbf{w}_i^k \tag{2.74}$$

Similarly, the loss can be updated as

$$\ell^{k+1} = \ell^k + \ell_i^{k+1} - \ell_i^k \tag{2.75}$$

where, $\ell_i^{k+1} = \ell_i^k + \gamma(\ell_i^s - \ell_i^k)$ and $\ell_i^s = \frac{1}{n}\Delta(\mathbf{y}_i^*, \mathbf{y}_i)$. Note that for BCFW the update depends on only one block, so it is much cheaper than an FW update.

Lagrange duality gap and linearization duality gap. Let us first derive the expression for the linearization duality gap. The FW linearization duality gap is defined as (see Appendix B.3 for details):

$$g_{lin}(\alpha;\lambda) := \max_{\mathbf{p}\in\mathscr{M}} \langle \alpha - \mathbf{p}, \nabla_\alpha f(\alpha)\rangle = \langle \alpha - \mathbf{s}, \nabla_\alpha f(\alpha)\rangle \tag{2.76}$$

where $\mathbf{s} = \text{argmin}_{\mathbf{p}\in\mathscr{M}}\langle \mathbf{p}, \nabla_\alpha f(\alpha)\rangle$. Recall that obtaining $\mathbf{s}$ is exactly the same as solving the linearlization problem, which we have already discussed. Using the gradient (2.67) we obtain the following expression for the linearization duality gap:

$$g_{lin}(\alpha;\lambda) = \lambda\big(A(\alpha - \mathbf{s})\big)^\top A\alpha - (\alpha - \mathbf{s})^\top \mathbf{b} \tag{2.77}$$

For FW algorithm the final expression can be written as:

$$g_{lin}(\alpha;\lambda) = \lambda(\mathbf{w} - \mathbf{w}^s)^\top \mathbf{w} - (\ell - \ell^s) \tag{2.78}$$

For BCFW, using $(\alpha - \mathbf{s}) = (\alpha_{[i]} - \mathbf{s}_{[i]})$, the linearization gap for the i-th block (known as the block-wise linearization gap) can be written as:

$$g_{lin}^i(\alpha;\lambda) = \lambda(\mathbf{w}_i - \mathbf{w}_i^s)^\top \mathbf{w} - (\ell_i - \ell_i^s) \tag{2.79}$$

Notice that $g_{lin}(\alpha;\lambda) = \sum_i g_{lin}^i(\alpha;\lambda)$. Another useful form of the block-wise duality gap can be obtained by using equation (2.68) into the block-wise form of equation (2.76) as follows:

$$g_{lin}^i(\alpha;\lambda) = \langle \alpha_{[i]} - \mathbf{s}_{[i]}, \nabla_{\alpha_{[i]}} f(\alpha)\rangle = \max_{\mathbf{y}\in\mathscr{Y}_i} H_i(\mathbf{y};\mathbf{w}) - \sum_{\mathbf{y}\in\mathscr{Y}_i} \alpha_i(\mathbf{y}) H_i(\mathbf{y};\mathbf{w}). \tag{2.80}$$

Similarly,

$$g_{lin}(\alpha;\lambda) = \frac{1}{n}\sum_i \left(\max_{\mathbf{y}\in\mathscr{Y}_i} H_i(\mathbf{y};\mathbf{w}) - \sum_{\mathbf{y}\in\mathscr{Y}_i} \alpha_i(\mathbf{y})H_i(\mathbf{y};\mathbf{w})\right) \tag{2.81}$$

The above forms of the duality gaps will used in chapter 5 for the derivation of the regularization path of SSVM. For the convergence criterion, we are interested in the Lagrange duality gap. The Lagrange duality gap is the difference between the primal and dual objective functions:

$$g_{Lag}(\alpha,\mathbf{w};\lambda) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n}\max_{\mathbf{y}\in\mathscr{Y}_i} H_i(\mathbf{y};\mathbf{w}) - \left(\mathbf{b}^\top\alpha - \frac{\lambda}{2}\|\mathbf{w}\|^2\right) \tag{2.82}$$

Using the fact that $\max_{\mathbf{y}\in\mathscr{Y}_i} H_i(\mathbf{y};\mathbf{w}) = \max_{\mathbf{p}\in\mathscr{M}}\langle\mathbf{p}, -\nabla_\alpha f(\alpha)\rangle$ we obtain

$$g_{Lag}(\alpha,\mathbf{w};\lambda) = \langle\alpha-\mathbf{s},\nabla_\alpha f(\alpha)\rangle = g_{lin}(\alpha;\lambda) \tag{2.83}$$

So for SSVM*, the linearization duality gap and the Lagrange duality gap are exactly the same [86].*

Optimal step size. Obtaining the optimal step size for the FW line search is equivalent to solving the problem $\gamma_{opt} := \operatorname{argmin}_{\gamma\in[0,1]} f(\alpha+\gamma(\mathbf{s}-\alpha))$, where

$$f(\alpha+\gamma(\mathbf{s}-\alpha)) \quad = \quad \frac{\lambda}{2}\|A(\alpha+\gamma(\mathbf{s}-\alpha))\|^2 - \mathbf{b}^\top(\alpha+\gamma(\mathbf{s}-\alpha)) \tag{2.84}$$

For the dual of SSVM this is a quadratic function in α, so the optimal step size can be obtained analytically by setting $\nabla_\gamma f(\alpha+\gamma(\mathbf{s}-\alpha)) = 0$. This leads to the following optimal step size:

$$\gamma_{opt} = \frac{\lambda\left(A(\alpha-\mathbf{s})\right)^\top A\alpha - (\alpha-\mathbf{s})^\top\mathbf{b}}{\lambda\|A\alpha - A\mathbf{s}\|^2} \tag{2.85}$$

Notice that the numerator of this equation is the linearization duality gap shown in equation (2.77). Therefore, for FW $\gamma_{opt} = \frac{g_{lin}(\alpha;\lambda)}{\lambda\|\mathbf{w}^k-\mathbf{w}^s\|^2}$ and for BCFW $\gamma_{opt} = \frac{g^i_{lin}(\alpha;\lambda)}{\lambda\|\mathbf{w}^k_i-\mathbf{w}^s_i\|^2}$.

2.5 Weakly Supervised SSVM

In many tasks related to structured prediction, it is difficult and very expensive to obtain a fully supervised dataset. For example, in the case of object detection where the task it to locate instances of the object in a given image, it is comparatively easy to get a training

Algorithm 4 The Frank-Wolfe algorithm for SSVM.

```
input $\mathscr{D}$, $K$, tolerance $\varepsilon$.
$\mathbf{w}^0 = 0$, $l^0 = 0$.
repeat
    Solve the 'max-oracle' problem for all the samples:
    for $i = 1, \cdots, n$ do
        $\mathbf{y}_i^* := \mathrm{argmax}_{\mathbf{y} \in \mathscr{Y}_i} H_i(\mathbf{y}; \mathbf{w}^k)$.
    end for
    $\mathbf{w}_s = \sum_i^n \frac{1}{\lambda n} \Psi(\mathbf{x}_i, \mathbf{y}_i^*)$ and $l_s = \frac{1}{n} \sum_i^n \Delta(\mathbf{y}_i, \mathbf{y}_i^*)$.
    Find the optimal step size $\gamma$ using equation (2.85) and clip to $[0, 1]$.
    Update: $\mathbf{w}^{k+1} = \gamma \mathbf{w}_s + (1 - \gamma)\mathbf{w}^k$, and $l^{k+1} = \gamma l_s + (1 - \gamma) l^k$.
    Convergence criterion – (1) use upper limit ($K$) on the number of iterations ($k$); or (2)
    use linearization duality gap $g(\alpha_k; \lambda)$ obtained using equation (2.78); or (3) use both.
until Converged.
return $\mathbf{w}^{k+1}$
```

Algorithm 5 The Block-Coordinate Frank-Wolfe algorithm for SSVM.

```
input $\mathscr{D}$, $K$.
$\mathbf{w}^0 = \mathbf{w}_i^0 = 0$, $l^0 = l_i^0 = 0$.
for $k = 0, \cdots, K$ do
    Pick $i$ at random in $\{1, \cdots, n\}$
    Solve the 'max-oracle' problem for the $i$-th sample: $\mathbf{y}_i^* := \mathrm{argmax}_{\mathbf{y} \in \mathscr{Y}_i} H_i(\mathbf{y}; \mathbf{w}^k)$.
    $\mathbf{w}^s = \frac{1}{\lambda n} \Psi(\mathbf{x}_i, \mathbf{y}_i^*)$ and $l^s = \frac{1}{n} \Delta(\mathbf{y}_i, \mathbf{y}_i^*)$.
    Find the optimal step size $\gamma$ using equation (2.85) and clip to $[0, 1]$.
    Update: $\mathbf{w}_i^{k+1} = \gamma \mathbf{w}_s + (1 - \gamma)\mathbf{w}_i^k$, and $l_i^{k+1} = \gamma l_s + (1 - \gamma) l_i^k$.
    Update: $\mathbf{w}^{k+1} = \mathbf{w}^k + \mathbf{w}_i^{k+1} - \mathbf{w}_i^k$, and $l^{k+1} = l^k + l_i^{k+1} - l_i^k$.
end for
return $\mathbf{w}^{k+1}$
```

dataset with millions of images with labels indicating the presence or the absence of the object in the image but much more expensive to supply bounding box information for each object in each image. There are many similar situations in which useful modeling annotations are missing from the dataset. We call this 'latent or missing or hidden' information and such datasets are called 'weakly labelled datasets'. Given such a dataset, a common practice is to introduce *latent* variables to represent the missing modeling information so that inference must recover the values of these. In the context of SSVM, such problems fall into the category of weakly supervised learning and can be formulated as *latent structured* SVM (LSSVM) ones [135].

2.5.1 Latent SSVM

Latent SSVM is an extension of SSVM that includes latent variables. Suppose we are given a weak labelled training set $\mathscr{D}$. Let us denote the missing (or hidden) modeling information as $\mathbf{h} \in \mathscr{H}$. In order to enrich the modeling using the missing information, the SSVM joint feature map $\Phi(\mathbf{x}, \mathbf{y})$ is further extended to include the latent variable $\mathbf{h}$, $\Phi(\mathbf{x}, \mathbf{y}, \mathbf{h})$.
Given the parameter vector $\mathbf{w}$, LSSVM prediction (or inference) is performed jointly over the output space and the latent space as follows:

$$(\hat{\mathbf{y}}, \hat{\mathbf{h}}) = \operatorname*{argmax}_{(\mathbf{y},\mathbf{h}) \in \mathscr{Y} \times \mathscr{H}} \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{y}, \mathbf{h}). \tag{2.86}$$

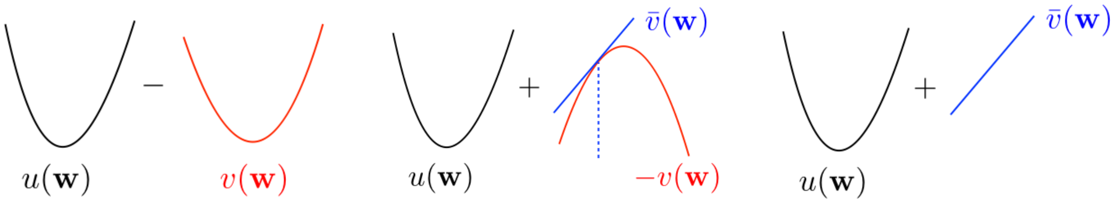
2.5.2 Difference of convex upper bounds of the empirical loss

For a given $\mathbf{w}$, ground truth lableling $\mathbf{y}$, predicted labeling $\hat{\mathbf{y}}$, and predicted latent variable $\hat{\mathbf{h}}$, the LSSVM loss function $\Delta(\mathbf{y}, \hat{\mathbf{y}}, \hat{\mathbf{h}})$ can be upper bounded as follows:

$$\begin{aligned} \Delta(\mathbf{y}, \hat{\mathbf{y}}, \hat{\mathbf{h}}) \leq & \quad \Delta(\mathbf{y}, \hat{\mathbf{y}}, \hat{\mathbf{h}}) + \max_{(\bar{\mathbf{y}},\bar{\mathbf{h}}) \in \mathscr{Y} \times \mathscr{H}} \mathbf{w}^\top \Phi(\mathbf{x}, \bar{\mathbf{y}}, \bar{\mathbf{h}}) - \max_{\bar{\mathbf{h}} \in \mathscr{H}} \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{y}, \bar{\mathbf{h}}) \\ \leq & \quad \max_{(\bar{\mathbf{y}},\bar{\mathbf{h}}) \in \mathscr{Y} \times \mathscr{H}} \Big(\Delta(\mathbf{y}, \bar{\mathbf{y}}, \bar{\mathbf{h}}) + \mathbf{w}^\top \Phi(\mathbf{x}, \bar{\mathbf{y}}, \bar{\mathbf{h}}) \Big) - \max_{\bar{\mathbf{h}} \in \mathscr{H}} \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{y}, \bar{\mathbf{h}}) \end{aligned} \tag{2.87}$$

Minimizing this bounded loss over the entire dataset under L_2 regularization leads to the following learning problem:

$$\begin{aligned} \min_{\mathbf{w}} \Bigg\{ \frac{\lambda}{2} \|\mathbf{w}\|^2 \quad + & \quad \frac{1}{n} \sum_i \max_{(\bar{\mathbf{y}}_i,\bar{\mathbf{h}}_i) \in \mathscr{Y}_i \times \mathscr{H}_i} \Big(\Delta(\mathbf{y}_i, \bar{\mathbf{y}}_i, \bar{\mathbf{h}}_i) + \mathbf{w}^\top \Phi(\mathbf{x}_i, \bar{\mathbf{y}}_i, \bar{\mathbf{h}}_i) \Big) \\ - & \quad \frac{1}{n} \sum_i^n \max_{\bar{\mathbf{h}}_i \in \mathscr{H}_i} \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{y}_i, \bar{\mathbf{h}}_i) \Bigg\} \end{aligned} \tag{2.88}$$



(a) Difference of Convex (b) Convex plus concave (c) Convex plus affine

Figure 2.11: Pictorial representation of one iteration of the CCCP algorithm [138]. Figure 2.11a shows the given difference of convex functions, which can be written as a sum of convex and concave functions. Figure 2.11b shows the linear upper bound for the concave function ($-v(\mathbf{w})$) at a given point, which results in an affine function $\bar{v}(\mathbf{w})$. Finally, Figure 2.11c shows the resulting convex function as a sum of convex $u(\mathbf{w})$ and affine $\bar{v}(\mathbf{w})$ ones.

Similarly to SSVM as discussed in section 2.4.1, the above learning problem can be written in constrained form as follows:

$$\min_{\mathbf{w},\xi} \quad \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n}\xi_i \tag{2.89}$$

$$\text{s.t.} \max_{\mathbf{h}_i \in \mathscr{H}_i} \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i) \geq \mathbf{w}^\top \Phi(\mathbf{x}_i, \bar{\mathbf{y}}_i, \bar{\mathbf{h}}_i) + \Delta(\mathbf{y}_i, \bar{\mathbf{y}}_i, \bar{\mathbf{h}}_i) - \xi_i, \tag{2.90}$$

$$\forall i, \forall \bar{\mathbf{y}}_i \in \mathscr{Y}_i, \forall \bar{\mathbf{h}}_i \in \mathscr{H}_i.$$

Let $s(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i; \mathbf{w}) = \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i)$ be the score for a given $\mathbf{w}$ when input $\mathbf{x}_i$ is assigned the output $\mathbf{y}_i$ and the latent variable $\mathbf{h}_i$. The constraints (2.90) introduce a margin between the score of the ground truth output together with the best corresponding value of the latent variable, and any other pair of output and latent variables, with the margin required being proportional to the loss.

2.5.3 Optimization of LSSVM using CCCP algorithm

The objective (2.88) optimizes an upper bound on the empirical risk but it is a non-convex program so it can not be optimized efficiently to obtain the optimal set of parameters. However, the objective belongs to a special family of non-convex functions known as difference of convex functions [55, 98], with $u(\mathbf{w})$ and $v(\mathbf{w})$ below as the two convex components:

$$f(\mathbf{w}) := \underbrace{\frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n} \max_{(\bar{\mathbf{y}}_i,\bar{\mathbf{h}}_i)\in\mathscr{Y}_i\times\mathscr{H}_i}\Big(\Delta(\mathbf{y}_i,\bar{\mathbf{y}}_i,\bar{\mathbf{h}}_i) + \mathbf{w}^\top\Phi(\mathbf{x}_i,\bar{\mathbf{y}}_i,\bar{\mathbf{h}}_i)\Big)}_{u(\mathbf{w})} - \underbrace{\frac{1}{n}\sum_{i=1}^{n}\max_{\bar{\mathbf{h}}_i\in\mathscr{H}_i}\mathbf{w}^\top\Phi(\mathbf{x}_i,\mathbf{y}_i,\bar{\mathbf{h}}_i)}_{v(\mathbf{w})} \tag{2.91}$$

For such families, the well known CCCP algorithm [138] can be used to obtain a local minimum or a saddle point. Broadly speaking, the CCCP algorithm consists of three steps — (1) upper bounding the concave part $(-v(\mathbf{w}))$ at a given $\mathbf{w}$, which produces an affine function in $\mathbf{w}$; (2) optimizing the resulting convex function (a sum of convex and affine functions being convex), which in the case of LSSVM is the objective function of SSVM; (3) repeating the above steps until the objective can no longer be decreased by more than a given tolerance of ε. A pictorial representation of the above mentioned steps is shown in Figure 2.11.

The complete CCCP algorithm for the optimization of LSSVM is shown in Algorithm 6. The first step of upper bounding the concave functions (Algorithm 6, Line 3) is called the *latent imputation step*. The second step is the optimization of the resulting convex subproblem (Algorithm 6, Line 4), which in this case is the optimization of SSVM and leads to the parameter update.

Algorithm 6 The CCCP algorithm for the optimization of LSSVM.

```
input $\mathscr{D}$, $\mathbf{w}_0$, $\lambda$, tolerance $\varepsilon$.
1: $t \leftarrow 0$
2: repeat
3:     Impute Latent Variables $\mathbf{h}_i^* = \mathrm{argmax}_{\mathbf{h}\in\mathscr{H}_i} \mathbf{w}^\top\Phi(\mathbf{x}_i,\mathbf{y}_i,\mathbf{h})$, for all $i \in [n]$.
4:     Update $\mathbf{w}_{t+1}$ by fixing the latent variables for the output $\mathbf{y}_i$ to $\mathbf{h}_i^*$ and solving the
       corresponding SSVM problem.
5:     $t \leftarrow t+1$
6: until $f(\mathbf{w}_t) - f(\mathbf{w}_{t+1}) \leq \varepsilon$.
7: return $\mathbf{w}_{t+1}$
```

Chapter 3

Parsimonious Labeling

3.1 Introduction

The *labeling problem*, also called the inference problem (chapter 2, section 2.3), provides an intuitive formulation for several tasks in computer vision and related areas. In this chapter, we talk about a new family of labeling problems, which we call *parsimonious labeling*, and propose move making algorithms with theoretical guarantees for the optimization.

Briefly, the labeling problem is defined using a set of random variables, each of which can take a value from a finite and discrete label set. The assignment of values to all the variables is referred to as a labeling. In order to quantitatively distinguish between the large number of putative labelings, we are provided with an energy function that maps a labeling to a real number. The energy function consists of two types of terms: (i) the unary potential, which depends on the label assigned to one random variable; and (ii) the clique potential, which depends on the labels assigned to a subset of random variables. The goal of the labeling problem is to obtain the labeling that minimizes the energy. A well-studied special case of the labeling problem is metric labeling [15, 65]. Here, the unary potentials are arbitrary. However, the clique potentials are specified by a user-defined metric distance function of the label space. Specifically, the clique potentials satisfy the following two properties: (i) each clique potential depends on two random variables; and (ii) the value of the clique potential (also referred to as the pairwise potential) is proportional to the metric distance between the labels assigned to the two random variables. Metric labeling has been used to formulate several problems in low-level computer vision, where the random variables correspond to image pixels. In such scenarios, it is natural to encourage two random variables that correspond to two nearby pixels in the image to take similar labels. However, by restricting the size of the cliques to two, metric labeling fails to capture more informative high-order cues. For example, it cannot encourage an arbitrary sized set of similar pixels (such as pixels that define a homogeneous superpixel) to take similar labels.

We propose a new family of discrete energy minimization problems, which we call parsimonious labeling. Parsimonious labeling is a natural generalization of the metric labeling problem for high-order potentials. The energy function of parsimonious labeling consists of unary potentials and high-order clique potentials. Similar to metric labeling, the unary potentials are arbitrary. However, the clique potentials can be defined on any subset of random variables, and their value depends on the set of unique labels assigned to the random variables in the clique. In more detail, the clique potential is defined using the recently proposed notion of a *diversity* [18, 19], which generalizes metric distance functions to all subsets of the label set. By minimizing the diversity, our energy function encourages the labeling to be parsimonious, that is, use as few labels as possible. This in turn allows us to capture useful cues for important low-level computer vision applications such as stereo correspondence and image denoising. Furthermore, in order to be practically useful, we propose an efficient graph-cuts based algorithm for the parsimonious labeling problem that provides strong theoretical guarantees on the quality of the solution. Our algorithm consists of three steps. First, we approximate a given diversity using a mixture of a novel *hierarchical P^n Potts model* (a generalization of the P^n Potts model [66]). Second, we use a divide-and-conquer approach for each mixture component, where each subproblem is solved using an efficient α-expansion algorithm. This provides us with a small number of putative labelings, one for each mixture component. Third, we choose the best putative labeling in terms of the energy value. Using both synthetic and standard real datasets, we show that our algorithm significantly outperforms other graph-cuts based approaches.

Summary of Contributions.

- Extension of metric labeling to high order cliques which we call *parsimonious labeling*.
- Hierarchical P^n-Potts model, which is a generalization of the P^n-Potts model defined over a given hierarchy of labels.
- Move making algorithm for the optimization of the hierarchical P^n-Potts model.
- Parsimonious labeling as a mixture of hierarchical P^n-Potts model, thus the above mentioned move making algorithm can be used for the optimization of the parsimonious labeling.
- Strong multiplicative bounds for the proposed move making algorithms for the optimization of hierarchical P^n-Potts model and parsimonious labeling.

3.2 Related Work

In last few years the research community has witnessed many successful applications of high-order random fields to solve low level vision related problems such as object segmentation [28, 30, 67, 87, 121, 129], disparity estimation [62, 131], and image restoration [89]. In this work, our focus is on methods that (i) rely on efficient move-making algorithms based on graph cuts; (ii) provide a theoretical guarantee on the quality of the solution. Below, we discuss the work most closely related to ours in more detail.

Kohli et al. [66] proposed the P^n Potts model, which enforces label consistency over a set of random variables. In [67], they presented a robust version of the P^n Potts model that takes into account the number of random variables that have been assigned an inconsistent label. Both the P^n Potts model and its robust version lend themselves to the efficient $\alpha-$expansion algorithm [66, 67]. Furthermore, the $\alpha-$expansion algorithm also provides a multiplicative bound on the energy of the estimated labeling with respect to the optimal labeling. While the robust P^n Potts model has been shown to be very useful for semantic segmentation, our generalization of the P^n Potts model offers a natural extension of the metric labeling problem and is therefore more widely applicable to several low-level vision tasks.

Delong et al. [28] proposed a global clique potential (label cost) that is based on the cost of using a label or a subset of labels in the labeling of the random variables. Similar to the P^n Potts model, the label cost based potential can also be minimized using $\alpha-$expansion. However, the theoretical guarantee provided by $\alpha-$expansion is an additive bound, which is not invariant to reparameterization of the energy function. Delong et al. [27] also proposed an extension of their work to hierarchical costs. However, the assumption of a given hierarchy over the label set limits its applications.

Ladicky *et al.* [87] proposed a global co-occurrence cost based high order model for a much wider class of energies that encourage the use of a small set of labels in the estimated labeling. Theoretically, the only constraint that [87] enforces in high order clique potential is that it should be monotonic in the label set. In other words, [87] can be regarded as a generalization of parsimonious labeling. However, they approximately optimize an upperbound on the actual energy function, which does not provide any optimality guarantees. In our experiments, we demonstrate that our move-making algorithm significantly outperforms their approach for the special case of parsimonious labeling.

Fix *et al.* [40] proposed an algorithm (SoSPD) for high-order random fields with arbitrary clique potentials. Each move of this algorithm requires us to approximately upperbound the clique potential into a sum of submodular functions and then optimize it using the submodular max-flow algorithm [71]. We show that our move making algorithm for parsimonious labeling has a much stronger multiplicative bound and better time complexity compared to [40].

3.3 Preliminaries

3.3.1 The labeling problem

As discussed in section 2.3, the labeling problem is to find the labeling that minimizes a given objective function, also known as the energy function. Hence, in this context, the inference problem and the labeling problems are the same. For a complete overview of the labeling problems please refer to the section 2.3. For the sake of completeness, below we give a brief description of the labeling problem. Consider a random field defined over a set of random variables $\mathbf{y} = \{y_1, \cdots, y_N\}$ arranged in a predefined lattice $V = \{1, \cdots, N\}$. Each random variable can take a value from a discrete label set $\mathscr{L} = \{l_1, \cdots, l_H\}$. Furthermore, let $\mathscr{C}$ denote the set of maximal cliques. Each maximal clique consists of a set of random variables that are all connected to each other in the lattice. A labeling is defined as the assignment or mapping of random variables to the labels. To assess the quality of each labeling $\mathbf{y}$ we define an energy function as:

$$E(\mathbf{y}) = \sum_{i \in V} \theta_i(y_i) + \sum_{c \in \mathscr{C}} \theta_c(\mathbf{y}_c). \tag{3.1}$$

where $\theta_i(y_i)$ is the unary potential of assigning a label y_i to the variable i, and $\theta_c(\mathbf{y}_c)$ is the clique potential for assigning the labels $\mathbf{y}_c$ to the variables in the clique c. Clique potentials are assumed to be non-negative. As will be seen shortly, this assumption is satisfied by the new family of energy functions proposed in this chapter. The total number of putative labelings is H^N, each of which can be assessed using its corresponding energy value. Within this setting, the labeling problem is to find the labeling corresponding to the minimum energy according to the function (3.1). Formally, the labeling problem is:

$$\mathbf{y}^* = \underset{\mathbf{y}}{\operatorname{argmin}} E(\mathbf{y}). \tag{3.2}$$

As already discussed in the section 2.3, the restrictions over the potentials, interactions among the random variables (structure), and compromise with the optimality guarantees leads to different models and their corresponding optimization algorithms. In this chapter, the most frequently used model is the P^n Potts model and the corresponding α-expansion algorithm for the optimization (discussed in section 2.3.3). For the sake of completeness, below we briefly define the P^n Potts model and give an idea of the corresponding $\alpha-$expansion algorithm for the optimization. For better understanding refer to the section 2.3.

P^n Potts model. An important special case of the labeling problem, which will be used throughout this paper, is defined by the P^n Potts model [66]. The P^n Potts model is a generalization of of the well known Potts model [106] for high-order energy functions (cliques can be of arbitrary sizes). For a given clique, the P^n Potts model is defined as:

$$\theta_c(\mathbf{y}_c) \propto \begin{cases} \gamma^k, & \text{if } y_i = l_k, \forall i \in c, \\ \gamma^{max}, & \text{otherwise,} \end{cases} \tag{3.3}$$

where γ^k is the cost of assigning all the nodes to the label $l_k \in \mathscr{L}$, and $\gamma^{max} > \gamma_k, \forall l_k \in \mathscr{L}$. Intuitively, the P^n Potts model enforces label consistency by assigning the cost of γ^{max} if there are more than one label in the given clique.

α-expansion for P^n Potts model. In order to solve the labeling problem corresponding to the P^n Potts model, Kohli *et al.* [66] proposed to use the $\alpha-$expansion algorithm [128]. The $\alpha-$expansion algorithm starts with an initial labeling, for example, by assigning each random variable to the label l_1. At each iteration, the algorithm moves to a new labeling by searching over a large *move space*. Here, the move space is defined as the set of labelings where each random variable is either assigned its current label or the label α. The key result that makes $\alpha-$expansion a computationally feasible algorithm for the P^n Potts model is that the minimum energy labeling within a move-space can be obtained using a single minimum st-cut operation on a graph that consists of a small number (linear in the size of the variables and the cliques) of vertices and arcs. The algorithm terminates when the energy cannot be reduced further for any choice of the label α. We refer the reader to [66] for further details.

3.3.2 Multiplicative Bound

An intuitive and commonly used measure of the accuracy of an approximation algorithm is the multiplicative bound. Formally, the multiplicative bound of a given algorithm is said to be B if the following condition is satisfied for all possible values of unary potential $\theta_i(.)$, and clique potentials $\theta_c(\mathbf{y}_c)$:

$$\sum_{i \in V} \theta_i(\hat{y}_i) + \sum_{c \in \mathscr{C}} \theta_c(\hat{\mathbf{y}}_c) \leq \sum_{i \in V} \theta_i(y_i^*) + B \sum_{c \in \mathscr{C}} \theta_c(\mathbf{y}_c^*). \tag{3.4}$$

Here, $\hat{\mathbf{y}}$ is the labeling estimated by the algorithm and $\mathbf{y}^*$ is the globally optimal labeling. By definition of an optimal labeling (one that has the minimum energy), the multiplicative bound will always be greater than or equal to one [83]. One interesting question that arise is *what is the best multiplicative bound an α-expansion type move making algorithm can* *provide for any arbitrary energy function?* Before answering to this question, let us first define λ as follows [50]:

$$\lambda = \max_{\mathbf{y}_c : |\mathbf{y}_c| > 1} \left(\frac{\max_{\mathbf{y}_c} \theta_c(\mathbf{y}_c)}{\min_{\mathbf{y}_c : \theta_c(\mathbf{y}_c) \neq 0} \theta_c(\mathbf{y}_c)} \right). \tag{3.5}$$

Using the above defined λ, the following theorem answers the above posed question.

Theorem 1. *Given a labeling problem defined using the energy function (3.1) over any arbitrary unary potentials, and arbitrary clique potentials defined over cliques of arbitrary sizes. Any α-expansion type move making algorithm with optimal expansion moves provide the multiplicative bound of $B = \lambda \min(\mathscr{M}, |\mathscr{L}|)$ for the labeling problem (3.2), where, $\mathscr{M}$ is the size of the largest maximal clique in the graph, $|\mathscr{L}|$ is the number of labels, and λ is defined in equation (3.5). Mathematically,*

$$\sum_{i \in V} \theta_i(\hat{y}_i) + \sum_{c \in \mathscr{C}} \theta_c(\hat{\mathbf{y}}_c) \leq \sum_{i \in V} \theta_i(y_i^*) + \lambda \min(\mathscr{M}, |\mathscr{L}|) \sum_{c \in \mathscr{C}} \theta_c(\mathbf{y}_c^*), \tag{3.6}$$

where, $\hat{\mathbf{y}}$ is the labeling corresponding to the local minima in the exansion move space obtained using the α-expansion and $\mathbf{y}^*$ is the labeling corresponding to the global minima.

Proof. The proof is mainly along the lines of thought of [50] and [128]. In [128], the multiplicative bound is provided for the energy function with pairwise clique potentials, specially in case of *metric labeling*. In [50], the proof is for the general high-order clique potentials. A small observation allows us to provide a stronger bound than [50]. The complete proof is as follows.

Let $\hat{\mathbf{y}}$ be the local minima. Let us define $\mathscr{Y}_\alpha = \{y_i : y_i^* = \alpha\}$ as the set of nodes having global minimum label as α. Therefore, we can define a labeling $\mathbf{y}^\alpha$, one α-expansion away from the local minima $\hat{\mathbf{y}}$ as follows:

$$\mathbf{y}^\alpha = \begin{cases} y_i^* \ \textit{if}\ y_i \in \mathscr{Y}_\alpha \\ \hat{y}_i \ \textit{otherwise.} \end{cases} \tag{3.7}$$

Since $\hat{\mathbf{y}}$ is the local minima, therefore, $E(\mathbf{y}^*) \leq E(\hat{\mathbf{y}}) \leq E(\mathbf{y}^\alpha)$. Let us define some set of cliques and nodes based on their relation with $\mathscr{Y}_\alpha$. Let $\mathscr{I}_\alpha = \{c : \mathbf{y}_c \subseteq \mathscr{Y}_\alpha\}$ be the set of *interior cliques* such that all the nodes in each clique are assigned the global optimal label of α. The set of *boundary cliques* $\mathscr{B}_\alpha = \{c : \mathbf{y}_c \cap \mathscr{Y}_\alpha \neq \emptyset, \mathbf{y}_c \not\subseteq \mathscr{Y}_\alpha\}$ such that in each clique there exist atleast one node having the optimal assignment of α and at least one node having optimal assignment from the label set $\mathscr{L} \setminus \{\alpha\}$. The set of *outside cliques* $\mathscr{O}_\alpha = \{c : \mathbf{y}_c \cap \mathscr{Y}_\alpha = \emptyset\}$. Similarly, we define $V_\alpha^I = \{i : y_i \in \mathscr{Y}_\alpha\}$ and $V_\alpha^O = \{i : y_i \notin \mathscr{Y}_\alpha\}$ as the set of interior and outside nodes. Since, $\{\mathscr{I}_\alpha, \mathscr{B}_\alpha, \mathscr{O}_\alpha, V^I_\alpha, V^O_\alpha\}$ covers all the cliques and the nodes in the given MRF, therefore, the energy can be decomposed over them as follows:

$$E(\mathbf{y}^\alpha) = \sum_{i \in V^I_\alpha} \theta_i(y_i) + \sum_{i \in V^O_\alpha} \theta_i(y_i) + \sum_{c \in \mathscr{I}_\alpha} \theta_c(\mathbf{y}_c) + \sum_{c \in \mathscr{B}_\alpha} \theta_c(\mathbf{y}_c) + \sum_{c \in \mathscr{O}_\alpha} \theta_c(\mathbf{y}_c). \tag{3.8}$$

Now, using the definition of $\mathbf{y}^\alpha$ from (3.7), we can say that:

$$\sum_{i \in V^I_\alpha} \theta_i(y^\alpha_i) = \sum_{i \in V^I_\alpha} \theta_i(y^*_i). \tag{3.9}$$

$$\sum_{i \in V^O_\alpha} \theta_i(y^\alpha_i) = \sum_{i \in V^O_\alpha} \theta_i(\hat{y}_i). \tag{3.10}$$

$$\sum_{c \in \mathscr{I}_\alpha} \theta_c(\mathbf{y}^\alpha_c) = \sum_{c \in \mathscr{I}_\alpha} \theta_c(\mathbf{y}^*_c). \tag{3.11}$$

$$\sum_{c \in \mathscr{O}_\alpha} \theta_c(\mathbf{y}^\alpha_c) = \sum_{c \in \mathscr{O}_\alpha} \theta_c(\hat{\mathbf{y}}_c). \tag{3.12}$$

$$\sum_{c \in \mathscr{B}_\alpha} \theta_c(\mathbf{y}^\alpha_c) \leq \lambda \sum_{c \in \mathscr{B}_\alpha} \theta_c(\mathbf{y}^*_c). \tag{3.13}$$

The first four equality are obvious from the definition (3.7), the proof of the last inequality is as follows. Let $\mathbf{y}^\alpha_{\mathscr{B}} = \cup_{c \in \mathscr{B}_\alpha} \mathbf{y}_c$ be the joint variable vector representing the set of nodes formed by taking the union of all the cliques in the set of boundary cliques. Thus,

$$\begin{aligned}
\sum_{c \in \mathscr{B}_\alpha} \theta_c(\mathbf{y}^\alpha_c) &\leq \max_{\mathbf{y}^\alpha_{\mathscr{B}}} \Big(\sum_{c \in \mathscr{B}_\alpha} \theta_c(\mathbf{y}_c) \Big) \\
&\leq \sum_{c \in \mathscr{B}_\alpha} \max_{\mathbf{y}_c} \theta_c(\mathbf{y}_c) && (3.14) \\
&= \sum_{c \in \mathscr{B}_\alpha} \left(\frac{\max_{\mathbf{y}_c} \theta_c(\mathbf{y}_c)}{\min_{\mathbf{y}_c : \theta_c(\mathbf{y}_c) \neq 0} \theta_c(\mathbf{y}_c)} \right) \min_{\mathbf{y}_c : \theta_c(\mathbf{y}_c) \neq 0} \theta_c(\mathbf{y}_c) && (3.15) \\
&\leq \lambda \sum_{c \in \mathscr{B}_\alpha} \min_{\mathbf{y}_c : \theta_c(\mathbf{y}_c) \neq 0} \theta_c(\mathbf{y}_c) && (3.16) \\
&\leq \lambda \sum_{c \in \mathscr{B}_\alpha} \theta_c(\mathbf{y}^*_c). && (3.17)
\end{aligned}$$

The first inequality is because of the fact that maximization is over the joint variable vector. The second inequality is because of the fact that $\max_x(g(x) + f(x)) \leq (\max_x g(x) + \max_x f(x))$. The last inequality is because of the fact that $\theta_c(\mathbf{y}_c) \neq 0$ for boundary cliques because the labeling for each clique contains more than one label. Since $E(\hat{\mathbf{y}}) \leq E(\mathbf{y}^\alpha)$, therefore, expanding the corresponding energy terms similar to the equation (3.8) and using the inequalities (3.9)-(3.13) we get

$$\sum_{i \in V_\alpha^I} \theta_i(\hat{y}_i) + \sum_{c \in \mathscr{I}_\alpha} \theta_c(\hat{\mathbf{y}}_c) + \sum_{c \in \mathscr{B}_\alpha} \theta_c(\hat{\mathbf{y}}_c) \leq \sum_{i \in V_\alpha^I} \theta_i(y_i^*) + \sum_{c \in \mathscr{I}_\alpha} \theta_c(\mathbf{y}_c^*) + \lambda \sum_{c \in \mathscr{B}_\alpha} \theta_c(\mathbf{y}_c^*) \tag{3.18}$$

In order to get the bound over total energy we sum the above equation over all $\alpha \in \mathscr{L}$.

$$\sum_\alpha \Big(\sum_{i \in V_\alpha^I} \theta_i(\hat{y}_i) + \sum_{c \in \mathscr{I}_\alpha} \theta_c(\hat{\mathbf{y}}_c) + \sum_{c \in \mathscr{B}_\alpha} \theta_c(\hat{\mathbf{y}}_c) \Big) \leq \sum_\alpha \Big(\sum_{i \in V_\alpha^I} \theta_i(y_i^*) + \sum_{c \in \mathscr{I}_\alpha} \theta_c(\mathbf{y}_c^*) + \lambda \sum_{c \in \mathscr{B}_\alpha} \theta_c(\mathbf{y}_c^*) \Big) \tag{3.19}$$

Let us have a look into the left and the right sides of the inequality (3.19) separately. The l.h.s can be written as:

$$\begin{aligned} \sum_\alpha \Big(\sum_{i \in V_\alpha^I} \theta_i(\hat{y}_i) + \sum_{c \in \mathscr{I}_\alpha} \theta_c(\hat{\mathbf{y}}_c) + \sum_{c \in \mathscr{B}_\alpha} \theta_c(\hat{\mathbf{y}}_c) \Big) &= \sum_{i \in V} \theta_i(\hat{y}_i) + \sum_\alpha \sum_{c \in \mathscr{I}_\alpha} \theta_c(\hat{\mathbf{y}}_c) + \sum_\alpha \sum_{c \in \mathscr{B}_\alpha} \theta_c(\hat{\mathbf{y}}_c) \\ &\geq \sum_{i \in V} \theta_i(\hat{y}_i) + \sum_{c \in \cup_\alpha \{\mathscr{I}_\alpha \cup \mathscr{B}_\alpha\}} \theta_c(\hat{\mathbf{y}}_c) \qquad (3.20) \\ &= \sum_{i \in V} \theta_i(\hat{y}_i) + \sum_{c \in \mathscr{C}} \theta_c(\hat{\mathbf{y}}_c). \qquad (3.21) \end{aligned}$$

The inequality in the equation (3.20) is due to the fact that $\cap_{\alpha \in \mathscr{L}} \mathscr{I}^\alpha = \emptyset$, $\cap_{\alpha \in \mathscr{L}} \mathscr{B}^\alpha$ is not necessarily an empty set, and $\theta_c(\mathbf{y}_c) \geq 0$. The equality in the equation (3.21) is due to the fact that $\cup_\alpha \{\mathscr{I}_\alpha \cup \mathscr{B}_\alpha\}$ is the union of interior cliques and the boundary cliques over all the $\alpha \in \mathscr{L}$, therefore, it is the set of all the cliques in the entire graph. Now let us have a look into the last term of the r.h.s of the inequality (3.19).

$$\begin{aligned} \sum_\alpha \sum_{c \in \mathscr{B}^\alpha} \theta_c(\mathbf{y}_c^*) &\leq \sum_{c \in \cup_\alpha \mathscr{B}^\alpha} \min(|\mathbf{y}_c|, |\mathscr{L}|) \theta_c(\mathbf{y}_c^*) \qquad (3.22) \\ &\leq \min(\mathscr{M}, |\mathscr{L}|) \sum_{c \in \cup_\alpha \mathscr{B}^\alpha} \theta_c(\mathbf{y}_c^*) \qquad (3.23) \end{aligned}$$

where, $\mathscr{M} = \max_c |\mathbf{y}_c|$. Notice that $\cup_\alpha \mathscr{B}^\alpha$ is the union of all the boundary cliques over all the labels $\alpha \in \mathscr{L}$. Since a given MRF can be divided into maximum of $\mathscr{L}$ unique partitions (uniqueness is in terms of the labels assigned to the nodes), therefore, if $|\mathbf{y}_c| \geq |\mathscr{L}|$, the clique $\mathbf{y}_c$ can not be shared in more than $|\mathscr{L}|$ unique partitions. Similarly, if $|\mathbf{y}_c| < |\mathscr{L}|$, the clique $\mathbf{y}_c$ can not be part of more than $|\mathbf{y}_c|$ unique partitions at a time. Hence, $\min(|\mathbf{y}_c|, |\mathscr{L}|)$ is the maximum possible times a clique $\mathbf{y}_c$ can be counted while summing over all the $\alpha \in \mathscr{L}$. This leads to the inequality (3.22). The inequality (3.23) is obvious as $\mathscr{M} = \max_{c \in \mathscr{C}} |\mathbf{y}_c|$ is the size of the largest maximal clique. For example, in case of multilabel pairwise MRF,

$\mathscr{M} = 2$ and $|\mathscr{L}| \geq 2$, therefore, $\min(\mathscr{M}, |\mathscr{L}|) = 2$, which is the same factor as shown in [128]. Using (3.23) into the r.h.s of the inequality (3.19):

$$
\begin{aligned}
\sum_{\alpha}\Big(\sum_{i \in V^I_\alpha} \theta_i(y_i^*) + \sum_{c \in \mathscr{I}_\alpha} \theta_c(\mathbf{y}_c^*) + \lambda \sum_{c \in \mathscr{B}_\alpha} \theta_c(\mathbf{y}_c^*)\Big) \\
&\leq \sum_{i \in V} \theta_i(y_i^*) + \sum_{c \in \cup_\alpha \mathscr{I}^\alpha} \theta_c(\mathbf{y}_c^*) + \lambda \min(\mathscr{M}, |\mathscr{L}|) \sum_{c \in \cup_\alpha \mathscr{B}^\alpha} \theta_c(\mathbf{y}_c^*) && (3.24) \\
&\leq \sum_{i \in V} \theta_i(y_i^*) + \lambda \min(\mathscr{M}, |\mathscr{L}|)\Big(\sum_{c \in \cup_\alpha \mathscr{I}^\alpha} \theta_c(\mathbf{y}_c^*) + \sum_{c \in \cup_\alpha \mathscr{B}^\alpha} \theta_c(\mathbf{y}_c^*)\Big) && (3.25) \\
&= \sum_{i \in V} \theta_i(y_i^*) + \lambda \min(\mathscr{M}, |\mathscr{L}|) \sum_{c \in \mathscr{C}} \theta_c(\mathbf{y}_c^*) && (3.26)
\end{aligned}
$$

The inequality (3.24) is due to the inequality (3.23). The inequality (3.25) is due to the fact that $\lambda \min(\mathscr{M}, |\mathscr{L}|) \geq 1$ and $\theta_c(\mathbf{y}_c) \geq 0$. Using the inequalities (3.21) and (3.26) in (3.19) we obtain the following multiplicative bound:

$$
\sum_{i \in V} \theta_i(\hat{y}_i) + \sum_{c \in \mathscr{C}} \theta_c(\hat{\mathbf{y}}_c) \leq \sum_{i \in V} \theta_i(y_i^*) + \lambda \min(\mathscr{M}, |\mathscr{L}|) \sum_{c \in \mathscr{C}} \theta_c(\mathbf{y}_c^*). \tag{3.27}
$$

$\square$

Note that the bound given in the Theorem 1 is practically useful if and only if the algorithm allows us to compute the optimal expansion move in polynomial time. However, *there is no known algorithm* that guarantees optimal expansion move in polynomial time for any arbitrary clique potential. In a restricted setting with arbitrary unary and pairwise clique potentials of the form of the Potts model ($\lambda = 1$), Theorem 1 provides the multiplicative bound of 2, which is a very well known result [128]. Under this setting, the expansion move can be computed in polynomial time using well known Graph-Cuts algorithm [74]. Similarly, in case of metric labeling, the multiplicative bound is 2λ. In case of the α-expansion algorithm for the P^n Potts model, the optimal expansion move can be computed in polynomial time. Therefore, it provides a multiplicative bound of $\lambda \min(\mathscr{M}, |\mathscr{L}|)$.

3.4 Parsimonious Labeling

The parsimonious labeling problem is defined using an energy function that consists of unary potentials and clique potentials defined over cliques of arbitrary sizes. While the parsimonious labeling problem places no restrictions on the unary potentials, the clique potentials are specified using a *diversity* function [18]. Before describing the parsimonious labeling problem in detail, we briefly define the diversity function for the sake of completion.

Definition 1. *A diversity is a pair* $(\mathscr{L}, \delta)$*, where* $\mathscr{L}$ *is the label set and* δ *is a non-negative function defined on subsets of* $\mathscr{L}$*,* $\delta : \Gamma \rightarrow \mathbb{R}, \forall \Gamma \subseteq \mathscr{L}$*, satisfying*

- *Non Negativity:* $\delta(\Gamma) \geq 0$*, and* $\delta(\Gamma) = 0$*, if and only if,* $|\Gamma| \leq 1$.
- *Triangular Inequality: if* $\Gamma_2 \neq \emptyset$*,* $\delta(\Gamma_1 \cup \Gamma_2) + \delta(\Gamma_2 \cup \Gamma_3) \geq \delta(\Gamma_1 \cup \Gamma_3), \forall \Gamma_1, \Gamma_2, \Gamma_3 \subseteq \mathscr{L}$.
- *Monotonicity:* $\Gamma_1 \subseteq \Gamma_2$ *implies* $\delta(\Gamma_1) \leq \delta(\Gamma_2)$.

Using a diversity function, we can define a clique potential as follows. We denote by $\Gamma(\mathbf{y}_c)$ the set of unique labels in the labeling of the clique c. Then, $\theta_c(\mathbf{y}_c) = w_c \delta(\Gamma(\mathbf{y}_c))$, where δ is a diversity function and w_c is the non-negative weight corresponding to the clique c. Formally, the parsimonious labeling problem amounts to minimizing the following energy function:

$$E(\mathbf{y}) = \sum_{i \in V} \theta_i(y_i) + \sum_{c \in \mathscr{C}} w_c \delta(\Gamma(\mathbf{y}_c)). \tag{3.28}$$

Therefore, given a clique $\mathbf{y}_c$ and the set of unique labels $\Gamma(\mathbf{y}_c)$ assigned to the random variables in the clique, the clique potential function for the parsimonious labeling problem is defined using $\delta(\Gamma(\mathbf{y}_c))$, where $\delta : \Gamma(\mathbf{y}_c) \rightarrow \mathbb{R}$ is a diversity function.

Intuitively, because of the monotonicity property diversities enforces parsimony by choosing a solution with fewer unique labels from a set of equally likely solutions. This is an essential property in many vision problems, for example, in the case of image segmentation, we would like to see label consistency within superpixels in order to preserve discontinuity. Unlike the P^n Potts model the diversity does not enforce the label consistency very rigidly. It gives monotonic rise to the cost based on the number of labels assigned to the given clique.

Metric labeling as Parsimonious labeling. An important special case of the parsimonious labeling problem is the *metric labeling problem*, as discussed in section 2.3.2. Metric labeling has been extensively studied in computer vision [15, 84] and theoretical computer science [20, 65]. In metric labeling, the maximal cliques are of size two (pairwise) and the clique potential function is a metric distance function defined over the labels. Recall that a distance function $d : \mathscr{L} \times \mathscr{L} \rightarrow \mathbb{R}$ is a metric if and only if: (i) $d(.,.) \geq 0$; (ii) $d(i,j) + d(j,k) \geq d(i,k), \forall i,j,k$; and (iii) $d(i,j) = 0$ if and only if $i = j$. Notice that there is a direct link between the metric distance function and the diversities. Specifically, metric distance functions are diversities defined on subsets of size 2. In other words, diversities are the generalization of the metric distance function and boil down to a metric distance function if the input set is restricted to the subsets with cardinality of at most two. Another way of understanding the connection between metrics and diversities is that *every diversity induces a metric*. In other words, consider $d(l_i, l_i) = \delta(l_i) = 0$ and $d(l_i, l_j) = \delta(\{l_i, l_j\})$. Using the properties of diversities, it can be shown that $d(\cdot, \cdot)$ is a metric distance function. Hence, in case of energy function defined over pairwise cliques, the parsimonious labeling problem reduces to the metric labeling problem.

In the remaining part of this section we talk about a specific type of diversity called *the diameter diversity*. We show its relation with the well known P^n Potts model. Furthermore, we propose a *hierarchical P^n Potts model* based on the diameter diversity defined over a hierarchical clustering (defined shortly). However, note that our approach is applicable to any general parsimonious labeling problem.

Diameter Diversity. In this work, we are primarily interested in the diameter diversity [18]. Let $(\mathscr{L}, \delta)$ be a diversity and $(\mathscr{L}, d)$ be the induced metric of $(\mathscr{L}, \delta)$, where $d : \mathscr{L} \times \mathscr{L} \to \mathbb{R}$ and $d(l_i, l_j) = \delta(\{l_i, l_j\}), \forall l_i, l_j \in \mathscr{L}$, then for all $\Gamma \subseteq \mathscr{L}$, the diameter diversity is defined as:

$$\delta^{dia}(\Gamma) = \max_{l_i, l_j \in \Gamma} d(l_i, l_j). \tag{3.29}$$

Clearly, given the induced metric function defined over a set of labels, diameter diversity over any subset of labels gives the measure of the dissimilarity (or diversity) of the labels. More the dissimilarity, based on the induced metric function, higher is the diameter diversity. Therefore, using diameter diversity as clique potentials enforces the similar labels to be together. Thus, a special case of parsimonious labeling in which the clique potentials are of the form of diameter diversity can be defined as below:

$$E(\mathbf{y}) = \sum_{i \in V} \theta_i(y_i) + \sum_{c \in \mathscr{C}} w_c \delta^{dia}(\Gamma(\mathbf{y}_c)). \tag{3.30}$$

Notice that the diameter diversity defined over uniform metric is nothing but the P^n Potts model where $\gamma_i = 0$. In what follows we define a generalization of the P^n Potts model, the hierarchical P^n Potts model, which will play a key role in the rest of the paper.

The Hierarchical P^n Potts Model. The hierarchical P^n Potts model is a diameter diversity defined over a special type of metric known as the r-HST metric. A rooted tree, as shown in Figure 3.1, is said to be an r-HST, or *r-hierarchically well separated* [3] if it satisfy the following properties: (i) all the leaf nodes are the labels; (ii) all edge weights are positive; (iii) the edge lengths from any node to all of its children are the same; and (iv) on any root to

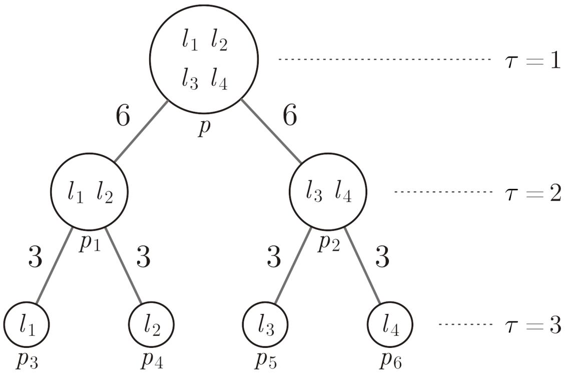


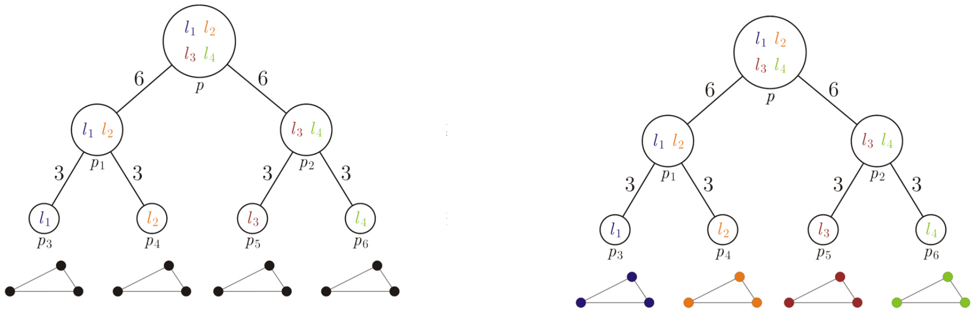
Figure 3.1: *An example of r-*HST *for $r = 2$. The cluster associated with root p contains all the labels. As we go down, the cluster splits into subclusters and finally we get the singletons, the leaf nodes (labels). The root is at depth of 1 ($\tau = 1$) and leaf nodes at $\tau = 3$. The metric defined over the r-*HST *is denoted as $d^t(.,.)$, the shortest path between the inputs. For example, $d^t(l_1, l_3) = 18$ and $d^t(l_1, l_2) = 6$. The diameter diversity for the subset of labels at cluster p is $\max_{\{l_i,l_j\} \in \{l_1,l_2,l_3,l_4\}} d^t(l_i, l_j) = 18$. Similarly, the diameter diversity at p_2 and p_3 are 6 and 0, respectively.*

leaf path the edge weight decrease by a factor of at least $r > 1$. We can think of a r-HST as a hierarchical clustering of the given label set $\mathscr{L}$. The root node is the cluster at the top level of the hierarchy and contains all the labels. As we go down in the hierarchy, the clusters break down into smaller clusters until we get as many leaf nodes as the number of labels in the given label set. The metric distance function defined on this tree $d^t(.,.)$ is known as the r-HST metric. In other words, the distance $d^t(\cdot,\cdot)$ between any two nodes in the given r-HST is the length of the unique path between these nodes in the tree. *The diameter diversity defined over $d^t(.,.)$ is called the hierarchical P^n Potts model.* Figure 3.1 shows an example of diameter diversity defined over an r-HST.

We will shortly talk about the algorithm to generate an r-HST for a given metric $d(.,.)$. Notice that, a node in r-HST can have any number of child between 0 to $|\mathscr{L}|$, and the leaf nodes can be at any depth greater than one.

3.5 The Hierarchical Move Making Algorithm

In the first part of this section we propose a move making algorithm for the hierarchical P^n Potts model (defined in the previous section). In the second part, we show how our hierarchical move making algorithm can be used to address the general parsimonious labeling problem with optimality guarantees (strong multiplicative bound).



(a) Subproblems at each leaf node.

(b) Trivial labelings at each subproblem.

Figure 3.2: An example of solving the labeling problems at leaf nodes. Consider a clique with three nodes which we intend to label. Figure 3.2a shows the subprobelms at each leaf node. Notice that each leaf node contains only one label, therefore, we have only one choice for the labeling. Figure 3.2b shows the corresponding trivial labelings (notice the colors).

3.5.1 The Hierarchical Move Making Algorithm for the Hierarchical P^n Potts Model

In the hierarchical P^n Potts model the clique potentials are of the form of the diameter diversity defined over a given r-HST metric function. The move making algorithm proposed in this section to minimize such an energy function is a divide-and-conquer based approach, inspired by the work of [84]. Instead of solving the actual problem, we divide the problem into smaller subproblems where each subproblem is a P^n Potts model, which can be solved efficiently using $\alpha-$expansion [66]. More precisely, given an r-HST, each node of the r-HST corresponds to a subproblem. We start with the bottom node of the r-HST, which is a leaf node, and go up in the hierarchy solving each subproblem associated with the nodes encountered.

In more detail, consider a node p of the given r-HST. Recall that any node p in the r-HST is a cluster of labels denoted as $\mathscr{L}^p \subseteq \mathscr{L}$ (Figure 3.1). In other words, the leaf nodes of the subtree rooted at p belongs to the subset $\mathscr{L}^p$. For example, in case of node p_1 in Figure 3.1, $\mathscr{L}^{p_1} = \{l_1, l_2\}$. Thus, the subproblem defined at any node p is to find the labeling $\mathbf{y}^p$ where the label set is restricted to $\mathscr{L}^p$, as defined below.

$$\mathbf{y}^p = \underset{\mathbf{y} \in (\mathscr{L}^p)^N}{\operatorname{argmin}} \left(\sum_{i \in V} \theta_i(y_i) + \sum_{c \in \mathscr{C}} w_c \delta^{dia}(\Gamma(\mathbf{y}_c)) \right). \tag{3.31}$$

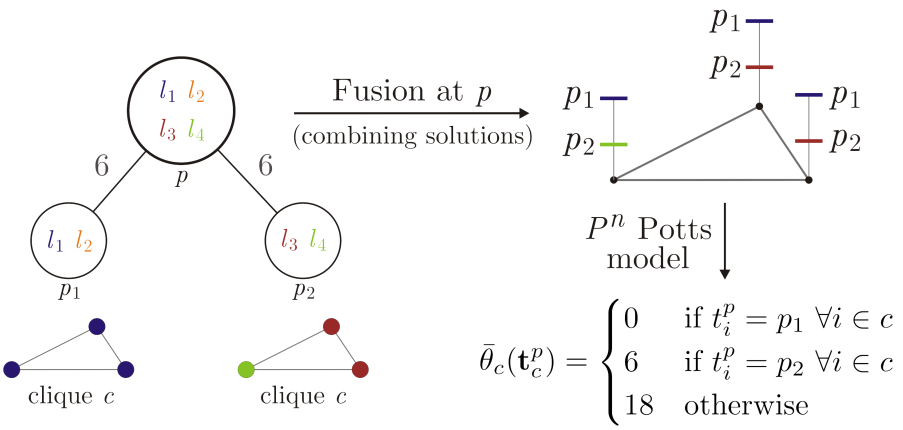

l_1 l_2
l_3 l_4
p
6
6
l_1 l_2
p_1
l_3 l_4
p_2
clique c
clique c
Fusion at p
(combining solutions)
p_1
p_2
p_1
p_2
p_1
p_2
P^n Potts model
$\bar{\theta}_c(\mathbf{t}_c^p) = \begin{cases} 0 & \text{if } t_i^p = p_1 \ \forall i \in c \\ 6 & \text{if } t_i^p = p_2 \ \forall i \in c \\ 18 & \text{otherwise} \end{cases}$

Figure 3.3: *An example of solving the labeling problem at non-leaf node* (p) *by combining the solutions of its child nodes* $\{p_1, p_2\}$*, given clique c and the labelings that it has obtained at the child nodes. The labeling fusion instance shown in this figure is the top two levels of the r-*HST *in the Figure 3.1. Note that the labelings shown at nodes* p_1 *and* p_2 *are an assumption. It could get different labelings as well but the algorithm for the fusion remains the same. The diameter diversity of the labeling of clique c at node* p_1 *is 0 as it contains only one unique label* l_1*. The diameter diversity of the labeling at* p_2 *is* $d^t(l_3, l_4) = 6$ *and at p is* $\max_{\{l_i,l_j\} \in \{l_1,l_3,l_4\}} d^t(l_i, l_j) = 18$. *Please notice the colors.*

If p is the root node, then the above problem (equation (3.31)) is as difficult as the original labeling problem defined in equation (3.30), since $\mathscr{L}^p = \mathscr{L}$. However, if p is the leaf node then the solution of the problem associated with p is trivial, $y_i^p = p$ for all $i \in V$, that is, assign the label p to all the random variables. An example of the labeling at leaf node is shown in the Figure 3.2. This insight leads to the design of our approximation algorithm, where we start by solving the simple problems corresponding to the leaf nodes, and use the labelings obtained to address the more difficult problem further up the hierarchy. In what follows, we describe how the labeling of the problem associated with the node p, when p is a non-leaf node, is obtained using the labelings of its child nodes.

Solving the Parent Labeling Problem. Before delving into the details, let us define some notations for the purpose of clarity. Let T be the depth (or the number of levels) of the given r-HST and $\mathscr{N}(\tau)$ be the set of nodes at level τ. The root node is at the top level $(\tau = 1)$. Let $\eta(p)$ denotes the set of child nodes associated with a non-leaf node p and $\eta(p,k)$ denotes its k^{th} child node. Recall that our approach is bottom up. Therefore, for each child node of p we already have an associated labeling. We denote the labeling associated with the k^{th} child of the node p as $\mathbf{y}^{\eta(p,k)}$. Thus, $y_i^{\eta(p,k)}$ denotes the label assigned to the i^{th} random variable by the labeling of the k^{th} child of the node p. We also define an N dimensional vector $\mathbf{t}^p \in \{1, \cdots, |\eta(p)|\}^N$, where $|\eta(p)|$ is the number of child nodes of node p. More

Algorithm 7 The Move Making Algorithm for the Hierarchical P^n Potts Model.

```
input r-HST Metric; $w_c, \forall c \in \mathscr{C}$; and $\theta_i(y_i), \forall i \in V$
1:  $\tau = T$, the leaf nodes
2:  repeat
3:      for each $p \in \mathscr{N}(\tau)$ do
4:          if $|\eta(p)| = 0$, leaf node then
5:              $y_i^p = p, \forall i \in V$
6:          else
7:              Fusion Move
                    $\hat{\mathbf{t}}^p = \underset{\mathbf{t}^p \in \{1,\cdots,|\eta(p)|\}^N}{\operatorname{argmin}} E(\mathbf{t}^p)$        (3.34)
8:              $y_i^p = y_i^{\eta(p,\hat{t}_i^p)}$.
9:          end if
10:     end for
11:     $\tau \leftarrow \tau - 1$
12: until $\tau > 0$.
```

precisely, for a given $\mathbf{t}^p$, $t_i^p = k$ denotes that the label for the i^{th} random variable comes from the k^{th} child of the node p. Therefore, the labeling problem at node p reduces to finding the optimal $\mathbf{t}^p$. Thus, the labeling problem at node p amounts to finding the best child index $k \in \{1, \cdots, |\eta(p)|\}$ for each random variable $i \in V$ so that the label assigned to the random variable comes from the labeling of the k^{th} child (step 7, Algorithm 7).

Using the above notations, associated with a $\mathbf{t}^p$ we define a new energy function as:

$$E(\mathbf{t}^p) = \sum_{i \in V} \bar{\theta}_i(t_i^p) + \sum_{c \in \mathscr{C}} w_c \bar{\theta}_c(\mathbf{t}_c^p). \tag{3.32}$$

where

$$\bar{\theta}_i(t_i^p) = \theta_i(y_i^{\eta(p,k)}) \quad if \quad t_i^p = k. \tag{3.33}$$

In other words, the unary potential for $t_i^p = k$ is the unary potential associated to the i^{th} random variable corresponding to the label $y_i^{\eta(p,k)}$.

The new clique potential $\bar{\theta}_c(\mathbf{t}_c^p)$ is as defined below:

$$\bar{\theta}_c(\mathbf{t}_c^p) = \begin{cases} \gamma_k^p, & \text{if } t_i^p = k, \forall i \in c, \\ \gamma_{max}^p, & \text{otherwise,} \end{cases} \tag{3.35}$$

where $\gamma_k^p = \delta_{dia}(\Gamma(\mathbf{y}_c^{\eta(p,k)}))$ is the diameter diversity of the set of *unique labels* associated with $\mathbf{y}_c^{\eta(p,k)}$ and $\gamma_{max}^p = \delta_{dia}(\bar{\mathscr{L}}^p)$. The set $\bar{\mathscr{L}}^p = \cup_{k\in\eta(p)}\Gamma(\mathbf{y}_c^{\eta(p,k)})$ is the union of the unique labels associated with the child nodes of p. Recall that, because of the construction of the r-HST, $\mathscr{L}^q \subset \bar{\mathscr{L}}^p \subseteq \mathscr{L}^p$ for all $q \in \eta(p)$. Hence, the monotonicity property of the diameter diversity (Definition 1) ensures that $\gamma_{max}^p > \gamma_k^p, \forall k \in \eta(p)$. This is the sufficient criterion to prove that the potential function defined by equation (3.35) is a P^n Potts model. Therefore, the $\alpha-$expansion algorithm can be used to obtain the locally optimal $\mathbf{t}^p$ for the energy function (3.32). Given the locally optimal $\hat{\mathbf{t}}^p$, the labeling $\mathbf{y}^p$ at node p can be trivially obtained as follows: $y_i^p = y_i^{\eta(p,\hat{t}_i^p)}$. In other words, the final label of the i^{th} random variable is the one assigned to it by the $(\hat{t}_i^p)^{th}$ child of node p.

Figure 3.3 shows an instance of the above mentioned algorithm to combine the labelings of the child nodes to obtain the labeling of the parent node. The complete hierarchical move making algorithm for the hierarchical P^n Potts model is shown in the Algorithm 7.

Multiplicative Bound. Theorem 2 gives the multiplicative bound for the move making algorithm for the hierarchical P^n Potts model.

Theorem 2. *The move making algorithm for the hierarchical P^n Potts model, Algorithm 7, gives the multiplicative bound of $\left(\frac{r}{r-1}\right)\min(\mathscr{M}, |\mathscr{L}|)$ with respect to the global minima. Here, $\mathscr{M}$ is the size of the largest maximal-clique, $|\mathscr{L}|$ is the number of labels, and $r > 1$ is the parameter of the r-*HST.

Proof. Briefly, the factor of $\min(\mathscr{M}, |\mathscr{L}|)$ comes from the fact that each subproblem amounts to solving $\alpha-$expansion for the P^n Potts models. The factor of $\left(\frac{r}{r-1}\right)$ comes from the fact that the edge lengths of the r-HST forms a geometric progression (refer to Figure 3.1), therefore, the distance between any two leaf node is upperbounded by $e^{max}\left(\frac{r}{r-1}\right)$, where e^{max} is the length of the longest edge. The detailed proof is given below.

Let $\mathbf{y}^*$ be the optimal labeling of the given hierarchical P^n Potts model based labeling problem. Recall that any node p in the underlying r-HST represents a cluster (subset) of labels. For each node p in the r-HST we define following sets using $\mathbf{y}^*$:

$$\begin{aligned}
\mathscr{L}^p &= \{l_i | l_i \in \mathscr{L}, i \in p\}, \\
V_I^p &= \{y_i : y_i^* \in \mathscr{L}^p\}, \\
V_O^p &= \{y_i : y_i^* \notin \mathscr{L}^p\}, \\
\mathscr{I}^p &= \{c : \mathbf{y}_c \subseteq V^p\}, \\
\mathscr{B}^p &= \{c : \mathbf{y}_c \cap V^p \neq \emptyset, \mathbf{y}_c \nsubseteq V^p\}, \\
\mathscr{O}^p &= \{c : \mathbf{y}_c \cap V^p = \emptyset\}.
\end{aligned} \tag{3.36}$$

In other words, $\mathscr{L}^p$ is the set of labels in the cluster at p^{th} node, V_I^p is the set of nodes whose optimal label lies in the subtree rooted at p, V_O^p is the set of nodes whose optimal label does not lie in the subtree rooted at p, $\mathscr{I}^p$ is the set of interior cliques such that the optimal labeling lies in the subtree rooted at p, $\mathscr{B}^p$ is the set of cliques (boundary cliques) such that $\forall \mathbf{y}_c \in \mathscr{B}^p, \exists \{y_i, y_j\} \in \mathbf{y}_c : y_i^* \in \mathscr{L}^p, y_j^* \notin \mathscr{L}^p$, and $\mathscr{O}^p$ is the set of outside cliques such that the optimal assignment for all the nodes belongs to the set $\mathscr{L} \setminus \mathscr{L}^p$. Let's define $\mathbf{y}^p$ as the labeling at node p. We prove the following lemma relating $\mathbf{y}^*$ and $\mathbf{y}^p$.

Lemma 1. *Let* $\mathbf{y}^p$ *be the labeling at node p,* $\mathbf{y}^*$ *be the optimal labeling of the given hierarchical* P^n *Potts model, and* $\delta^{dia}(\Gamma(\mathbf{y}_c^p))$ *be the diameter diversity based clique potential defined as* $\max_{l_i,l_j \in \mathscr{L}^p} d^t(l_i,l_j), \forall p$*, where* $d^t(.,.)$ *is the tree metric defined over the given* r-HST. *Then the following bound holds true at any node p of the* r-HST.

$$\sum_{i \in V_I^p} \theta_i(y_i^p) + \sum_{c \in \mathscr{I}^p} \delta^{dia}(\Gamma(\mathbf{y}_c^p)) \leq \sum_{i \in V_I^p} \theta_i(y_i^*) + \left(\frac{r}{r-1}\right) \min(\mathscr{M}, |\mathscr{L}|) \sum_{c \in \mathscr{I}^p} \delta^{dia}(\Gamma(\mathbf{y}_c^*)). \tag{3.37}$$

Proof. We prove the above lemma by mathematical induction. Clearly, when p is a leaf node, $y_i = p, \forall i \in V$. For a non-leaf node p, we assume that the lemma holds true for the labeling $\mathbf{y}^q$ and all its children q. Given the labeling $\mathbf{y}^p$ and $\mathbf{y}^q$, we define a new labeling $\mathbf{y}^{pq}$ such that

$$\mathbf{y}^{pq} = \begin{cases} y_i^q \ \textit{if}\ y_i^* \in \mathscr{L}^q, \\ y_i^p \ \textit{otherwise.} \end{cases} \tag{3.38}$$

Note that $\mathbf{y}^{pq}$ lies within one α-expansion iteration away from $\mathbf{y}^p$. Since $\mathbf{y}^p$ is the local minima, therefore, $E(\mathbf{y}^p) \leq E(\mathbf{y}^{pq})$. Using the energy decomposition similar to the one in equation (3.8) we can write:

$$\sum_{i \in V_I^q} \theta_i(y_i^p) + \sum_{i \in V_O^q} \theta_i(y_i^p) + \sum_{c \in \mathscr{I}^q} \delta^{dia}(\Gamma(\mathbf{y}_c^p)) + \sum_{c \in \mathscr{B}^q} \delta^{dia}(\Gamma(\mathbf{y}_c^p)) + \sum_{c \in \mathscr{O}^q} \delta^{dia}(\Gamma(\mathbf{y}_c^p)) \leq$$

$$\sum_{i \in V_I^q} \theta_i(y_i^{pq}) + \sum_{i \in V_O^q} \theta_i(y_i^{pq}) + \sum_{c \in \mathscr{I}^q} \delta^{dia}(\Gamma(\mathbf{y}_c^{pq})) + \sum_{c \in \mathscr{B}^q} \delta^{dia}(\Gamma(\mathbf{y}_c^{pq})) + \sum_{c \in \mathscr{O}^q} \delta^{dia}(\Gamma(\mathbf{y}_c^{pq}))$$

Using the definition (3.38) and the fact that $\theta_i(y_i^{pq}) = \theta_i(y_i^q), \forall i \in V_I^q$ in the above inequality we get:

$$\begin{align}
&\sum_{i\in V_I^q}\theta_i(y_i^p) + \sum_{c\in\mathscr{I}^q}\delta^{dia}(\Gamma(\mathbf{y}_c^p)) + \sum_{c\in\mathscr{B}^q}\delta^{dia}(\Gamma(\mathbf{y}_c^p)) \nonumber \\
&\qquad \le \sum_{i\in V_I^q}\theta_i(y_i^q) + \sum_{c\in\mathscr{I}^q}\delta^{dia}(\Gamma(\mathbf{y}_c^q)) + \sum_{c\in\mathscr{B}^q}\delta^{dia}(\Gamma(\mathbf{y}_c^{pq})) \tag{3.39} \\
&\qquad \le \sum_{i\in V_I^q}\theta_i(y_i^*) + \left(\frac{r}{r-1}\right)\min(\mathscr{M},|\mathscr{L}|)\sum_{c\in\mathscr{I}^q}\delta^{dia}(\Gamma(\mathbf{y}_c^*)) + \sum_{c\in\mathscr{B}^q}\delta^{dia}(\Gamma(\mathbf{y}_c^{pq})) \tag{3.40}
\end{align}$$

The inequality (3.40) comes from the assumption that the lemma holds true for the labeling $\mathbf{y}^q$ and all its children q. Now consider a clique $c \in \mathscr{B}^q$. Let e^p be the length of edges from node p to its children q. Since $c \in \mathscr{B}^q$, there must exist atleast two nodes y_i and y_j in $\mathbf{y}_c$ such that $y_i^* \in \mathscr{L}^q$ and $y_j^* \notin \mathscr{L}^q$, therefore, by construction of r-HST

$$\delta^{dia}(\Gamma(\mathbf{y}_c^*)) \ge 2e^p \tag{3.41}$$

Furthermore, by the construction of $\mathbf{y}^{pq}$, $\mathscr{L}^{pq} \subseteq \mathscr{L}^p$, therefore, in worst case (leaf nodes), we can write

$$\begin{align}
\delta^{dia}(\Gamma(\mathbf{y}_c^{pq})) = max_{l_i,l_j\in\mathscr{L}^{pq}} d^t(l_i,l_j) &\le 2e^p\left(1+\frac{1}{r}+\frac{1}{r^2}+\cdots\right) \nonumber \\
&= 2e^p\left(\frac{r}{r-1}\right) \nonumber \\
&\le \delta^{dia}(\Gamma(\mathbf{y}_c^*))\left(\frac{r}{r-1}\right). \tag{3.42}
\end{align}$$

Plugging inequality (3.42) into (3.40) we get:

$$\begin{align}
&\sum_{i\in V_I^q}\theta_i(y_i^p) + \sum_{c\in\mathscr{I}^q}\delta^{dia}(\Gamma(\mathbf{y}_c^p)) + \sum_{c\in\mathscr{B}^q}\delta^{dia}(\Gamma(\mathbf{y}_c^p)) \le \nonumber \\
&\quad \sum_{i\in V_I^q}\theta_i(y_i^*) + \left(\frac{r}{r-1}\right)\min(\mathscr{M},|\mathscr{L}|)\sum_{c\in\mathscr{I}^q}\delta^{dia}(\Gamma(\mathbf{y}_c^*)) + \left(\frac{r}{r-1}\right)\sum_{c\in\mathscr{B}^q}\delta^{dia}(\Gamma(\mathbf{y}_c^*)). \tag{3.43}
\end{align}$$

In order to get the bound over the total energy we sum over all the children q of p, denoted as $\eta(p)$. Therefore, summing the inequality (3.43) over $\eta(p)$ we get

$$\sum_{q\in\eta(p)}\sum_{i\in V_I^q}\theta_i(y_i^p)+\sum_{q\in\eta(p)}\sum_{c\in\mathscr{I}^q}\delta^{dia}(\Gamma(\mathbf{y}_c^p))+\sum_{q\in\eta(p)}\sum_{c\in\mathscr{B}^q}\delta^{dia}(\Gamma(\mathbf{y}_c^p))\leq\sum_{q\in\eta(p)}\sum_{i\in V_I^q}\theta_i(y_i^*)$$
$$+\min(\mathscr{M},|\mathscr{L}|)\left(\frac{r}{r-1}\right)\sum_{q\in\eta(p)}\sum_{c\in\mathscr{I}^q}\delta^{dia}(\Gamma(\mathbf{y}_c^*))+\left(\frac{r}{r-1}\right)\sum_{q\in\eta(p)}\sum_{c\in\mathscr{B}^q}\delta^{dia}(\Gamma(\mathbf{y}_c^*)).\tag{3.44}$$

The l.h.s of the above inequality can be written as

$$\sum_{q\in\eta(p)}\sum_{i\in V_I^q}\theta_i(y_i^p)+\sum_{q\in\eta(p)}\sum_{c\in\mathscr{I}^q}\delta^{dia}(\Gamma(\mathbf{y}_c^p))+\sum_{q\in\eta(p)}\sum_{c\in\mathscr{B}^q}\delta^{dia}(\Gamma(\mathbf{y}_c^p))$$
$$=\sum_{i\in V_I^p}\theta_i(y_i^p)+\sum_{q\in\eta(p)}\sum_{c\in\mathscr{I}^q}\delta^{dia}(\Gamma(\mathbf{y}_c^p))+\sum_{q\in\eta(p)}\sum_{c\in\mathscr{B}^q}\delta^{dia}(\Gamma(\mathbf{y}_c^p))\tag{3.45}$$
$$\geq\sum_{i\in V_I^p}\theta_i(y_i^p)+\sum_{c\in\cup_{q\in\eta(p)}\mathscr{I}^q}\delta^{dia}(\Gamma(\mathbf{y}_c^p))+\sum_{c\in\cup_{q\in\eta(p)}\mathscr{B}^q}\delta^{dia}(\Gamma(\mathbf{y}_c^p))\tag{3.46}$$
$$=\sum_{i\in V_I^p}\theta_i(y_i^p)+\sum_{c\in\mathscr{I}^p}\delta^{dia}(\Gamma(\mathbf{y}_c^p))\tag{3.47}$$

The equality (3.45) is because of the fact that summing over all the interior nodes of all the child nodes of p is the same as summing over all the interior nodes of p. The inequality (3.46) is due to the fact that $\cap_{q\in\eta(p)}\mathscr{I}^q=\emptyset$ but $\cap_{q\in\eta(p)}\mathscr{B}^q$ is not necessarily an empty set, and $\delta^{dia}(\Gamma(\mathbf{y}_c))\geq 0$. The equality (3.47) is due to the fact that $\mathscr{I}^p=\{\cup_{q\in\eta(p)}\mathscr{I}^q\}\cup\{\cup_{q\in\eta(p)}\mathscr{B}^q\}$. Now let us have a look into the last term of the r.h.s of the inequality (3.44)

$$\sum_{q\in\eta(p)}\sum_{c\in\mathscr{B}^q}\delta^{dia}(\Gamma(\mathbf{y}_c^*))\leq\sum_{c\in\cup_{q\in\eta(p)}\mathscr{B}^q}\min(|\eta(p)|,|\mathbf{y}_c|)\delta^{dia}(\Gamma(\mathbf{y}_c^*))\tag{3.48}$$
$$\leq\min\left(\max_{p\in\eta(p)}|\eta(q)|,\max_c|\mathbf{y}_c|\right)\sum_{c\in\cup_{q\in\eta(p)}\mathscr{B}^q}\delta^{dia}(\Gamma(\mathbf{y}_c^*))$$
$$=\min(\mathscr{L},|\mathscr{M}|)\sum_{c\in\cup_{q\in\eta(p)}\mathscr{B}^q}\delta^{dia}(\Gamma(\mathbf{y}_c^*)).\tag{3.49}$$

The inequality (3.48) is due to the fact that $\cup_{q\in\eta(p)}\mathscr{B}^q$ can not count a clique more than $\min(|\eta(p)|,|\mathbf{y}_c|)$ times. Therefore, using the inequality (3.49) the r.h.s of the inequality (3.44)

Algorithm 8 The Move Making Algorithm for the Parsimonious Labeling Problem.

```
input Diversity $(\mathscr{L},\delta)$; $w_c, \forall c \in \mathscr{C}$; $\theta_i(x_i), \forall i \in V$; $\mathscr{L}$; $k$
1: Approximate the given diversity as the mixture of $k$ hierarchical $P^n$ Potts model using Algorithm 9.
2: for each hierarchical $P^n$ Potts model in the mixture do
3:     Use the hierarchical move making algorithm defined in the Algorithm 7.
4:     Compute the corresponding energy.
5: end for
6: Choose the solution with the minimum energy.
```

can be written as:

$$\sum_{q \in \eta(p)} \left(\sum_{i \in V_I^q} \theta_i(y_i^*) + \min(\mathscr{M}, |\mathscr{L}|) \left(\frac{r}{r-1} \right) \sum_{c \in \mathscr{I}^q} \delta^{dia}(\Gamma(\mathbf{y}_c^*)) + \left(\frac{r}{r-1} \right) \sum_{c \in \mathscr{B}^q} \delta^{dia}(\Gamma(\mathbf{y}_c^*)) \right)$$

$$\leq \sum_{i \in V_I^p} \theta_i(y_i^*) + \min(\mathscr{M}, |\mathscr{L}|) \left(\frac{r}{r-1} \right) \left(\sum_{c \in \cup_{q \in \eta(p)} \mathscr{I}^q} \delta^{dia}(\Gamma(\mathbf{y}_c^*)) + \sum_{c \in \cup_{q \in \eta(p)} \mathscr{B}^q} \delta^{dia}(\Gamma(\mathbf{y}_c^*)) \right) \tag{3.50}$$

$$= \sum_{i \in V_I^p} \theta_i(y_i^*) + \min(\mathscr{M}, |\mathscr{L}|) \left(\frac{r}{r-1} \right) \sum_{c \in \mathscr{I}^p} \delta^{dia}(\Gamma(\mathbf{y}_c^*)). \tag{3.51}$$

Finally, using inequalities (3.44), (3.47) and (3.51) we get

$$\sum_{i \in V_I^p} \theta_i(y_i^p) + \sum_{c \in \mathscr{I}^p} \delta^{dia}(\Gamma(\mathbf{y}_c^p)) \leq \sum_{i \in V_I^p} \theta_i(y_i^*) + \min(\mathscr{M}, |\mathscr{L}|) \left(\frac{r}{r-1} \right) \sum_{c \in \mathscr{I}^p} \delta^{dia}(\Gamma(\mathbf{y}_c^*)). \tag{3.52}$$

□

Applying the above lemma to the root node proves the theorem. □

3.5.2 The Move Making Algorithm for the Parsimonious Labeling

In the previous subsection, we proposed a hierarchical move making algorithm for the hierarchical P^n Potts model (Algorithm 7). This restricted us to a small class of clique potentials. In this section, we generalize our approach to the much more general parsimonious labeling problem defined using the energy function (3.28).

The move making algorithm for the parsimonious labeling problem is shown in Algorithm 8. Given diversity based clique potentials, non-negative clique weights, and arbitrary unary

Algorithm 9 Diversity to Mixture of Hierarchical P^n Potts Model.

```
input Diversity $(\mathscr{L}, \delta)$, $k$
1: Compute the induced metric, $d(.,.)$, where $d(l_i, l_j) = \delta(\{l_i, l_j\}), \forall l_i, l_j \in \mathscr{L}$.
2: Approximate $d(.,.)$ into mixture of $k$ r-HST metrics $d^t(.,.)$ using the algorithm proposed in [32].
3: for each r-HST metrics $d^t(.,.)$ do
4:     Obtain the corresponding hierarchical $P^n$ Potts model by defining the diameter diversity over $d^t(.,.)$
5: end for
```

potentials, Algorithm 8 approximates the diversity into a mixture of hierarchical P^n Potts models (using Algorithm 9) and then use the previously defined Algorithm 7 on each of the hierarchical P^n Potts models.

The algorithm for approximating a given diversity into a mixture of hierarchical P^n Potts models is shown in Algorithm 9. The first and the third steps of the Algorithm 9 have already been discussed in the previous sections. The second step, which amounts to finding the mixture of r-HST metrics for a given metric, can be solved using the randomized algorithm proposed in [32]. We refer the reader to [32] for further details of the algorithm for approximating a metric using a mixture of r-HST metrics.

Multiplicative Bound. Theorem 3 gives the multiplicative bound for the parsimonious labeling problem, when the clique potentials are any general diversity. Notice that the multiplicative bound of our algorithm is significantly better than the multiplicative bound of SoSPD [40], which is $\mathscr{M}\frac{\max_c \delta(\Gamma(\mathbf{y}_c))}{\min_c \delta(\Gamma(\mathbf{y}_c))}$.

Theorem 3. *The move making algorithm defined in Algorithm 8 gives the multiplicative bound of* $\left(\frac{r}{r-1}\right)(|\mathscr{L}|-1)O(\log|\mathscr{L}|)\min(\mathscr{M}, |\mathscr{L}|)$ *for the* parsimonious labeling *problem (equation (3.28)). Here,* $\mathscr{M}$ *is the size of the largest maximal-clique,* $|\mathscr{L}|$ *is the number of labels, and* $r > 1$ *is the parameter of the r-*HST.

Proof. Briefly, the additional factor of $(|\mathscr{L}|-1)$ and $O(\log|\mathscr{L}|)$ comes from the inequalities $\delta(\mathscr{L}) \leq (|\mathscr{L}|-1)\delta^{dia}(\mathscr{L})$ [19] and $d(.,.) \leq O(\log|\mathscr{L}|)d^t(.,.)$ [32], respectively. Mathematically, let us say that $d(.,.)$ is the induced metric of the given diversity $(\delta, \mathscr{L})$ and δ^{dia} be it's diameter diversity. We first approximate $d(.,.)$ as a mixture of r-HST metrics $d^t(.,.)$. Using Theorem 4 we get the following relationship

$$d(.,.) \leq O(\log|\mathscr{L}|)d^t(.,.). \tag{3.53}$$

For a given clique $\mathbf{y}_c$, using Proposition 1 we get the following relationship

$$\delta^{dia}(\Gamma(\mathbf{y}_c)) \leq \delta(\Gamma(\mathbf{y}_c)) \leq (|\Gamma(\mathbf{y}_c)|-1)\delta^{dia}(\Gamma(\mathbf{y}_c)). \tag{3.54}$$

Therefore, using equations (3.54) and (3.53), we get the following inequality

$$\begin{aligned}\delta^{dia}(\Gamma(\mathbf{y}_c)) \leq \delta(\Gamma(\mathbf{y}_c)) &\leq (|\Gamma(\mathbf{y}_c)|-1)\delta^{dia}(\Gamma(\mathbf{y}_c)) \\ &\leq O(\log|\Gamma(\mathbf{y}_c)|)(|\Gamma(\mathbf{y}_c)|-1)\delta_t^{dia}(\Gamma(\mathbf{y}_c)).\end{aligned} \tag{3.55}$$

where, $\delta_t^{dia}(\Gamma(\mathbf{y}_c))$ is the diameter diversity defined over the tree metric $d^t(.,.)$ which is obtained using the randomized algorithm [32] on the induced metric $d(.,.)$. Notice that $\delta_t^{dia}(\Gamma(\mathbf{y}_c))$ is the hierarchical P^n Potts model. Hence, combing the inequality (3.55) and the previously proved Theorem 2 proves Theorem 3. In case the clique potential is the diameter diversity, we do not need the inequality (3.54). Therefore, the multiplicative bound reduces to $\left(\frac{r}{r-1}\right) O(\log|\mathscr{L}|)\min(\mathscr{M},|\mathscr{L}|)$. □

Theorem 4. *Given any distance metric function $d(.,.)$ defined over a set of labels $\mathscr{L}$, the randomized algorithm given in [32] produces a mixture of r-HST tree metrics $d^t(.,.)$ such that $d(.,.) \leq O(\log|\mathscr{L}|)d^t(.,.)$.*
Proof: Please see the reference [32].

Proposition 1. *Let $(\mathscr{L},\delta)$ be a diversity with induced metric space $(\mathscr{L},d)$, then the following inequality holds $\forall\Gamma \subseteq \mathscr{L}$.*

$$\delta^{dia}(\Gamma) \leq \delta(\Gamma) \leq (|\Gamma|-1)\delta^{dia}(\Gamma). \tag{3.56}$$

Proof: Please see the reference [19].

Time Complexity. Each expansion move of our Algorithm 8 amounts to solving a graph-cut on a graph with $2|\mathscr{C}|$ auxiliary variables and $|\mathscr{C}|(2\mathscr{M}+2)$ edges (in worst case), therefore, the time complexity is $O((|V|+|\mathscr{C}|)^2|\mathscr{C}|\mathscr{M})$. In addition, each subproblem in our algorithm is defined over a much smaller label set (number of child nodes). Furthermore, Algorithm 8 can be parallelized over the trees and over the subproblems at any level of the hierarchy. In contrast, each expansion move of SoSPD [40] amounts to solving submodular max-flow, which is $O(|V|^2|\mathscr{C}|2^{\mathscr{M}})$ [39], exponential in the size of the largest clique. Furthermore, each expansion move of Ladicky *et al.* [87] amounts to solving a graph-cut on a graph with $|\mathscr{C}||\mathscr{L}|$ auxiliary nodes and $|\mathscr{C}|(2\mathscr{M}+|\mathscr{L}|)$ edges having a time complexity of $O((|V|+|\mathscr{C}||\mathscr{L}|)^2|\mathscr{C}|(\mathscr{M}+|\mathscr{L}|))$ [49]. As can be seen from the above discussion, our move-making algorithm is significantly more efficient for the parsimonious labeling problem.

3.6 Experiments

We demonstrate the utility of parsimonious labeling on both synthetic and real data. In the case of synthetic data, we perform experiments on a large number of grid lattices and evaluate our method based on the energy and the time taken. We show the modeling capabilities of the parsimonious labeling by applying it on two challenging real problems: (i) stereo matching, and (ii) image inpainting. We use the move-making algorithm for the co-occurrence based energy function proposed by Ladicky *et al.* [87] as our baseline. Based on the synthetic and the real data results, supported by the theoretical guarantees, we show that the move making algorithm proposed in our work outperforms [87].

Recall that the energy function we are interested in minimizing is defined in the equation (3.28). In our experiments, we frequently use the truncated linear metric. We define it below for the sake of completeness.

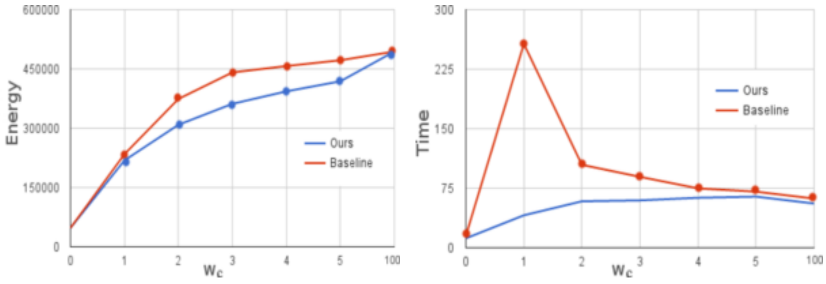
$$\theta_{i,j}(l_a, l_b) = \nu \min(|l_a - l_b|, M), \forall l_a, l_b \in \mathcal{L}. \tag{3.57}$$

where ν is the weight associated with the metric and M is the truncation constant.

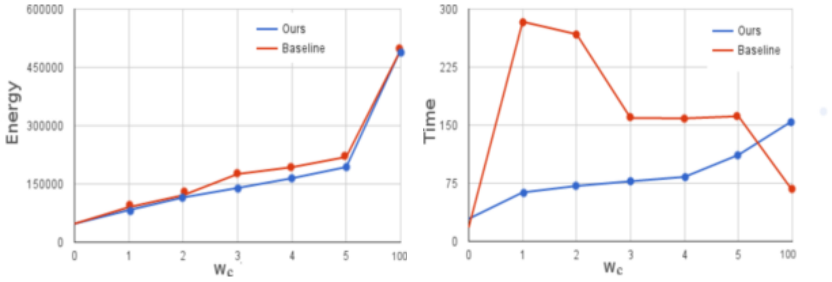
3.6.1 Synthetic Data

We consider following two cases: (i) given the hierarchical P^n Potts model, and (ii) given a general diversity based clique potential. In each of the two cases, we generate lattices of size 100×100, 20 labels, and use $\nu = 1$. The cliques are generated using a window of size 10×10 in a sliding window fashion. The unary potentials are randomly sampled from the uniform distribution defined over the interval $[0, 100]$. In the first case, we randomly generate 100 lattices and random r-HST trees associated with each lattice, ensuring that they satisfy the properties of the r-HST. Each r-HST is then converted into hierarchical P^n Potts model by defining diameter diversity over each of them. The hierarchical P^n Potts model is then used as the actual clique potential. We performed 100 such experiments. On the other hand, in the second case, for a given value of the truncation M, we generate a truncated linear metric and 100 lattices. We treat this metric as the induced metric of a diameter diversity and apply Algorithm 7 for the energy minimization. We used four different values of the truncation factor $M \in \{1, 5, 10, 20\}$. For both the experiments, we used 7 different values of w_c: $w_c \in \{0, 1, 2, 3, 4, 5, 100\}$.

The average energy and the time taken for both the methods and both the cases are shown in the Figure 3.4. It is evident from the figures that our method outperforms [87] both in terms of time and the energy. In case (ii), despite the fact that our method first approximates the given diversity into mixture of hierarchical P^n Potts models, it outperforms [87]. This can be best supported by the fact that our algorithm has a strong multiplicative bound.



(a) Energy (b) Time (in seconds)



(c) Energy (d) Time (in seconds)

Figure 3.4: *Synthetic (Blue: Our, Red: Co-occ [87]). The x-axis of all the figures is the weight associated with the cliques* (w_c)*. Figures (a) and (b) are the plots when the hierarchical* P^n *Potts model is known. Figures (c) and (d) are the plots when a diversity (diameter diversity over truncated linear metric) is given as the clique potentials which is then approximated using the mixture of hierarchical* P^n *Potts model. Notice that in both the cases our method outperforms the baseline [87] both in terms of energy and time. Also, for very high value of* $w_c = 100$*, both the methods converges to the same labeling. This is expected as a very high value of* w_c *enforces rigid smoothness by assigning everything to the same label.*

3.6.2 Real Data

In case of real data, the high-order cliques are the superpixels obtained using the mean-shift method [24], the clique potentials are the diameter diversity of the truncated linear metric (equation (3.57)). A truncated linear metric enforces smoothness in the pairwise setting, therefore, its diameter diversity will naturally enforce smoothness in the high-order cliques, which is a desired cue for the two applications we are dealing with. In both the real experiments we use $w_c = \exp^{-\frac{\rho(\mathbf{y}_c)}{\sigma^2}}$ (for high order cliques), where $\rho(\mathbf{y}_c)$ is the variance of the intensities of the pixels in the clique $\mathbf{y}_c$ and σ is a hyperparameter.

3.6.2.1 Stereo Matching

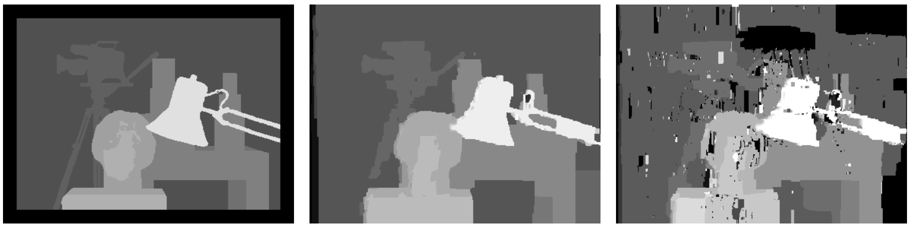
Given two rectified stereo pair of images, the problem of stereo matching is to find the disparity (gives the notion of depth) of each pixel in the reference image [118, 120]. We extend the standard setting of the stereo matching [118] to high-order cliques and test our method to the images, 'tsukuba' and 'teddy', from the widely used Middlebury stereo data set [111]. The unaries are the $L1-$norm of the difference in the RGB values of the left and the right image pixels. Notice that the index for the right image pixel is the index for the left image pixel minus the disparity, where disparity is the label. For 'tsukuba' and 'teddy' we used 16 and 60 labels respectively. In case of 'teddy' the unaries are trucated at 16. The weights w_c for the pairwise cliques are set to be proportional to the $L1-$norm of the gradient of the intensities of the neighboring pixels $\|\nabla\|_1$. In case of 'tsukuba', if $\|\nabla\|_1 < 8$, $w_c = 2$, otherwise $w_c = 1$. In case of 'teddy', if $\|\nabla\|_1 < 10$, $w_c = 3$, otherwise $w_c = 1$. As mentioned earlier, w_c for the high-order cliques is set to be proportional to the variance. We used different values of σ, ν, and M. We are showing results for the following setting: for 'tsukuba', $\nu = 20$, $\sigma = 100$, and $M = 5$; for 'teddy', $\nu = 20$, $\sigma = 100$, and $M = 1$. Figure 3.5 shows the results obtained. Notice that our method significantly outperforms [87] in terms of energy and the visual quality for both 'tsukuba' and 'teddy'.

In order to further validate our results, we visually compare our method with other well known methods in Figures 3.6 and 3.7. Notice that the energy function used in the α-expansion and the TRWS based optimization is the pairwise energy function. Therefore, the comparison is not very fair. Still we show these results in order to highlight the importance of using high order cliques and suitable optimization algorithm. It can be clearly seen that the *parsimonious labeling* gives better visual results compared to all the other three methods.

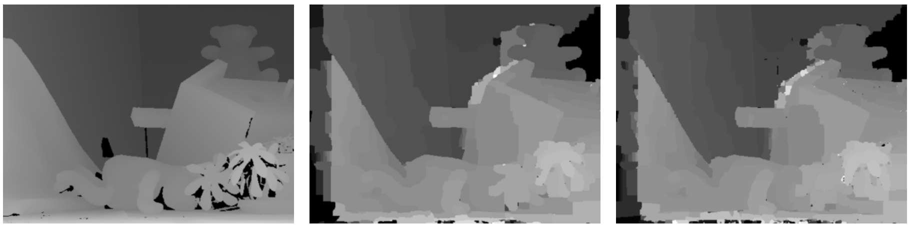
Effect of cliques and their weights. The parameter w_c is the weight associated with each clique. As we increase the weight, the algorithm emphasizes more in decreasing the clique potential corresponding to the clique which results in over smoothing. Thus, finding the best setting of w_c is very important. The effects of the parameter w_c, achieved by changing σ, is shown in the Figure 3.8. Similarly, the cliques have great impact on the overall result. Large cliques and high value of w_c will result in over smoothing. We show the effect of clique size in Figure 3.9.

3.6.2.2 Image Inpainting and Denoising

Given an image with added noise and obscured regions (regions with missing pixels), the problem is to denoise the image and fill the obscured regions such that it is consistent with the

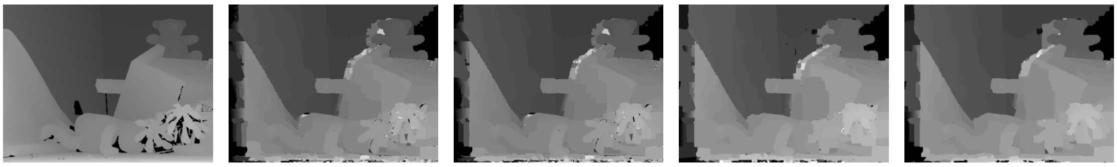


(a) Tsukuba (Energy, Time (sec)) (b) Our (**1387300**, 773) (c) Co-occ [87] (2117804, **306**)



(d) Teddy (Energy, Time (sec)) (e) Our (**3196400**, 2918) (f) Co-occ [87] (3256200, **1392**)

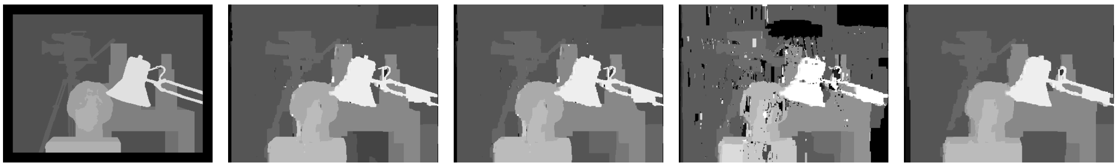
Figure 3.5: *Stereo Matching Results. Figures (a) and (d) are the ground truth disparity for the 'tsukuba' and 'teddy' respectively. Our method significantly outperforms the baseline Co-ooc [87] in both the cases in terms of energy. Our results are visually more appealing also. Figures (b) and (e) clearly shows the influence of 'parsimonious labeling' as the regions are smooth and the discontinuity is preserved. Recall that we use super-pixels obtained using the mean-shift as the cliques.*



(a) Gnd Truth (b) $\alpha-$ exp (c) TRWS (d) Co-occ (e) Our Method

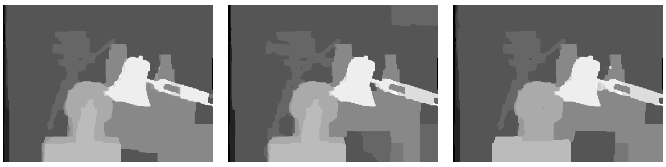
Figure 3.6: *Comparison of all the methods for the stereo matching of 'teddy'. We used the optimal setting of the parameters proposed in the well known Middlebury webpage and [118]. The above results are obtained using $\sigma = 10^2$ for the Co-occ and our method. Clearly, our method gives much smooth results while keeping the underlying shape intact. This is because of the cliques and the corresponding potentials (diversities) used. The diversities enforces smoothness over the cliques while σ controls this smoothness in order to avoid over smooth results.*

surroundings. We perform this experiment on the images, 'penguin' and 'house', from the widely used Middlebury data set. The images under consideration are gray scale, therefore, there are 256 labels in the interval $[0, 255]$, each representing an intensity value. The unaries for each pixel (or node) corresponding to a particular label is the squared difference between the label and the intensity value at that pixel. The weights w_c for the pairwise cliques are all



(a) Gnd Truth (b) $\alpha-$ exp (c) TRWS (d) Co-occ (e) Our Method

Figure 3.7: *Comparison of all the methods for the stereo matching of 'tsukuba'. We used the optimal setting of the parameters proposed in the well known Middlebury webpage and [118]. The above results are obtained using* $\sigma = 10^2$ *for the Co-occ and our method. We can see that the disparity obtained using our method is closest to the ground truth compared to all other methods. In our method, the background is uniform (under the table also), the camera shape is closest to the ground truth camera, and the face disparity is also closest to the ground truth compared to other methods.*



(a) $\sigma = 10^2$ (b) $\sigma = 10^3$ (c) $\sigma = 10^4$

Figure 3.8: *Effect of* σ *in the* parsimonious labeling. *All the parameters are same except for the* σ. *Note that as we increase the* σ, *the* w_c *increases, which in turn results in over smoothing.*

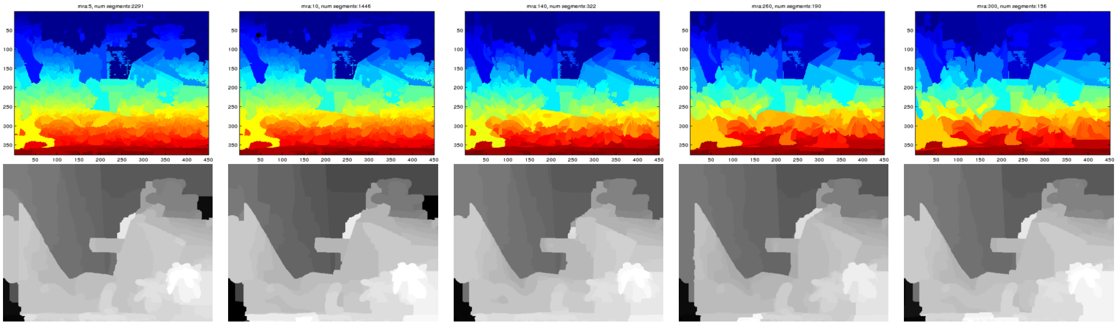
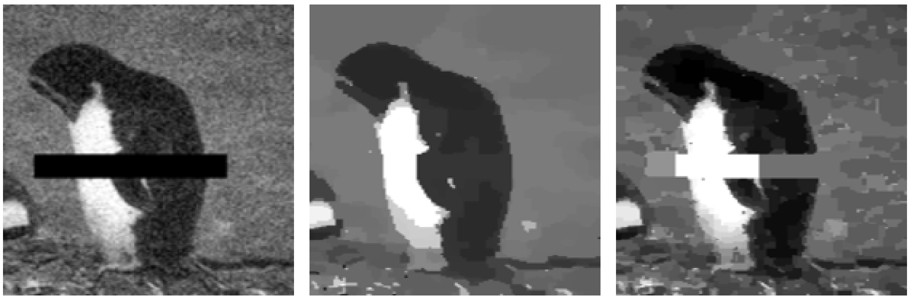
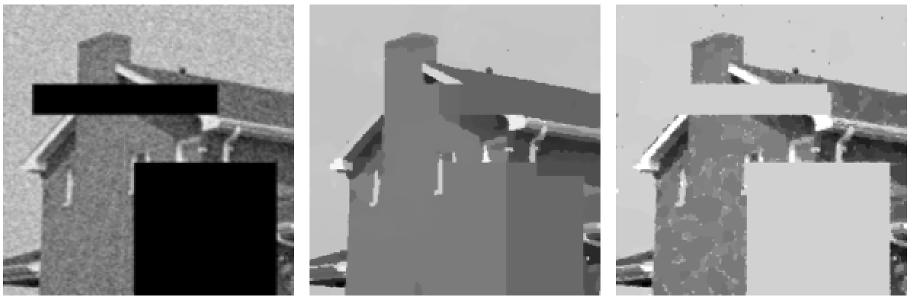

mra:5, num segments:2291
mra:10, num segments:1446
mra:140, num segments:322
mra:260, num segments:190
mra:300, num segments:156

Figure 3.9: *Effect of clique size (superpixels). The top row shows the cliques (superpixels) used and the bottom row shows the stereo matching using these cliques. As we go from left to right, the minimum number of pixels that a superpixel must contain increases. All the other parameters are the same. In order to increase the weight* w_c, *we use high value of* σ, *which is* $\sigma = 10^5$ *in all the above cases.*

set to one. We used different values of σ, ν, and the truncation M and showing results for the following setting: 'penguin', the $\nu = 40$, $\sigma = 10000$ and $M = 40$; for 'house', the $\nu = 50$, $\sigma = 1000$ and $M = 50$. Notice that our method (Figure 3.10) significantly outperforms [87] in terms of energy and visual quality for both 'penguin' and 'house'.



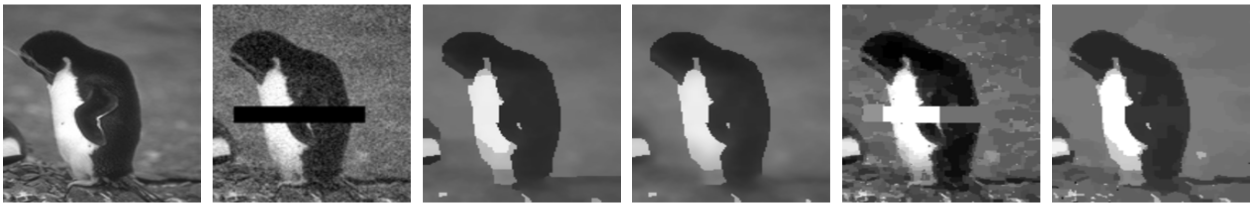
(a) Penguin (Energy, Time (sec))
(b) Our (**12547000**, 1964)
(c) Co-occ [87] (14710302, **358**)



(d) House (Energy, Time (sec))
(e) Our (**36840076**, 2838)
(f) Co-occ [87] (41980784, **912**)

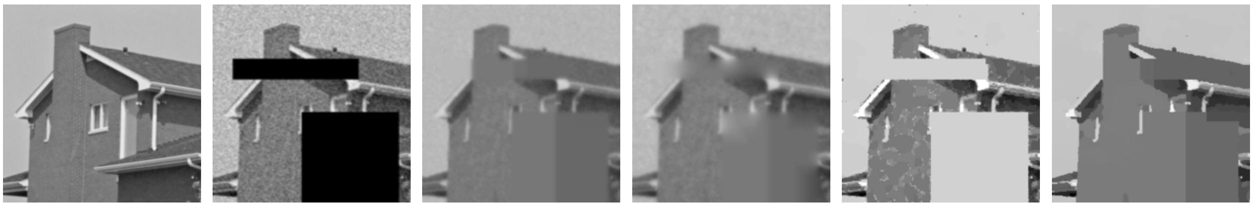
Figure 3.10: *Image inpainting results. Figures (a) and (d) are the input images of 'penguin' and 'house' with added noise and obscured regions. Our method, (b) and (e), significantly outperforms the baseline [87] in both the cases in terms of energy. Visually, our method gives much more appealing results. We use super-pixels obtained using the mean-shift as the cliques.*

In order to further validate our results, we visually compare our method with other well known methods in Figures 3.11 and 3.12 for the 'penguin' and the 'house' examples, respectively. It can be clearly seen that the *parsimonious labeling* gives visually highly promising results compared to all other methods.



(a) Original (b) Input (c) $\alpha-$ exp (d) TRWS (e) Co-occ (f) Our

Figure 3.11: *Comparison of all the methods for the image inpainting and denoising problem of the 'penguin'. Notice that our method recovers the hand of the penguin very smoothly. In other methods, except Co-oc, the ground is over-smooth while our method recovers the ground quite well compared to others.*



(a) Original (b) Input (c) $\alpha-$ exp (d) TRWS (e) Co-occ (f) Our

Figure 3.12: *Comparison of all the methods for the image inpainting and denoising problem of the 'house'.*

3.7 Discussion

We proposed a new family of labeling problems, called parsimonious labeling, where the energy function is defined using a diversity measure. Our energy function includes the novel hierarchical P^n Potts model, which allows us to design an efficient and accurate move-making algorithm based on iteratively solving the minimum st-cut problem. The large class of energy functions covered by parsimonious labeling can be used for various computer vision tasks such as semantic segmentation (where diversity function can be used to favor certain subsets of semantic classes), or 3D reconstruction (where diversity function can be used to favor certain subsets of depth values). An interesting direction for future research would be to explore different diversities and propose specific algorithms for them, which may provide better theoretical guarantees. Another interesting work would be to directly approximate diversities into a mixture of hierarchical P^n Potts model, without the use of the intermediate r-HST metrics. This may also lead to better multiplicative bounds.

In general, the models related to the inference problems, for example the model of the parsimonious labeling as shown in the equation 3.28, contains hyperparameters (w_c in case of parsimonious labeling). As evident from the Figures 3.4 and 3.8, these hyperparameters greatly influence the outcome of an inference algorithm. The standard approach is to hand tune them in order to obtain the best results. This is exactly what we did in this chapter as there were very few hyperparameters. However, hand tuning quickly becomes infeasible as the number of hyperparameters increases. In order to circumvent this problem, a risk minimization based approach, structured SVM (SSVM), can be used to automatically learn these hyperparameters using a training dataset. In the following chapter, we propose a new SSVM framework which we call HOAP-SVM to learn these parameters. The risk function used in HOAP-SVM is based on an upperbound on the average precision, which is a very frequently used performance measure in many vision related tasks.

Chapter 4

Learning to Rank using High-Order Information

4.1 Introduction

In chapter 3 we talked about a new family of high-order inference problem which we call *parsimonious labeling*. We saw that, in general, the models related to inference problems contains hyperparameters which greatly influences the outcome of the inference algorithm. Normally, these hyperparameters are hand tuned which quickly becomes infeasible as the number of hyperparameters increases. In this chapter, we talk about an empirical risk minimization based framework, structured SVM (SSVM), to automatically learn these parameters given a training dataset. The standard approach is to minimize an accuracy based risk in order to learn these parameters. However, many tasks in computer vision are evaluated based on average precision which is a measure of ranking. Motivated by this fact, in this chapter, we propose a new learning framework HOAP-SVM that can optimize an average precision based risk function while using high-order information. In what follows, we talk about the HOAP-SVM in details.

Many tasks in computer vision require the development of automatic methods that sort (or rank) a given set of visual samples according to their relevance to a query. For example, consider the problem of action classification (or more precisely action ranking). The input is a set of samples corresponding to bounding boxes of persons, and an action such as ‘jumping’. The desired output is a ranking where a sample representing a jumping person is ranked higher than a sample representing a person performing a different action. Other related problems include image classification (sorting images according to their relevance to a user query) and object detection (sorting all the windows in a set of images according to their relevance to an object category). As the desired output of the aforementioned problems is a ranking, the accuracy of an approach is typically reported using a ranking-based measure such as the average precision (AP).

A popular way of solving a problem that requires us to rank a set of samples is to train a binary classifier. The positive class of the classifier corresponds to the relevant samples and the negative class corresponds to the non-relevant samples. Once a classifier is learned on a training set, a new set of samples is sorted according to the scores assigned to the samples by the classifier. Perhaps the most commonly used classifier is the support vector machine (SVM) [127]. However, the SVM framework has two main drawbacks. First, an SVM minimizes an upper bound on the 0-1 loss function (that is, the fraction of misclassifications) instead of a ranking-based loss function that depends on the quantitative measure of the quality of the ranking (for example, average precision (AP)). Second, an SVM only uses first-order information to classify a sample, that is, the score of a sample depends only on itself and not on other samples in the dataset. In other words, an SVM does not explicitly incorporate *a priori* high-order information, which can be very useful in improving the accuracy of ranking. For example, in action classification, most of the persons present in the same image tend to perform the same action. In object detection, two objects of the same category tend to have the similar aspect ratio. In pose estimation, people in the same scene tend to have similar poses (sitting down to watch movie). In document retrieval, documents containing same or similar words are more likely to belong to the same class.

At first glance, the two drawbacks seem to be easily fixable using a generalization of the SVM framework, known as structured output support vector machines (SSVM) [122, 126]. Given a structured input, the SSVM framework provides a linear prediction rule to obtain a structured output. Specifically, the score of a putative output is the inner product of the parameters of an SSVM with the joint feature vector of the input and the output. The prediction requires us to maximize the score over all possible outputs for an input. Given a training dataset, the parameters of an SSVM are learned by minimizing a regularized convex upper bound on a user-specified loss function. In the past decade, several customized algorithms have been developed to solve the optimization problem that learns the SSVM parameters [42, 61, 75, 122, 126]. While the optimization algorithms for SSVM differ significantly in their details, they share the common characteristic of iteratively performing *loss-augmented inference*. In other words, given the current estimate of the parameters, they compute the output that jointly maximizes the sum of the score and the loss function. Loss-augmented inference can be viewed as the optimal cutting-plane or the subgradient of the learning objective, which exploits its central role in the optimization.

The SSVM framework places no restriction on the form of the loss function and on the structure of the input and the output. Thus, it appears the ideal framework to (i) optimize the loss based on AP (AP loss); and (ii) incorporate high-order information. However, in order to successfully employ an SSVM, we need an efficient algorithm for loss-augmented inference. In that regard, the first drawback can be addressed using AP-SVM [137], which is a special form of SSVM. In the AP-SVM framework, the input is a set of samples and the output is a ranking. The loss value for a putative output is one minus the AP of the corresponding ranking with respect to the ground truth ranking. The joint feature vector of the input and the output is a weighted sum of the feature vectors for all samples, where the weights are governed by the ranking. Yue *et al.* [137] showed that, for this choice of joint feature vector, loss-augmented inference can be performed optimally using an efficient greedy algorithm. Furthermore, they showed that the prediction of AP-SVM is exactly the same as the prediction of the standard SVM, that is, to sort the samples according to their individual scores. Since the joint feature vector of AP-SVM depends only on the feature vectors of the individual samples, AP-SVM does not incorporate high-order information. A straightforward way to address this deficiency would be to modify the joint feature vector such that it depends on feature vectors of pairs of samples, or more generally, on feature vectors of subsets of samples. For example, Rosenfeld *et al.* [109] recently proposed a framework to optimize the area under curve (AUC) while considering the high-order information. However, a similar approach cannot be used for optimizing AP based loss function since it does not decompose over single variable. Therefore, such a modification can not be introduced trivially into the AP-SVM formulation.

We present two alternate frameworks to incorporate high-order information for ranking. The first framework, which we call high-order binary SVM (HOB-SVM), takes its inspiration from the standard SVM. The input of HOB-SVM is a set of samples. The output is a binary label for each sample, where the label 1 indicates that the sample is relevant and 0 indicates that the sample is not relevant. The joint feature vector of HOB-SVM depends not only on the feature vectors of the individual samples, but also on the feature vectors of subsets of samples. In this work, we restrict the subsets to be of size two, but our frameworks can easily be generalized to other subset sizes. The loss function of HOB-SVM is a weighted 0-1 loss, which allows us to efficiently perform loss-augmented inference using graph cuts [74]. Practically speaking, the difficulty with employing HOB-SVM is that it provides a single score for the entire labeling of a dataset, whereas we need scores corresponding to each sample in order to find the ranking. To address this difficulty, we propose to rank the samples using the difference between the *max-marginal* for assigning a sample to the relevant class and the max-marginal for assigning it to the non-relevant class. Intuitively, difference of max-marginals measure the positivity of a particular sample while capturing high-order information. Empirically, we show that the difference of max-marginals provides an accurate ranking. The main advantage of HOB-SVM is that its parameters can be estimated efficiently by solving a convex optimization problem. However, its main disadvantage is that, similar to SVM, it optimizes a surrogate loss function instead of the AP loss.

The second framework, which we call high-order AP-SVM (HOAP-SVM), takes its inspiration from AP-SVM and HOB-SVM. Similar to AP-SVM, the input of HOAP-SVM is a set of samples, its output is a ranking of the samples, and its loss function is the AP loss. However, unlike AP-SVM, the score of a ranking is equal to the weighted sum of the difference of max-marginals of the individual samples. Since the max-marginals capture high-order information, and the loss function depends on the AP, HOAP-SVM addresses both the aforementioned deficiencies of traditional classifiers such as SVM. The main disadvantage of HOAP-SVM is that estimating its parameters requires solving a difference-of-convex program [55]. While we cannot obtain an optimal set of parameters for HOAP-SVM, we show how a local optimum of the HOAP-SVM learning problem can be computed efficiently by the concave-convex procedure [138]. Using standard, publicly available datasets, we empirically demonstrate that HOAP-SVM outperforms the baselines by effectively utilizing high-order information while optimizing the correct loss function.

Contributions.

- In this chapter we propose HOB-SVM that can incorporate high-order information and allow us to rank the samples based on difference of max-marginal scores. The HOB-SVM optimizes upperbound on accuracy based loss function which leads to a convex objective function.
- We propose another novel framework, HOAP-SVM, which can optimize average precision based loss function while incorporating high-order information. The ranking, in the end, is obtained by sorting the scores based on the difference of max-marginals. However, the objective function of HOAP-SVM is a special type of non-convex function known as the difference of convex function. Local minima of such functions can be obtained using the well known concave-convex procedure (CCCP algorithm).

4.2 Preliminaries

4.2.1 Structured Output SVM

A detailed introduction to SSVM is given in the chapter 2. For the sake of completeness, below we give a brief introduction to SSVM.

An SSVM, parameterized by $\mathbf{w}$, provides a linear prediction rule to obtain a structured output $\mathbf{y} \in \mathscr{Y}$ from a structured input $\mathbf{x} \in \mathscr{X}$. Formally, let $\Phi(\mathbf{x},\mathbf{y})$ denote the joint feature vector of the input $\mathbf{x}$ and the output $\mathbf{y}$. The prediction for a given input $\mathbf{x}$ is obtained by maximizing the score over all possible outputs, that is, $\hat{\mathbf{y}} = \mathrm{argmax}_{\mathbf{y}\in\mathscr{Y}} \mathbf{w}^\top \Phi(\mathbf{x},\mathbf{y})$.

Given a dataset that consists of n samples, that is, $\mathscr{D} = \{(\mathbf{x}_i, \mathbf{y}_i^*), i = 1, \cdots, n\}$, the parameters of an SSVM are estimated by minimizing a regularized upper bound on the empirical risk. The risk is measured using a user-specified loss function $\Delta(\cdot,\cdot)$. In more detail, the parameters are estimated by solving the following convex optimization problem:

$$\min_{\mathbf{w},\xi} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i \tag{4.1}$$

$$\text{s.t.} \quad \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{y}_i^*) \geq \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{y}_i) + \Delta(\mathbf{y}_i, \mathbf{y}_i^*) - \xi_i, \quad \forall i, \forall \mathbf{y}_i \in \mathscr{Y}_i. \tag{4.2}$$

Intuitively, the above problem encourages a margin (proportional to $\Delta(\mathbf{y}_i, \mathbf{y}_i^*)$) between the score of the ground-truth output $\mathbf{y}_i^*$ and all other outputs $\mathbf{y}_i$. The hyperparameter C controls the trade-off between the training error and the model complexity. Notice that the hyperparameter λ defined in equation (2.38) is inversely proportional to the the hyperparameter C. Inspite very large number of constraints, it has been shown that the above problem can be optimized efficiently using cutting-plane algorithm [61] which requires iteratively solving the loss-augmented inference problem (to find the most-violated constraint), that is, $\hat{\mathbf{y}}_i = \mathrm{argmax}_{\mathbf{y}_i} \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{y}_i) + \Delta(\mathbf{y}_i^*, \mathbf{y}_i)$.

4.2.2 AP-SVM

The AP-SVM (or Average Precision SVM) classifier [137] optimizes an upperbound on the ranking loss. The ranking loss in this case depends on the average precision. The AP based loss function is non decomposable over the samples. This makes the loss-augmented problem hard to solve. However, using the powerful SSVM formulation to encode the structure in the loss function allows us to solve the loss-augmented problem efficiently [96, 137]. This makes the AP-SVM classifier a special case of SSVM. In what follows we give a brief mathematical introduction to AP-SVM.

The input of an AP-SVM is a set of n samples, which we denote by $\mathbf{x} = \{x_i, i = 1, \cdots, n\}$. Each sample can either belong to the positive class (that is, the sample is relevant) or the negative class (that is, the sample is not relevant). The indices for the positive and negative samples are denoted by the sets $\mathscr{P}$ and $\mathscr{N}$ respectively. In other words, if $i \in \mathscr{P}$ and $j \in \mathscr{N}$ then x_i belongs to positive class and x_j belongs to the negative class. The desired output is a ranking matrix $\mathbf{R}$ of size $n \times n$, such that (i) $\mathbf{R}_{ij} = 1$ if x_i is ranked higher than x_j; (ii)

$\mathbf{R}_{ij} = -1$ if x_i is ranked lower than x_j; and (iii) $\mathbf{R}_{ij} = 0$ if x_i and x_j are assigned the same rank. During training, the ground-truth ranking matrix $\mathbf{R}^*$ is defined as: (i) $\mathbf{R}^*_{ij} = 1$ and $\mathbf{R}^*_{ji} = -1$ for all $i \in \mathscr{P}$ and $j \in \mathscr{N}$; (ii) $\mathbf{R}^*_{ii'} = 0$ and $\mathbf{R}^*_{jj'} = 0$ for all $i, i' \in \mathscr{P}$ and $j, j' \in \mathscr{N}$.

Joint Feature Vector. For a sample x_i, let $\phi(x_i)$ denote its feature vector. For example, in action classification, $\phi(x_i)$ can represent poselet [9] or bag-of-visual-words [26]. Similar to [137], we specify a joint feature vector as

$$\Phi(\mathbf{x}, \mathbf{R}) = \gamma \sum_{i \in \mathscr{P}} \sum_{j \in \mathscr{N}} \mathbf{R}_{ij}(\phi(x_i) - \phi(x_j)), \gamma = \frac{1}{|\mathscr{P}||\mathscr{N}|} \tag{4.3}$$

In other words, the joint feature vector is the scaled sum of the difference between the features of all pairs of samples having different classes.

Parameters and Prediction. The parameter vector of the classifier is denoted by $\mathbf{w}$. Given the parameters $\mathbf{w}$, the ranking of an input $\mathbf{x}$ is predicted by maximizing the score, that is, $\mathbf{R} = \text{argmax}_{\overline{\mathbf{R}}} \mathbf{w}^\top \Phi(\mathbf{x}, \overline{\mathbf{R}})$. Yue *et al.* [137] showed that the above optimization can be performed efficiently by sorting the samples x_k in descending order of the score $\mathbf{w}^\top \phi(x_k)$.

Loss Function. Given a training dataset, our aim is to learn a classifier that provides a high AP measure. Let $\text{AP}(\mathbf{R}^*, \mathbf{R})$ denote the AP of the ranking matrix $\mathbf{R}$ with respect to the ground truth ranking $\mathbf{R}^*$. The $\text{AP}(\mathbf{R}^*, \mathbf{R})$ is defined as: $\text{AP}(\mathbf{R}, \mathbf{R}^*) = \frac{1}{|\mathscr{P}|} \sum_k Prec(k) \delta(Rec(k))$, where $|\mathscr{P}|$ is the number of positive samples in the ground truth $\mathbf{R}^*$, $Prec(k)$ is the precision upto top k samples given by $\mathbf{R}$, and $\delta(Rec(k))$ is the change in recall when moving from $(k-1)^{th}$ to k^{th} sample. The value of the $\text{AP}(\cdot, \cdot)$ lies between 0 and 1, where 0 corresponds to a completely incorrect ranking $-\mathbf{R}^*$ and 1 corresponds to the correct ranking $\mathbf{R}^*$. In order to maximize the AP, we will minimize a loss function defined as $\Delta(\mathbf{R}^*, \mathbf{R}) = 1 - \text{AP}(\mathbf{R}^*, \mathbf{R})$.

Parameter Estimation. Given the input $\mathbf{x}$ and the ground-truth ranking matrix $\mathbf{R}^*$, we would like to learn the parameters of the classifier such that regularized upper bound on the empirical AP loss is minimized. Specifically, the model parameters are obtained by solving the following convex optimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\xi, \\ & \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{R}^*) - \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{R}) \geq \Delta(\mathbf{R}^*, \mathbf{R}) - \xi, \forall \mathbf{R} \end{aligned} \tag{4.4}$$

Problem (4.4) is specified over an exponential number of $\mathbf{R}$. Nonetheless, Yue *et al.* [137] showed that it can be optimized efficiently by providing an optimal greedy algorithm to solve the corresponding loss-augmented inference problem, that is, $\hat{\mathbf{R}} = \text{argmax}_{\overline{\mathbf{R}}} \mathbf{w}^\top \Phi(\mathbf{x}, \overline{\mathbf{R}}) + \Delta(\mathbf{R}^*, \overline{\mathbf{R}})$.

4.3 High-Order Binary SVM (HOB-SVM)

We now describe our two frameworks for ranking while incorporating high-order information. As mentioned earlier, we will restrict our description to second-order information. However, extending our frameworks to general high-order information is trivial. We start with the simpler framework, which we call *High-Order Binary* SVM (HOB-SVM). This will allow us to define the terminology necessary to develop a more principled framework (HOAP-SVM) in the next section.

The input of a HOB-SVM is a set of n samples $\mathbf{x} = \{x_i | i = 1, \cdots, n\}$. Similar to the AP-SVM, a sample can either belong to the positive class or a negative class. However, the output of HOB-SVM is not a ranking, but an assignment of a class for each sample. In other words, the output is a list $\mathbf{y} = \{y_i | i = 1, \cdots, n\}$ where $y_i \in \{0, 1\}$. The label '0' implies that the sample has been assigned to the negative class, whereas the label '1' implies that the sample has been assigned to the positive class. During training, the ground-truth output $\mathbf{y}^*$ assigns all relevant samples to the positive class and all non-relevant samples to the negative class. Once again, given $\mathbf{y}^*$, we denote the indices of the positive and the negative samples as $\mathscr{P}$ and $\mathscr{N}$ respectively.

Joint Feature Vector. The joint feature vector of the input $\mathbf{x}$ and the output $\mathbf{y}$ consists of two parts. The first part $\Phi_1(\mathbf{x}, \mathbf{y})$ captures first-order information, and is henceforth referred to as the unary joint feature vector. The second part $\Phi_2(\mathbf{x}, \mathbf{y})$ captures second-order information, and is henceforth referred to as the pairwise joint feature vector. In more detail, let $\phi(x_i) \in \mathbb{R}^d$ denote the feature vector of the sample x_i. The unary joint feature vector is defined as follows:

$$\Phi_1(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} \sum_{i, y_i = 1} \phi(x_i) \\ \sum_{i, y_i = 0} \phi(x_i) \end{pmatrix}. \tag{4.5}$$

The unary joint feature vector is of dimensionality $2d$. The first d dimensions correspond to the sum of the feature vectors of the samples belonging to the positive class. The last d dimensions correspond to the sum of the feature vectors of the samples belonging to the negative class. Clearly, $\Phi_1(\mathbf{x}, \mathbf{y})$ only captures the first-order information.

As mentioned earlier, our aim is to use second-order information to improve ranking. In other words, if we know *a priori* that two samples x_i and x_j are more likely to belong to the same class (henceforth referred to as similar samples), then we would like to encourage them to either both be labeled as relevant or as non-relevant. Let $\mathscr{E}$ denote the set of all pairs of similar samples. In other words, if samples x_i and x_j are similar, then $(i,j) \in \mathscr{E}$. We define the pairwise joint feature vector as follows:

$$\Phi_2(\mathbf{x},\mathbf{y}) = \eta \left(\sum_{(i,j)\in\mathscr{E}, y_i \neq y_j} \theta(\phi(x_i),\phi(x_j)) \right) \tag{4.6}$$

where $\theta(\phi(x_i),\phi(x_j))$ is a vector such that each of its elements is inversely proportional to the difference between the corresponding elements of its two input vectors and η controls the trade-off between the first-order and high-order information. In our work, we define $\theta(\mathbf{z}_i,\mathbf{z}_j) = \exp(-(\mathbf{z}_i - \mathbf{z}_j)^2)$. All the operations are performed in an element-wise manner. In other words, $\Phi_2(\mathbf{x},\mathbf{y})$ is a d dimensional vector that is the sum of pairwise feature vectors over all pairs of similar samples having different classes.

Notice that the $\mathscr{E}$ represents the underlying structure or the interaction between the variables in the output space, which otherwise is missing in the AP-SVM formulation. Obtaining $\mathscr{E}$ is part of the modeling of the task in hand. For example, while solving semantic segmentation problem in images, a four neighborhood connectivity models the problem quite well. Therefore, the four neighborhood connectivity hypothesis leads to a promising set of $\mathscr{E}$ for the semantic segmentation problem. In the experiment section we talk about the hypothesis we used to generate the $\mathscr{E}$ for the action classification problem.

Parameters and Prediction. Similar to the joint feature vector, the parameters of a HOB-SVM consist of two parts: the unary parameters $\mathbf{w}_1 \in \mathbb{R}^{2d}$ and the pairwise parameters $\mathbf{w}_2 \in \mathbb{R}^d$. Given an input $\mathbf{x}$, the output $\mathbf{y}$ is predicted by maximizing the score, that is,

$$\mathbf{y} = \operatorname*{argmax}_{\mathbf{y}} \mathbf{w}^\top \Phi(\mathbf{x},\mathbf{y}), \mathbf{w} = \begin{pmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \end{pmatrix}, \Phi(\mathbf{x},\mathbf{y}) = \begin{pmatrix} \Phi_1(\mathbf{x},\mathbf{y}) \\ \Phi_2(\mathbf{x},\mathbf{y}) \end{pmatrix}. \tag{4.7}$$

Note that, in general, the problem in the equation (4.7), which is the inference problem, is NP-hard. A description of such problems is given the section 2.2.1. However, when $\mathbf{w}_2 \leq 0$, it can be optimized efficiently using graph cuts [74] (please refer to the section 2.3 for detailed explanation). This follows from the fact that each element of the pairwise joint feature vector is non-negative (see equation (4.6)), and hence the score of an output $\mathbf{y}$ is a supermodular (negative of submodular) function of $\mathbf{y}$. In what follows, we will always estimate the parameters of a HOB-SVM under the constraint that $\mathbf{w}_2 \leq 0$. Moreover, our approaches can also be used without this constraint by employing approximate inference algorithms such as [70, 76, 105].

Loss Function. Although we would ideally like to optimize the AP loss, as mentioned earlier, this results in a difficult loss-augmented inference problem when the joint feature vector captures high-order information. Hence, inspired by the success of SVM for ranking, we use a surrogate loss function defined as follows:

$$\Delta(\mathbf{y}^*,\mathbf{y}) = \frac{J\sum_{i,y_i^*=1}\delta(y_i=0)+\sum_{j,y_j^*=0}\delta(y_j=1)}{J|\mathscr{P}|+|\mathscr{N}|}, \tag{4.8}$$

where $\delta(\cdot)$ is 1 if its argument is true and 0 otherwise. The terms $|\mathscr{P}|$ and $|\mathscr{N}|$ are the total number of positive and negative samples (as specified by the ground-truth assignment $\mathbf{y}^*$) respectively. The hyperparameter J is set to $|\mathscr{N}|/|\mathscr{P}|$. This helps in data balancing. In other words, $\Delta(\mathbf{y}^*,\mathbf{y})$ is the weighted fraction of misclassifications.

Parameter Estimation. Given the dataset $(\mathbf{x},\mathbf{y}^*)$, the parameters of HOB-SVM are obtained by solving the following convex optimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\xi, \\ & \mathbf{w}^\top\Phi(\mathbf{x},\mathbf{y}^*) - \mathbf{w}^\top\Phi(\mathbf{x},\mathbf{y}) \geq \Delta(\mathbf{y}^*,\mathbf{y}) - \xi, \forall \mathbf{y} \in \mathscr{Y}, \mathbf{w}_2 \leq 0. \end{aligned} \tag{4.9}$$

Even though the number of constraints in the above problem are exponential in the number of samples n, it can be optimized efficiently by iteratively solving the loss-augmented inference problem, that is, $\hat{\mathbf{y}} = \text{argmax}_{\bar{\mathbf{y}}}(\mathbf{w}^\top\Phi(\mathbf{x},\bar{\mathbf{y}}) + \Delta(\mathbf{y}^*,\bar{\mathbf{y}}))$. The restriction $\mathbf{w}_2 \leq 0$ allows us to solve the above problem efficiently using graph cuts [74]. The problem (4.9) is similar to training graphical models with approximate inference [37, 42, 82, 123].

Using HOB-SVM for Ranking. From a theoretical point of view, the main disadvantage of HOB-SVM is that it optimizes a surrogate loss function instead of the AP loss. In the next section, we will describe a novel framework that addresses this disadvantage. From a practical point of view, the main disadvantage of HOB-SVM is that it provides a single score for the entire assignment $\mathbf{y}$. In other words, instead of assigning an individual score for each sample x_k, it assigns one score $\mathbf{w}^\top\Phi(\mathbf{x},\mathbf{y})$ for all the samples taken together. This prevents us from specifying a ranking of the samples. To address this issue, we propose a simple yet intuitive solution: (i) compute the difference between the max-marginal of a sample being assigned to the positive class and the max-marginal of it being assigned to the negative class; and (ii) sort the samples according to the difference in max-marginals. Max-marginal captures high-order information and the difference of max-marginals measures our confidence on a particular sample belonging to the positive class. Formally, we define the max-marginal of a sample x_i belonging to the positive class $m_i^+(\mathbf{w})$ and negative class $m_i^-(\mathbf{w})$ as:

$$m_i^+(\mathbf{w}) = \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{y}_i^+), \mathbf{y}_i^+ = \underset{\mathbf{y}, y_i = 1}{\operatorname{argmax}} \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{y}). \tag{4.10}$$

$$m_i^-(\mathbf{w}) = \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{y}_i^-), \mathbf{y}_i^- = \underset{\mathbf{y}, y_i = 0}{\operatorname{argmax}} \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{y}). \tag{4.11}$$

The max-marginals for all the samples can be computed efficiently using the *dynamic graph cuts* algorithm [68, 69]. Given the max-marginals $m_i^+(\mathbf{w})$ and $m_i^-(\mathbf{w})$, the score of a sample x_i is defined as

$$s_i(\mathbf{w}) = m_i^+(\mathbf{w}) - m_i^-(\mathbf{w}). \tag{4.12}$$

Note that, if the two labelings $\mathbf{y}_i^+$ and $\mathbf{y}_i^-$ defined in equations (4.10)-(4.11) respectively differ only in the label assigned to the sample x_i, this implies that the sample x_i has no influence in determining the labels of the other samples in the dataset. In this case, the difference in max-marginals does not depend on the feature vectors of any other samples except the sample x_i. However, if the sample x_i does influence the labels of the other samples (that is, $\mathbf{y}_i^+$ and $\mathbf{y}_i^-$ differ significantly), then the difference in the max-marginals depends on several samples in the dataset. The ranking is obtained by sorting the samples in descending order of their scores $s_i(\mathbf{w})$. As will be seen in the experiments section, this intuitive way of scoring a sample provides an improved ranking over the baselines.

4.4 High-Order Average Precision SVM (HOAP-SVM)

While HOB-SVM allows us to incorporate high-order information via the pairwise joint feature vector, it suffers from the deficiency of using a surrogate loss function. Specifically, instead of optimizing the AP loss in order to estimate the parameters, it optimizes a weighted 0-1 loss. However, the way that HOB-SVM obtains a ranking points us to the direction of resolving this deficiency. We begin by presenting the high-level overview of our approach. We observe that the score of a ranking according to an AP-SVM is the weighted sum of the scores of the individual samples. The reason why AP-SVM fails to capture high-order information is that the score of the individual sample depends on no other sample in the dataset. This is in contrast to the score employed by HOB-SVM (see equation (4.12)). Hence, it would be desirable to extend AP-SVM such that the score of the ranking is the weighted sum of the difference of max-marginals of individual samples. This is precisely our next learning framework, which we call *High-Order* AP-SVM (HOAP-SVM). In what follows, we describe HOAP-SVM in detail.

The input of HOAP-SVM is a set of n samples $\mathbf{x} = \{x_i, i = 1, \cdots, n\}$. Similar to AP-SVM, a sample can belong to the positive class or the negative class. The output of HOAP-SVM is a ranking matrix $\mathbf{R}$, defined in a similar manner to AP-SVM. During training, the ground-truth ranking matrix $\mathbf{R}^*$ assigns each positive sample to a higher rank than all negative samples. Once again, the indices of positive and negative samples is represented as $\mathscr{P}$ and $\mathscr{N}$ respectively.

Score of a Ranking. The parameters of HOAP-SVM are denoted by $\mathbf{w}$. Given an input $\mathbf{x}$ and a ranking $\mathbf{R}$, the score for the ranking specified by HOAP-SVM is defined as follows:

$$S(\mathbf{x}, \mathbf{R}; \mathbf{w}) = \gamma \sum_{i \in \mathscr{P}} \sum_{j \in \mathscr{N}} \mathbf{R}_{ij}(s_i(\mathbf{w}) - s_j(\mathbf{w})), \gamma = \frac{1}{|\mathscr{P}||\mathscr{N}|}, \tag{4.13}$$

where $s_i(\mathbf{w})$ is as specified in equation (4.12). In other words, the score of a ranking is the weighted sum of the difference of max-marginals for each sample, where the weights are specified by the ranking.

Prediction. Given an input $\mathbf{x}$, the ranking $\mathbf{R}$ is predicted by maximizing the score over all possible rankings, that is,

$$\mathbf{R} = \operatorname*{argmax}_{\overline{\mathbf{R}}} S(\mathbf{x}, \mathbf{R}; \mathbf{w}). \tag{4.14}$$

Proposition 2. *Problem (4.14) can be solved efficiently by sorting the samples in descending order of their scores $s_i(\mathbf{w})$.*

Proof. Predicting ranking is basically picking each $\mathbf{R}_{ij}$ to maximize $S(\mathbf{x}, \mathbf{R}; \mathbf{w})$. Since $\mathbf{R}_{ij} = sign(s_i(\mathbf{w}) - s_j(\mathbf{w}))$, therefore, maximizing $S(\mathbf{x}, \mathbf{R}; \mathbf{w})$ w.r.t. $\mathbf{R}$ is basically reshuffling the scores such that $sign(s_i(\mathbf{w}) - s_j(\mathbf{w})) > 0$, which can be achieved by sorting the scores. Refer to [58] for more details. $\square$

In other words, the prediction for HOAP-SVM is the same as the prediction for HOB-SVM. Recall that the score $s_i(\mathbf{w})$ can be computed efficiently using dynamic graph cuts [68, 69].

Parameter Estimation. Given the input $\mathbf{x}$ and the ground-truth ranking $\mathbf{R}^*$, the parameters of HOAP-SVM are learned by optimizing the AP loss. To this end, we propose to estimate $\mathbf{w}$

by solving the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\xi, \\ & S(\mathbf{x}, \mathbf{R}^*; \mathbf{w}) - S(\mathbf{x}, \mathbf{R}; \mathbf{w}) \geq \Delta(\mathbf{R}^*, \mathbf{R}) - \xi, \forall \mathbf{R}, \mathbf{w}_2 \leq 0. \end{aligned} \tag{4.15}$$

Here, $\Delta(\mathbf{R}^*, \mathbf{R})$ is the AP loss (discussed in section 4.2.2), that is, one minus the AP of the ranking $\mathbf{R}$ with respect to $\mathbf{R}^*$. The following proposition establishes the suitability of the above problem for learning an HOAP-SVM.

Proposition 3. *Problem (4.15) minimizes a regularized upper bound on the* AP *loss of the predicted ranking*

Proof. The proof is similar to the derivation of the convex upperbound of empirical loss as discussed in the section 2.4. For a given $\mathbf{w}$ and predicted ranking $\mathbf{R}(\mathbf{w})$ (see equation 4.14), the AP loss can be upperbounded as follows:

$$\Delta(\mathbf{R}(\mathbf{w}), \mathbf{R}^*) = \Delta(\mathbf{R}(\mathbf{w}), \mathbf{R}^*) + S(\mathbf{x}, \mathbf{R}(\mathbf{w}); \mathbf{w}) - S(\mathbf{x}, \mathbf{R}(\mathbf{w}); \mathbf{w}) \tag{4.16}$$

$$\leq \Delta(\mathbf{R}(\mathbf{w}), \mathbf{R}^*) + S(\mathbf{x}, \mathbf{R}(\mathbf{w}); \mathbf{w}) - S(\mathbf{x}, \mathbf{R}^*; \mathbf{w}) \tag{4.17}$$

$$\leq \max_{R} (\Delta(\mathbf{R}, \mathbf{R}^*) + S(\mathbf{x}, \mathbf{R}; \mathbf{w})) - S(\mathbf{x}, \mathbf{R}^*; \mathbf{w}) \tag{4.18}$$

In expression (4.16) we add and subtract the same term, therefore, it is valid. Expression (4.17) follows from the fact that the score for the ground truth ranking $\mathbf{R}^*$ must be less than or equal to the score corresponding to the predicted ranking $\mathbf{R}(\mathbf{w})$ (see equation (4.14)). Expression (4.18) follows from the fact that the first two terms of the expression (4.18) are always greater than or equal to the first two terms of the expression (4.17). Using this upperbound on the AP loss (4.18), the regularized objective function of HOAP-SVM can be written as:

$$\min_{\mathbf{w}} \quad \frac{1}{2}||\mathbf{w}||^2 + C\left\{ \max_{R} \Big(S(\mathbf{x}, \mathbf{R}; \mathbf{w}) + \Delta(\mathbf{R}^*, \mathbf{R}) \Big) - S(\mathbf{x}, \mathbf{R}; \mathbf{w}) \right\}, \mathbf{w}_2 \leq 0. \tag{4.19}$$

Note that the objective function (4.15) is the constrained form of the objective function (4.19), therefore, they represent exactly the same optimization problem. □

Optimization. While problem (4.15) provides a valid upper bound on the AP loss, it is not a convex program. Hence, it cannot be optimized efficiently to obtain an optimal set of parameters for HOAP-SVM. However, in what follows, we show that problem (4.15) is a difference-of-convex program. By identifying the convex and the concave part of problem (4.15), we show how a locally optimal set of parameters can be obtained efficiently using the concave-convex procedure (CCCP) [138].

We begin by specifying the following shorthand notation that will be useful in simplifying problem (4.15). Given a ranking $\mathbf{R}$ we define functions $f(\mathbf{w};\mathbf{R})$ and $g(\mathbf{w};\mathbf{R})$ of the parameters $\mathbf{w}$ as

$$f(\mathbf{w};\mathbf{R}) = \gamma\sum_{i\in\mathscr{P}} m_i^-(\mathbf{w})\left(\sum_{j\in\mathscr{N}}(\mathbf{R}^*_{ij}-\mathbf{R}_{ij})\right)+\gamma\sum_{j\in\mathscr{N}} m_j^+(\mathbf{w})\left(\sum_{i\in\mathscr{P}}(\mathbf{R}^*_{ij}-\mathbf{R}_{ij})\right). \tag{4.20}$$

$$g(\mathbf{w};\mathbf{R}) = \gamma\sum_{i\in\mathscr{P}} m_i^+(\mathbf{w})\left(\sum_{j\in\mathscr{N}}(\mathbf{R}^*_{ij}-\mathbf{R}_{ij})\right)+\gamma\sum_{j\in\mathscr{N}} m_j^-(\mathbf{w})\left(\sum_{i\in\mathscr{P}}(\mathbf{R}^*_{ij}-\mathbf{R}_{ij})\right). \tag{4.21}$$

Using our shorthand notation problem (4.15) can be rewritten as follows:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\xi, \\ & \xi \geq \Delta(\mathbf{R}^*,\mathbf{R}) + f(\mathbf{w};\mathbf{R}) - g(\mathbf{w};\mathbf{R}), \forall\mathbf{R}. \end{aligned} \tag{4.22}$$

Proposition 4. *For any valid ranking matrix $\mathbf{R}$, the functions $f(\mathbf{w};\mathbf{R})$ and $g(\mathbf{w};\mathbf{R})$ are convex in $\mathbf{w}$. Therefore, the problem (4.22) represents a difference of convex program.*

Proof. Substituting different terms (e.g. $m_i^+(\mathbf{w})$, $m_j^-(\mathbf{w})$) in the objective function (4.15) leads to the following form:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\xi, \\ s.t. \quad & \frac{1}{|\mathscr{P}||\mathscr{N}|}\sum_{i\in\mathscr{P}}\sum_{j\in\mathscr{N}}\mathbf{R}^*_{ij}\{(m_i^+(\mathbf{w})-m_i^-(\mathbf{w}))-(m_j^+(\mathbf{w})-m_j^-(\mathbf{w}))\} \\ - \quad & \frac{1}{|\mathscr{P}||\mathscr{N}|}\sum_{i\in\mathscr{P}}\sum_{j\in\mathscr{N}}\mathbf{R}_{ij}\{(m_i^+(\mathbf{w})-m_i^-(\mathbf{w}))-(m_j^+(\mathbf{w})-m_j^-(\mathbf{w}))\} \\ \geq \quad & \Delta(\mathbf{R},\mathbf{R}^*)-\xi,\ \xi\geq 0, \forall R. \end{aligned} \tag{4.23}$$

Rearranging different terms:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\xi, \\ \xi \quad \geq \quad & \frac{1}{|\mathscr{P}||\mathscr{N}|}\sum_{i\in\mathscr{P}}\sum_{j\in\mathscr{N}}(\mathbf{R}^*_{ij}-\mathbf{R}_{ij})(m_i^-(\mathbf{w})+m_j^+(\mathbf{w}))+\Delta(\mathbf{R},\mathbf{R}^*) \\ - \quad & \frac{1}{|\mathscr{P}||\mathscr{N}|}\sum_{i\in\mathscr{P}}\sum_{j\in\mathscr{N}}(\mathbf{R}^*_{ij}-\mathbf{R}_{ij})(m_i^+(\mathbf{w})+m_j^-(\mathbf{w})), \forall R. \end{aligned} \tag{4.24}$$

Using shorthand notations, the objective function (4.24) can be further written as:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\xi, \\ & \xi \geq \Delta(\mathbf{R}^*,\mathbf{R}) + f(\mathbf{w};\mathbf{R}) - g(\mathbf{w};\mathbf{R}), \forall\mathbf{R}. \end{aligned} \tag{4.25}$$

Algorithm 10 The CCCP algorithm for learning HOAP-SVM parameters.

```
input Samples $\mathbf{x}$, ranking $\mathbf{R}^*$, tolerance $\varepsilon$, initial parameters $\mathbf{w}_0$.
$t \leftarrow 0$.
repeat
    For all $\mathbf{R}$, find a linear lower bound $l(\mathbf{w};\mathbf{R})$ tight at $\mathbf{w}_t$ using Proposition 5.
    Update the parameters by solving the following convex optimization problem (Algorithm 11):
        $\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w}} \quad \frac{1}{2}||\mathbf{w}||^2 + C\xi, \quad (4.29)$
        $\xi \geq \Delta(\mathbf{R}^*,\mathbf{R}) + f(\mathbf{w};\mathbf{R}) - l(\mathbf{w};\mathbf{R}), \forall \mathbf{R}.$
    The loss-augmented inference can be solved efficiently using Proposition 6.
    $t \leftarrow t+1$.
until Objective of problem (4.22) does not decrease more than $\varepsilon$.
```

where,

$$f(\mathbf{w};\mathbf{R}) = \gamma \sum_{i\in\mathscr{P}} m_i^-(\mathbf{w}) \left(\sum_{j\in\mathscr{N}} (\mathbf{R}^*_{ij} - \mathbf{R}_{ij}) \right) + \gamma \sum_{j\in\mathscr{N}} m_j^+(\mathbf{w}) \left(\sum_{i\in\mathscr{P}} (\mathbf{R}^*_{ij} - \mathbf{R}_{ij}) \right), \quad (4.26)$$

$$g(\mathbf{w};\mathbf{R}) = \gamma \sum_{i\in\mathscr{P}} m_i^+(\mathbf{w}) \left(\sum_{j\in\mathscr{N}} (\mathbf{R}^*_{ij} - \mathbf{R}_{ij}) \right) + \gamma \sum_{j\in\mathscr{N}} m_j^-(\mathbf{w}) \left(\sum_{i\in\mathscr{P}} (\mathbf{R}^*_{ij} - \mathbf{R}_{ij}) \right), \quad (4.27)$$

$$\gamma = \frac{1}{|\mathscr{P}||\mathscr{N}|}. \quad (4.28)$$

Since max-marginals in itself are convex ($\max_{\mathbf{y},y_i=+1} \mathbf{w}^\top \Phi(\mathbf{x},\mathbf{y})$ is the max over affine functions which is convex) and the summation of convex functions with positive coefficients is also convex, therefore, the convexity or concavity of the terms in (4.26) and (4.27) depends on the sign of $\sum_i(\mathbf{R}^*_{ij} - \mathbf{R}_{ij})$ and $\sum_j(\mathbf{R}^*_{ij} - \mathbf{R}_{ij})$. We can observe that for the fixed values of i, $0 \leq \sum_j \mathbf{R}_{ij} \leq |\mathscr{N}|$, similarly, $0 \leq \sum_i \mathbf{R}_{ij} \leq |\mathscr{P}|$. Also, since $\mathbf{R}^*$ is the true ranking matrix, therefore, $\sum_j \mathbf{R}^*_{ij} = |\mathscr{N}|$ and $\sum_i \mathbf{R}^*_{ij} = |\mathscr{P}|$. Hence, we can conclude that $\sum_j(\mathbf{R}^*_{ij} - \mathbf{R}_{ij}) \geq 0$ and $\sum_i(\mathbf{R}^*_{ij} - \mathbf{R}_{ij}) \geq 0$. Therefore, for any valid $\mathbf{R}$, the constraints of the optimization problem (4.22) clearly represents a difference-of-convex functions [55, 138] with $f(\mathbf{w};\mathbf{R})$ and $-g(\mathbf{w};\mathbf{R})$ as the the convex and concave functions respectively. □

Using proposition 4, it follows that problem (4.22) is a difference-of-convex program. This allows us to obtain a locally optimal set of parameters for the HOAP-SVM formulation using the CCCP approach outlined in Algorithm 10. The CCCP algorithm (also discussed in section 2.5.3) consists of two steps. In the first step, given the current set of parameters $\mathbf{w}_t$, we obtain a linear approximation $l(\mathbf{w};\mathbf{R})$ of the function $g(\mathbf{w};\mathbf{R})$ such that $l(\mathbf{w}_t;\mathbf{R}) = g(\mathbf{w}_t;\mathbf{R}), l(\mathbf{w};\mathbf{R}) \leq g(\mathbf{w};\mathbf{R}), \forall \mathbf{w}$. In other words, the linear function $l(\mathbf{w};\mathbf{R})$ is a lower bound on the function $g(\mathbf{w};\mathbf{R})$ (or upperbound on $-g(\mathbf{w};\mathbf{R})$ which is concave) such that the lower bound is tight at the current parameters $\mathbf{w}_t$. While at first sight the problem of obtaining the linear approximation for each ranking matrix $\mathbf{R}$ may appear to be highly expensive, the following proposition shows how this step can be performed in a computationally efficient manner.

Proposition 5. *Given the current set of parameters* $\mathbf{w}_t$*, let*

$$\bar{\mathbf{y}}_i^+ = \operatorname*{argmax}_{\mathbf{y}, y_i=1} \mathbf{w}_t^\top \Phi(\mathbf{x},\mathbf{y}), \bar{\mathbf{y}}_j^- = \operatorname*{argmax}_{\mathbf{y}, y_j=-1} \mathbf{w}_t^\top \Phi(\mathbf{x},\mathbf{y}). \tag{4.30}$$

The following linear function is a lower bound on $g(\mathbf{w};\mathbf{R})$ *that is tight at* $\mathbf{w}_t$*:*

$$l(\mathbf{w};\mathbf{R}) = \gamma \sum_{i\in\mathcal{P}} \mathbf{w}^\top \Phi(\mathbf{x},\bar{\mathbf{y}}_i^+) \left(\sum_{j\in\mathcal{N}} (\mathbf{R}^*_{ij} - \mathbf{R}_{ij}) \right) + \gamma \sum_{j\in\mathcal{N}} \mathbf{w}^\top \Phi(\mathbf{x},\bar{\mathbf{y}}_j^-) \left(\sum_{i\in\mathcal{P}} (\mathbf{R}^*_{ij} - \mathbf{R}_{ij}) \right)$$

Proof. Upperbounding the concave function $-g$ is equivalent to lower bounding the convex function g. Given the current set of parameters $\mathbf{w}_t$, we obtain a linear approximation $l(\mathbf{w};\mathbf{R})$ of the function $g(\mathbf{w};\mathbf{R})$ such that

$$l(\mathbf{w}_t;\mathbf{R}) = g(\mathbf{w}_t;\mathbf{R}), l(\mathbf{w};\mathbf{R}) \leq g(\mathbf{w};\mathbf{R}), \forall \mathbf{w}. \tag{4.31}$$

In other words, the linear function $l(\mathbf{w};\mathbf{R})$ is a lower bound on the function $g(\mathbf{w};\mathbf{R})$ such that the lower bound is tight at the current parameters $\mathbf{w}_t$. Since $g(\mathbf{w};\mathbf{R})$ is a convex function, therefore, the linear approximation of g at a given $\mathbf{w}_t$ lowerbounds g. Mathematically,

$$\underbrace{(\mathbf{w}-\mathbf{w}_t)^\top \frac{\partial g(\mathbf{w};\mathbf{R})}{\partial \mathbf{w}}\bigg|_{\mathbf{w}_t} + g(\mathbf{w}_t;\mathbf{R})}_{l(\mathbf{w};\mathbf{R})} \leq g(\mathbf{w};\mathbf{R}), \forall \mathbf{w} \tag{4.32}$$

Note that, $g(\mathbf{w};\mathbf{R})$ is the linear combination of max-marginals with positive coefficients. Also, max-marginals are the functions of $\mathbf{w}$, therefore, finding gradient of $g(\mathbf{w};\mathbf{R})$ is equivalent to finding gradients of max-marginals independently and summing them up with their respective coefficients. Gradients for the max-marginals are not defined as max-marginals are the *max* over affine functions, therefore, non-smooth. Instead, we compute the subgradients. One way to define the subgradient of the max-marginal at $\mathbf{w}_t$ is:

$$\begin{aligned} \frac{\partial(m_i^+(\mathbf{w}))}{\partial \mathbf{w}}\bigg|_{\mathbf{w}_t} &= \frac{\partial(\operatorname{argmax}_{\mathbf{y},y_i=1} \mathbf{w}^\top \Phi(\mathbf{x},\mathbf{y}))}{\partial \mathbf{w}}\bigg|_{\mathbf{w}_t}, \\ &= \Phi(\mathbf{x},\bar{\mathbf{y}}_i^+), \bar{\mathbf{y}}_i^+ = \operatorname*{argmax}_{\mathbf{y},y_i=1} \mathbf{w}_t^\top \Phi(\mathbf{x},\mathbf{y}). \end{aligned} \tag{4.33}$$

Similarly, subgradient of $m_j^-(\mathbf{w})$ can be defined. From equations (4.21) and (4.33) it is clear that:

$$g(\mathbf{w}_t;\mathbf{R}) = (\mathbf{w}_t)^\top \frac{\partial g(\mathbf{w};\mathbf{R})}{\partial \mathbf{w}}\Big|_{\mathbf{w}_t} \tag{4.34}$$

Combining equations (4.34) and (4.32), the lower bound can be defined as:

$$\begin{align} l(\mathbf{w}_t;\mathbf{R}) &= \mathbf{w}^\top \frac{\partial g(\mathbf{w};\mathbf{R})}{\partial \mathbf{w}}\Big|_{\mathbf{w}_t} \tag{4.35}\\ &\leq g(\mathbf{w};\mathbf{R}), \forall \mathbf{w}. \tag{4.36}\end{align}$$

Also from equation (4.34) it is clear that $l(\mathbf{w}_t;\mathbf{R}) = g(\mathbf{w}_t;\mathbf{R})$. Therefore, combining equations (4.35), (4.33) and (4.21) leads to the following form of $l(\mathbf{w};\mathbf{R})$:

$$\begin{align} l(\mathbf{w};\mathbf{R}) = &\frac{1}{|\mathscr{P}||\mathscr{N}|}\sum_{i\in\mathscr{P}} \mathbf{w}^\top\Phi(\mathbf{x},\bar{\mathbf{y}}_i^+)\left(\sum_{j\in\mathscr{N}}(\mathbf{R}_{ij}^* - \mathbf{R}_{ij})\right)\\ &+\frac{1}{|\mathscr{P}||\mathscr{N}|}\sum_{j\in\mathscr{N}} \mathbf{w}^\top\Phi(\mathbf{x},\bar{\mathbf{y}}_j^-)\left(\sum_{i\in\mathscr{P}}(\mathbf{R}_{ij}^* - \mathbf{R}_{ij})\right). \tag{4.37}\end{align}$$

While at first sight the problem of obtaining the linear approximation for each ranking matrix $\mathbf{R}$ may appear to be highly expensive, but a little insight to the function $g(\mathbf{w};\mathbf{R})$ reveals that each constraint corresponding to each $\mathbf{R}$ has exactly the same convex terms, $m_i^+(\mathbf{w})$ and $m_j^-(\mathbf{w})$, independent of $\mathbf{R}$. Therefore, we do not have to lower bound $g(\mathbf{w};\mathbf{R})$ again and again for each $\mathbf{R}$. Also, lowerbounding $g(\mathbf{w};\mathbf{R})$ at a given $\mathbf{w}_t$ is equivalent to finding the max-marginal labelings of the entire dataset at $\mathbf{w}_t$ such that each sample is marginalized to get its ground truth label (see equation 4.33)). Each labeling problem can be solved using graph cut [15]. Solving graph cuts n times naively would be computationally expensive, therefore, in order to improve the efficiency, we use dynamic graph cuts [69]. In the case of max-marginals, the structure of the graph remains the same for all the samples, and only unary potentials change. Therefore, the time complexity of the dynamic graph cuts is mainly dominated by the time complexity of the graph construction for the first time and the first cut in the graph, which is computationally very efficient [68]. ☐

An upshot of the above proposition is that the linear lower bound of $g(\mathbf{w};\mathbf{R})$ can be computed efficiently for any $\mathbf{R}$ by pre-computing the labelings $\bar{\mathbf{y}}_i^+$ and $\bar{\mathbf{y}}_j^-$, which are independent of $\mathbf{R}$. The labelings $\bar{\mathbf{y}}_i^+$ and $\bar{\mathbf{y}}_j^-$ can be obtained efficiently using dynamic graph cuts [68, 69].

In the second step of CCCP, we obtain a convex optimization problem by substituting the linear approximation $l(\mathbf{w};\mathbf{R})$ in place of the convex function $g(\mathbf{w};\mathbf{R})$. We update the parameters by solving the resulting convex optimization problem. To this end, we use the cutting-plane algorithm [61] (Algorithm 11), discussed in details in section 2.4.3, in order to handle exponentially many constraints. Cutting-plane algorithm requires us to iteratively find

Algorithm 11 The Cutting Plane algorithm for optimizing the resulting convex function of HOAP-SVM.

```
input Samples $\mathbf{x}$, ranking $\mathbf{R}^*$, tolerance $\varepsilon$, initial parameters $\mathbf{w}_0$.
$t \leftarrow 0$.
Start with empty constraint set: $\mathscr{C} \leftarrow \{\}$.
repeat
    Find the most violated constraint using following equation:
        $\hat{R} \leftarrow \operatorname*{argmax}_{R} \Big\{\eta \sum_{i\in\mathscr{P}} \sum_{j\in\mathscr{N}} R_{ij}(\bar{s}_i(\mathbf{w}) - \bar{s}_j(\mathbf{w})) + \Delta(R,R^*)$
        $\quad - \eta \sum_{i\in\mathscr{P}} \sum_{j\in\mathscr{N}} R^*_{ij}(\bar{s}_i(\mathbf{w}) - \bar{s}_j(\mathbf{w}))\Big\}$    (4.38)
    Update the constraint set with the new most violated constraint.
    Update the parameters $\mathbf{w}$ by solving the following convex optimization problem with
    all the constraints in the constraint set $\mathscr{C}$:
        $\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w}} \quad \frac{1}{2}||\mathbf{w}||^2 + C\xi,$    (4.39)
until No additional constraint can be added.
```

the most-violated ranking $\hat{\mathbf{R}}$. The following proposition makes the cutting-plane algorithm efficient.

Proposition 6. *Given the upperbounded scores* $\bar{m}_i^+(\mathbf{w}) = \mathbf{w}^\top \Phi(\mathbf{x}, \bar{\mathbf{y}}_i^+)$, $\bar{m}_j^-(\mathbf{w}) = \mathbf{w}^\top \Phi(\mathbf{x}, \bar{\mathbf{y}}_j^-)$, *and the scores for the current parameters* $m_i^+(\mathbf{w}) = \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{y}_i^+)$, $m_j^-(\mathbf{w}) = \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{y}_j^-)$, *the following problem gives the most violated ranking.*

$$\hat{\mathbf{R}} \leftarrow \operatorname*{argmax}_{\mathbf{R}} \Big\{\eta \sum_{i\in\mathscr{P}, j\in\mathscr{N}} \mathbf{R}_{ij}(\bar{s}_i(\mathbf{w}) - \bar{s}_j(\mathbf{w})) + \Delta(\mathbf{R}, \mathbf{R}^*) - \eta \sum_{i\in\mathscr{P}, j\in\mathscr{N}} \mathbf{R}^*_{ij}(\bar{s}_i(\mathbf{w}) - \bar{s}_j(\mathbf{w}))\Big\}$$

where, $\bar{s}_i(\mathbf{w}) = (\bar{m}_i^+(\mathbf{w}) - m_i^-(\mathbf{w}))$ *and* $\bar{s}_j(\mathbf{w}) = (m_j^+(\mathbf{w}) - \bar{m}_j^-(\mathbf{w}))$. *The greedy algorithm of [137] can be used to find the* $\hat{\mathbf{R}}$ *efficiently.*

Proof. In the second step of the CCCP algorithm (Algorithm 10), we obtain a convex optimization problem by substituting the linear approximation $l(\mathbf{w}; \mathbf{R})$ in place of the convex function $g(\mathbf{w}; \mathbf{R})$. We update the parameters by solving the convex optimization problem. Since the space of $\mathbf{R}$ is very large, therefore, in order to handle exponentially many constraints we use the cutting plane algorithm (Algorithm 11) to optimize the resulting convex objective function [61] . The algorithm iteratively introduces *most violated constraints* until we have solved the original problem within a desired tolerance ε. In case of bounded joint feature map and bounded loss function, the number of iterations to solve this convex optimization is independent of the number of training samples and is linear in the desired precision (ε) and the regularization parameter (C) [61]. Let us define some short hand notations in order to derive the final formulation for finding the most violated constraint. For given $\bar{\mathbf{y}}_i^+$ and $\bar{\mathbf{y}}_j^-$ (see equation 4.33), the scores for the lower bounded functions corresponding to the current $\mathbf{w}$ can be defined as:

$$\bar{m}_i^+(\mathbf{w}) = \mathbf{w}^\top \Phi(\mathbf{x}, \bar{\mathbf{y}}_i^+), \bar{m}_j^-(\mathbf{w}) = \mathbf{w}^\top \Phi(\mathbf{x}, \bar{\mathbf{y}}_j^-). \tag{4.40}$$

Substituting the upper bounded scores (4.40) into (4.24) in order to obtain the convex problem leads to the following objective function:

$$\begin{aligned}
\min_{\mathbf{w}} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\xi, \\
s.t. \quad & \eta \sum_{i\in\mathscr{P}} \sum_{j\in\mathscr{N}} \mathbf{R}^*_{ij}\{(\bar{m}_i^+(\mathbf{w}) - m_i^-(\mathbf{w})) - (m_j^+(\mathbf{w}) - \bar{m}_j^-(\mathbf{w}))\} \\
- \quad & \eta \sum_{i\in\mathscr{P}} \sum_{j\in\mathscr{N}} \mathbf{R}_{ij}\{(\bar{m}_i^+(\mathbf{w}) - m_i^-(\mathbf{w})) - (m_j^+(\mathbf{w}) - \bar{m}_j^-(\mathbf{w}))\} \\
\geq \quad & \Delta(\mathbf{R}, \mathbf{R}^*) - \xi, \ \xi \geq 0, \forall R.
\end{aligned} \tag{4.41}$$

Therefore, *finding the most violated constraint* problem can be defined as:

$$\begin{aligned}
\hat{R} \ \leftarrow \ & \operatorname*{argmax}_R \Big\{ \eta \sum_{i\in\mathscr{P}} \sum_{j\in\mathscr{N}} R_{ij}(\bar{s}_i(\mathbf{w}) - \bar{s}_j(\mathbf{w})) + \Delta(R, R^*) \\
- \ & \eta \sum_{i\in\mathscr{P}} \sum_{j\in\mathscr{N}} R^*_{ij}(\bar{s}_i(\mathbf{w}) - \bar{s}_j(\mathbf{w})) \Big\}
\end{aligned} \tag{4.42}$$

where, $\bar{s}_i(\mathbf{w}) = (\bar{m}_i^+(\mathbf{w}) - m_i^-(\mathbf{w}))$ and $\bar{s}_j(\mathbf{w}) = (m_j^+(\mathbf{w}) - \bar{m}_j^-(\mathbf{w}))$. Note that the problem (4.42) is equivalent to the standard problem of finding the most violated constraint as defined in [137], therefore, the most violated constraint $\hat{R}$ can be computed using the optimal greedy procedure proposed in [137]. This step is mainly dominated by the sorting of the scores $\bar{s}_i(\mathbf{w})$ and $\bar{s}_j(\mathbf{w})$, which has the time complexity of $O(n \log n)$. □

Upon convergence, the CCCP algorithm provides a locally optimal set of parameters for the HOAP-SVM framework.

4.5 Experiments

We now demonstrate the efficacy of our learning frameworks on the challenging problem of action classification [26, 93]. The input for action classification is an action class such as ‘jumping’ or ‘running’ and a set of samples $\mathbf{x} = \{x_i = (\mathbf{I}_i, \mathbf{b}_i), i = 1, \cdots, n\}$. Here, $\mathbf{I}_i$ is the image corresponding to the i-th sample, and $\mathbf{b}_i$ is a tight bounding box around a person present in the image. The desired output is a ranking of the samples according to their relevance to the action. Recall that our main hypothesis is that high-order information can

Table 4.1: *The AP over five folds for the best setting of the hyperparameters obtained using the cross-validation. Our frameworks outperforms* SVM *and* AP-SVM *in all the 10 action classes. Note that* HOAP-SVM *is initialized with* HOB-SVM.

Actions/ Methods	Jump	Phone	Play inst	Read	Ride bike	Run	Take photo	Use comp	Walk	Ride horse	Average
SVM	56.0	35.5	42.6	33.8	81.9	78.4	33.9	37.2	61.7	85.9	54.7
AP-SVM	57.5	34.4	46.3	35.5	83.0	79.3	33.3	42.7	63.1	86.6	56.2
HOB-SVM	60.9	**36.1**	48.1	35.7	84.1	**81.5**	35.1	45.8	63.0	**87.9**	57.8
HOAP-SVM	**63.4**	34.5	**48.8**	**38.3**	**84.3**	81.0	**36.5**	**48.7**	**65.3**	87.7	**58.9**

Table 4.2: *The AP of all the four methods. The training is performed over the entire 'trainval' dataset of PASCAL VOC 2011 using the best hyperparameters obtained during 5-fold cross-validation. The testing is performed on the 'test' dataset and evaluated on the PASCAL VOC server. Note that* HOAP-SVM *is initialized using* HOB-SVM.

Actions/ Methods	Jump	Phone	Play inst	Read	Ride bike	Run	Take photo	Use comp	Walk	Ride horse	Average
SVM	51.1	29.7	40.5	20.6	81.1	76.7	20.0	27.7	56.7	84.2	48.82
AP-SVM	54.0	**33.8**	42.3	26.5	82.5	76.7	23.7	32.8	**57.7**	84.2	51.42
HOB-SVM	56.3	**33.8**	42.8	24.3	82.5	80.5	**27.7**	32.8	53.6	84.5	51.88
HOAP-SVM	**59.5**	**33.8**	**47.5**	**27.2**	**84.0**	**82.6**	26.1	**36.4**	55.1	**85.3**	**53.75**

help improve the ranking accuracy. To test our hypothesis, we require a set of similar samples such that samples x_i and x_j are more likely to belong to the same class (relevant or non-relevant) if $(i,j) \in \mathscr{E}$. *In the action classification experiments, we define* $\mathscr{E} = \{(i,j), \mathbf{I}_i = \mathbf{I}_j\}$, *that is, the set of all pairs of bounding boxes that are present in the same image.* Note that one could use any other similarity criterion in the proposed frameworks. Below we describe our experimental setup in detail.

Dataset. We use PASCAL VOC 2011 [31] action classification dataset, which consists of 4846 images depicting 10 action classes. The dataset is divided into two subsets: 2424 'trainval' images for which we are provided the bounding boxes of the person in the image together with their action classes; and 2422 'test' images for which we are only provided with the person bounding boxes.

Features. Given a sample $x_i = (\mathbf{I}_i, \mathbf{b}_i)$, we use the concatenation of standard poselet-based feature vector [9] of the bounding box $\mathbf{b}_i$ and GIST feature vector [101] of the image $\mathbf{I}_i$ to specify the sample features $\phi(x_i)$. The poselet feature consists of 2400 activation scores of action-specific poselets and 4 object activation scores. The GIST feature is a 512 dimensional feature vector that captures the overall scene depicted in the image. This results in a sample feature of size 2916. The sample features used to specify the unary and the pairwise joint feature vectors are shown in equations (4.5) and (4.6). As each pair of similar samples comes from the same image, we defined the joint pairwise feature vector using only the poselet features. The size of the joint feature vector is therefore 8748.

Methods. We compare our proposed approaches, namely HOB-SVM and HOAP-SVM, with the standard binary SVM (obtained by setting $\mathbf{w}_2 = 0$ in HOB-SVM) and AP-SVM (obtained by setting $\mathbf{w}_2 = 0$ in HOAP-SVM) that ignore high-order information. The baselines, SVM and AP-SVM requires one hyperparameter C, and HOB-SVM and HOAP-SVM requires two hyperparameters C and η. The common hyperparameter C is the trade-off between the regularization and the empirical loss, and η is the trade-off between the first order information and the high-order information. Note that the 'test' dataset was not used for cross-validation. We obtained the best setting of the hyperparameters for each method independently via a 5-fold cross-validation on the entire 'trainval' dataset. We consider the following putative values: $C \in \{10^{-1}, 10^{0}, \ldots, 10^{4}\}$ and $\eta \in \{10^{-4}, 10^{0}, \ldots, 10^{4}\}$. The J parameter in (4.8) is fixed to $|\mathscr{N}|/|\mathscr{P}|$.

Results. Table 4.1 shows the average AP over all the five folds for the best hyperparameter setting. By incorporating high-order information HOB-SVM provides an improvement in the ranking compared to the commonly used SVM classifier for all 10 action classes. Furthermore, even though HOB-SVM employs a surrogate loss function, it provides more accurate rankings compared to AP-SVM for 9 action classes. By optimizing the AP loss function, while incorpoating high-order information, HOAP-SVM outperforms SVM in 9 action classes, AP-SVM in all 10 action classes, and HOB-SVM in 7 action classes. Table 4.2 shows the AP values obtained for the 'test' set when the methods are trained using the best hyperparameter setting over the entire 'trainval' set. Table 4.2 clearly shows that HOB-SVM outperforms SVM classifier in 9 action classes and AP-SVM in 5 along with 3 ties. On the other hand, HOAP-SVM outperforms SVM classifier in all the 10 classes, AP-SVM in 8 along with 1 tie, and HOB-SVM in 8 along with 1 tie.

The paired t-test shows that: (a) HOB-SVM is statistically better than SVM for 6 action classes, (b) HOB-SVM is not statistically better than AP-SVM, (c) HOAP-SVM is statistically better than SVM for 6 action classes, and (d) HOAP-SVM is statistically better than AP-SVM for 4 action classes.

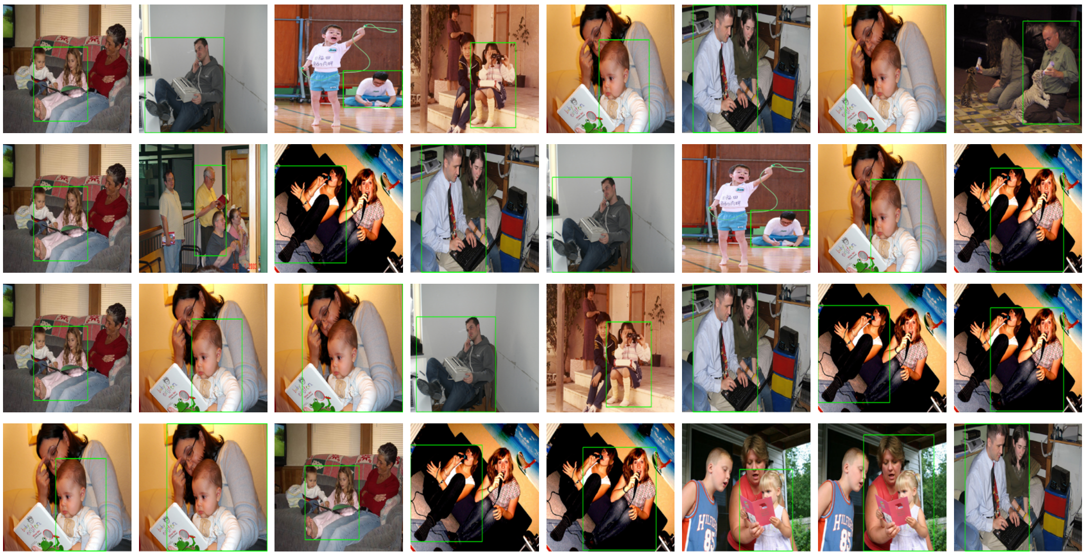


Figure 4.1: *Top 8 samples ranked by all the four methods for 'reading' action class. First row –* SVM*, Second row –* AP-SVM*, Third row –* HOB-SVM*, and Fourth row –* HOAP-SVM*. Note that, the first false positive is ranked 2nd in case of* SVM *(first row) and 3rd in case of* AP-SVM *(second row), this shows the importance of optimizing the* AP *loss. On the other hand, in case of* HOB-SVM *(third row), the first false positive is ranked 4th and the 'similar samples' (2nd and 3rd) are assigned similar scores, this illustrates the importance of using high-order information. Furthermore,* HOAP-SVM *(fourth row) has the best* AP *among all the four methods, this shows the importance of using high-order information and optimizing the correct loss. Note that, in case of* HOAP-SVM*, the 4th and 5th ranked samples are false positives (underlying action is close to reading) and they both belong to the same image (our similarity criterion). This indicates that high-order information sometimes may lead to poor test* AP *in case of confusing classes (such as 'playinginstrument' vs 'usingcomputer') by assigning all the connected samples to the wrong label. Same effect can be seen in* HOB-SVM *for 7th and 8th ranked samples.*

The effects of incorporating high-order information and optimizing the AP based loss function is illustrated in Fig. 4.1. While high-order information can introduce errors in the ranking, in general it provides boost in the overall performance.

4.6 Discussion

We proposed two new learning frameworks that incorporate high-order information to improve the accuracy of ranking. The first framework, HOB-SVM, incorporates high-order information while optimizing a surrogate loss function, which allows us to compute its parameters by solving a convex optimization problem. The second framework, HOAP-SVM, incorporates high-order information while optimizing an AP based loss function, which results in a difference-of-convex optimization problem. Both HOB-SVM and HOAP-SVM outperform baseline methods that do not make use of high-order information. By minimizing the correct loss function, HOAP-SVM outperforms HOB-SVM. An interesting direction for future work would be to allow for weakly supervised learning by extending the recently proposed *latent* AP-SVM [5] formulation to use high-order information. While such a learning formulation can be easily obtained with the introduction of latent variables, it is not clear whether the resulting optimization problem can be solved efficiently.

Another interesting direction of research work would be to explore the whole range of the regularization parameter C in order to obtain the best model. In this chapter we chose few putative values of the regularization parameter $C \in \{10^{-1}, 10^{0}, \ldots, 10^{4}\}$ and used five fold cross validation based approach to find the best C among the chosen putative values. The hope thus was that one of the C among the chosen putative values would lead to a robust model which would perform well at the test time. However, there are infinitely possible values of the regularization parameter which clearly is not reflected by the chosen putative values. Therefore, the regularization parameter chosen using the cross validation may lead to a suboptimal model. This is exactly the problem we talk about in the following chapter. We propose new algorithm, regularization path for SSVM, that can efficiently explore the whole range of the regularization parameter in order to obtain the best model.

Chapter 5

Regularization Path for SSVM using BCFW and its variants

All the research presented in this chapter were conducted under the supervision of Dr. Simon Lacoste-Julien at the SIERRA Team of INRIA (Paris) during my visit from 15^{th} *June 2015 to* 15^{th} *September 2015.*

5.1 Introduction

Structured output prediction [88, 122, 126] is one of the key problems in the machine learning and computer vision community. As we have seen in chapter 2 and chapter 4, the objective function of an SSVM is parametric and depends on the regularization parameter λ (also referred as C which is inversely proportional to λ), which controls the trade-off between the model complexity and an upperbound on the empirical risk. It has been seen in various experiments in chapter 4 that the value of the regularization parameter have a significant impact on the performance or the generalization of the SSVM learning method. Thus, we must choose it very carefully in order to obtain the best model. Finding an appropriate value of the regularization parameter often requires us to tune it. However, lack of knowledge about the structure of the regularization parameter compel us to cross validate it over the entire parameter space, which is practically not feasible because there are infinitely many possible values of λ. To circumvent this problem, the standard approach is to resort to a sub optimal solution by cross validating a small set of regularization parameters on a given training dataset. This is exactly what we did while performing experiments to find the best regularization parameter for models discussed in the chapter 4. Doing this is tedious and quickly becomes infeasible as we increase the number of regularization parameters. Therefore, in this chapter, we propose an algorithm which we call SSVM-RP, to obtain an ε-optimal regularization path of SSVM. By definition, regularization path is the set of solutions for all possible values of the regularization parameter in the parameter space [29]. In order to make the SSVM-RP algorithm efficient, we propose different intuitive variants of the Block-Coordinate Frank-Wolfe (BCFW) algorithm. As we have seen in chapter 4, it is sometimes necessary to put additional constraints on the SSVM objective function (for example, positivity/negativity constraints) in order to obtain the global optimum of the inference problem. Another such example can be seen in [119]. Motivated by this fact, we also propose a principled approach to optimize SSVM with general box constraints (SSVM-B) using BCFW algorithm. Lastly, we propose regularization path algorithm of SSVM with positivity constraints, which can be trivially modified to obtain the regularization path algorithm of SSVM with negativity constraints. In what follows, we build the intuition behind the concept of the regularization path and then talk about the algorithms in details.

As mentioned earlier, obtaining regularization path for any parametric model implies exploring the whole space of the regularization parameter in an highly efficient manner to provide optimal learned model for any given value of the regularization parameter $\lambda \in [0, \infty]$. This allows us to obtain the best possible model. We are interested in ε-optimal regularization path for SSVM, which means that for any value of $\lambda \in [0, \infty]$, the algorithm must provide a learned model which is guaranteed to have a maximum duality gap of ε. The key idea behind the regularization path algorithm is to break the regularization parameter space into segments such that learning an ε optimal model for any value of λ in a particular segment guarantees that the same learned model is ε optimal for all the values of λ in that particular segment. Mathematically, let us say that the model under consideration has the following form:

$$\begin{aligned} \min_{\mathbf{w}\in\mathbb{R}^n} \quad & f_\lambda(\mathbf{w}) := l(\mathbf{w}) + \lambda r(\mathbf{w}) \\ s.t. \quad & f_i(\mathbf{w}) \leq 0, \forall i \in [1, \cdots, m]. \end{aligned} \tag{5.1}$$

where, $l(\mathbf{w})$ is the loss term, $f_i(\mathbf{w})$ are the constraints, and $r(\mathbf{w})$ is the regularization term. The regularization term $r(\mathbf{w})$ can be L_1 norm, L_2 norm, or any other norm suitable for the task under consideration. The Lagrange dual of the above problem can be written as:

$$\mathbb{L}_\lambda(\mathbf{w}, \alpha) = l(\mathbf{w}) + \lambda r(\mathbf{w}) + \sum_i^m \alpha_i f_i(\mathbf{w}). \tag{5.2}$$

where, $\alpha \in \mathbb{R}^m_{\geq 0}$ is the dual variable vector. Let $d_\lambda(\alpha) = \min_{\mathbf{w}} \mathbb{L}_\lambda(\mathbf{w}, \alpha)$ be the dual problem of the above mentioned Lagrange dual. Therefore, the duality gap can be defined as $g(\mathbf{w}, \alpha; \lambda) = f_\lambda(\mathbf{w}) - d_\lambda(\alpha)$, where $\mathbf{w}$ is the minimizer of the primal problem 5.1 and α

is the maximizer of the dual problem $d_\lambda(\alpha)$. A given pair of primal and dual solutions, $\mathbf{w}$ and α, is ε-optimal at a predefined λ if $g(\mathbf{w},\alpha;\lambda) \leq \varepsilon$. As mentioned earlier, the idea behind the ε-optimal regularization path is to break the regularization parameter space $\lambda \in [0,\infty]$ into segments. This can be achieved by finding breakpoints or kinks $\{\lambda_k,\lambda_{k-1},\lambda_1\}$ such that if the pair $(\mathbf{w},\alpha)$ is ε-optimal for any $\lambda \in [\lambda_k,\lambda_{k-1})$, then it is ε-optimal for all $\lambda \in [\lambda_k,\lambda_{k-1})$. In another words, if $\exists\lambda \in [\lambda_k,\lambda_{k-1})$ such that $g(\mathbf{w},\alpha;\lambda) \leq \varepsilon$, then $g(\mathbf{w},\alpha;\lambda) \leq \varepsilon, \forall\lambda \in [\lambda_k,\lambda_{k-1})$. Now the challenge remains to find these breakpoints and corresponding optimal models in an efficient manner. In this chapter we show how to efficiently find these breakpoints for SSVM. We also show how BCFW algorithm and its variants allow us to warm start the optimization process at each breakpoint in order to find the complete regularization path highly efficiently.

Contributions

- ε-optimal regularization path algorithm for SSVM.
- Principled approach for the optimization of SSVM with additional box constraints.
- ε-optimal regularization path algorithm for SSVM with additional positivity/negativity constraints.

5.2 Related Work

In the past, there have been many works for developing regularization path algorithms for different useful learning methods such as LASSO [124] (model with squared errors and L_1 regularization), grouped LASSO [136], graphical LASSO [44], elastic net [141] (regularization with the combination of L_1 and L_2 norms), SVM [127], SVM with L_1 norm, and multiple kernel learning [2]. The work on regularization path primarily started with the seminal work of [29] which proposed an algorithm popularly known as the *least angle regression*. This work showed that the regularization path of LASSO [124] is piece wise linear. The regularization path for elastic net was proposed by [141]. The work by [46, 104] can be seen as the generalization of the previous two works in which the authors proposed the regularization path algorithm for generalized linear model with L_1, L_2, or elastic net type penalties. Furthermore, the work by [94, 136] and [45] proposed the regularization path algorithms for grouped LASSO and graphical LASSO, respectively. The regularization path for SVM was first proposed by [52], which was further revisited by [102]. On the other hand, because of the useful intrinsic properties of the sparsity inducing L_1 norm, the regularization path for SVM with L_1 norm is proposed in the work by [140]. All the above mentioned works provides regularization path algorithms for models with only one regularization parameter. Recently, [8] proposed algorithm to obtain regularization path for models with more than one regularization parameters.

5.3 Structured SVM

A detailed introduction to SSVM, related inference algorithms, and the optimization algorithms is given in the chapter 2. In this chapter we mainly focus on the Frank Wolfe (FW) and Block Coordinate Frank Wolfe (BCFW) algorithms for the optimization of the SSVM objective function and show how these algorithms can be used to obtain the regularization path for SSVM. We briefly define the objective function of SSVM, its dual, and the optimization algorithms (FW and BCFW), in order to make the reader familiar with the notations and terminologies which will be useful in the remaining sections of the chapter.

Notations. Let $\mathbf{x} \in \mathscr{X}$ (patterns or vectors) and $\mathbf{y} \in \mathscr{Y}$ (strings, graphs, trees, or sequences) be the structured input and the structured output, respectively. The dataset $\mathscr{D} = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \cdots, n\}$ represents n training samples, each with an input-output pair. The parameter vector of the SSVM is represented as $\mathbf{w}$ and the joint feature map as $\Phi(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^d$. The loss function is represented as $\Delta(\cdot, \cdot) \in \mathbb{R}_{\geq 0}$. As seen in chapter 2, the dual variable vector of the SSVM is the Cartesian product of the dual variable vector corresponding to each input-output pair (or block). Thus, the dual variable vector can be represented as $\alpha = (\alpha_1, \cdots, \alpha_n) \in \mathbb{R}^{|\mathscr{Y}_1|} \times \cdots \mathbb{R}^{|\mathscr{Y}_n|} \in \mathbb{R}^m$, where, $\alpha_i \in \mathbb{R}^{|\mathscr{Y}_i|}$ is the dual vector corresponding to the i-th block (or i-th input-output pair). The term $\alpha_i(\mathbf{y})$ represents a particular element of the dual variable vector α_i corresponding to the output $\mathbf{y}$. Similarly, $\mathbf{w}_i$ denotes the learning parameter associated with the i-th block. The terms ℓ and ℓ_i represents the average loss over the entire dataset and the loss corresponding to the i-th block, respectively. The notation $\alpha_{[i]}$ represents the dual variable vector with zeros everywhere except for the indices corresponding to the i-th block. Notice that the vectors α and $\alpha_{[i]}$ are of the same dimension.

5.3.1 Objective Function

An SSVM, parametrized by $\mathbf{w}$, provides a linear *prediction* rule to obtain a structured output $\mathbf{y} \in \mathscr{Y}$ for a given input $\mathbf{x} \in \mathscr{X}$. The prediction for a given input $\mathbf{x}$ is obtained by maximizing the score over all possible outputs, that is, $\hat{\mathbf{y}} = \operatorname{argmax}_{\mathbf{y} \in \mathscr{Y}} \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{y})$. In order to *learn* the parameters $\mathbf{w}$, SSVM minimizes an objective function that models the trade-off between the model complexity and an upperbound on the empirical risk. The risk is measured using a user-specified loss function $\Delta(\cdot,\cdot) \in \mathbb{R}_{\geq 0}$. In more detail, given a training dataset $\mathscr{D}$, the parameters of SSVM are estimated by solving the following convex optimization problem:

$$\begin{aligned} \min_{\mathbf{w},\xi} \quad & \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n}\xi_i && (5.3) \\ \text{s.t.} \quad & \mathbf{w}^\top \Psi(\mathbf{x}_i,\mathbf{y}) \geq \Delta(\mathbf{y}_i,\mathbf{y}) - \xi_i, \quad \forall i, \forall \mathbf{y} \in \mathscr{Y}_i. && (5.4) \end{aligned}$$

where, $\Psi(\mathbf{x}_i,\mathbf{y}) = \Phi(\mathbf{x}_i,\mathbf{y}_i) - \Phi(\mathbf{x}_i,\mathbf{y})$, and $\Delta(\mathbf{y}_i,\mathbf{y}) \in \mathbb{R}_{\geq 0}$ is the loss function that quantitatively measures the difference between the prediction $\mathbf{y}$ and the ground truth $\mathbf{y}_i$. The regularization parameter (or the hyperparameter) λ controls the trade-off between the training error and the model complexity.

The Lagrange Dual. The derivation of the Lagrange dual of SSVM is shown in the section 2.4 of chapter 2. Below we briefly talk about it for building the intuition. Notice that the primal objective function (5.3) has $|\mathscr{Y}_i|$ constraints per sample. In total there are $m = \sum_i |\mathscr{Y}_i|$ constraints. Relaxing each constraint using the dual variable $\alpha_i(\mathbf{y})$ leads to the following form of the dual objective function:

$$\begin{aligned} \min_{\alpha} \quad & f(\alpha) := \frac{\lambda}{2}\|A\alpha\|^2 - \mathbf{b}^\top \alpha && (5.5) \\ s.t. \quad & \sum_{\mathbf{y}\in\mathscr{Y}_i} \alpha_i(\mathbf{y}) = 1, \forall i \in [n], \\ & \alpha_i(\mathbf{y}) \geq 0, \forall i \in [n], \forall \mathbf{y} \in \mathscr{Y}_i. \end{aligned}$$

where, $\alpha = (\alpha_1,\cdots,\alpha_n) \in \mathbb{R}^{|\mathscr{Y}_1|} \times \cdots \mathbb{R}^{|\mathscr{Y}_n|} \in \mathbb{R}^m$ is the dual variable vector, $A \in \mathbb{R}^{d\times m}$ is a matrix with column entries as $\frac{\Psi(\mathbf{x}_i,\mathbf{y})}{\lambda n}$, and $\mathbf{b} \in \mathbb{R}^m$ is a vector with elements as $\frac{\Delta(\mathbf{y}_i,\mathbf{y})}{n}$, as shown below:

$$A = \frac{1}{\lambda n}\begin{pmatrix} & | & \\ \cdots & \Psi(\mathbf{x}_i,\mathbf{y}) & \cdots \\ & | & \end{pmatrix} \in \mathbb{R}^{d\times m}, \quad \mathbf{b} = \frac{1}{n}\begin{pmatrix} \vdots \\ \Delta(\mathbf{y}_i,\mathbf{y}) \\ \vdots \end{pmatrix} \in \mathbb{R}^m. \tag{5.6}$$

From the first order KKT condition of the Lagrangian of the objective function (5.3) (please refer to the section 2.4.2 for details) we obtain the following relationship between the primal and the dual variables:

$$\mathbf{w} = \sum_{i\in[n],\mathbf{y}\in\mathscr{Y}_i} \frac{1}{n\lambda}\alpha_i(\mathbf{y})\Psi(\mathbf{x}_i,\mathbf{y}) = A\alpha. \tag{5.7}$$

The above relationship between the primal and the dual variables is very important. It allows us to work on the smooth dual objective with compact domain without explicitly manipulating the very high dimensional dual variable vector α.

5.3.2 Optimization of SSVM using FW and BCFW

Different methods for the optimization of the SSVM have been discussed in the section 2.4. In this chapter we are mainly interested in the FW and its variants for the optimization of SSVM. An overview of the Frank-Wolfe (FW) algorithm [43, 59, 60] is given in the Appendix B.3. The general FW algorithm is shown in the Algorithm 19 in the Appendix B.3. The FW and BCFW algorithm for the optimization of SSVM have been discussed in detail in section 2.4. To summarize, the FW algorithm is used to optimize a continuously differentiable convex function over a convex and compact domain. In the context of the optimization of the SSVM, the FW algorithm based optimization approach can be divided in to following four steps: (1) solving the linearization problem; (2) convex combination to update the variables; (3) obtaining the optimal step size; and (4) computing the duality gap. Recall that in FW algorithm, the linearization duality gap (or the Fenchel duality gap) is obtained as a by product without additional computation. In case of SSVM, the linearization duality gap and the Lagrange duality gap turns out to be exactly the same, therefore, can be used as the convergence criterion. The standard FW algorithm and the BCFW algorithm for the optimization of SSVM is given in chapter 2, Algorithm 4 and Algorithm 5, respectively. In this chapter we modify these algorithms from implementation point of view (no mathematical modification) which will be useful in following sections. These modified and more general algorithms are given in the Algorithm 12 and Algorithm 13.

Briefly, the BCFW algorithm is an online and faster variant of the FW algorithm. Recall that each sample forms a simplex in the dual of the SSVM. These simplices are independent to each other. Based on this insight, the BCFW algorithm stores the parameter vector $\mathbf{w}_i$ and the loss ℓ_i for each i-th sample and update them individually. The final parameter vector $\mathbf{w}$ and the loss ℓ are approximately updated using the parameter vector $\mathbf{w}_i$ and the loss l_i corresponding to a single sample. Thus, it requires only one 'max-oracle' call for each update of the parameter vector. This is much cheaper than the parameter updates of the FW algorithm which requires us to call the ‘max-oracle’ procedure for all the n samples in the dataset.

Algorithm 12 The Frank-Wolfe algorithm for SSVM.

```
input $\mathcal{D}$, K, $\lambda$, tolerance $\varepsilon$, initialization $\{\mathbf{w}^0, l^0\}$, .
If no initialization is given $\mathbf{w}^0 = 0$, $l^0 = 0$.
$k \leftarrow 0$
repeat
    Solve the ‘max-oracle’ problem for all the samples:
    for $i = 1, \cdots, n$ do
        $\mathbf{y}_i^* := \text{argmax}_{\mathbf{y}\in\mathcal{Y}_i} H_i(\mathbf{y};\mathbf{w}^k)$, $\mathbf{w}_i^s = \frac{1}{\lambda n}\Psi(\mathbf{x}_i,\mathbf{y}_i^*)$, $\ell_i^s = \frac{1}{n}\Delta(\mathbf{y}_i,\mathbf{y}_i^*)$
    end for
    $\mathbf{w}^s = \sum_i^n \mathbf{w}_i^s$ and $\ell^s = \sum_i^n \ell_i^s$.
    Compute gap: $g(\alpha^k;\lambda) = \lambda(\mathbf{w}^k - \mathbf{w}^s)^\top \mathbf{w}^k - (\ell^k - \ell^s)$.
    if $g(\alpha^k;\lambda) \le \varepsilon$ or $k \ge K$ then
        Converged
    else
        Find the optimal step size $\gamma = \frac{g(\alpha^k;\lambda)}{\lambda \|\mathbf{w}^k - \mathbf{w}^s\|^2}$ and clip to $[0,1]$.
        Update: $\mathbf{w}^{k+1} = \gamma\mathbf{w}^s + (1-\gamma)\mathbf{w}^k$, and $\ell^{k+1} = \gamma\ell^s + (1-\gamma)\ell^k$.
        $k \leftarrow k+1$
    end if
until Converged.
return $\mathbf{w}^k$, $\{\mathbf{w}_i^s\}_{i\in[n]}$, $\{\ell_i^s\}_{i\in[n]}$, $g(\alpha^k;\lambda)$
```

5.4 Variants of BCFW Algorithm

In this section we talk about different variants of the BCFW algorithm. The variants are intuitive yet faster compared to the standard BCFW algorithm for the optimization of the SSVM dual objective function. The first variant is the non uniform sampling based BCFW. More specifically, BCFW *with gap based sampling* [92] as shown in the Algorithm 13 (please note the sampling step of the algorithm). Recall that the standard BCFW [86] uniformly picks a block at each iteration. The block we choose decides the descent direction of the algorithm. Thus, intuitively, choosing a good descent direction might result in faster convergence. Based on this insight, different sampling methods for randomized algorithms were explored in some recent works [99, 100, 139]. The main idea of those works is to use Lipschitz constants of the gradients to pick more often functions for which gradients change quickly. This approach presents two main drawbacks. First, Lipschitz constants are most of the time unknown and one has to propose heuristic to estimate them. Second, such schemes are not adaptive.

Algorithm 13 The general BCFW algorithm for SSVM.

```
input $\mathscr{D}$, $K$, $\lambda$, $p$, tolerance $\varepsilon$, initialization $\{\mathbf{w}^0, l^0, \mathbf{w}_i^0, l_i^0, \tilde{g}\}$, where $\tilde{g} \in \mathbb{R}^n$ is the gap vector and $\tilde{g}(i)$ represents the gap corresponding to the $i$-th block.
If no initialization is given $\mathbf{w}^0 = \mathbf{w}_i^0 = 0$, $l^0 = l_i^0 = 0$, $\tilde{g}(i) = \infty, \forall i$.
for $k = 0, \cdots, K$ do
    if gap based sampling then
        Pick $i$ at random in $\{1, \cdots, n\}$ based on the probabilities proportional to the block-wise gaps $\{\tilde{g}(1), \cdots, \tilde{g}(n)\}$.
    else
        Pick $i$ at random in $\{1, \cdots, n\}$
    end if
    Solve the 'max-oracle' problem for the $i$-th sample: $\mathbf{y}_i^* := \text{argmax}_{\mathbf{y} \in \mathscr{Y}_i} H_i(\mathbf{y}; \mathbf{w}^k)$.
    $\mathbf{w}_i^s = \frac{1}{\lambda n} \Psi(\mathbf{x}_i, \mathbf{y}_i^*)$ and $l_i^s = \frac{1}{n} \Delta(\mathbf{y}_i, \mathbf{y}_i^*)$.
    Update the gap for $i$-th block $\tilde{g}(i) = g^i(\alpha^k; \lambda)$, where $g^i(\alpha^k; \lambda) = \lambda (\mathbf{w}_i^k - \mathbf{w}_i^s)^\top \mathbf{w}^k - (\ell_i^k - \ell_i^s)$.
    Find the optimal step size $\gamma = \frac{\tilde{g}(i)}{\lambda ||\mathbf{w}_i^k - \mathbf{w}_i^s||^2}$ and clip to $[0, 1]$.
    Update: $\mathbf{w}_i^{k+1} = \gamma \mathbf{w}_i^s + (1-\gamma) \mathbf{w}_i^k$, and $l_i^{k+1} = \gamma l_i^s + (1-\gamma) l_i^k$.
    Update: $\mathbf{w}^{k+1} = \mathbf{w}^k + \mathbf{w}_i^{k+1} - \mathbf{w}_i^k$, and $l^{k+1} = l^k + l_i^{k+1} - l_i^k$.
    Convergence criteria
        • Iteration based: Use upper limit ($K$) on the number of iterations ($k$).
        • Exact: Obtain the linearization duality gap $g(\alpha^k; \lambda) = \lambda (\mathbf{w}^k - \mathbf{w}^s)^\top \mathbf{w}^k - (\ell^k - \ell^s)$ by making a full pass on the data (similar to the FW algorithm) after every $p$ updates of the parameter vector $\mathbf{w}$.
        • Heuristic: Use approximate duality gap $g_{app} = \sum_i \tilde{g}(i)$ for the convergence criteria. This is possible after all the samples have been visited at least once.
end for
return $\mathbf{w}^{k+1}$, $\ell^{k+1}$, $\{\mathbf{w}_i^{k+1}\}_{i \in [n]}$, $\{\ell_i^{k+1}\}_{i \in [n]}$, $\tilde{g}$
```

More precisely, the criterion for choosing a function to optimize does not change with the optimization. Since our goal is to minimize the duality gap, therefore, it make sense to pick a block with high value of the duality gap. The hope thus is that if we choose blocks with high duality gaps, we may successfully reach the minima faster. This is exactly what a gap based BCFW algorithm is based on. Instead of choosing the blocks uniformly in random, the blocks are chosen based on the block wise duality gaps. Note the sampling step in Algorithm 13.

Another variant of the BCFW algorithm is the combination of the BCFW and FW algorithms. We call it the *hybrid* BCFW *algorithm*, shown in the Algorithm 14. Recall that, in case of BCFW, the block wise gaps can not lead to the exact duality gap since after each update of the parameters based on the selected block, the block wise gaps of other blocks changes. In order to guarantee duality gap based convergence of the BCFW algorithm, we must compute the exact duality gap using the current set of parameters. Computing exact duality gap requires us to call the ‘max-oracle’ procedure for all the samples in the dataset, which is computationally expensive. However, computing the exact duality gap is similar to one pass of the FW algorithm (Algorithm 12). Therefore, instead of using this step only to compute the exact duality gap, as is done in the standard BCFW algorithm, we update the parameters if the exact duality gap is higher than the given threshold. Thus, instead of wasting the computation, the hybrid algorithm uses it to further update the parameters. Hence, it exploits the best of both BCFW and FW algorithms.

The last variant that we talk about in this chapter is the BCFW algorithm with *heuristic based convergence*. The idea is to use the approximate block wise duality gaps to decide the convergence as shown in the convergence step of the Algorithm 13. This convergence criterion does not guarantee optimality. However, in practice, approximate duality gap based convergence of BCFW is much faster than the exact duality gap based convergence, and it leads to highly promising results (will be seen in the experiments section).

5.5 SSVM with box constraints (SSVM-B)

There are several important problems in which additional inequality constraints are required over $\mathbf{w}$ in order to solve the inference problem exactly. For example, if there are two labels and the output structure forms a graph with loops, then if the pairwise potentials are submodular, graph cuts [74] can be used to solve the inference problem exactly. In order to ensure that the pairwise potentials are submodular, additional constraints (for example, $\mathbf{w} \leq 0$, or $\mathbf{w} \geq 0$) are normally used in the objective function of the SSVM. One example of such objective function can be seen in [119] where the parameters are learned for the task of image segmentation. Below we present the SSVM objective function with additional box

Algorithm 14 The hybrid BCFW algorithm for SSVM.

```
input $\mathscr{D}$, $K_1$, $K_2$, $\lambda$, tolerance $\varepsilon$, initialization $\{\mathbf{w}^0, l^0, \mathbf{w}_i^0, l_i^0, \tilde{g}\}$, where $\tilde{g} \in \mathbb{R}^n$ is the gap vector and $\tilde{g}(i)$ represents the gap corresponding to the $i$-th block.
If no initialization is given $\mathbf{w}^0 = \mathbf{w}_i^0 = 0$, $l^0 = l_i^0 = 0$, $\tilde{g}(i) = \infty, \forall i$.
$k \leftarrow 0$
repeat
    $(\mathbf{w}^k, \mathbf{w}_i^k, \ell^k, \ell_i^k) \leftarrow$ BCFW ( $\mathscr{D}, \lambda, \mathbf{w}^k, \mathbf{w}_i^k, \ell^k, \ell_i^k, \tilde{g}, K = K_2$)
    $(\mathbf{w}^k, \{\mathbf{w}_i^s\}_{i\in[n]}, \{\ell_i^s\}_{i\in[n]}, gap\ ) \leftarrow$ FW ($\mathscr{D}, \mathbf{w}^k, \ell^k, K = 0$)
    if $gap \leq \varepsilon$ or $k \geq K_2$ then
        Converged
    else
        Find the optimal step size $\gamma_{opt} = \frac{gap}{\lambda\|\mathbf{w}^k - \mathbf{w}^s\|^2}$ and clip to $[0, 1]$.
        Compute $\mathbf{w}^s = \sum_i^n \mathbf{w}_i^s$ and $\ell^s = \sum_i^n \ell_i^s$.
        Update gaps for all blocks: $\tilde{g}(i) = g^i(\alpha^k;\lambda)$, where $g^i(\alpha^k;\lambda) = \lambda(\mathbf{w}_i^k - \mathbf{w}_i^s)^\top \mathbf{w}^k - (\ell_i^k - \ell_i^s)$.
        Update: $\mathbf{w}^{k+1} = \gamma \mathbf{w}^s + (1-\gamma)\mathbf{w}^k$, and $\ell^{k+1} = \gamma \ell^s + (1-\gamma)\ell^k$.
        Update: $\mathbf{w}_i^{k+1} = \gamma \mathbf{w}_i^s + (1-\gamma)\mathbf{w}_i^k$, and $l_i^{k+1} = \gamma l_i^s + (1-\gamma) l_i^k$, for all $i \in [n]$.
        $k \leftarrow k+1$
    end if
until Converged.
return $\mathbf{w}^{k+1}$, $\ell^{k+1}$, $\{\mathbf{w}_i^{k+1}\}_{i\in[n]}$, $\{\ell_i^{k+1}\}_{i\in[n]}$, $\tilde{g}$
```

constraints (SSVM-B). We also present its dual and the corresponding optimization algorithm using BCFW (Algorithm 15).

5.5.1 Objective Function

The primal objective function of SSVM with additional box constraints is:

$$\min_{\mathbf{w},\xi} \quad \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n} \xi_i \tag{5.8}$$

$$\text{s.t.} \quad \mathbf{w}^\top \Psi(\mathbf{x}_i, \mathbf{y}) \geq \Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i, \quad \forall i, \forall \mathbf{y} \in \mathscr{Y}_i, \tag{5.9}$$

$$\mathbf{l} \preceq \mathbf{w} \preceq \mathbf{u}. \tag{5.10}$$

where, **u** and **l** are the upper and lower bounds, respectively.

The Lagrange Dual. The Lagrangian of the problem (5.8) is:

$$L(\mathbf{w},\xi,\alpha,\beta_{\mathbf{l}},\beta_{\mathbf{u}}) = \frac{\lambda}{2}\langle \mathbf{w},\mathbf{w}\rangle + \frac{1}{n}\sum_{i=1}^{n}\xi_i + \sum_{i\in[n],\mathbf{y}\in\mathscr{Y}_i}\frac{1}{n}\alpha_i(\mathbf{y})\left(-\xi_i + \langle \mathbf{w}, -\Psi_i(\mathbf{y})\rangle + \Delta(\mathbf{y},\mathbf{y}_i)\right)$$

$$\lambda\langle \beta_u, \mathbf{w}-\mathbf{u}\rangle + \lambda\langle \beta_l, -\mathbf{w}+\mathbf{l}\rangle \tag{5.11}$$

where, β_l and β_u are the additional dual variables corresponding to the box constraints. Using the first order KKT condition over $\mathbf{w}$ we get the following expression:

$$\lambda\mathbf{w} = \sum_{i\in[n],\mathbf{y}\in\mathscr{Y}_i}\frac{1}{n}\alpha_i(\mathbf{y})\Psi_i(\mathbf{y}) - \lambda(\beta_u - \beta_l)$$

$$\implies \mathbf{w} = V(\alpha) - (\beta_u - \beta_l). \tag{5.12}$$

where, $V(\alpha) = A\alpha$, $A := \left\{\frac{1}{\lambda n}\Psi_i(\mathbf{y}) \in \mathbb{R}^d \;\middle|\; i\in[n], \mathbf{y}\in\mathscr{Y}_i\right\}$. Similarly, using $\nabla_{\xi_i}L(\mathbf{w},\boldsymbol{\xi},\alpha,\beta_l,\beta_u) = 0$, we get:

$$\sum_{\mathbf{y}\in\mathscr{Y}_i}\alpha_i(\mathbf{y}) = 1 \quad \forall i\in[n]\,. \tag{5.13}$$

Plugging equations (5.12) and (5.13) into the Lagrangian (5.11), we obtain the Lagrange dual problem as follows:

$$\begin{aligned}
\max_{\alpha,\beta_u,\beta_l}\; & -\frac{\lambda}{2}\|V(\alpha)-(\beta_u-\beta_l)\|^2 + \mathbf{b}^\top\alpha - \lambda(\beta_{\mathbf{u}}^\top\mathbf{u} - \beta_{\mathbf{l}}^\top\mathbf{l}) \qquad (5.14)\\
\text{s.t. } & \sum_{\mathbf{y}\in\mathscr{Y}}\alpha_i(\mathbf{y}) = 1 \quad \forall i\in[n],\\
& \alpha_i(\mathbf{y}) \ge 0 \quad \forall i\in[n], \forall\mathbf{y}\in\mathscr{Y}_i\\
& \beta_u \ge 0, \beta_l \ge 0.
\end{aligned}$$

where, $\mathbf{b}\in\mathbb{R}^m$ is given by $\mathbf{b} := \left(\frac{1}{n}\Delta(\mathbf{y},\mathbf{y}_i)\right)_{i\in[n],\mathbf{y}\in\mathscr{Y}_i}$. This is equivalent to solving the following minimization problem:

$$\begin{aligned}
\min_{\alpha,\beta_u,\beta_l} f(\alpha,\beta_l,\beta_u) :=\; & \frac{\lambda}{2}\|V(\alpha)-(\beta_u-\beta_l)\|^2 - \mathbf{b}^\top\alpha + \lambda(\beta_u^\top\mathbf{u} - \beta_l^\top\mathbf{l}) \qquad (5.15)\\
\text{s.t. } & \sum_{\mathbf{y}\in\mathscr{Y}}\alpha_i(\mathbf{y}) = 1 \quad \forall i\in[n],\\
& \alpha_i(\mathbf{y}) \ge 0 \quad \forall i\in[n], \forall\mathbf{y}\in\mathscr{Y}_i\\
& \beta_u \ge 0, \beta_l \ge 0.
\end{aligned}$$

5.5.2 Optimization of SSVM-B

Ideally, we should optimize $f(\alpha,\beta_l,\beta_u)$ jointly over all the dual variables. This can be done using the cutting plane algorithm [61] explained in section 2.4. However, FW or its variants can not be used to optimize it jointly over all the dual variables. This is because of the fact that the constraints of the problem (5.15) do not form a compact domain because of the additional constraints over the dual variables corresponding to the box constraints ($\beta_l \geq \mathbf{0}$ and $\beta_u \geq \mathbf{0}$). However, we have already seen that the domain is compact over the α. Therefore, we resort to optimizing in a block-wise fashion where the blocks consists of α, β_u, and β_l, respectively.

5.5.2.1 Optimizing blocks β_u and β_l

While optimizing the objective function (5.15) with respect to β_u, the variables β_l and α are kept constant. The optimization can be performed by putting $\nabla_{\beta_u} f(\alpha,\beta_l,\beta_u) = 0$ and projecting the solution in to feasible set of $\beta_u \geq 0$ as follows:

$$\beta_u = [V(\alpha) + \beta_l - \mathbf{u}]_+ \tag{5.16}$$

where, β_u is the optimal solution and $[.]_+$ is the truncation function. In case of scalar $[a]_+ = 0$ if $a < 0$. In case of vector $\mathbf{a}$, the operation $[\mathbf{a}]_+$ is the element wise truncation. Similarly, the optimal β_l can be obtained as follows:

$$\beta_l = [-V(\alpha) + \beta_u + \mathbf{l}]_+ \tag{5.17}$$

Let us say that $\beta_u(i)$ and $\beta_l(i)$ denotes the i-th index of the optimal dual variables β_u and β_l respectively. Notice that, when both β_u and β_l are optimal, for any index i, $\beta_u(i)$ and $\beta_l(i)$ can not be non-zero simultaneously. This is because of the fact that if any one of the constraint is violated (either the upperbound or the lowerbound) for that particular index then the other constraint must be satisfied. Hence, if $\beta_u(i) \neq 0$, it implies $\beta_l(i) = 0$, vice versa. Therefore, the final update equations can be written as:

$$\beta_u = [V(\alpha) - \mathbf{u}]_+ \tag{5.18}$$

$$\beta_l = [-V(\alpha) + \mathbf{l}]_+ \tag{5.19}$$

5.5.2.2 Optimizing block α

Given the optimal α, $\beta_u(i)$ and $\beta_l(i)$, the optimal primal variable can be obtained using the KKT condition (equation 5.12) as follows:

$$\mathbf{w} = V(\alpha) - (\beta_u - \beta_l) = V(\alpha) - ([V(\alpha) - \mathbf{u}]_- - [-V(\alpha) + \mathbf{l}]_+) \tag{5.20}$$

Notice that the above operation is simply the projection of the vector $V(\alpha) = A\alpha$ into the feasible space corresponding to the box constraints. This projection can be performed very cheaply as follows.

$$\mathbf{w}(i) = [V_i(\alpha)]_B = \begin{cases} \mathbf{l}(i), & \text{if } V_i(\alpha) \leq \mathbf{l}(i) \\ \mathbf{u}(i), & \text{if } V_i(\alpha) \geq \mathbf{u}(i) \\ V_i(\alpha), & \text{otherwise.} \end{cases} \tag{5.21}$$

where, $\mathbf{w}(i)$ and $V_i(\alpha)$ are the i-th index of the vectors $V(\alpha)$ and $\mathbf{w}$ respectively. The projection of $V(\alpha)$ into the box constraint ($[V(\alpha)]_B$) is performed element wise in the similar manner. In what follows, we talk about the algorithm to obtain the optimal α.

In order to optimize over the block α we use the FW and the BCFW algorithm. As discussed previously, there are four major steps for the FW algorithm – (1) solving the linearization problem; (2) convex combination to update the variables; (3) obtain the optimal step size; and (4) compute the duality gap. Below we present these steps for the optimization of the SSVM-B dual.

Solving the linearization problem. The linearization problem is defined as:

$$\mathbf{s} = \underset{\mathbf{s}' \in \mathscr{D}}{\operatorname{argmin}} \langle \mathbf{s}', \nabla_\alpha f(\alpha, \beta_l, \beta_u) \rangle \tag{5.22}$$

where,

$$\nabla_\alpha f(\alpha, \beta_l, \beta_u) = \lambda A^\top (V(\alpha) - (\beta_u - \beta_l)) - \mathbf{b} \tag{5.23}$$

Following the similar arguments as given in the section 2.4.3.3, solving the linearization problem in the dual is the same as solving the 'max-oracle' problem in the primal. The only difference is that the primal variable in case of SSVM-B is $\mathbf{w} = [V(\alpha)]_B = V(\alpha) - (\beta_u - \beta_l)$, which is the projection of $V(\alpha)$ over the box constraints. Therefore, the linearization problem

Algorithm 15 BCFW for SSVM-B

```
input Given $\mathbf{u}$, $\mathbf{l}$, $K$
Initialization $V(\alpha^0) := V(\alpha^0_{[i]}) := \mathbf{0}$, $\ell^{(0)} := \ell_i{}^{(0)} := 0$ (Notice that, we assume that the zero vector is a feasible solution.)
for $k = 0 \dots K$ do
    Pick $i$ at random in $\{1, \dots, n\}$
    $\mathbf{w}^k = [V(\alpha^k)]_B$.
    Solve $\mathbf{y}_i^* := \underset{\mathbf{y} \in \mathcal{Y}_i}{\text{argmax}}\ H_i(\mathbf{y}; \mathbf{w}^k)$
    Let $V(\mathbf{s}_{[i]}) := \frac{1}{\lambda n}\Psi_i(\mathbf{y}_i^*)$ and $\ell_i^{(\mathbf{s})} := \frac{1}{n}\Delta(\mathbf{y}_i^*, \mathbf{y}_i)$
    Let $g^i_{lin} = \lambda\big(V(\alpha^k_{[i]}) - V(\mathbf{s}_{[i]})\big)^\top \mathbf{w}^k - (\ell_i^k - \ell_i^{(\mathbf{s})})$
    Let $\gamma := \frac{g^i_{lin}}{\lambda \left\| V(\alpha^k_{[i]}) - V(\mathbf{s}_{[i]}) \right\|^2}$ and clip to $[0,1]$
    Update $V(\alpha^{(k+1)}_{[i]}) := (1-\gamma)V(\alpha^{(k)}_{[i]}) + \gamma V(\alpha^{\mathbf{s}}_{[i]})$
        and $\ell_i{}^{(k+1)} := (1-\gamma)\ell_i{}^{(k)} + \gamma \ell_{\mathbf{s}}$
    Update $V(\alpha^{(k+1)}) := V(\alpha^k) + V(\alpha^{(k+1)}_{[i]}) - V(\alpha^k_{[i]})$
        and $\ell^{(k+1)} := \ell^{(k)} + \ell_i{}^{(k+1)} - \ell_i{}^{(k)}$
end for
output $\mathbf{w}^k = [V(\alpha^k)]_B$.
```

can be solved as follows:

$$\mathbf{y}_i^* = \underset{\mathbf{y}}{\text{argmax}}\, H_i(\mathbf{y}; \mathbf{w}) \tag{5.24}$$

Convex combination. Let γ be the optimal step size, α^k be the current solution, and $\mathbf{s}$ be the corner obtained by solving the linearization problem. In case of FW algorithm, the dual variable can be updated as follows:

$$\begin{aligned} \alpha^{k+1} &= \gamma \mathbf{s} + (1-\gamma)\alpha^k \\ A\alpha^{k+1} &= \gamma A\mathbf{s} + (1-\gamma)A\alpha^k \\ V(\alpha^{k+1}) &= \gamma V(\mathbf{s}) + (1-\gamma)V(\alpha^k) \end{aligned} \tag{5.25}$$

Similarly, in case of FW algorithm, the loss can be updated as follows:

$$\begin{aligned} \ell^{k+1} =& \mathbf{b}^\top \alpha^{k+1} = \mathbf{b}^\top(\gamma \mathbf{s} + (1-\gamma)\alpha^k) \\ =& \gamma \ell^s + (1-\gamma)\ell^k \end{aligned} \tag{5.26}$$

where, $\ell^s = \frac{1}{n}\sum_i^n \Delta(\mathbf{y}_i^*, \mathbf{y}_i)$. In case of BCFW algorithm, since we update only one block, therefore, the variables corresponding to the other blocks remains unchanged. Hence,

$(\alpha - \mathbf{s}) = (\alpha_{[i]} - \mathbf{s}_{[i]})$, where, $\mathbf{a}_{[i]}$ denotes the vector with zero everywhere except for the i-th block. Therefore,

$$\begin{aligned} \alpha^{k+1} &= \alpha^k + \gamma(\mathbf{s}_{[i]} - \alpha^k_{[i]}) \\ V(\alpha^{k+1}) &= V(\alpha^k) + \gamma(V(\mathbf{s}_{[i]}) - V(\alpha^k_{[i]})) \end{aligned} \tag{5.27}$$

Now, since $\alpha^{k+1}_{[i]} = \alpha^k_{[i]} + \gamma(\mathbf{s}_{[i]} - \alpha^k_{[i]})$, therefore, using $V(\alpha^{k+1}_{[i]}) = V(\alpha^k_{[i]}) + \gamma(V(\mathbf{s}_{[i]}) - V(\alpha^k_{[i]}))$ in the above equation we obtain

$$V(\alpha^{k+1}) = V(\alpha^k) + V(\alpha^{k+1}_{[i]}) - V(\alpha^k_{[i]}) \tag{5.28}$$

Similarly, the loss can be updated as

$$\ell^{k+1} = \ell^k + \ell^{k+1}_i - \ell^k_i \tag{5.29}$$

where, $\ell^{k+1}_i = \ell^k_i + \gamma(\ell^s_i - \ell^k_i)$ and $\ell^s_i = \frac{1}{n}\Delta(\mathbf{y}^*_i, \mathbf{y}_i)$.

Lagrange duality gap and the Linearization duality gap. Let us first derive the expression for the linearlization duality gap. The linearization duality gap is defined as (refer to Appendix B.3 for details):

$$g_{lin}(\alpha;\lambda) := \max_{\mathbf{s}' \in \mathcal{M}} \langle \alpha - \mathbf{s}', \nabla_\alpha f(\alpha, \beta_l, \beta_u) \rangle = \langle \alpha - \mathbf{s}, \nabla_\alpha f(\alpha, \beta_l, \beta_u) \rangle \tag{5.30}$$

where, $\mathbf{s} = \operatorname{argmin}_{\mathbf{s}' \in \mathcal{D}} \langle \mathbf{s}', \nabla_\alpha f(\alpha, \beta_l, \beta_u) \rangle$. Recall that the problem of obtaining $\mathbf{s}$ is exactly the same as solving the linearlization problem, which we have already discussed. Using the expression for the gradient (equation (5.23)) we obtain the following expression for the linearization duality gap:

$$g_{lin}(\alpha;\lambda) = \lambda \big(A(\alpha - \mathbf{s})\big)^\top \big(V(\alpha) - (\beta_u - \beta_l)\big) - (\alpha - \mathbf{s})^\top \mathbf{b} \tag{5.31}$$

In case of FW algorithm, the final expression can be written as:

$$g_{lin}(\alpha;\lambda) = \lambda \big(V(\alpha) - V(\mathbf{s})\big)^\top [V(\alpha)]_B - (\ell - \ell^s) \tag{5.32}$$

In case of BCFW, using $(\alpha - \mathbf{s}) = (\alpha_{[i]} - \mathbf{s}_{[i]})$, the linearization gap for the i-th block can be written as:

$$g^i_{lin}(\alpha;\lambda) = \lambda \big(V(\alpha_{[i]}) - V(\mathbf{s}_{[i]})\big)^\top [V(\alpha)]_B - (\ell_i - \ell^s_i) \tag{5.33}$$

Notice that, $g_{lin}(\alpha;\lambda) = \sum_i g^i_{lin}(\alpha;\lambda)$. For the convergence criterion, we are interested in the Lagrange duality gap. As already discussed, in case of SSVM without the box constraints, the Lagrange duality gap and the linearization duality gap turns out to be exactly the same [86]. In what follows, we show that the Lagrange and the linearization duality gaps are exactly the same for the SSVM-B objective function also. The Lagrange duality gap for the SSVM-B is defined as:

$$g_{Lag}(\mathbf{w},\alpha,\beta_u,\beta_l;\lambda) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n}\max_{\mathbf{y}\in\mathscr{Y}_i} H_i(\mathbf{y};\mathbf{w}) - \Big(\mathbf{b}^\top\alpha - \frac{\lambda}{2}\|\mathbf{w}\|^2 - \lambda(\beta_u^\top\mathbf{u} - \beta_l^\top\mathbf{l})\Big) \tag{5.34}$$

Using the fact that $\frac{1}{n}\sum_{i=1}^{n}\max_{\mathbf{y}\in\mathscr{Y}_i} H_i(\mathbf{y};\mathbf{w}) = \max_{\mathbf{s}'\in\mathscr{M}}\langle\alpha-\mathbf{s}',\nabla_\alpha f(\alpha,\beta_l,\beta_u)\rangle = \langle\alpha-\mathbf{s},\nabla_\alpha f(\alpha,\beta_l,\beta_u)\rangle$ we obtain

$$g_{Lag}(\mathbf{w},\alpha,\beta_u,\beta_l;\lambda) = \langle\alpha-\mathbf{s},\nabla_\alpha f(\alpha,\beta_l,\beta_u)\rangle + \lambda(\beta_u^\top(\mathbf{u}-\mathbf{w}) - \beta_l^\top(\mathbf{l}-\mathbf{w})) \tag{5.35}$$

Noticing that at the optimal β_u and β_l with respect to α, $\beta_u^\top(\mathbf{u}-\mathbf{w}) = 0$ and $\beta_l^\top(\mathbf{l}-\mathbf{w}) = 0$, we obtain the following expression:

$$g_{Lag}(\mathbf{w},\alpha,\beta_u,\beta_l;\lambda) = \langle\alpha-\mathbf{s},\nabla_\alpha f(\alpha,\beta_l,\beta_u)\rangle = g_{lin}(\alpha;\lambda) \tag{5.36}$$

Therefore, similar to the SSVM, the Lagrange and the linearization duality gap of SSVM-B are exactly the same. In what follows we always use the linearization duality gap.

Optimal step size. Obtaining the optimal step size for the line search is equivalent to solving the problem $\gamma_{opt} := \operatorname{argmin}_{\gamma\in[0,1]} f(\alpha+\gamma(\mathbf{s}-\alpha),\beta_l,\beta_u)$, where

$$\begin{aligned} f(\alpha+\gamma(\mathbf{s}-\alpha),\beta_l,\beta_u) &= \frac{\lambda}{2}\|A(\alpha+\gamma(\mathbf{s}-\alpha)) - (\beta_u-\beta_l)\|^2 - \mathbf{b}^\top(\alpha+\gamma(\mathbf{s}-\alpha)) \\ &+ \lambda(\beta_u^\top\mathbf{u} - \beta_l^\top\mathbf{l}) \end{aligned} \tag{5.37}$$

Since our aim is to optimize the block corresponding to α and the dual of the SSVM-B is a quadratic function in α, therefore, the optimal step size can be obtained analytically by setting $\nabla_\gamma f(\alpha+\gamma(\mathbf{s}-\alpha),\beta_l,\beta_u) = 0$, which leads to the following form of the optimal step size:

$$\gamma_{opt} = \frac{\lambda\,(A(\alpha-\mathbf{s}))^\top\big(V(\alpha) - (\beta_u-\beta_l)\big) - (\alpha-\mathbf{s})^\top\mathbf{b}}{\lambda\,\|V(\alpha)-V(\mathbf{s})\|^2} \tag{5.38}$$

Notice that the numerator of the above equation is the same as the expression for the linearization duality gap as shown in equation (5.31). Therefore, in case of FW algorithm $\gamma_{opt} = \frac{g_{lin}(\alpha;\lambda)}{\lambda \|V(\alpha)-V(\mathbf{s})\|^2}$ and in case of BCFW algorithm $\gamma_{opt} = \frac{g^i_{lin}(\alpha;\lambda)}{\lambda \left\|V(\alpha_{[i]})-V(\mathbf{s}_{[i]})\right\|^2}$.

Algorithm 16 INITIALIZE-RP-SSVM: Initialization of the regularization path algorithm for SSVM.

```
input $\mathscr{D}$, A, tolerance $\varepsilon$
Initialization: $\mathbf{w} = \mathbf{w}_i = 0$, $\ell = \ell_i = 0, \tilde{\Psi} = 0$.
Compute the loss maximizer
for $i = 1, \cdots, n$ do
    $\tilde{\mathbf{y}}_i := \text{argmax}_{\mathbf{y} \in \mathscr{Y}_i} H_i(\mathbf{y}; \mathbf{w} = 0)$, $\ell_i = \frac{1}{n}\Delta(\mathbf{y}_i, \tilde{\mathbf{y}}_i)$.
    $\ell \leftarrow \ell + \ell_i$.
    $\tilde{\Psi} \leftarrow \tilde{\Psi} + \frac{1}{n}\Psi(\mathbf{x}_i, \tilde{\mathbf{y}}_i)$
end for
Compute $\theta$ for each sample (same as the inference problem)
for $i = 1, \cdots, n$ do
    $\theta(i) = \max_{\mathbf{y} \in \mathscr{Y}_i} \left( -\tilde{\Psi}^\top \Psi(\mathbf{x}_i, \mathbf{y}) \right)$
end for
Compute $\lambda = \frac{\|\tilde{\Psi}\|^2 + \frac{\sum_i \theta(i)}{n}}{A\varepsilon}$
Compute the parameters: $\mathbf{w} = \frac{\tilde{\Psi}}{\lambda}$, $\mathbf{w}_i = \frac{\Psi(\mathbf{x}_i, \tilde{\mathbf{y}}_i)}{n\lambda}$.
Compute gaps
for $i = 1, \cdots, n$ do
    $g(i) = \frac{\theta(i)}{n\lambda} + \lambda \mathbf{w}_i^\top \mathbf{w}$
    $\Omega(i) = \ell_i - \lambda \mathbf{w}_i^\top \mathbf{w}$
end for
return $\mathbf{w}$, $\mathbf{w}_i$, $\ell$, $\ell_i$, $g$, $\Omega$, $\lambda$
```

5.6 Regularization path for SSVM

As mentioned earlier, the key idea behind the regularization path algorithm is to break the regularization parameter space into segments such that learning an ε-optimal model for any value of the λ in a particular segment guarantees that the same learned model is ε-optimal for all the values of the λ in that particular segment. The regularization path algorithm to obtain the entire regularization path of SSVM is shown in Algorithm 17. Broadly speaking, the algorithm has three main parts: (1) progressively finding the breakpoints starting from $\lambda = \infty$, (2) warm starting the optimization for each segment using the solution of the previous segment (for faster convergence); and (3) finding the initialization parameters of the algorithm (Algorithm 16). Let us talk about these three parts of the algorithm in detail.

Algorithm 17 SSVM-RP: The regularization path algorithm for SSVM.

```
input $\mathscr{D}$, $\kappa_1$, $\kappa_2$, tolerance $\varepsilon$, $\lambda_{min}$.
Get the initialization parameters using Algorithm 16.
$\{\mathbf{w}^0, \mathbf{w}_i^0, \ell^0, \ell_i^0, g^0, \Omega^0, \lambda^0\} \leftarrow$ INITIALIZE-RP-SSVM $(\mathscr{D}, A, \varepsilon)$
$k \leftarrow 0$.
repeat
    Compute excess gap: $\tau = (\kappa_1 + \kappa_2)\varepsilon - \sum_i g^k(i)$.
    Compute $\mu = \sum_i^n \Omega^k(i)$.
    if $\mu > 0$ then
        Compute: $\nu = \frac{1}{1-\eta}$, where $\eta = \frac{\tau}{\mu}$
    else
        $\mathbf{w}^k$ is $\kappa_1\varepsilon$ optimal for any $\nu > 1$. Thus, we can trace the path from $\lambda = \lambda^k$ to $\lambda = 0$,
        and still guarantee to be $\kappa_1\varepsilon$ optimal with the parameter $\mathbf{w}^k$. Therefore, there is no
        further breakpoint. Hence converged.
    end if
    Update: $\lambda^{k+1} = \frac{\lambda^k}{\nu}$, $\ell^{k+1} = \frac{\ell^k}{\nu}$, $\ell_i^{k+1} = \frac{\ell_i^k}{\nu}$, $g^{k+1}(i) = g^k(i) + \big(1 - \frac{1}{\nu}\big)\Omega^k(i)$
    $(\mathbf{w}^{k+1}, \mathbf{w}_i^{k+1}, \ell^{k+1}, \ell_i^{k+1}, g^{k+1}) \leftarrow$ SSVM-OPTIMIZER ( $\mathscr{D}$, $\lambda^{k+1}$, $\mathbf{w}^k$, $\mathbf{w}_i^k$, $\ell^{k+1}$, $\ell_i^{k+1}$,
    $g^{k+1}$, $\kappa_1\varepsilon$). The optimizer can be any optimizer that returns gaps also, for example,
    BCFW hybrid. Notice that the optimality condition is $\kappa_1\varepsilon$.
    Update $\Omega$ as follows:
    for $i = 1, \cdots, n$ do
        $\Omega^{k+1}(i) = \ell_i^{k+1} - \lambda^{k+1}(\mathbf{w}_i^{k+1})^\top \mathbf{w}^{k+1}$
    end for
    $k \leftarrow k + 1$.
until $\lambda^{k+1} \leq \lambda_{min}$.
```

5.6.1 Finding the breakpoints

The linearization duality gap, which in case of SSVM is the same as the Lagrange duality gap, can be written as (refer chapter 2, equation (2.80) for proof):

$$g(\alpha;\lambda) = \frac{1}{n}\sum_i \big(\max_{\mathbf{y}\in\mathscr{Y}_i} H_i(\mathbf{y};\mathbf{w}) - \sum_{\mathbf{y}\in\mathscr{Y}_i} \alpha_i(\mathbf{y}) H_i(\mathbf{y};\mathbf{w})\big) \tag{5.39}$$

where, $H_i(\mathbf{y};\mathbf{w}) = \Delta(\mathbf{y}_i,\mathbf{y}) - \mathbf{w}^\top \Psi_i(\mathbf{y})$ is the hinge loss. From the KKT condition we obtain the following relationship between the primal and the dual variables:

$$\mathbf{w} = \sum_{i\in[n],\mathbf{y}\in\mathscr{Y}_i} \frac{1}{\lambda n}\alpha_i(\mathbf{y})\Psi_i(\mathbf{y}) \tag{5.40}$$

Let us assume that we are given a $\mathbf{w}$ which is $\kappa_1\varepsilon$ optimal for problem associated with λ_{old}, where, $\kappa_1 \in \mathbb{R}_{>0}$ is a user defined parameter. The goal is to find $\lambda_{new} (< \lambda_{old})$ such that

$\mathbf{w}$ is at least $(\kappa_1 + \kappa_2)\varepsilon$ optimal solution to the new problem associated with λ_{new}, where, $\kappa_2 \in \mathbb{R}_{>0}$ is another user defined parameter. In order to obtain a complete ε optimal path we must use κ_1 and κ_2 such that $\kappa_1 + \kappa_2 = 1$. Notice that, for ε optimal path, $\mathbf{w}$ is $\kappa_1\varepsilon(< \varepsilon)$ optimal solution to the problem associated with λ_{old}. We will discuss about the effects of choosing κ_1 and κ_2 in the later part of this section. Let us see how to decrease λ_{old} such that $\mathbf{w}$ does not change and the duality gap does not increase beyond $(\kappa_1 + \kappa_2)\varepsilon$.

For the given λ_{old}, let us say that the dual variables are α^{old} corresponding to the $\kappa_1\varepsilon$ optimal $\mathbf{w}$. Now we decrease λ_{old} such that $\frac{\alpha_i^{new}(\mathbf{y})}{\lambda_{new}} = \frac{\alpha_i^{old}(\mathbf{y})}{\lambda_{old}}, \forall \mathbf{y} \neq \mathbf{y}_i$. Therefore, $\alpha_i^{new}(\mathbf{y}) = \alpha_i^{old}(\mathbf{y})\frac{\lambda_{new}}{\lambda_{old}} < \alpha_i^{old}(\mathbf{y}), \forall \mathbf{y} \neq \mathbf{y}_i$. Hence, the values of the dual variables corresponding to the non ground truth labels decreases. This results in some extra mass since sum of the dual variables must be one (KKT condition). This extra mass is completely shifted into the dual variable corresponding to the ground truth label $\alpha_i^{new}(\mathbf{y}_i)$. Therefore, $\alpha_i^{new}(\mathbf{y}_i) = 1 - \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_i^{new}(\mathbf{y}_i)$ increases. From the KKT condition (equation 5.40), we can clearly see that this increase in mass does not change $\mathbf{w}$ since $\Psi_i(\mathbf{y}_i) = 0$. Now we need to bound the decrease in λ_{old} so that the duality gap corresponding to the problem with λ_{new} does not go beyond $(\kappa_1 + \kappa_2)\varepsilon$. The new duality gap corresponding to λ_{new} and α^{new} is:

$$g(\alpha^{new}; \lambda_{new}) = \frac{1}{n}\sum_i \big(\max_{\mathbf{y}\in\mathscr{Y}_i} H_i(\mathbf{y};\mathbf{w}) - \frac{\lambda_{new}}{\lambda_{old}} \sum_{\mathbf{y}\in\mathscr{Y}_i} \alpha_i^{old}(\mathbf{y}) H_i(\mathbf{y};\mathbf{w})\big) \tag{5.41}$$

Let $\lambda_{new} = \frac{\lambda_{old}}{\nu}, \nu > 1$, then $g(\alpha^{new}; \lambda_{new})$ can be written as:

$$\begin{aligned} g(\alpha^{new}; \lambda_{new}) &= \frac{1}{n}\sum_i \Big(\max_{\mathbf{y}\in\mathscr{Y}_i} H_i(\mathbf{y};\mathbf{w}) - \frac{1}{\nu}\sum_{\mathbf{y}\in\mathscr{Y}_i} \alpha_i^{old}(\mathbf{y}) H_i(\mathbf{y};\mathbf{w}) - \sum_{\mathbf{y}\in\mathscr{Y}_i} \alpha_i^{old}(\mathbf{y}) H_i(\mathbf{y};\mathbf{w}) \\ &\quad + \sum_{\mathbf{y}\in\mathscr{Y}_i} \alpha_i^{old}(\mathbf{y}) H_i(\mathbf{y};\mathbf{w})\Big) \\ &= g(\alpha^{old}; \lambda_{old}) + \Big(1 - \frac{1}{\nu}\Big) \underbrace{\sum_i \sum_{\mathbf{y}\in\mathscr{Y}_i} \frac{\alpha_i^{old}(\mathbf{y}) H_i(\mathbf{y};\mathbf{w})}{n}}_{\Omega(\alpha^{old}, \lambda_{old})} \end{aligned} \tag{5.42}$$

In order to obtain an $(\kappa_1 + \kappa_2)\varepsilon$ optimal path the new gap must be less than or equal to $(\kappa_1 + \kappa_2)\varepsilon$. Thus, using the shorthand notations, the above equation can be written as:

$$g(\alpha^{new}; \lambda_{new}) = \underbrace{g(\alpha^{old}; \lambda_{old})}_{\leq \kappa_1\varepsilon} + \Big(1 - \frac{1}{\nu}\Big)\Omega(\alpha^{old}, \lambda_{old}) \leq (\kappa_1 + \kappa_2)\varepsilon \tag{5.43}$$

The above inequality leads to the following inequality over ν:

$$1 \leq \nu \leq \eta \tag{5.44}$$

where,

$$\eta = \frac{1}{1 - \frac{(\kappa_1+\kappa_2)\varepsilon - g(\alpha^{old};\lambda_{old})}{\Omega(\alpha^{old},\lambda_{old})}} \tag{5.45}$$

In another words, $\nu = \max\{1, \eta\}$. Let us have a closer into the equation (5.43). If $\Omega(\alpha^{old}, \lambda_{old}) \leq 0$, it implies that for all values of $\nu > 1$, the new gap is at least $\kappa_1\varepsilon$ optimal $(g(\alpha^{new}; \lambda_{new}) \leq \kappa_1\varepsilon)$. Therefore, in this case, we can trace the path of the regularization parameter from λ_{old} to $\lambda_{new} = 0$, and still guarantee that the optimal $\mathbf{w}$ at λ_{new} is at least $\kappa_1\varepsilon$ optimal for any value of $\lambda_{new} \in [\lambda_{old}, 0]$. Similarly, if the excess gap $(\kappa_1 + \kappa_2)\varepsilon - g(\alpha^{old}; \lambda_{old}) \leq 0$, it implies that the $\mathbf{w}$ at λ_{old} is suboptimal.

In order to compute η, we should be able to compute $\Omega(\alpha^{old}, \lambda_{old})$ efficiently. Notice that, $\Omega(\alpha^{old}, \lambda_{old}) = \sum_i \sum_{\mathbf{y}\in\mathscr{Y}_i} \frac{\alpha_i^{old}(\mathbf{y})H_i(\mathbf{y};\mathbf{w})}{n} = \mathbf{b}^\top\alpha^{old} - \lambda^{old}\mathbf{w}^\top\mathbf{w} = \ell^{\alpha^{old}} - \lambda^{old}\mathbf{w}^\top\mathbf{w}$. In case of FW or its variants, $\ell^{\alpha^{old}}$ is already known (without knowing α^{old}) as it is being adaptively updated in the algorithm itself, therefore, computing $\Omega(\alpha^{old}, \lambda_{old})$ is highly efficient. Once we have computed ν, we decrease the regularization parameter from λ^{old} to $\lambda_{new} = \frac{\lambda_{old}}{\nu}$. At this point we have a new problem with the regularization parameter as λ_{new} and dual variables as α^{new}. Recall that, we modify the dual variables and the regularization parameter such that the weight vector $\mathbf{w}$ is not modified. Therefore, the parameter vector $\mathbf{w}$, which is $\kappa_1\varepsilon$ optimal for the problem associated with λ_{old} and α^{old}, is $(\kappa_1 + \kappa_2)\varepsilon$ optimal for the new problem. We need to find $\kappa_1\varepsilon$ optimal solution to this new problem so that λ_{new} can be further decreased in order to trace the complete path upto $\lambda = 0$. For faster convergence of this new problem, we can warm start the optimization algorithm with $\mathbf{w}$, which already is $(\kappa_1 + \kappa_2)\varepsilon$ optimal solution to this problem. In order to warm start with $\mathbf{w}$, we must have the loss corresponding to the α^{new} (please refer to the FW algorithm to see the initialization parameters), which can be computed as follows:

$$\ell^{\alpha^{new}} = \sum_i \sum_{\mathbf{y}\in\mathscr{Y}_i} \frac{\alpha_i^{new}(\mathbf{y})\Delta(\mathbf{y}_i, \mathbf{y})}{n} = \frac{\ell^{\alpha^{old}}}{\nu} \leq \ell^{\alpha^{old}}. \tag{5.46}$$

Effects of the parameters $\kappa_1 \in \mathbb{R}_{>0}$ and $\kappa_2 \in \mathbb{R}_{>0}$.

- If $\kappa_1 + \kappa_2 = 1$, we a regularization path which is throughout ε-optimal.

- In order to obtain ε-optimal regularization path ($\kappa_1 + \kappa_2 = 1$), we must have $\kappa_1 < 1$, therefore, each time we are at a breakpoint (λ^{new}), we have to get $\kappa_1\varepsilon(< \varepsilon)$ optimal solution instead of ε-optimal solution. This requires more number of iterations to converge. However, as will be seen in the experimental results, the warm start helps to converge the algorithm very fast.

- If $\kappa_1 + \kappa_2 > 1$, we obtain an approximate regularization path which is not ε-optimal. The approximation depends on the values of the κ_1 and κ_2.

5.6.2 Initialization

The algorithm for the initialization of the regularization path algorithm is shown in Algorithm 16. In order to start the regularization path algorithm, we must find a λ and corresponding optimal $\mathbf{w}$ such that for any regularization parameter less than or equal to λ, $\mathbf{w}$ is the $\kappa_1\varepsilon$ optimal solution. Let us start with $\lambda = \infty$. From the KKT condition, $\mathbf{w} = 0$ is one of the solutions for $\lambda = \infty$. Please note that we are not sure about the duality gap at this point (refer to the equation (5.39)). However, in what follows, we start with this solution and move to a new solution for which we can bound the regularization parameter such that the duality gap is always less than or equal to $\kappa_1\varepsilon$. In order to move to a new point (or solution) we solve the linearization problem (the 'max-oracle' problem) at $\mathbf{w} = \mathbf{0}$, which simply is the loss-maximizer as shown below:

$$\tilde{\mathcal{Y}}_i = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}_i} H_i(\mathbf{y}; \mathbf{w} = \mathbf{0}) = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}_i} \Delta(\mathbf{y}, \mathbf{y}_i) \tag{5.47}$$

where, $\tilde{\mathcal{Y}}_i$ is the set of possible outputs for which the loss can attain its maximum value, therefore, $|\tilde{\mathcal{Y}}_i| \geq 1, \forall i$. Let us say that $\tilde{\mathbf{y}}_i \in \tilde{\mathcal{Y}}_i, \forall i$, is the randomly selected loss maximizer label. Putting all the mass to this label, which means $\alpha_i(\tilde{\mathbf{y}}_i) = 1$, gives the new corner point (or new solution) as follows (using the KKT condition):

$$\mathbf{w} = \frac{1}{\lambda}\sum_i \frac{\Psi_i(\tilde{\mathbf{y}}_i)}{n} = \frac{\tilde{\Psi}}{\lambda}. \tag{5.48}$$

where, $\tilde{\Psi} = \frac{\sum_i \Psi_i(\tilde{\mathbf{y}}_i)}{n}$. For this new solution we would like to bound the duality gap and find a λ such that the $\mathbf{w} = \frac{\tilde{\Psi}}{\lambda}$ is $\kappa_1\varepsilon$ optimal. Let us assume that we know the dual variable vector α which would lead to the primal variable vector $\mathbf{w}$ using the KKT condition. Thus, the duality gap at this new solution using equation (5.39) can be defined as (notice that directly using equation (5.43) does not give a bound on λ):

$$g(\alpha;\lambda) = \frac{1}{n}\sum_i \Big(\max_{\mathbf{y}\in\mathcal{Y}_i} \big(\Delta(\mathbf{y},\mathbf{y}_i) - \mathbf{w}^\top\Psi(\mathbf{y})\big) - \Delta(\tilde{\mathbf{y}}_i,\mathbf{y}_i) + \mathbf{w}^\top\Psi(\tilde{\mathbf{y}}_i)\Big) \tag{5.49}$$

Using the inequality $\max_x(f(x)+g(x)) \leq \max_x f(x) + \max_x g(x)$, the gap can be upper-bounded as follows:

$$\begin{aligned}
g(\alpha;\lambda) &\leq \frac{1}{n}\sum_i \Big(\max_{\mathbf{y}\in\mathcal{Y}_i} \Delta(\mathbf{y},\mathbf{y}_i) + \max_{\mathbf{y}\in\mathcal{Y}_i}(-\mathbf{w}^\top\Psi(\mathbf{y})) - \Delta(\tilde{\mathbf{y}}_i,\mathbf{y}_i) + \mathbf{w}^\top\Psi(\tilde{\mathbf{y}}_i)\Big)\\
&= \frac{1}{n}\sum_i \Big(\max_{\mathbf{y}\in\mathcal{Y}_i}(-\mathbf{w}^\top\Psi(\mathbf{y})) + \mathbf{w}^\top\Psi(\tilde{\mathbf{y}}_i)\Big)\\
&= \frac{1}{n\lambda}\sum_i \max_{\mathbf{y}\in\mathcal{Y}_i}(-\tilde{\Psi}^\top\Psi(\mathbf{y})) + \frac{1}{n\lambda}\tilde{\Psi}^\top\sum_i\Psi(\tilde{\mathbf{y}}_i)\\
&= \frac{1}{n\lambda}\sum_i \theta(i) + \frac{1}{\lambda}\left\|\Psi\right\|^2
\end{aligned} \tag{5.50}$$

where, $\theta(i) = \max_{\mathbf{y}\in\mathcal{Y}_i}\big(-\tilde{\Psi}^\top\Psi(\mathbf{y})\big)$. We want the gap to be always less than $\kappa_1\varepsilon$, this leads to the following bound over the regularization parameter:

$$\begin{aligned}
&\frac{1}{n\lambda}\sum_i \theta(i) + \frac{1}{\lambda}\left\|\tilde{\Psi}\right\|^2 \leq \kappa_1\varepsilon\\
&\implies \lambda \geq \frac{\left\|\tilde{\Psi}\right\|^2 + \frac{1}{n}\sum_i\theta(i)}{\kappa_1\varepsilon}
\end{aligned} \tag{5.51}$$

Hence, for any value of the regularization parameter satisfying the above inequality, $\mathbf{w} = \frac{\tilde{\Psi}}{\lambda}$ is the $\kappa_1\varepsilon$ optimal solution. In order to compute λ, we should be able to compute $\theta(i)$ efficiently. Computing $\theta(i)$ is equivalent to the 'inference' problem, which is very efficient for many real world applications. In order to compute ν (defined in equation (5.44)) at $\mathbf{w} = \frac{\tilde{\Psi}}{\lambda}$, we can compute $\Omega(\alpha,\lambda)$ (defined in equation (5.43)) efficiently as follows:

$$\begin{aligned}
\Omega(\alpha,\lambda) &= \sum_i\sum_{\mathbf{y}\in\mathcal{Y}_i}\frac{\alpha_i(\mathbf{y})H_i(\mathbf{y};\mathbf{w})}{n}\\
&= \frac{1}{n}\sum_i\Delta(\tilde{\mathbf{y}}_i,\mathbf{y}) - \mathbf{w}^\top\frac{\sum_i\Psi_i(\tilde{\mathbf{y}}_i)}{n}\\
&= \frac{1}{n}\sum_i\Delta(\tilde{\mathbf{y}}_i,\mathbf{y}) - \frac{\left\|\tilde{\Psi}\right\|^2}{\lambda}
\end{aligned} \tag{5.52}$$

The above equalities comes from the fact that in order to compute $\mathbf{w} = \frac{\tilde{\Psi}}{\lambda}$, we had put all the mass on $\tilde{\mathbf{y}}_i$, which means $\alpha_i(\tilde{\mathbf{y}}_i) = 1$.

5.7 Regularization path for SSVM with positivity constraints

As already discussed in section 5.5, the SSVM with additional box constraints is very useful in many vision tasks. In this section we provide regularization path algorithm for SSVM with additional positivity constraints. However, the same procedure can be trivially modified to obtain the regularization path algorithm for SSVM with negativity constraints. In case of only positivity constraints, $\mathbf{l} = \mathbf{0}$, $\mathbf{u} = \infty$. Therefore, $\beta_u = \mathbf{0}$. From the KKT condition (equation (5.12)),

$$\mathbf{w} = A\alpha + \beta_l = \sum_{i\in[n],\mathbf{y}\in\mathscr{Y}_i} \frac{1}{\lambda n}\alpha_i(\mathbf{y})\Psi_i(\mathbf{y}) + \beta_l = [V(\alpha)]_+ \tag{5.53}$$

where, $\beta_l = [-V(\alpha)]_+$ (refer to the equation (5.19), and $[V(\alpha)]_+$ is the element wise truncation of $V(\alpha) = A\alpha$ into the positive numbers. The linearization duality gap, which is the same as the Lagrange duality gap (refer section 5.5), can be written as (refer chapter 2, equation (2.80) for proof):

$$g(\alpha;\lambda) = \frac{1}{n}\sum_i \big(\max_{\mathbf{y}\in\mathscr{Y}_i} H_i(\mathbf{y};\mathbf{w}) - \sum_{\mathbf{y}\in\mathscr{Y}_i} \alpha_i(\mathbf{y})H_i(\mathbf{y};\mathbf{w})\big). \tag{5.54}$$

where, $H_i(\mathbf{y};\mathbf{w}) = \Delta(\mathbf{y}_i,\mathbf{y}) - \mathbf{w}^\top\Psi_i(\mathbf{y})$.

5.7.1 Finding the breakpoints

The derivation to find the breakpoints follows exactly the same as in section 5.6.1. The only difference is that in this case the $\mathbf{w}$ is truncated. In another words, we use $\mathbf{w} = [V(\alpha)]_+$ in case of additional positivity constraints.

5.7.2 Initialization

We follow the same arguments as given in section 5.6.2. In order to move to a new point (or solution) we solve the linearization problem (the 'max-oracle' problem) at $\mathbf{w} = \mathbf{0}$, which simply is the loss-maximizer as shown below:

$$\tilde{\mathscr{Y}_i} = \underset{\mathbf{y}\in\mathscr{Y}_i}{\operatorname{argmax}} H_i(\mathbf{y};\mathbf{w}=\mathbf{0}) = \underset{\mathbf{y}\in\mathscr{Y}_i}{\operatorname{argmax}} \Delta(\mathbf{y},\mathbf{y}_i) \tag{5.55}$$

where, $\tilde{\mathscr{Y}_i}$ is the set of possible outputs for which the loss can attain its maximum value, therefore, $|\tilde{\mathscr{Y}_i}| \geq 1, \forall i$. Let us say that $\tilde{\mathbf{y}}_i \in \tilde{\mathscr{Y}_i}, \forall i$, is the randomly selected loss maximizer label. Putting all the mass to this label, which means $\alpha_i(\tilde{\mathbf{y}}_i) = 1$, gives the new corner point (or new solution) as follows (using the KKT condition):

$$\mathbf{w} = V(\alpha) + \beta_l = \frac{1}{\lambda}\sum_i \frac{\Psi_i(\tilde{\mathbf{y}}_i)}{n} + \beta_l = \frac{\tilde{\Psi}}{\lambda} + \beta_l \tag{5.56}$$

where, $\tilde{\Psi} = \frac{\sum_i \Psi_i(\tilde{\mathbf{y}}_i)}{n}$, $V(\alpha) = A\alpha$, and $\beta_l = [-V(\alpha)]_+$. For this new solution we would like to bound the duality gap and find a λ such that the $\mathbf{w}$ is $\kappa_1\varepsilon$ optimal. Let us assume that we know the dual variable vector α which would lead to the primal variable vector $\mathbf{w}$ using the KKT condition. Thus, the duality gap at this new solution using equation (5.54) can be defined as:

$$g(\alpha;\lambda) = \frac{1}{n}\sum_i \left(\max_{\mathbf{y}\in\mathscr{Y}_i} \left(\Delta(\mathbf{y},\mathbf{y}_i) - \mathbf{w}^\top\Psi(\mathbf{y})\right) - \Delta(\tilde{\mathbf{y}}_i,\mathbf{y}_i) + \mathbf{w}^\top\Psi(\tilde{\mathbf{y}}_i)\right) \tag{5.57}$$

Using the inequality $\max_x(f(x)+g(x)) \leq \max_x f(x) + \max_x g(x)$ and the fact that $[\frac{\tilde{\Psi}}{\lambda}]_+ = \frac{\tilde{\Psi}_+}{\lambda}$ (since λ is always positive), the gap can be upperbounded as follows:

$$\begin{aligned} g(\alpha;\lambda) &\leq \frac{1}{n}\sum_i \left(\max_{\mathbf{y}\in\mathscr{Y}_i} \Delta(\mathbf{y},\mathbf{y}_i) + \max_{\mathbf{y}\in\mathscr{Y}_i}(-\mathbf{w}^\top\Psi(\mathbf{y})) - \Delta(\tilde{\mathbf{y}}_i,\mathbf{y}_i) + \mathbf{w}^\top\Psi(\tilde{\mathbf{y}}_i)\right) \\ &= \frac{1}{n}\sum_i \left(\max_{\mathbf{y}\in\mathscr{Y}_i}(-\mathbf{w}^\top\Psi(\mathbf{y})) + \mathbf{w}^\top\Psi(\tilde{\mathbf{y}}_i)\right) \\ &= \frac{1}{n\lambda}\sum_i \max_{\mathbf{y}\in\mathscr{Y}_i}(-\tilde{\Psi}_+^\top\Psi(\mathbf{y})) + \frac{1}{n\lambda}\tilde{\Psi}_+^\top\sum_i\Psi(\tilde{\mathbf{y}}_i) \\ &= \frac{1}{n\lambda}\sum_i \theta(i) + \frac{1}{\lambda}\left\|\tilde{\Psi}_+\right\|^2 \end{aligned} \tag{5.58}$$

where, $\theta(i) = \max_{\mathbf{y}\in\mathcal{Y}_i}\big(-\tilde{\Psi}_+^\top\Psi(\mathbf{y})\big)$. We want the gap to be always less than $\kappa_1\varepsilon$ which leads to the following bound over the regularization parameter:

$$\begin{aligned}
&\frac{1}{n\lambda}\sum_i \theta(i) + \frac{1}{\lambda}\left\|\tilde{\Psi}_+\right\|^2 \leq \kappa_1\varepsilon \\
\implies &\lambda \geq \frac{\left\|\tilde{\Psi}_+\right\|^2 + \frac{1}{n}\sum_i \theta(i)}{\kappa_1\varepsilon}
\end{aligned} \tag{5.59}$$

Hence, for any value of the regularization parameter satisfying the above inequality, $\mathbf{w}$ is the $\kappa_1\varepsilon$ optimal solution. In order to compute λ, we should be able to compute $\theta(i)$ efficiently. Computing $\theta(i)$ is equivalent to the 'inference' problem, which is very efficient for many real world applications. In order to compute ν (defined in equation (5.44)) at $\mathbf{w} = \frac{\tilde{\Psi}_+}{\lambda}$, we can compute $\Omega(\alpha,\lambda)$ (defined in equation (5.43)) efficiently as follows:

$$\begin{aligned}
\Omega(\alpha,\lambda) &= \sum_i \sum_{\mathbf{y}\in\mathcal{Y}_i} \frac{\alpha_i(\mathbf{y})H_i(\mathbf{y};\mathbf{w})}{n} \\
&= \frac{1}{n}\sum_i \Delta(\tilde{\mathbf{y}}_i,\mathbf{y}) - \mathbf{w}^\top \frac{\sum_i \Psi_i(\tilde{\mathbf{y}}_i)}{n} \\
&= \frac{1}{n}\sum_i \Delta(\tilde{\mathbf{y}}_i,\mathbf{y}) - \frac{\left\|\tilde{\Psi}_+\right\|^2}{\bar{\lambda}}
\end{aligned} \tag{5.60}$$

The above equalities comes from the fact that in order to compute $\mathbf{w}$, we had put all the mass on $\tilde{\mathbf{y}}_i$, which means $\alpha_i(\tilde{\mathbf{y}}_i) = 1$.

5.8 Experiments and Analysis

In this section we talk about the experiments we performed on the OCR dataset [122] with 6251 train samples and 626 test samples. The joint feature map is exactly the same as defined in [122] which allows us to solve the 'max-oracle' problem very efficiently using dynamic algorithm. Broadly speaking, this section is mainly divided into three parts. In the first part, we use 20 different values of the regularization parameter equally spaced in the range of $[10^{-4}, 10^{3}]$ and discuss about the number of passes taken for the training, and the test losses. In the second part, we show results for the $\varepsilon = 0.1$ optimal regularization path using different variants of the BCFW algorithm for $\kappa_1 = \{0.1, 0.3, 0.5, 0.7, 0.9, 0.95\}$. We discuss the results in terms of duality gaps, number of passes, and test losses. In the third part, we talk about computational feasible strategies to obtain the best model using the regularization path algorithm. Notice that we only show results for ε optimal paths, therefore, $\kappa_2 = 1.0 - \kappa_1$. In order to avoid clutter we use short hand notations for different algorithms as follows:

- BCFW-STD-U and BCFW-STD-G: The BCFW-STD part represents the standard BCFW algorithm [86] with $\mathbf{w} = 0$ as the initialization. The parts U/G represents the sampling methods, uniform and gap based, respectively (refer Algorithm 13). Notice that the algorithm in [86] did not talk about the gap based sampling. We also trivially modify the convergence criterion of [86]. Instead of running the BCFW for p complete passes through the data before computing the exact duality gap, we rather keep on checking the the approximate duality gap (by product of BCFW). If the approximate duality gap is less than the ε, we compute the exact duality gap even before p passes. In our experiments we use $p = 10$. Notice that after computation of the exact duality gap, the approximate duality gaps (per block) are updated accordingly. This helps in selecting better blocks for the gap based sampling algorithm.

- BCFW-HYB-U and BCFW-HYB-G: It represents the hybrid version of the BCFW algorithm with uniform/gap based sampling (refer Algorithm 14). Everything remains the same as in BCFW-STD-U and BCFW-STD-G except for the fact that after computing the exact duality gap, the weight vectors are also updated along with the approximate duality gaps.

- RP-BCFW-STD-U and RP-BCFW-STD-G: The RP represents the regularization path. Therefore, it represents the regularization path algorithm using BCFW-STD-U and BCFW-STD-G as the SSVM-OPTIMIZER (refer to Algorithm 17).

- RP-BCFW-HEU-U and RP-BCFW-HEU-G: The RP represents the regularization path and HEU represents the heuristic based convergence criterion (converged if approximate duality gap is less than ε). Therefore, it represents the regularization path algorithm using BCFW-HEU-U and BCFW-HEU-G as the SSVM-OPTIMIZER (refer Algorithm 17). Notice that these two algorithms are approximate and does not guarantee exact convergence.

No regularization path. In order to perform the experiments without regularization path, we use 20 different values of λ equally spaced in the range of $[10^{-4},\ 10^{3}]$ and four variants of the BCFW algorithm (as shown in Table 5.1). It is clear from the table that the gap based sampling outperforms the uniform sampling based algorithms. However, contrary to the intuition, we see that the BCFW-STD-G marginally outperforms the BCFW-HYB-G. One possible explanation is the fact that once we obtain gaps at a particular $\mathbf{w}$, the gaps are exact

Table 5.1: No regularization path: Initialization with $\mathbf{w} = 0$. We used 20 different values of λ equally spaced in the range of $[10^{-4},\ 10^{3}]$ and four variants of the BCFW algorithm. Total passes represents the total number of passes through the entire dataset using all the 20 values of λ. Min test loss represents the minimum value of the test loss obtained by using all the 20 trained models. Clearly, the gap based sampling method (BCFW-STD-G) requires less number of passes to achieve the same generalization.

	BCFW-HYB-U	**BCFW-HYB-G**	**BCFW-STD-U**	**BCFW-STD-G**
Min test loss	0.121	0.121	0.122	0.121
Total passes	1646.786	1153.439	1614.769	**1138.872**

Table 5.2: Regularization path: κ_1 vs total passes for four different variants of the BCFW algorithm.The total passes represents the total number of passes through the entire dataset to obtain the complete regularization path.

κ_1	**RP-BCFW-HEU-G**	**RP-BCFW-HEU-U**	**RP-BCFW-STD-G**	**RP-BCFW-STD-U**
0.1	2711.946	5102.513	4405.881	5487.914
0.3	1708.943	2939.686	2287.803	4078.553
0.5	1301.869	2485.238	2120.969	2649.019
0.7	1247.567	2109.278	1786.437	2306.304
0.9	**1076.005**	1874.513	2100.304	2122.717
0.95	**1070.634**	1851.846	2572.735	2174.990

at that $\mathbf{w}$. However, if we further update $\mathbf{w}$ at this step (in case of the hybrid algorithm), the gaps no longer represents the true gaps corresponding to the updated $\mathbf{w}$. Therefore, in case of hybrid algorithm, the gap based sampling strategy may not pick the right samples.

Regularization path. In order to better understand the behavior of different variants of BCFW algorithm and the effects of parameters κ_1 and κ_2 on the regularization path algorithm, we performed several experiments using four variants of the BCFW algorithm and 6 different combinations of κ_1 and κ_2. The variants of the BCFW algorithm used are BCFW-HEU-U/G

Table 5.3: Regularization path: κ_1 vs number of regularization parameters (λ) for four different variants of the BCFW algorithm. The number of regularization parameters is basically the number of kinks that divide the complete regularization space into segments.

κ_1	**RP-BCFW-HEU-G**	**RP-BCFW-HEU-U**	**RP-BCFW-STD-G**	**RP-BCFW-STD-U**
0.1	142	142	133	132
0.3	168	168	135	134
0.5	225	226	153	148
0.7	364	365	193	182
0.9	1060	1065	349	306
0.95	2105	2115	501	420

Table 5.4: Regularization path: κ_1 vs minimum test loss. For a given regularization path, the test losses are obtained for models corresponding to 20 different values of λ equally spaced in the range of $[10^{-4},\ 10^{3}]$ (same as in case of no regularization path experiments). Notice that the models are trained using the regularization path algorithm. For each path, the minimum test loss is the minimum over the 20 different values.

κ_1	**RP-BCFW-HEU-G**	**RP-BCFW-HEU-U**	**RP-BCFW-STD-G**	**RP-BCFW-STD-U**
0.1	0.113	0.113	0.114	0.113
0.3	0.118	0.113	0.115	0.116
0.5	0.121	0.116	0.118	0.119
0.7	0.120	0.120	0.120	0.120
0.9	0.125	0.123	0.118	0.122
0.95	0.125	0.121	0.120	0.121

and BCFW-STD-U/G. Notice that, we do not use the hybrid variants of BCFW as it was outperformed by the standard variants (refer to the Table 5.1).

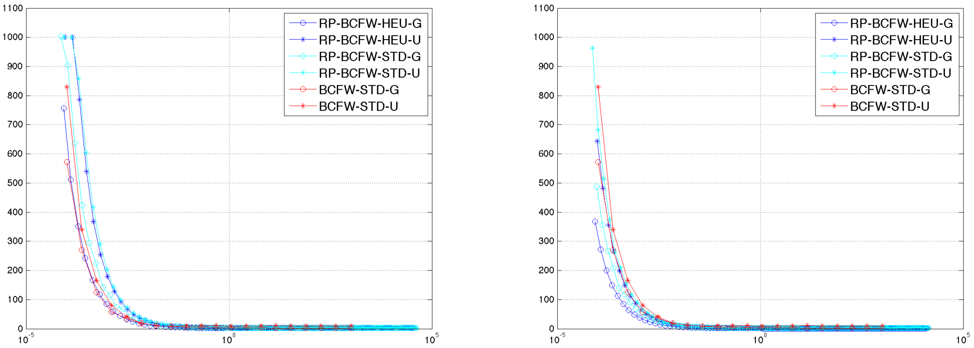
Tables 5.2 and 5.3 shows the total number of passes and total number of regularization parameters (kinks) to obtain the complete regularization path. As shown in the Table 5.3, as we increase κ_1, the gap corresponding to $\kappa_1\varepsilon$ increases, which in turn decreases the excess gap (refer to the equation (5.45)). Therefore, the length of the segments in the regularization path also decreases. This leads to higher number of kinks (thus segments) to obtain the complete regularization path. However, increasing κ_1 decreases κ_2 (as $\kappa_1 + \kappa_2 = 1$). Therefore, each time we are at a breakpoint (or kink) with ε optimal solution, the optimizer has to put very less efforts, and thus less number of passes (because of the warm start), to reach $\kappa_1\varepsilon$ optimal solution as $\kappa_1\varepsilon \approx \varepsilon$ for κ_1 closer to one. The effects of warm start is well captured in the Figure 5.1. For example, in Figure 5.1f at $\lambda = 10^{-4}$, the warm start with ε optimal solution (RP algorithms) almost takes 90 passes through the data to converge to the $\kappa_1\varepsilon$ optimal solution. However, BCFW-STD-G/U takes almost 590/820 passes to converge to the ε optimal solution. As we decrease κ_1, the warm start goes further away from the desired optimality criterion, thus, leads to more number of passes to converge. Let us now have a look into the optimality of the regularization path obtained using various variants of the BCFW algorithm as the SSVM optimizer. Figure 5.2 shows duality gaps for different combinations of κ_1 and κ_2. Recall that BCFW-HEU-U/G provides approximate duality gap. Therefore, in order to see how far the approximate duality gap is compared to the true duality gap, we also show the true duality for the purpose of analysis. Notice that, as shown in Figures 5.2b-5.2d, for smaller values of κ_1 (upto 0.5), the true duality gap for BCFW-HEU-U/G is always less than the ε. Thus, the regularization path obtained is ε-optimal even if the convergence criterion was based on approximate duality gap. However, as we increase κ_1 (beyond 0.5), the true duality gap for smaller values of λ is higher than the ε. Thus, the path obtained in this region is suboptimal. Hence, for higher values of κ_1, the regularization path obtained using the heuristic algorithm as the SSVM optimizer is not entirely ε-optimal. In another words, using BCFW-HEU-U/G algorithms as the optimizer is not guaranteed to provide ε-optimal regularization path. On the other hand, for all other algorithms, the entire path is ε-optimal (as expected).

In order to better understand the generalization of the models obtained using the experiments, we show the test losses in Figure 5.3. It is evident from the figure that models obtained using all the algorithms provides almost the same test losses. However, for lower values of κ_1, the models obtained using the regularization path algorithm seems to provide better test losses for lower values of λ. One of the possible explanation of this observation is that, in case of lower values of κ_1, the models obtained are closer to the global minima as the convergence criterion is more strict $(\kappa_1\varepsilon \ll \varepsilon)$. Thus provides better generalization. Table 5.4 shows the minimum test loss obtained using all the methods. Notice that the test losses are obtained only for 20 values of the regularization parameter. As explained earlier, the regularization path algorithm for lower values of κ_1 provides better minimum test loss compared to the experiments without regularization path. Also, the optimizers RP-BCFW-HEU-U/G, even being suboptimal, provides promising generalization in terms of the minimum test loss.

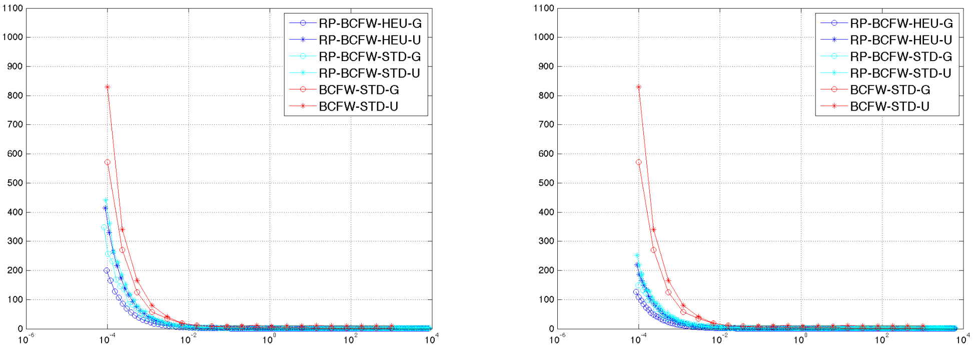
Obtaining the best model. In order to obtain the best model using the regularization path algorithm, we have to test (solving 'max-oracle') the models at each breakpoint. If there are many breakpoints, this task is computationally expensive. For example, as shown in Tables 5.2 and 5.3, the RP-BCFW-HEU-G optimizer at $\kappa_1 = 0.95$, even being the fastest (nearly 1071 passes) gives 2105 breakpoints. Thus requires 2105 max-oracles to find the best λ, which is highly expensive task to perform. On the other hand, the RP-BCFW-STD-U at $\kappa_1 = 0.1$ provides 132 breakpoints. However, obtaining the entire regularization at this setting requires almost 5488 passes through the data, which is highly expensive compared to other settings. A computationally affordable approach to obtain a highly promising model is to use the fastest setting to obtain the entire regularization path. For example, RP-BCFW-STD-G with $\kappa_1 = 0.7$ to obtain the ε-optimal path, or RP-BCFW-HEU-G with $\kappa_1 = 0.95$ to obtain a suboptimal path. Once the path is obtained, use a grid of regularization parameters over this path, store the corresponding models and perform testing. This is exactly what we do. We choose 20 values of λ even spaced in the range of $[10^{-4},\ 10^{3}]$. Notice that, in this approach, with each additional value of the λ, we perform only one extra 'max-oracle' for the testing. However, without the regularization path, for each additional λ, we have to first train the model (which is expensive) and then perform the 'max-oracle' for the testing.

5.9 Discussion

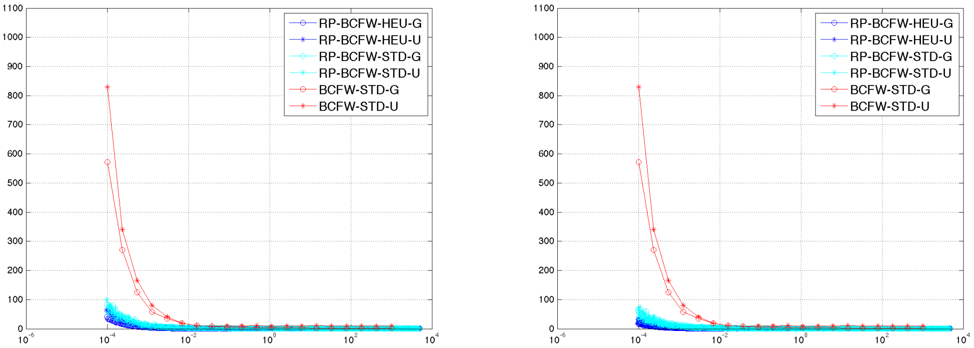
We proposed an algorithm to obtain the entire regularization path, ε-optimal, for SSVM and SSVM with additional positivity/negativity constraints. We also provided a principled algorithm to optimize SSVM with additional box constraints using BCFW and its variants. Obtaining the regularization path allows us to obtain the best model for the SSVM. On one hand, if computational efficiency is the deciding factor, the heuristic variant of the BCFW with gap based sampling outperforms all other variants of the BCFW and provide regularization path (suboptimal) very efficiently. However, if optimality is the prime focus, the gap sampling based BCFW algorithm outperforms all other variants and provide regularization path which is ε-optimal. An interesting direction of future work would be to efficiently obtain the exact regularization path for the SSVM and SSVM with additional box constraints. Another interesting direction of research would be to devise some strategies to obtain less number of breakpoints with the theoretical guarantees of being optimal.



(a) $\kappa_1 = 0.1$ and $\kappa_2 = 0.9$ (b) $\kappa_1 = 0.3$ and $\kappa_2 = 0.7$

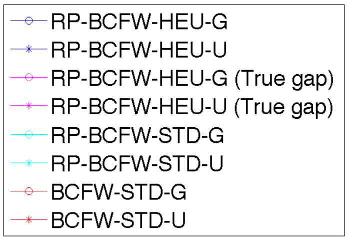


(c) $\kappa_1 = 0.5$ and $\kappa_2 = 0.5$ (d) $\kappa_1 = 0.7$ and $\kappa_2 = 0.3$

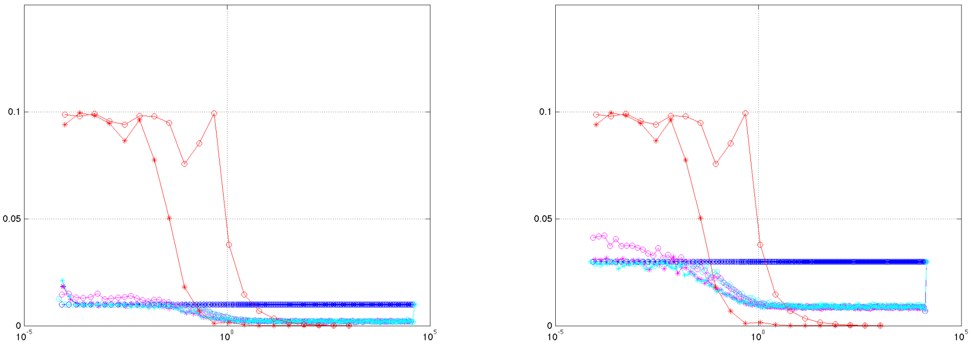


(e) $\kappa_1 = 0.9$ and $\kappa_2 = 0.1$ (f) $\kappa_1 = 0.95$ and $\kappa_2 = 0.05$

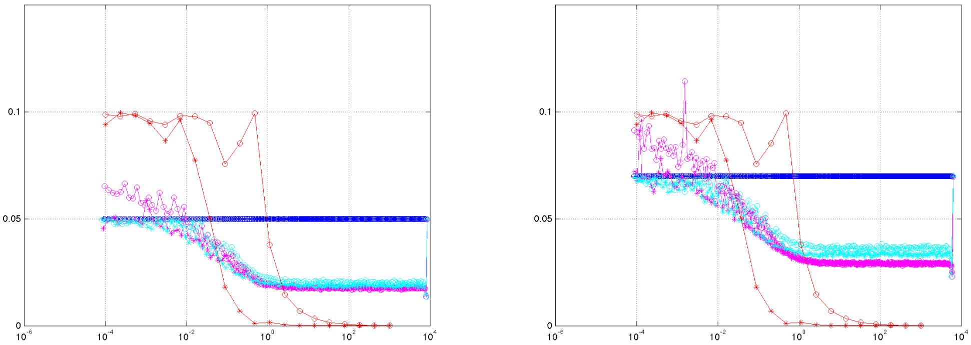
Figure 5.1: In each figure, x-axis is the λ and y-axis is the number of passes. In case of regularization path algorithms (starting with RP), x-axis represents all the breakpoints for the entire path, and y-axis represents the number of passes required to reach $\kappa_1\varepsilon$ optimal solution at each breakpoint (warm start with ε optimal solution). In other two algorithms (no regularization path), x-axis represents 20 values of λ evenly spaced in the range of $[10^{-4}, 10^{3}]$, and y-axis represents the number of passes to reach ε optimal solution (initialization with $\mathbf{w} = 0$).



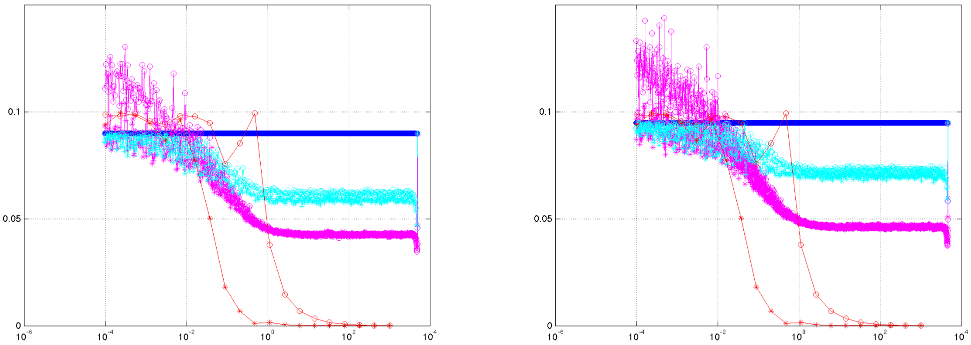
(a) Legend



(b) $\kappa_1 = 0.1$ and $\kappa_2 = 0.9$ (c) $\kappa_1 = 0.3$ and $\kappa_2 = 0.7$

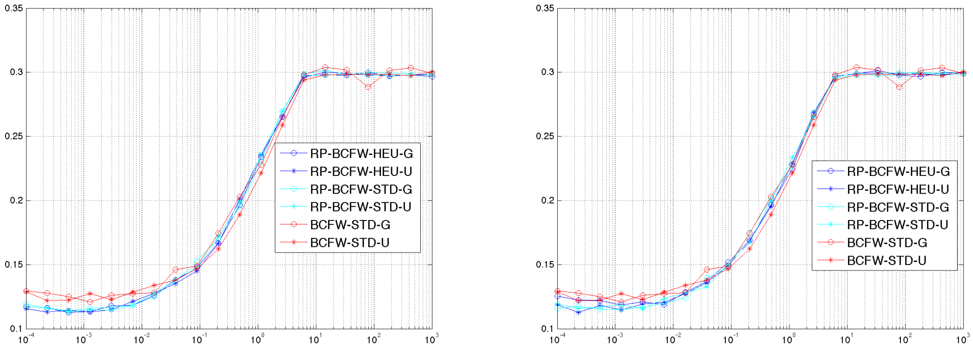


(d) $\kappa_1 = 0.5$ and $\kappa_2 = 0.5$ (e) $\kappa_1 = 0.7$ and $\kappa_2 = 0.3$

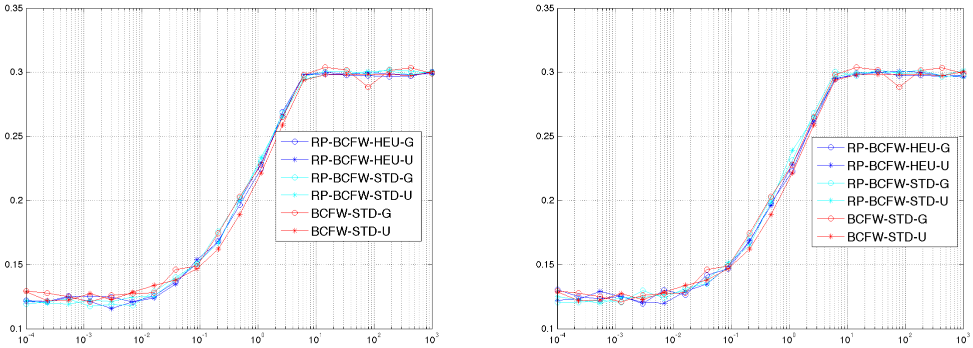


(f) $\kappa_1 = 0.9$ and $\kappa_2 = 0.1$ (g) $\kappa_1 = 0.95$ and $\kappa_2 = 0.05$

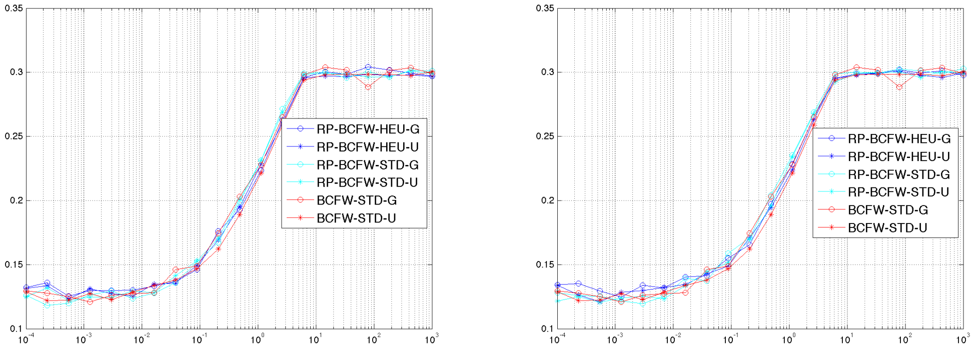
Figure 5.2: Figure 5.2a shows the 'legend' for the purpose of clarity. In all other figures, x-axis is the λ and y-axis is the duality gap. In case of regularization path algorithms (starting with RP), x-axis represents all the breakpoints for the entire path. In other two algorithms (no regularization path), x-axis represents 20 values of λ evenly spaced in the range of $[10^{-4}, 10^{3}]$. Recall that in case of regularization path algorithms the solution we seek must be $\kappa_1 \varepsilon$ optimal.



(a) $\kappa_1 = 0.1$ and $\kappa_2 = 0.9$ (b) $\kappa_1 = 0.3$ and $\kappa_2 = 0.7$



(c) $\kappa_1 = 0.5$ and $\kappa_2 = 0.5$ (d) $\kappa_1 = 0.7$ and $\kappa_2 = 0.3$



(e) $\kappa_1 = 0.9$ and $\kappa_2 = 0.1$ (f) $\kappa_1 = 0.95$ and $\kappa_2 = 0.05$

Figure 5.3: In all figures, x-axis is the λ and y-axis is the test loss. In all the experiments, the test losses are obtained for models corresponding to 20 different values of λ equally spaced in the range of $[10^{-4}, 10^{3}]$.

Chapter 6

Discussion

6.1 Contributions of the Thesis

In this thesis we developed inference and machine learning algorithms several tasks related to computer vision and medical imaging. The major contributions are:

- In chapter 3 we investigated a higher-order Markov field problem. Specifically, we presented a novel family of discrete energy minimization problems, that we call *parsimonious labeling*. It is a natural generalization of the well known metric labeling problem to higher-order potentials. Similarly to metric labeling, the unary potentials of parsimonious labeling can be arbitrary. The clique potentials are defined using the recently proposed notion of a *diversity* [18], defined over the set of unique labels assigned to the random variables in the clique. Intuitively, diversities enforce parsimony by assigning lower potentials to assignments with fewer distinct labels. In addition to this we proposed a generalization of the P^n-Potts model [66], that we call Hierarchical P^n-Potts. We showed how parsimonious labeling can be represented as a mixture of hierarchical P^n-Potts models. Finally, we proposed parallelizable move-making algorithms with strong multiplicative bounds for the optimization of both the hierarchical P^n-Potts model and parsimonious labeling. We showed the efficacy of parsimonious labeling in some challenging image denoising and stereo matching tasks, along with synthetic experiments.

- In chapter 4 we introduced two alternate frameworks to incorporate higher-order information into ranking. The first framework, which we call high-order binary SVM (HOB-SVM), takes its inspiration from standard SVM. The input of HOB-SVM is a set of samples. The output is a binary vector of labels for the samples, where the label 1 indicates that the sample is relevant and 0 indicates that it is not relevant. The joint feature vector of HOB-SVM depends not only on the features of the individual samples, but also on the feature vectors of subsets of them. In this work we restricted the subsets to be of size two but our framework can easily be generalized to larger subsets. The loss function of HOB-SVM is a weighted 0-1 loss, which allows us to efficiently perform loss-augmented inference using graph cuts [74]. Practically speaking, one difficulty with HOB-SVM is that it provides a single score for the entire labeling of a dataset, whereas we need scores for each sample in order to find the ranking. To address this, we proposed to rank the samples using the difference between the *max-marginal score* for assigning the sample to the relevant class and the *max-marginal score* for assigning it to the non-relevant class. Intuitively, the difference of max-marginal scores measures the 'positivity' of a particular sample while still capturing higher-order information. Empirically we showed that differences of max-marginal scores provide accurate rankings. The main advantage of HOB-SVM is that its parameters can be estimated efficiently by solving a convex optimization problem. Its main disadvantage is that, similarly to SVM, it optimizes a surrogate loss function instead of the average precision based loss (AP is one of the measures of ranking). The second framework, which we call high-order AP-SVM (HOAP-SVM), takes its inspiration from AP-SVM and HOB-SVM. Similarly to AP-SVM, the input of HOAP-SVM is a set of samples, the output is a ranking of them, and the loss function is the AP loss. However unlike AP-SVM, the score of a ranking is equal to a weighted sum of the differences of max-marginal scores of the individual samples. As the max-marginal scores capture higher-order information and the loss function depends on the AP, HOAP-SVM addresses both of the aforementioned deficiencies of traditional classifiers such as SVM. The main disadvantage of HOAP-SVM is that estimating its parameters requires solving a difference-of-convex program [55]. While we cannot obtain a globally optimal set of parameters for HOAP-SVM, we showed that a local optimum can be computed efficiently using the concave-convex procedure [138]. Using standard publicly available datasets we empirically demonstrated that HOAP-SVM out-performs SVM, AP-SVM and HOB-SVM by effectively utilizing higher-order information while optimizing a ranking based loss function.

- Notably, SSVM methods normally contain a regularization parameter that controls the trade-off between the model complexity and an upper bound on the empirical risk. The regularization value has a significant impact on the generalization of the SSVM. However, lack of knowledge about the structure of the regularization parameter compel us to cross validate it over the entire parameter space, which is practically not feasible because of infinitely many possibilities. To circumvent this problem, the standard approach is to resort to a sub optimal solution by cross validating or tuning a large number of possible values of regularization parameter on a validation or cross-validation set. Doing this is tedious and it quickly becomes prohibitively costly as we increase the number of regularization values tested. In order to handle this issue, in chapter 5 we proposed a new algorithm (SSVM-RP) that obtains an ε-optimal regularization path for SSVM. By definition, the regularization path is the set of solutions $\mathbf{w}(\lambda)$ for all possible values of the regularization parameter λ [29]. This allows us to obtain the best model by efficiently searching the entire regularization parameter space. We proposed several intuitive variants of the Block-Coordinate Frank-Wolfe algorithm for the rapid solution of the SSVM-RP problem. In addition to this we proposed a principled approach to solving the SSVM with additional box constraints using BCFW and its variants. These additional constraints are useful in many important problems in order to solve the corresponding inference problem exactly. For example, for binary labels and an output structure that forms a graph with loops, if the pairwise potentials are submodular graph cuts [74] can be used to solve the inference problem exactly. However in order to ensure that the pairwise potentials are submodular, additional positivity/negativity constraints are needed in the objective function of SSVM. Finally, we proposed a regularization path algorithm for SSVM with such additional positivity/negativity constraints.

- In Appendix A we proposed a novel weakly-supervised discriminative algorithm for learning context specific registration metrics as a linear combination of conventional metrics. We adopted a popular graphical model framework [48] to cast deformable registration as a discrete inference problem. This involves data and smoothness terms. The data term is an application-specific similarity measure or metric, such as mutual information, normalized cross correlation, sum of absolute difference, or discrete wavelet coefficients. Depending on the clinical context conventional metrics can only cope partially with tissue anatomical properties. Metric learning representation seeks to determine a mapping between the source and target representations in the registration task. In this work our goal was to determine anatomy/tissue specific metrics as context-specific aggregations (linear combinations) of known metrics. To achieve this, we proposed a weakly-supervised learning algorithm for estimating these parameters conditionally on the semantic data classes, using a weakly labelled training dataset. The learning objective in our formulation was a special kind of non-convex program, known as a difference of convex programs. We used the concave-convex procedure to obtain a local minimum or saddle point of the optimization problem. In order to estimate the unknown ground truth deformation vectors – which we treated as latent variables – we used ‘segmentation consistent’ inference endowed with a loss function. We showed the efficacy of our approach on three challenging datasets from medical imaging with different anatomical structures and image modalities.

6.2 Future Work

We now discuss some possible future directions that could extend the practical and theoretical implications of our work.

High-Order Inference

- In parsimonious labeling we approximated a given *diversity* with a *diameter diversity*. However, there exist many other diversities such as Steiner tree diversity, L_1 diversity, truncated diversity, etc [18].It would be interesting to explore these options and propose specific algorithms for them.

- Another interesting direction would be to directly approximate diversities as mixtures of hierarchical P^n-Potts models, without the use of intermediate r-HST metrics. This might also lead to better multiplicative bounds.

- Parsimonious labeling assumes that the given clique potentials are diversities which induces metrics. In order to apply it to problems in which the labels are symbolic, for example semantic segmentation, it would be interesting to learn these metrics from the data and then define diversities over them.

Structured SVM

- An interesting direction for future work would be to propose weakly-supervised extension of HOAP-SVM. While such a learning objective can easily be formulated with the introduction of latent variables, it is not clear whether the resulting optimization problem can be solved efficiently.

- In general, in SSVM frameworks, the structure and interactions between the outputs are assumed to be known a priori. However, experience shows that the underlying structure has a considerable impact on the overall performance of the algorithm. Therefore, it would interesting to design a framework that can simultaneously discover the most suitable underlying structure and minimize an upper bounded loss over it.

- The stochastic optimization algorithms for SSVM work by randomly picking samples to optimize, where each sample has a structure. However, if the samples themselves are complex, for example huge graphs with millions of edges and cliques, then solving the ‘loss augmented inference’ problem even for one sample can be difficult. It would be useful to devise stochastic algorithms that can decompose the samples into smaller graphs for which the ‘loss augmented inference’ problem can be solved more easily, then sample these smaller graphs to update the learning parameters. There are two main challenges with this: (i) how to decompose the samples such that the context encoded is not destroyed (or if it is then how to quantify that); and (ii) how to guarantee that while doing this we can still obtain the global minimum of the actual problem.

- Normally, for latent SSVM [135] the joint feature map encodes the interactions among the output variables. However, no such interactions are encoded between the latent variables themselves. It would be interesting to devise frameworks that can encode interactions between the latent variables as well as ones between the output variables.

Regularization path

- An interesting direction for future work would be to efficiently obtain the *exact* regularization path for SSVM and SSVM with additional box constraints.

- Another interesting direction would be to devise strategies to reduce the number of breakpoints while maintaining the same theoretical guarantees.

Appendix A

Deformable Registration through Learning of Context-Specific Metric Aggregation

A.1 Introduction

Registration [17, 115, 117] is a highly challenging problem frequently encountered in many vision and medical imaging related tasks. Few examples of the applications of the registration problems are – (1) registration between 2D and 3D point sets [38], (2) alignment of medical acquisitions [33, 36, 115, 116], (3) range data integration [53], and (4) registration of 3D points to 3D planes [107]. In general, the registration problem involves aligning a source image (or source data) to a target image (or target data) such that a predefined energy function (or model) is minimized. In other words, the registration task involves the estimation of spatial transformation that establishes meaningful links between the source and the target images. Broadly speaking, we can divide the registration problems into two categories – (1) linear registration, and (2) deformable one. Linear registration involves transformation that are global in nature (same for all the voxels in the source image). Deformable registration adopts spatial transformation that is non-linear, which significantly increases the complexity of this task [115]. In this work, we focus on the deformable registration problem. The energy function related to this problem usually consist of the data terms and the smoothness terms. Such energy functions can be optimized using off-the-shelf inference algorithms such as the well known FastPD [77], Loopy belief propagation [97, 105], and Graph Cuts [74]. We adopt a popular graphical model framework [48] to cast deformable registration as a discrete inference problem. It is known that the data terms in the energy function has great influence on the accuracy of the solution of the registration problem. It refers to a function that measures similarity/(dis)similarity between images such as the mutual information (MI), sum of absolute differences (SAD), normalized cross correlation (NCC), and many other. A particular data term is thus chosen based on the application. Metric learning [16, 21, 90, 91, 95, 125, 130] is an alternative that consists of determining from labeled visual correspondences the most efficient means of image comparison. By *metric* we mean the similarity measures (for example, MI and NCC) used as the data terms in the registration problem. In the context of registration, metric learning can simply be interpreted as learning a domain specific matching criterion that allows the comparison of any two given image modalities. Our approach can be considered as a specific case of metric learning where the idea is to efficiently combine existing/well known/well studied mono and multi-modal metrics to a single one depending on the local context. In other words, we would like to learn the relative weights from a given training dataset using a learning framework spatially conditioned on vague semantic knowledge that we must have. We propose a novel discriminative learning framework, based on the well known structured support vector machines (SSVM) [122, 126], to learn the relative weights (or the parameters). The SSVM and its extension to latent models LSSVM [135] have received considerable attention in the recent years for developing discriminative frameworks for parameter learning [4, 85].

Our focus is mainly on the 3D to 3D deformable registration problem where the input and the output images are 3D volumes. However, the same framework can be used for other registration problems as well. One of the key problem we face is that the ground truth deformations are not known a priori. This leads us to design a weakly supervised learning framework based on the latent structured support vector machines LSSVM. We treat the ground truth deformations as the latent variables. We model the latent variable imputation problem as the deformable registration problem with additional constraints. These additional set of constraints ensures that when the imputed latent deformation vectors are applied to the source image, the deformed source image is maximally aligned with the target image. The alignment accuracy is measured based on a user defined loss function. Our learning framework, similar to the LSSVM, is a special family of non-convex optimization problems, known as the difference of convex functions. The local optimum or the saddle point of such non-convex function can be obtained using the well known CCCP algorithm [138]. We demonstrate the efficacy of our framework using three challenging datasets related to the medical imaging.

A.2 Related Work

Broadly speaking, metric learning methods can be classified as supervised or unsupervised. Supervised methods require a set of annotated data to learn a distance metric function. Instead, in unsupervised methods (also referred by [133] as manifold learning approaches), the main idea is to learn an underlying low-dimensional manifold where geometric relationships (e.g. distance) between most of the observed data are preserved. This allows us to map the original data into a simpler representation endowed with a metric that correctly represents the data similarity [16].

In supervised learning setting, Lee et al. [90] proposed a rigid multi-modality registration algorithm where the similarity measure is learned in a discriminative manner such that the target and the correctly deformed source image receive high similarity scores. The training data consisted of pre-aligned images and the learning is performed at the patch level with an assumption that the similarity measure of two images decompose over the patches. Another work by Bronstein et al. [16] proposed the use of sensitive hashing to learn a multi-modality distance metric that can be applied to data coming from two different spaces. The idea is to embed the input data from two arbitrary spaces into the Hamming space. This mapping is expressed as a binary classification problem and can be efficiently learned using boosting algorithms. Similar to [90], Bronstein et al. adopted a patch-wise approach. The dataset consisted of pairs of perfectly aligned images and a collection of positive and negative pairs of patches. A detailed version of [16], plugged into the standard graph-based deformable registration framework was presented in [95]. Another patch-based alternative was presented by [125]. However, in [125], the training set consisted of non-aligned images with manually annotated patch pairs (landmarks). [125] used a coarse-to-fine strategy where a global similarity measure was learned followed by a fine similarity measure that captured the fine level point correspondences. Recently, approaches based on deep learning have started to gain popularity. An interesting discussion about this topic was presented in [91], suggesting that features learned using convnets are at least as useful (sometimes more useful) as the conventional ones when performing alignment of elements of the same class.

Unsupervised learning methods for image registration have also been well studied. Wachinger and Navab [130] proposed to apply manifold learning (through Laplacian eigenmaps) to learn structural representations of multi-modal images. The idea is to calculate dense descriptors that represent the structural information of image patches, which does not depend on the intensity values of the images but on the structures in the patch. Ou et al. [103] proposed an online metric learning framework based on mutual saliency. The central idea was to consider a voxel-based linear combination of metrics (SAD over filter responses) as single metric and then locally adjust the coefficients based on the correspondence in terms of 'saliency' expression between the source and the target image. A different approach based on unsupervised deep learning was proposed by Wu et al. in [132]. The authors directly learned the basis filters that can effectively represent the observed image patches.

In contrast to the aforementioned approaches that aim at learning a completely new similarity measure, our method aggregates standard metrics in a domain specific and smart way. The work by [21] showed, in fact, that using a multichannel registration method where a set of features is considered instead of a single similarity measure, produced more robust registration results when compared to using the features individually. However, they did not discuss how these features should be weighted. In a posterior work, [22] proposed a methodology, independent of the need of explicitly weighting the features, by estimating different deformation fields from each feature independently, and then composing them into a final diffeomorphic transformation. Such a strategy produces multiple deformation models (as many as the number of metrics) which might be locally inconsistent. Therefore, their combination will not be that trivial and in the general case not anatomical meaningful.

A.3 The Deformable Registration Problem

Let us assume that we are given a source $3D$ volume (or image) I, source $3D$ segmentation mask S^I, and the target $3D$ volume (or image) J. The size of the segmentation mask is the same as that of the corresponding image. The segmentation mask is formed by the elements (or voxels) $s_k \in \mathscr{C}$, where $\mathscr{C}$ is the set of classes. For example, in case of medical imaging, $\mathscr{C}$ can be the set of organs. Without loss of generality, we assume that the elements in the class set $\mathscr{C}$ are the natural numbers starting from one.

We model the deformable registration problem as an inference problem on an MRF [48]. A deformation field is sparsely represented by a regular grid graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. Each node $i \in V$ corresponds to a control point p_i. Each control point p_i is allowed to move in the $3D$ space, therefore, can be assigned a label $\boldsymbol{d_i}$ from the set of the $3D$ displacement vectors $\mathscr{L}$. Notice that each $3D$ displacement vector is a tuple defined as $\boldsymbol{d_i} = \{dx_i, dy_i, dz_i\}$, where dx, dy, and dz are the displacements in the x, y, and z directions, respectively. The deformation, which is the labeling of the graph G, is denoted as D. In another words, the deformation $D \in \mathscr{L}^{|V|}$ represents a set of nodes V where each node is assigned a displacement vector $\boldsymbol{d_i}$ from the set of displacement vectors $\mathscr{L}$. We denote the control point $\bar{p}_i$ as the new control point when the displacement $\boldsymbol{d_i}$ is applied to the original control point p_i. Let us define a patch $\bar{\Omega}_i^I$ on the source image I centered at the displaced control point $\bar{p}_i$. Similarly, we define Ω_i^J as the patch on the target image J centered at the original control point p_i, and $\bar{\Omega}_i^{S^I}$ as the patch on the input segmentation mask centered at the displaced control point $\bar{p}_i$. Using the above notations, we define the unary feature vector corresponding to the i^{th} node for a given displacement vector $\boldsymbol{d_i}$ as $\mathscr{U}_i(\boldsymbol{d_i}, I, J) = (u_1(\bar{\Omega}_i^I, \Omega_i^J), \cdots, u_n(\bar{\Omega}^I, \Omega_i^J)) \in \mathbb{R}^n$, where n is the number of metrics and $u_j(\bar{\Omega}_i^I, \Omega_i^J)$ is the unary feature corresponding to the j^{th} metric evaluated using the patches $\bar{\Omega}_i^I$ and Ω_i^J. For example, in case the j^{th} metric is the mutual information (MI) then the unary feature $u_{MI}(\bar{\Omega}_i^I, \Omega_i^J)$ is the mutual information between the patches $\bar{\Omega}_i^I$ and Ω_i^J. In case of single metric, $n = 1$. Recall that we have $|\mathscr{C}|$ number of classes. Therefore, given a weight matrix $W \in \mathbb{R}^{n \times |\mathscr{C}|}$, where $W(i, j)$ denote the weight for the i^{th} metric corresponding to the class j, the unary potential for the i^{th} node for a given displacement vector $\boldsymbol{d_i}$ is computed as follows:

$$\bar{\mathscr{U}_i}(\boldsymbol{d_i}, I, J, S^I; W) = \mathbf{w}(\bar{c})^\top \mathscr{U}_i(\boldsymbol{d_i}, I, J) \in \mathbb{R}. \tag{A.1}$$

where, $\mathbf{w}(\bar{c}) \in \mathbb{R}^n$ is the $\bar{c}^{th}$ column of the weight matrix W and $\bar{c}$ is the most dominant class in the patch on the source segmentation mask $\bar{\Omega}_i^{S^I}$. The dominant class $\bar{c}$ is obtained as follows:

$$\bar{c} = \underset{c \in \mathscr{C}}{\operatorname{argmax}} f(\bar{\Omega}_i^{S^I}, c). \tag{A.2}$$

where, $f(\bar{\Omega}_i^{S^I}, c)$ is the number of voxels of class c in the patch $\bar{\Omega}_i^{S^I}$. Notice that one can use any other criteria to find the dominant class.

We now begin to define the pairwise clique potentials. The pairwise clique potential between the control points p_i and p_j is defined as $\mathscr{V}(\boldsymbol{d_i}, \boldsymbol{d_j})$, where $\mathscr{V}(.,.)$ is the L_1 norm between the two input arguments. Under this setting, the multi-class energy function corresponding to the deformable registration task is defined as:

$$\mathscr{E}(I, J, S^I, D; W) = \sum_{i \in V} \bar{\mathscr{U}_i}(\boldsymbol{d_i}, I, J, S^I; W) + \sum_{(i,j) \in E} \mathscr{V}(\boldsymbol{d_i}, \boldsymbol{d_j}) \tag{A.3}$$

Therefore, we aim at finding the optimal deformation $\hat{D}$ by solving the following problem:

$$\hat{D} = \underset{D \in \mathscr{L}^{|V|}}{\operatorname{argmin}} \mathscr{E}(I, J, S^I, D; W). \tag{A.4}$$

The Approximate Inference Algorithm. Similar to [48], we adopt a pyramidal approach that allows us to refine the search space at every level and, at the same time, capture a big range of deformations. We use the well known FastPD [77] as the inference algorithm at every level of the pyramid. Notice that the energy function (A.3) is defined over the nodes and the edges of the sparse graph G which represents the deformation field, and not over the dense voxels and the neighbourhood system defined over the input image I. The reason being that the input images are too big and thus can not be optimized efficiently. Once we obtain the optimal deformation $\hat{D}$, we estimate the dense deformation field using the free form deformation interpolation model [110] in order to warp the input image.

A.4 Learning the Parameters

In the previous section we assumed that the weight matrix W is given to us. However, knowing the weight matrix a priori is non-trivial and hand tuning it becomes infeasible very quickly as the number of metrics and the classes increases. In order to circumvent this problem, we propose an algorithm to learn the weights using a given dataset. Our algorithm is based on the well known latent structured SVM framework [122, 126, 135] which optimizes an upperbound on the empirical risk, discussed in detail in section 2.5. Instead of learning the complete weight matrix at once, we learn the weights for each class $c \in \mathscr{C}$ individually. From now onwards, the weight vector $\mathbf{w}_c$ denotes a particular column of the weight matrix W, which represents the weights corresponding to a particular class. We use the words ‘parameters’ and ‘weights’ interchangeably. In what follows we talk about the learning algorithm in details.

A.4.1 Preliminaries

Dataset. We consider a dataset $\mathscr{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1,\cdots,N}$. Each $\mathbf{x}_i$ is a pair represented as $\mathbf{x}_i = (I_i, J_i)$, where I_i is the source volume (or the source image) and J_i is the target volume (or the target image). Similarly, each $\mathbf{y}_i$ is a pair represented as $\mathbf{y}_i = (S_i^I, S_i^J)$, where S_i^I and S_i^J are the segmentation masks for the source and target images, respectively. The size of each segmentation mask is the same as that of the corresponding images. As stated earlier, the segmentation mask is formed by the elements (or voxels) $s_k \in \mathscr{C}$, where $\mathscr{C}$ is the set of classes.

The Loss Function. The loss function $\Delta(S^I, S^J) \in \mathbb{R}_{\geq 0}$ evaluates the similarity between the two segmentation masks S^I and S^J. Higher value of $\Delta(.,.)$ implies higher dissimilarity between the segmentations. Since our final evaluation is based on the DICE, therefore, we use a DICE based loss function as:

$$\Delta(S^I, S^J) = 1 - DICE(S^I, S^J). \tag{A.5}$$

We approximate the dice between the segmentation masks as defined below:

$$DICE(S^I, S^J) = 2 \sum_{i \in V} \frac{|\phi(p_i^I) \cap \phi(p_i^J)|}{|\phi(p_i^I)| + |\phi(p_i^J)|}, \tag{A.6}$$

where, $\phi(p_i^I)$ and $\phi(p_i^J)$ are the patches at the control point p_i on the segmentation masks S^I and S^J, respectively. The function $|.|$ represents the cardinality of a given set. The above approximation of the dice makes it decomposable over the nodes of the graph G. As will be discussed shortly, this decomposition allows us to train our algorithm very efficiently.

Joint Feature Map. Given the parameters $\mathbf{w}_c$ for a particular class, the deformation D, and the input tuple $\mathbf{x}$, the mutli-class energy function (A.3) can be trivially converted into class based energy function as follows:

$$\mathscr{E}_c(\mathbf{x}, D; \mathbf{w}) = \mathbf{w}_c^\top \sum_{i \in V} \mathscr{U}_i(\boldsymbol{d_i}, \mathbf{x}) + w_p \sum_{(i,j) \in E} \mathscr{V}(\boldsymbol{d_i}, \boldsymbol{d_j}). \tag{A.7}$$

where, $w_p \in \mathbb{R}_{\geq 0}$ is the parameter associated with the pairwise term. Let us denote the parameter vector $\mathbf{w} \in \mathbb{R}^{n+1}$ as the concatenation of $\mathbf{w}_c$ and w_p. Clearly, the energy function (A.7) is linear in $\mathbf{w}$ and can be written as:

$$\mathscr{E}_c(\mathbf{x}, D; \mathbf{w}) = \mathbf{w}^\top \Psi(\mathbf{x}, D). \tag{A.8}$$

where, $\Psi(\mathbf{x}, D) \in \mathbb{R}^{n+1}$ is the joint feature map defined as:

$$\Psi(\mathbf{x}, D) = \begin{pmatrix} \sum_{i \in V} \mathscr{U}_i^1(\boldsymbol{d_i}, \mathbf{x}) \\ \sum_{i \in V} \mathscr{U}_i^2(\boldsymbol{d_i}, \mathbf{x}) \\ \vdots \\ \sum_{i \in V} \mathscr{U}_i^n(\boldsymbol{d_i}, \mathbf{x}) \\ \sum_{(i,j) \in E} \mathscr{V}(\boldsymbol{d_i}, \boldsymbol{d_j}) \end{pmatrix} \tag{A.9}$$

Notice that the energy function (A.7) does not depend on the source segmentation mask S^I. The only use of source segmentation mask in the energy function (A.3) is to obtain the dominant class using the equation (A.2), which in this case is not required. However, we will shortly see that the source segmentation mask S^I plays a crucial role in the learning algorithm to compute the loss function.

Latent Variables. Ideally, the dataset $\mathscr{D}$ must contain the ground truth deformations D corresponding to the source image I in order to compute the energy term defined in the equation (A.7). Since annotating the dataset with the ground truth deformation is non-trivial, we use them as the latent variables in our algorithm. As will be seen shortly, we impute these deformations using the given dataset ensuring that the loss (as defined in the equation (A.5)) between the source segmentation mask when deformed using the imputed deformation field, and the target segmentation mask is minimized.

A.4.2 The Objective Function

Given the dataset $\mathscr{D}$, we would like to learn the parameter vector $\mathbf{w}$ such that minimizing the energy function (A.7) leads to a deformation field which when applied to the source segmentation mask gives minimum loss with respect to the target segmentation mask. Let us denote $g(S,D)$ as the deformed segmentation when the deformation D is applied to the segmentation mask S. Therefore, ideally, we would like to learn $\mathbf{w}$ such that:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{N} \sum_i \Delta(g(S_i^I, \bar{D}), S_i^J). \tag{A.10}$$

where, $\bar{D} = \operatorname{argmin}_D \mathscr{E}(\mathbf{x}_i, D; \mathbf{w})$. The above objective function is the empirical risk minimization based formulation. However, the objective function is highly non-convex in $\mathbf{w}$, therefore, minimizing it directly makes the algorithm prone to bad local minima. In order to circumvent this problem, we optimize a regularized upper bound on the loss as follows:

$$\begin{aligned}
\min_{\mathbf{w},\{\xi_i\}} \quad & \frac{1}{2}||\mathbf{w}||^2 + \alpha||\mathbf{w} - \mathbf{w}_0||^2 + \frac{C}{N}\sum_i \xi_i, \\
s.t. \quad & \min_{D, \Delta(g(S_i^I, D), S_i^J) = 0} \quad \mathbf{w}^\top \Psi(\mathbf{x}_i, D) \le \mathbf{w}^\top \Psi(\mathbf{x}_i, \bar{D}) - \Delta(g(S_i^I, \bar{D}), S_i^J) + \xi_i, \forall \bar{D}, \forall i \\
& w_p \ge 0, \xi_i \ge 0, \forall i.
\end{aligned} \tag{A.11}$$

The above objective function minimizes an upper bound on the dice based loss function denoted as the variable ξ_i, known as the slack. The first term in the objective function $||\mathbf{w}||^2$, is the regularization term used to avoid overfitting. The effect of the regularization term is controlled by the hyper-parameter C. The second term is the proximity term. This ensures that the learned $\mathbf{w}$ is close to the initialization $\mathbf{w}_0$. The effect of the proximity term can be controlled by the hyperparameter α. Intuitively, for a given input-output pair, the constraints of the above objective function enforce that the energy corresponding to the best possible deformation field, in terms of both energy and loss (in order to be semantically meaningful),

must always be less than or equal to the energy corresponding to any other deformation field with a margin proportional to the loss and some positive slack.
Notice that, the term $\min_{D,\Delta(g(S_i^I,D),S_i^J)=0} \mathbf{w}^\top \Psi(\mathbf{x}_i, D)$, which is the minumum over affine functions, is concave. Therefore, the above objective function can be easily shown to be the sum of convex and concave functions, which is nothing but difference of convex functions. Hence, non-convex. Shortly we will see how to upperbound the concave term, also known as the latent variable imputation, in order to make the problem convex which allows us to iteratively and efficiently optimize the above objective function.

A.4.3 The Learning Algorithm

While the objective function (A.11) optimizes an upper bound on the empirical risk, it is a non-convex program. Hence, it can not be optimized efficiently to obtain the optimal set of parameters. However, it can be shown that the objective function is a special family of non-convex functions known as the difference of convex functions [98], which can be seen as the sum of the convex and the concave functions. For such family of non-convex functions, the well known CCCP algorithm [138] can be used to obtain a local minima or a saddle point, discussed in detail in section 2.5. Broadly speaking, the CCCP algorithm consist of three steps — (1) upperbounding the concave part at a given $\mathbf{w}$, which leads to an affine function in $\mathbf{w}$; (2) optimizing the resultant convex function (sum of convex and affine functions is convex); (3) repeating the above steps until the objective can not be decreased beyond a given tolerance of ε.
The complete CCCP algorithm for the optimization of the objective function (A.11) is shown in the Algorithm 18. The first step of upperbounding the concave functions (Algorithm 18, Line 3) is the same as the latent imputation step, which we call the *segmentation consistent registration* problem. The second step is the optimization of the resultant convex problem (Algorithm 18, Line 4), which, in this case, is the optimization of the SSVM. The optimization leads to updating the parameters. We use the well known cutting plane algorithm [61] for this purpose. In what follows we discuss these steps in detail.

Segmentation Consistent Registration. As already discussed, the ground truth deformation is not known a priori. Thus, in this step, we generate the best possible ground truth deformation field at a given $\mathbf{w}$. This is same as the latent imputation step of the CCCP algorithm. Recall that we are interested in learning the parameters $\mathbf{w}$ such that the upper bound on the loss function, defined in equation (A.5), is minimized. This leads us to formulate the latent imputation step as an inference problem with additional constraints. These additional constraints ensure that the imputed deformation field deforms the input image such that the loss between the deformed input image and the target image is minimized. Mathematically, for a given parameter vector $\mathbf{w}$, the latent deformation is imputed by solving the following problem:

$$\hat{D}_i = \underset{D \in \mathscr{L}^{|V|}, \Delta(g(S_i^I, D), S_i^J) = 0}{\operatorname{argmin}} \mathbf{w}^\top \Psi(\mathbf{x}_i, D). \tag{A.12}$$

The above problem can be relaxed as follows:

$$\hat{D}_i = \underset{D \in \mathscr{L}^{|V|}}{\operatorname{argmin}} \mathbf{w}^\top \Psi(\mathbf{x}_i, D) + \eta \Delta(g(S_i^I, D), S_i^J). \tag{A.13}$$

where, η controls the relaxation trade-off parameter. Since the loss function used is decomposable, therefore, the above problem is equivalent to the inference problem for the deformable registration with trivial modifications on the unary potentials. Thus, can be solved efficiently in polynomial time using the FastPD based approximate inference algorithm briefly discussed in the section A.3.

Updating the Parameters. Once the latent variables have been imputed or the concave functions have been upperbounded, the resultant objective function can be written as:

$$\begin{aligned}
\min_{\mathbf{w}, \{\xi_i\}} \quad & \frac{1}{2}||\mathbf{w}||^2 + \alpha ||\mathbf{w} - \mathbf{w}_0||^2 + \frac{C}{N} \sum_i \xi_i, \\
s.t. \quad & \mathbf{w}^\top \Psi(\mathbf{x}_i, \hat{D}_i) \leq \mathbf{w}^\top \Psi(\mathbf{x}_i, \bar{D}) - \Delta(g(S_i^I, \bar{D}), S_i^J) + \xi_i, \forall \bar{D}, \forall i \\
& w_p \geq 0, \xi_i \geq 0, \forall i.
\end{aligned} \tag{A.14}$$

where, $\hat{D}_i$ is the latent deformation field imputed by solving the problem (A.13). Intuitively, the above objective function tries to learn the parameters $\mathbf{w}$ such that the energy corresponding to the imputed deformation field is always less than the energy for any other deformation field with a margin proportional to the loss function with some positive slack. Notice that the above objective function has exponential number of constraints, one for each possible deformation field $\bar{D} \in \mathscr{L}^{|V|}$. In order to alleviate this problem we use the well known cutting plane algorithm [61]. Let us briefly talk about the idea behind the cutting plane algorithm. For a given $\mathbf{w}$, each deformation field $\bar{D}$ gives a slack. Therefore, instead of minimizing all the slacks for a particular sample at once, we rather find the deformation field that leads to the maximum value of the slack and store this in a set known as the working set. This is known as finding the most violated constraint. Therefore, instead of using exponentially many constraints, we now optimize our algorithm over the constraints stored in the working set. This process is repeated till no constraints can be added to the working

Algorithm 18 The CCCP Algorithm.

```
input $\mathscr{D}$, $\mathbf{w}_0$, $C$, $\alpha$, $\eta$, the tolerance $\varepsilon$.
t = 0, w_t = w_0.
repeat
    Segmentation consistent registration. For a given $\mathbf{w}_t$, impute the latent variables $\hat{D}_i$ for each sample by solving the problem (A.13).
    Obtain the updated parameters $\mathbf{w}_{t+1}$ by solving the convex optimization problem (A.14). The most violated constraint can be found by solving the problem (A.15).
    $t = t + 1$
until The objective function of the problem (A.11) does not decrease more than $\varepsilon$.
```

set. The main ingredient of the above discussed cutting plane algorithm is *finding the most violated constraint*. As discussed earlier, the most violated constraint for the i^{th} sample is the deformation field that maximizes the slack corresponding to this sample. Rearranging the terms in the constraints of the objective function (A.14) to obtain the slack, ignoring the constant term $\mathbf{w}^\top \Psi(\mathbf{x}_i, \hat{D}_i)$, and maximizing it with respect to the possible deformations (which is equivalent to minimizing the negative of it), leads to the following problem solving which gives the most violated constraint:

$$\bar{D}_i = \underset{D \in \mathscr{L}^{|V|}}{\operatorname{argmin}} \left(\mathbf{w}^\top \Psi(\mathbf{x}_i, \bar{D}) - \Delta(g(S_i^I, \bar{D}), S_i^J) \right). \tag{A.15}$$

Since the loss function is decomposable, therefore, the above problem is equivalent to the inference problem for the deformable registration with trivial modifications on the unary potentials. Thus, can be solved efficiently in polynomial time using the FastPD based approximate inference algorithm discussed in the section A.3.

A.4.4 Prediction

Once we obtain the learned parameters $\mathbf{w}_c$ for each class $c \in \mathscr{C}$ using the Algorithm 18, we form the matrix W where each column of the matrix represents the learned parameter for a specific class. This learned matrix is then used to solve the registration problem defined in the equation (A.3) using the approximate inference algorithm discussed in the section A.3.

A.5 Experiments and Results

We evaluated our registration algorithm and the learning framework on three different challenging medical datasets – (1) RT Parotids, (2) RT Abdominal, and (3) IBSR. These datasets

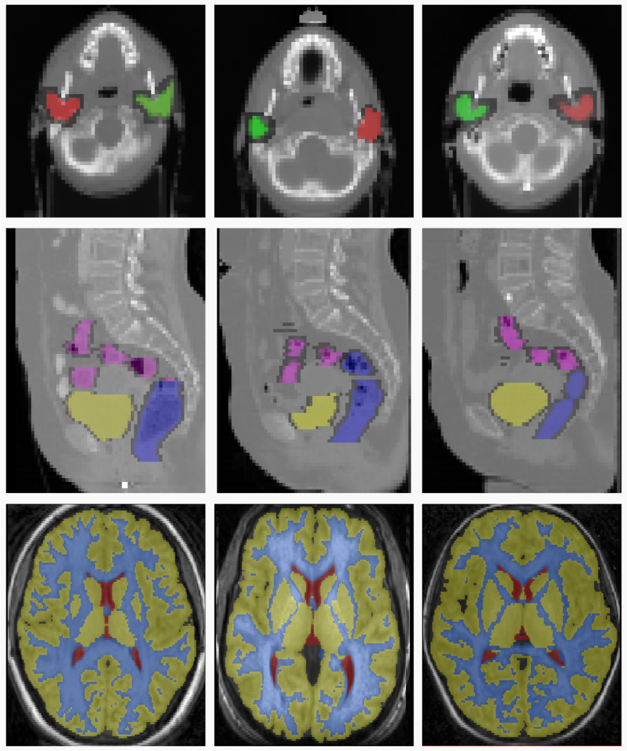


Figure A.1: The top row represents the sample slices from three different volumes of the RT Parotids dataset. The middle row represents the sample slices of the RT Abdominal dataset, and the last row represents the sample slices from the IBSR dataset.

involve several anatomical structures, different image modalities, and inter/intra patient images, which makes the deformable registration task on these dataset highly challenging. Figure A.1 shows the examples of the slices from the $3D$ volumes corresponding to each dataset. In all the experiments, we cross validate the hyper parameters C and α, and use $\eta = 50$. We use four different metrics in all the experiments: (1) sum of absolute differences (SAD), (2) mutual information (MI), (3) normalized cross correlation (NCC), and (4) discrete wavelet coefficients (DWT). In all the experiments (single and multi-metric) we used the same set of parameters for the pyramidal approach based inference algorithm (discussed in the section A.3). These parameters are as follows: 2 pyramid levels, 5 refinement steps per pyramid level, 125 labels, and distance between control points of 25mm in the finer level. The running time for each registration case is around 12 seconds. For the training, we initialized $\mathbf{w}_0$ with the hand tuned values for each metric: $\mathbf{w}_0 = (0.1, 10, 10, 10)$, for SAD, MI, NCC, and DWT, respectively. Figures A.2 and A.3 show visual results on all the three datasets. Below we give details about the different datasets and discuss the results obtained in each of them.

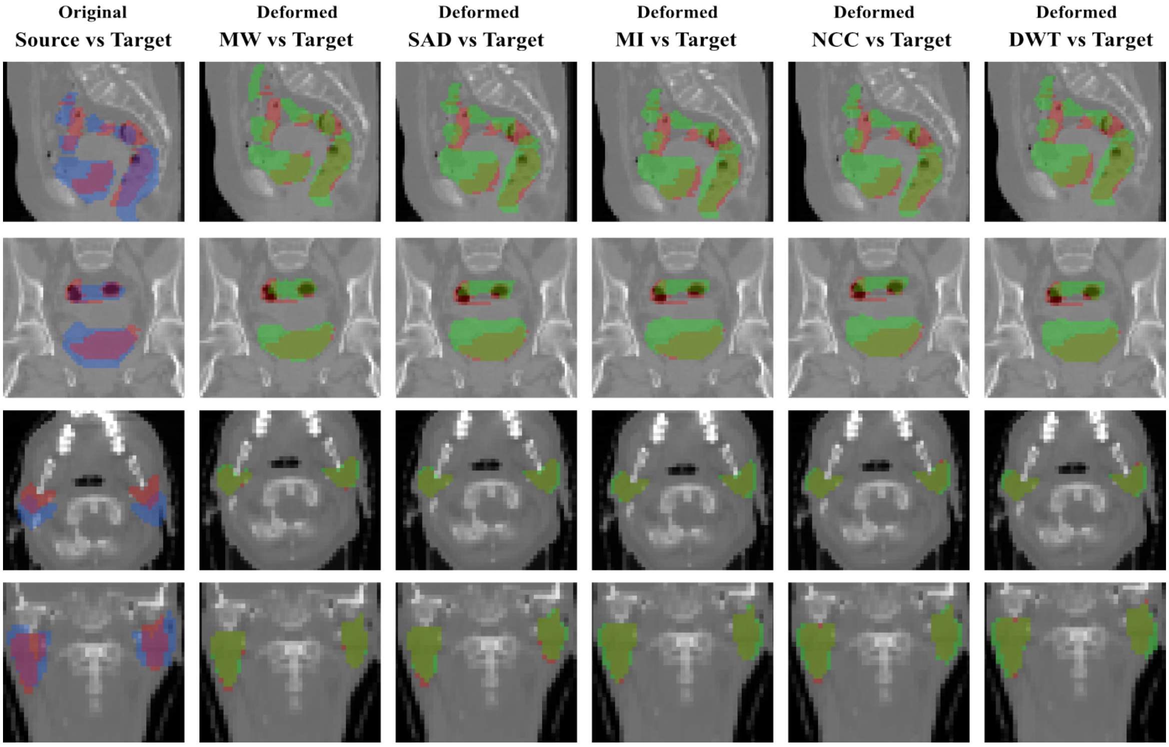


Figure A.2: Overlapping of the segmentation masks in different views for one registration case from **RT Abdominal** (first and second rows) and **RT Parotids** (third and fourth rows) datasets. The first column corresponds to the overlapping before registration between the source (in blue) and target (in red) segmentation masks of the different anatomical structures. From second to sixth column, we observe the overlapping between the warped source (in green) and the target (in red) segmentation masks, for the multiweight algorithm (MW) and the single metric algorithm using SAD, MI, NCC and DWT). We observe that MW gives a better fit between the deformed and ground truth structures than the rest of the single similarity measures, which are over segmenting most of the structures showing a poorer registration performance.

RT Parotids. The first dataset (RT Parotids) contains 8 CT volumes of head, obtained from 4 different patients, 2 volumes per patient. The volumes are captured in two different stages of a radiotherapy treatment in order to estimate the radiation dose. Right and left parotid glands were segmented by the specialists in every volume. The dimensions of the volumes are $56 \times 62 \times 53$ voxels with a physical spacing of 3.45mm, 3.45mm, and 4mm, in x, y, and z axes, respectively. We generated 8 pairs of source and target volumes using the given dataset. Notice that, while generating the source and target pairs, we did not mix the volumes coming from different patients. We splitted the dataset into train and test, and cross validated the hyperparameters C and α on the train dataset. The average results on the test dataset are shown in the Figure A.4.

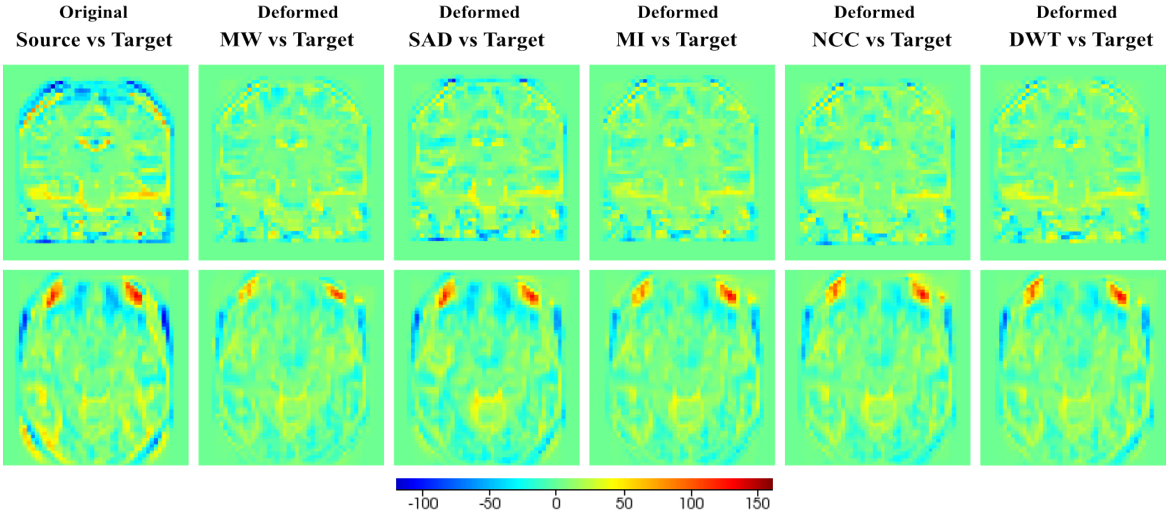


Figure A.3: Qualitative results for one slice of one registration case from **IBSR dataset**. Since showing overlapped structures in the same image is too ambiguous given that the segmentation masks almost cover the complete image, we are showing the intensity difference between the two volumes. The first column shows the difference of the original volumes before registration. From second to sixth column we observe the difference between the warped source and the target images, for the MW and the single metric algorithm using SAD, MI, NCC or DWT). According to the scale in the bottom part of the image, extreme values (which mean high differences between the images) correspond to blue and red colors, while green indicates no difference in terms of intensity. Note how most of the big differences observed in the first column (before registration) are reduced by the MW algorithm, while some of them (specially in the peripheral area of the head) remain when using single metrics.

RT Abdominal. The second dataset (RT Abdominal) contains 5 CT volumes of abdomen for a particular patient captured with a time window of about 7 days during a radiotherapy treatment. Three organs have been manually segmented by the specialists: (1) sigmoid, (2) rectum, and (3) bladder. The dimensions of the volumes are $90 \times 60 \times 80$ voxels with a physical spacing of 3.67mm, 3.67mm, and 4mm, in x, y, and z axes, respectively (there are small variations depending on the volume). We generated a train dataset of 6 pairs and test dataset of 4 pairs. The results on the test dataset are shown in the Figure A.5.

IBSR. The third dataset (IBSR) is the well known Internet Brain Segmentation Repository dataset, which consists of 18 brain MRI volumes, coming from different patients. Segmentations of three different brain structures are provided by the specialists: white mater (WM), gray mater (GM), and cerebrospinal fluid (CSF). We used a downsampled version of the dataset to reduce the computation. The dimension of the volumes are $64 \times 64 \times 64$ voxels with a physical spacing of 3.75mm, 3.75mm, and 3mm in x, y, and z axes, respectively. To

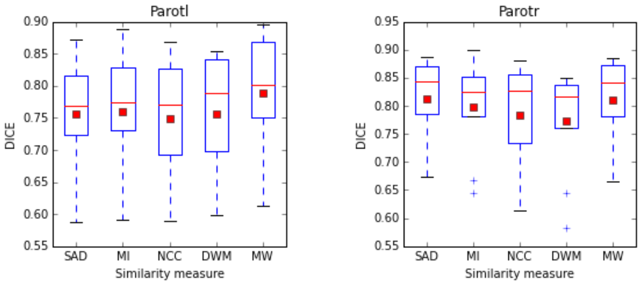


Figure A.4: Results for the RT parotids dataset for the single-metric registration (SAD, MI, NCC, DWT) and the multi-metric registration (MW). The weights for the multi-metric registration are learned using the framework proposed in this work. 'Parotl' and 'Parotr' are the left and the right organs. The red square is the mean and the red bar is the median. It is evident from the results that using the learned linear combination of the metrics outperforms the single-metric based deformable registration.

perform the experiments, we divided the 18 volumes in 2 folds of 9 volumes on each fold. This gave a total of 72 pairs per fold. We used an stochastic approach for the learning process, where we sample 10 different pairs from the training set, and we tested on the 72 pairs of the other fold. We run this experiment 3 times per fold, giving a total of 6 different experiments, with 72 testing samples and 10 training samples randomly chosen. The results on the test dataset are shown in the Figure A.6.

Results. As evident from the Figures A.4, A.5, and A.6, the linear combination of similarity measures weighted using the learned coefficients outperforms the single metric based registration. In all the cases the dice for the multi-metric is higher than the dice for the single metric based registration, or it is as good as the best dice obtained using the single metric registration (please refer to the Figure A.5, 'Parotr' to see the case in which the multi-metric is atleast as good as the best obtained using the single metric). The results for the 'Sigmoid' organ in the Figure A.5 show that in some cases the multi-metric based registration can significantly outperform the single metric based registration.

A.6 Discussions and Conclusions

In this work we introduced a novel and sophisticated framework for learning aggregations of image similarities in the context of deformable image registration. We also proposed a multi-metric MRF based image registration algorithm that incorporates such metric aggregations by weighting different similarity measures depending on the anatomical regions. We showed that associating different similarity criteria to every anatomical region yields results superior

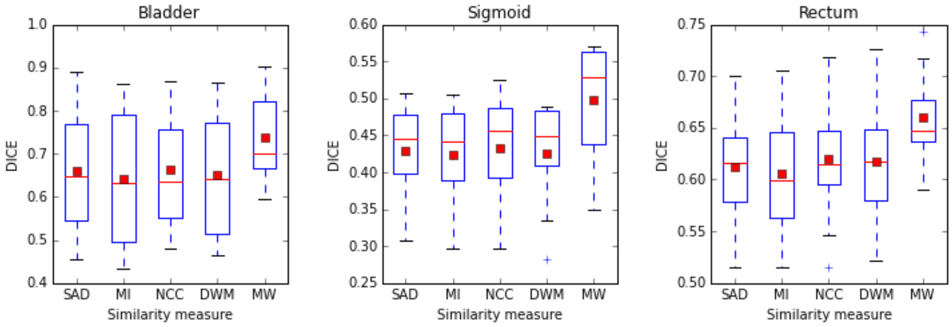


Figure A.5: Results for the RT abdominal dataset for the single-metric registration (SAD, MI, NCC, DWT) and the multi-metric registration (MW). The weights for the multi-metric registration are learned using the framework proposed in this work. ‘Bladder’, ‘Sigmoid’, and ‘Rectum’ are the three organs on the dataset. The red square is the mean and the red bar is the median. It is evident from the results that using the learned linear combination of the metrics outperforms the single-metric based deformable registration.

to the classic single metric approach. In order to learn this mapping in real scenarios where ground truth is generally given in the form of segmentation masks, we proposed to conceive deformation fields as latent variables and implement our problem using the LSSVM framework. The main limitation of our method is the need of segmentation masks for the source images in testing time. However, different real scenarios like radiation therapy or atlas-based segmentation methods fulfill this condition and can be improved through this technique. Note that, at prediction (testing) time, the segmentation mask is simply considered to determine the metrics weights combination per control node (as indicated in equation A.2). The segmentation labels are not used explicitly at testing time to guide the registration process that is purely image based. In our multi-metric registration approach, segmentation masks are only required (at testing time) for the source image and are only used to choose the best learned metric aggregation. The idea could be further extended to unlabeled data (as it concerns the source image at testing time) where the dominant label class per control node is the output of a classification/learning method.

Rather than employing highly customized solutions that suffer from scalability, portability and modularity, our approach relies on producing class-dependent metrics as linear combination of widely known and conventional mono-modal and multi-modal metrics. Consequently, the proposed registration method is scalable and modular and can be adjusted in any setting by simply changing the linear weights. These weights could be learned trivially offline using partially annotated data or could easily integrate new metrics. Extensive experimental validation on various challenging datasets demonstrated the potentials of the proposed method.

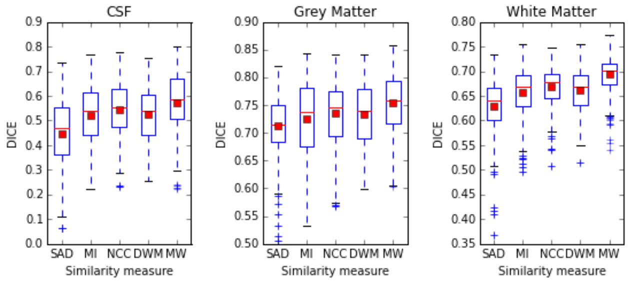


Figure A.6: Results for the IBSR dataset for the single-metric registration (SAD, MI, NCC, DWT) and the multi-metric registration (MW). The weights for the multi-metric registration are learned using the framework proposed in this work. 'CSF', 'Grey Mater', and 'White Mater' are different structures in the brain. The red square is the mean and the red bar is the median. It is evident from the results that using the learned linear combination of the metrics outperforms the single-metric based deformable registration.

The registration weights were learned by minimizing an upperbound on the DICE based loss function. DICE is conventionally used but does not offer a very convincing picture as it concerns registration performance. The integration of alternative accuracy measures such as the Hausdorff distance between surfaces or even real geometric distances for anatomical landmarks could further enhance the performance and the robustness of the method. The use of alternative parameter learning mechanisms [78] is also another interesting approach to explore. Last but not least, the use of the method on clinical applications where domain knowledge is present (for example, radiotherapy) could be considered to improve existing patient positioning practices.

Appendix B

Optimization

B.1 Brief Introduction to Lagrangian Theory

In case of unconstrained smooth convex optimization problem, the optimality condition can be obtained by equating the first derivative of the objective function equal to zero. However, in case of constrained convex optimization problems, this is not the case. We use the concept of Lagrange dual [6, 10] to study the optimality conditions of the constrained optimization problems (we talk about convex optimization problems) of the following form:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \quad f_0(\mathbf{w}) \tag{B.1}$$

$$\text{s.t.} \quad f_i(\mathbf{w}) \leq 0, i = 1, \cdots, m, \tag{B.2}$$

$$h_i(\mathbf{w}) = 0, i = 1, \cdots, p, \tag{B.3}$$

where, $\mathbf{w} \in \mathbb{R}^d$ is the optimization variable (or the primal variable), $f_0 : \mathbb{R}^d \to \mathbb{R}$ and $f_i : \mathbb{R}^d \to \mathbb{R}$ are differentiable convex functions, and $h_i : \mathbb{R}^d \to \mathbb{R}$ are the affine functions. Notice that the equality constraints are not any general convex functions. They are the affine functions, a special type of convex functions. The above problem is also called the *primal* problem. A given primal variable $\mathbf{w}$ is said to be *primal feasible* if all the constraints are satisfied at this point. Let $\mathbf{p}^*$ be the optimal solution of the above problem.

B.1.1 Lagrangian

The Lagrangian $\mathbb{L} : \mathbb{R}^d \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ of the constrained problem (B.1) is defined as:

$$\mathbb{L}(\mathbf{w}, \alpha, \beta) = f_0(\mathbf{w}) + \sum_i^m \alpha_i f_i(\mathbf{w}) + \sum_i^p \beta_i h_i(\mathbf{w}) \tag{B.4}$$

where, $\alpha \in \mathbb{R}^m$ and $\beta \in \mathbb{R}^p$ are called the *dual variables* or the *Lagrange multipliers*. Intuitively, the objective function (B.4) is the unconstrained form of the objective function (B.1). In another words, it is the weighted sum of the objective and the constraints of the problem (B.1). The multipliers α_i and β_i are the costs assigned to the constraint $g_i(\mathbf{w}) \leq 0$ and $h_i(\mathbf{w}) = 0$, respectively.

B.1.2 Lagrange dual objective

The function $g : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$, defined below, is known as the Lagrange dual objective.

$$g(\alpha,\beta) = \inf_{\mathbf{w}} \quad \mathbb{L}(\mathbf{w},\alpha,\beta) = \inf_{\mathbf{w}} \quad (f_0(\mathbf{w}) + \sum_i^m \alpha_i f_i(\mathbf{w}) + \sum_i^p \beta_i h_i(\mathbf{w})) \tag{B.5}$$

The above function is obtained by the minimization of the Lagrangian over the primal variables. Notice that the dual objective is an affine function of the dual variables.

Theorem 5. *(The Lower Bound Property:) If $\alpha_i \geq 0, \forall i$, then, $g(\alpha,\beta) \leq \mathbf{p}^*$. In another words, if $\alpha_i \geq 0$ then the primal objective is always lower bounded by the dual objective.*

Proof. The proof is rather simple. Let $\mathbf{w}$ be primal feasible (satisfies all the constraints of the primal problem), and $\alpha \geq 0$. Then,

$$f_0(\mathbf{w}) \geq f_0(\mathbf{w}) + \sum_i^m \alpha_i g_i(\mathbf{w}) + \sum_i^p \beta_i h_i(\mathbf{w}) = \mathbb{L}(\mathbf{w},\alpha,\beta) \tag{B.6}$$

$$\geq \inf_{\mathbf{w}} \quad \mathbb{L}(\mathbf{w},\alpha,\beta) = g(\alpha,\beta). \tag{B.7}$$

The inequality (B.6) comes from the fact that $\alpha_i f_i(\mathbf{w}) \leq 0$ (since $\alpha_i \geq 0$ and $f_i(\mathbf{w}) \leq 0$) and $\beta_i h_i(\mathbf{w}) = 0$ (since $h_i(\mathbf{w}) = 0$). □

B.1.3 Lagrange dual problem

Theorem 5 states that the primal objective is always lower bounded by the dual objective. Therefore, the goal of the dual problem is to find the best (or tightest possible) lower bound of the primal by solving the following maximization problem:

$$\max_{\alpha,\beta} \quad g(\alpha,\beta) \tag{B.8}$$

$$s.t. \quad \alpha \geq 0. \tag{B.9}$$

B.1.4 Strong duality and complementary slackness

Let $\mathbf{d}^*$ be the optimal solution of the dual problem (B.8). In case of convex optimization problem, the strong duality ($\mathbf{p}^* = \mathbf{d}^*$) holds if following *constraint qualification* condition (also known as the feasibility condition) is satisfied:

$$\exists \mathbf{w} : f_i(\mathbf{w}) \leq 0, \forall i = 1, \cdots, m, \tag{B.10}$$

$$h_i(\mathbf{w}) = 0, \forall i = 1, \cdots, p. \tag{B.11}$$

The above *constraint qualification* states that there must exist a primal variable such that all the constraints in the primal problem are satisfied. Let us assume that the strong duality holds with primal and dual optimal solutions as $\mathbf{w}^*$, α^*, and β^*, respectively. Thus,

$$\begin{aligned}
f_0(\mathbf{w}^*) = g(\alpha^*, \beta^*) &= \inf_{\mathbf{w}} \quad \mathbb{L}(\mathbf{w}, \alpha^*, \beta^*) \\
&= \inf_{\mathbf{w}} \quad (f(\mathbf{w}) + \sum_i^m \alpha_i^* f_i(\mathbf{w}) + \sum_i^p \beta_i^* h_i(\mathbf{w})) \\
&\leq f_0(\mathbf{w}^*) + \sum_i^m \alpha_i^* f_i(\mathbf{w}^*) + \sum_i^p \beta_i^* h_i(\mathbf{w}^*) && \text{(B.12)} \\
&\leq f_0(\mathbf{w}^*). && \text{(B.13)}
\end{aligned}$$

The strong duality is possible if and only if the inequalities (B.12) and (B.13) satisfy with equality. This is guaranteed by the following two conditions:

- *First Order Condition:* The inequality (B.12) is satisfied with equality if $\mathbf{w}^*$ is the optimal solution of $\mathbb{L}(\mathbf{w}, \alpha^*, \beta^*)$. Therefore, $\nabla_{\mathbf{w}} \mathbb{L}(\mathbf{w}, \alpha^*, \beta^*) = 0$ must be satisfied at $\mathbf{w}^*$.

- *Complementary Slackness:* The inequality (B.13) is satisfied with equality if $\alpha_i^* f_i(\mathbf{w}^*) = 0, \forall i$. Notice that $h_i(\mathbf{w}^*) = 0$ is already satisfied because of the fact that $\mathbf{w}^*$ is assumed to be primal feasible. The condition $\alpha_i^* f_i(\mathbf{w}^*) = 0, \forall i$ is known as the complementary slackness. If $\alpha_i > 0$, then $f_i(\mathbf{w}^*) = 0$, and if $f_i(\mathbf{w}^*) < 0$, then $\alpha_i = 0$.

B.1.5 Karush-Kuhn-Tucker (KKT) conditions

Combining the feasibility conditions, first order condition, and the complementary slackness condition leads to the well known KKT conditions. If strong duality holds and $\mathbf{w}$, α, and β are optimal, then following KKT conditions must be satisfied:

1. *Primal Feasibility:* $f_i(\mathbf{w}) \le 0, \forall i = 1, \cdots, m$ and $h_i(\mathbf{w}) = 0, \forall i = 1, \cdots, p$.
2. *Dual Feasibility:* $\alpha \ge 0$.
3. *First Order Condition:* $\nabla_{\mathbf{w}} \mathbb{L}(\mathbf{w}, \alpha, \beta) = 0$.
4. *Complementary Slackness:* $\alpha_i f_i(\mathbf{w}) = 0, \forall i$.

B.2 SSVM as quadratic program

In this section we present the SSVM primal and dual problems in the form of the standard quadratic programs (QP). These forms are very helpful for using off-the-shelf QP solvers. A standard quadratic program has the following form:

$$\begin{aligned} \min_{\mathbf{q}} \quad & \frac{1}{2}\mathbf{q}^\top P \mathbf{q} + \mathbf{q}^\top \mathbf{c} && \text{(QP)} \\ s.t. \quad & G\mathbf{q} \le h, \\ & H\mathbf{q} = z. && \text{(B.14)} \end{aligned}$$

where, $\mathbf{q}$ is the variable and P (also known as the Gram matrix) is a positive semidefinite matrix. Few solvers expect the matrix G to be full rank, however, this is not a necessary condition for the problem (QP).

SSVM Primal as QP The primal objective function of SSVM (equation (2.36)) can be written as:

$$\begin{aligned} \min_{\mathbf{w}, \xi} \quad & \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n} \xi_i && \text{(B.15)} \\ \text{s.t.} \quad & -\mathbf{w}^\top \Psi(\mathbf{x}_i, \mathbf{y}) - \xi_i \le -\Delta(\mathbf{y}_i, \mathbf{y}), \forall i, \forall \mathbf{y} \in \mathscr{Y}_i. \end{aligned}$$

The variables in this case are the concatenation of $\mathbf{w}$ and ξ, therefore, $\mathbf{q} = (\mathbf{w}, \xi) \in \mathbb{R}^{d+n}$. The other terms of the QP are:

$$P = \lambda \begin{pmatrix} \mathbf{1}_{d\times d} & \mathbf{0}_{d\times n} \\ \mathbf{0}_{n\times d} & \mathbf{0}_{n\times n} \end{pmatrix}, \mathbf{c} = \frac{1}{n}\begin{pmatrix} \mathbf{0}_{d\times 1} \\ \mathbf{1}_{n\times 1} \end{pmatrix}, G = \begin{pmatrix} \vdots \\ G(\mathbf{x}_i, \mathbf{y}) \\ \vdots \end{pmatrix} \in \mathbb{R}^{m\times(d+n)}, h = \begin{pmatrix} \vdots \\ -\Delta(\mathbf{y}_i, \mathbf{y}) \\ \vdots \end{pmatrix}$$

where, $G(\mathbf{x}_i, \mathbf{y}) = (-\Psi(\mathbf{x}_i, \mathbf{y}), -\mathbf{e}_i) \in \mathbb{R}^{d+n}$. The vector $\mathbf{e}_i \in \{0,1\}^n$ has $i-$th element as one and all other elements are zeros. There are no equality constraints. In the above QP, the dimension of the variable is $d+n$, however, the matrices G and h corresponding to the inequality constraints have m rows each, one row per constraint. Since m is exponentially large, no off-the-shelf QP solver can be used to optimize the above QP.

SSVM **dual as** QP The dual problem of the SSVM (equation (2.49)) can be written as:

$$\min_{\alpha} \quad \frac{\lambda}{2}\alpha^\top A^\top A\alpha - \mathbf{b}^\top \alpha \tag{B.16}$$

$$s.t. \quad \sum_{\mathbf{y}\in\mathscr{Y}_i} \alpha_i(\mathbf{y}) = 1, \forall i \in [n], \tag{B.17}$$

$$-\alpha_i(\mathbf{y}) \leq 0, \forall i \in [n], \forall \mathbf{y} \in \mathscr{Y}_i. \tag{B.18}$$

Here, α is the variable of the QP. The other terms of the QP are:

$$P = A^\top A,\ \mathbf{c} = -\mathbf{b},\ G = -\mathbb{I}_m, h = \mathbf{0}_{m\times 1},\ H = \begin{pmatrix} \mathbf{1}_{1\times|y_1|}, \mathbf{0}_{1\times|y_2|}, \cdots, \mathbf{0}_{1\times|y_n|} \\ \mathbf{0}_{1\times|y_1|}, \mathbf{1}_{1\times|y_2|}, \cdots, \mathbf{0}_{1\times|y_n|} \\ \vdots \\ \mathbf{0}_{1\times|y_1|}, \mathbf{0}_{1\times|y_2|}, \cdots, \mathbf{1}_{1\times|y_n|} \end{pmatrix}_{n\times m},\ z = \mathbf{1}_{n\times 1},$$

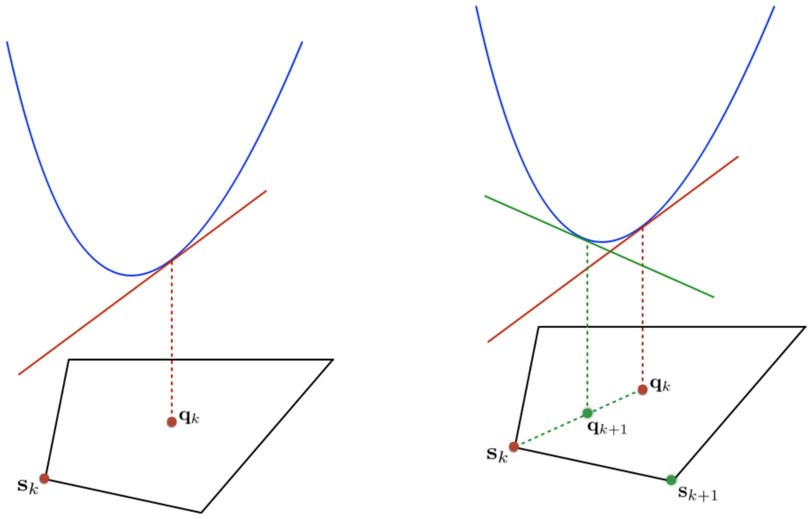
where, $\mathbb{I}_m$ is the identity matrix of size $m\times m$. Notice that, in the above QP, the variable $\alpha \in \mathbb{R}^m$ and the constraint matrices are exponentially large, therefore, no off-the-shelf QP solver can be used to optimize it.

B.3 Frank-Wolfe Algorithm

The well known Frank-Wolfe (FW) algorithm [43] is an iterative method for the optimization of the general differentiable convex functions defined over convex and compact domain:

$$\operatorname*{argmin}_{\mathbf{q}\in\mathbb{D}} f(\mathbf{q}) \tag{B.19}$$

where, f is a convex and continuously differentiable function, and the domain $\mathbb{D}$ is convex and compact. The FW algorithm is also known as the *conditional gradient method*. This algorithm is very simple yet very powerful and useful. The algorithm has following steps – (1) at a given $\mathbf{q}_k$, find the strict lowerbound of the function f by linearizing it; (2) minimize the linearized function over the compact domain to obtain a new point $\mathbf{s}$ (also called an 'atom'); (3) obtain $\mathbf{q}_{k+1}$ as the linear combination of $\mathbf{q}_k$ and $\mathbf{s}$; (4) repeat the above processes till convergence. The FW algorithm is shown in Algorithm 19 in this Appendix. For the purpose of better understanding, the steps of the FW algorithm are shown pictorially in the Figure B.1.



(a) Solving the linearization problem at $\mathbf{q}_k$.

(b) Obtained new point $\mathbf{q}_{k+1}$ and repeating the process.

Figure B.1: The blue curve shows the convex function and the black polygon shows the convex compact domain. At a given $\mathbf{q}_k$, the Figure B.1a shows the linearization and the minimization of the linearized function over the domain to obtain the atom $\mathbf{s}_k$. Figure B.1b shows the new point $\mathbf{q}_{k+1}$ obtained using the linear combination of $\mathbf{q}_k$ and $\mathbf{s}_k$. The green line shows the linearization at the new point $\mathbf{q}_{k+1}$, and $\mathbf{s}_{k+1}$ is the new atom obtained as a result of the minimization of this function.

Let us have a look into each step one by one. The *linearization problem* can be solved by finding the first order Taylor series expansion of f at a given $\mathbf{q}_k$:

$$\begin{aligned} f(\mathbf{p}) &\geq f(\mathbf{q}_k) + \langle \nabla f(\mathbf{q}_k), (\mathbf{p} - \mathbf{q}_k) \rangle && \text{(B.20)} \\ &= \underbrace{f(\mathbf{q}_k) + \langle \nabla f(\mathbf{q}_k), \mathbf{p} \rangle - \langle \nabla f(\mathbf{q}_k), \mathbf{q}_k \rangle}_{f_l(\mathbf{p})} && \text{(B.21)} \end{aligned}$$

where, $f_l(\mathbf{p})$ is the linearization (a strict under estimator) of the function f at $\mathbf{q}_k$. The minimization of the linearized function over the domain $\mathbb{D}$ is:

$$\begin{aligned} \mathbf{s} &= \underset{\mathbf{p} \in \mathbb{D}}{\operatorname{argmin}} f_l(\mathbf{p}) := f(\mathbf{q}_k) + \langle \nabla f(\mathbf{q}_k), \mathbf{p} \rangle - \langle \nabla f(\mathbf{q}_k), \mathbf{q}_k \rangle && \text{(B.22)} \\ &= \underset{\mathbf{p} \in \mathbb{D}}{\operatorname{argmin}} \langle \nabla f(\mathbf{q}_k), \mathbf{p} \rangle && \text{(B.23)} \end{aligned}$$

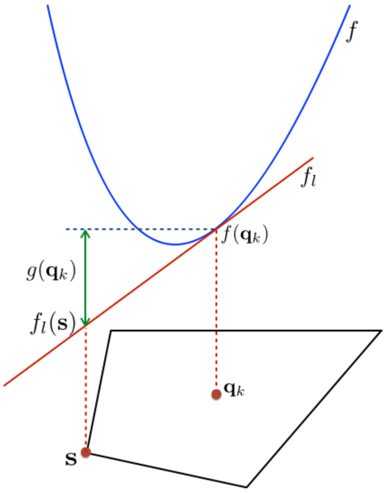


Figure B.2: Visualization of the *linearization duality gap*. The blue curve is the convex function f. The red line is the linearization of f at $\mathbf{q}_k$, denoted as f_l. The atom $\mathbf{s}$ is obtained as the result of the minization of f_l over the domain. The duality gap is $g(\mathbf{q}_k) = f(\mathbf{q}_k) - f_l(\mathbf{s})$.

The equality (B.23) comes from the fact that the quantities $f(\mathbf{q}_k)$ and $\langle \nabla f(\mathbf{q}_k), \mathbf{q}_k \rangle$ are constants at a given $\mathbf{q}_k$. Notice that solving the problem (B.23) is equivalent to solving a linear program over the compact domain $\mathbb{D}$, which can be solved very efficiently using many off-the-shelf LP solvers. Taking the convex combination of $\mathbf{q}_k$ and $\mathbf{s}$ to obtain the new point $\mathbf{q}_{k+1}$ is another crucial step. In order to ensure that the new point $\mathbf{q}_{k+1}$ is always an improvement over the old point $\mathbf{q}_k$, the step size γ must be obtained using the well known *line-search* based approach:

$$\gamma = \operatorname*{argmin}_{\gamma \in [0,1]} f(\gamma \mathbf{s} + (1-\gamma)\mathbf{q}_k) \tag{B.24}$$

In [43] γ was chosen to be $\frac{2}{k+2}$. The last and the very important step is to decide when to stop the algorithm, the *convergence criteria*. The easiest way is to decide based on an upper limit on the number of iterations. However, this may lead to suboptimal results. To get the certificate about the quality of the solution obtained, one must use the *Lagrange duality gap* based criteria. Unfortunately, it may not be very efficient or simple to obtain the Lagrange duality gap in many optimization problems. However, the Frank-Wolfe algorithm inherently allows us to obtain the *linearization duality gap*, a special case of *Fenchel duality gap*, very efficiently [60, 86]. The linearization duality gap, shown in the Figure B.2, can also be used as a convergence criteria. In what follows we give an expression of the linearization duality gap. Let $\mathbf{s}$ be the solution of the linearization problem (B.23) at a given $\mathbf{q}_k$. Then the

Algorithm 19 The Frank-Wolfe Algorithm.

```
input $K$, tolerance $\varepsilon$, $\mathbf{q}_0 \in \mathbb{D}$.
$k = 0$.
repeat
    Solve the linearized problem (B.23) to obtain the atom $\mathbf{s}$.
    Find the step size $\gamma$ – (1) either use $\gamma = \frac{2}{k+2}$; or (2) find optimal step size by solving the 'line-search' problem (B.24).
    Update: $\mathbf{q}_{k+1} = \gamma\mathbf{s} + (1-\gamma)\mathbf{q}_k,\ k = k+1$.
    Convergence criteria – (1) use upper limit ($K$) on the number of iterations ($k$); or (2) use linearization duality gap $g(\mathbf{q}_k)$ obtained using equation (B.29); or (3) use both.
until Converged.
return $\mathbf{q}_{k+1}$
```

linearization duality gap $g(\mathbf{q}_k)$ at point $\mathbf{q}_k$ is $f(\mathbf{q}_k) - f_l(\mathbf{s})$, where f_l is the linearlization of f at $\mathbf{q}_k$ (refer to the equation (B.21)). This gap can be obtained as follows:

$$g(\mathbf{q}_k) = f(\mathbf{q}_k) - \min_{\mathbf{p}\in\mathbb{D}} f_l(\mathbf{p}) \tag{B.25}$$

$$= f(\mathbf{q}_k) - \min_{\mathbf{p}\in\mathbb{D}} \Big(f(\mathbf{q}_k) + \langle \nabla f(\mathbf{q}_k), (\mathbf{p} - \mathbf{q}_k) \rangle \Big) \tag{B.26}$$

$$= -\min_{\mathbf{p}\in\mathbb{D}} \langle \nabla f(\mathbf{q}_k), (\mathbf{p} - \mathbf{q}_k) \rangle \tag{B.27}$$

$$= \langle \nabla f(\mathbf{q}_k), \mathbf{q}_k \rangle - \min_{\mathbf{p}\in\mathbb{D}} \langle \nabla f(\mathbf{q}_k), \mathbf{p} \rangle \tag{B.28}$$

$$= \langle \nabla f(\mathbf{q}_k), (\mathbf{q}_k - \mathbf{s}) \rangle \tag{B.29}$$

where, $\mathbf{s} = \mathrm{argmin}_{\mathbf{p}\in\mathbb{D}} \langle \nabla f(\mathbf{q}_k), \mathbf{p} \rangle$ is obtained by solving the linearlization problem (B.23), which is the first step of the Frank-Wolfe algorithm (please refer to the Figure B.2). Therefore, the linearization duality gap $g(\mathbf{q}_k)$ can be obtained as the byproduct of the Frank-Wolfe algorithm. Notice that, because of the fact that f_l is an strict under estimator of the function f, the linearization duality gap $g(\mathbf{q}_k)$ is always positive. Another important point to remember is that $g(\mathbf{q}_k)$ is the gap at point $\mathbf{q}_k$, not at $\mathbf{q}_{k+1}$ or $\mathbf{s}$.

References

[1] Bach, F. (2013). *Learning with Submodular Functions: A Convex Optimization Perspective*. Foundations and Trends in Machine Learning.

[2] Bach, F., Lanckriet, G. R. G., and Jordan, M. I. (2004). Multiple kernel learning, conic duality, and the smo algorithm. In *ICML*.

[3] Bartal, Y. (1998). On approximating arbitrary metrics by tree metrics. In *STOC*.

[4] Baudin, P. Y., Goodman, D., Kumar, P., Azzabou, N., Carlier, P. G., Paragios, N., and Kumar, M. P. (2013). Discriminative parameter estimation for random walks segmentation. In *MICCAI*.

[5] Behl, A., Jawahar, C. V., and Kumar, M. P. (2014). Optimizing average precision using weakly supervised data. In *CVPR*.

[6] Bertsekas, D. P. (2009). *Convex optimization theory*. Athena Scientific Belmont, MA.

[7] Besag, J. (1974). Spatial interaction and the statistical analysis of lattice systems. *Royal Statistical Society*, 36(2):192–236.

[8] Blechschmidt, K., Giesen, J., and Laue, S. (2015). Tracking approximate solutions of parameterized optimization problems over multi-dimensional (hyper-)parameter domains. In *ICML*.

[9] Bourdev, L. and Malik, J. (2009). Poselets: Body part detectors trained using 3D human pose annotations. In *IJCV*.

[10] Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

[11] Boykov, Y. and Jolly, M., P. (2000). Interactive organ segmentation using graph cuts. In *MICCAI*.

[12] Boykov, Y. and Jolly, M., P. (2001). Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *ICCV*.

[13] Boykov, Y. and Kolmogorov, V. (2003). Computing geodesics and minimal surfaces via graph cuts. In *ICCV*.

[14] Boykov, Y., Veksler, O., and Zabih, R. (1998). Markov random fields with efficient approximations. In *CVPR*.

[15] Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. In *PAMI*.

[16] Bronstein, M., Bronstein, A., Michel, F., and Paragios, N. (2010). Data fusion through cross-modality metric learning using similarity-sensitive hashing. In *CVPR*.

[17] Brown, L. G. (1992). A survey of image registration techniques. In *ACM Computing Surveys*.

[18] Bryant, D. and Tupper, P. F. (2012). Hyperconvexity and tight-span theory for diversities. In *Advances in Mathematics*.

[19] Bryant, D. and Tupper, P. F. (2014). Diversities and the geometry of hypergraphs. In *Discrete Mathematics and Theoretical Computer Science*.

[20] Chekuri, C., Khanna, S., Naor, J., and Zosin, L. (2001). Approximation algorithms for the metric labeling problem via a new linear programming formulation. In *SODA*.

[21] Cifor, A., Risser, L., Chung, D., Anderson, E. M., and Schnabel, J. A. (2012). Hybrid feature-based Log-Demons registration for tumour tracking in 2-D liver ultrasound images. In *ISBI*.

[22] Cifor, A., Risser, L., Chung, D., Anderson, E. M., and Schnabel, J. A. (2013). Hybrid feature-based diffeomorphic registration for tumor tracking in 2-D liver ultrasound images. *IEEE Transactions on Medical Imaging*.

[23] Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP*.

[24] Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. In *PAMI*.

[25] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *CVPR*.

[26] Delaitre, V., Laptev, I., and Sivic, J. (2010). Recognizing human actions in still images: a study of bag-of-features and part-based representations. In *BMVC*.

[27] Delong, A., Gorelick, L., Veksler, O., and Boykov, Y. (2012). Minimizing energies with hierarchical costs. In *IJCV*.

[28] Delong, A., Osokin, A., Isack, H., and Boykov, Y. (2010). Fast approximate energy minimization with label costs. In *CVPR*.

[29] Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression. *Annals of Statistics*.

[30] El-Zehiry, N. and Grady, L. (2010). Fast global optimization of curvature. In *CVPR*.

[31] Everingham, M., Gool, L., Williams, C., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. In *IJCV*.

[32] Fakcharoenphol, J., Rao, S., and Talwar, K. (2003). A tight bound on approximating arbitrary metrics by tree metrics. In *STOC*.

[33] Feldmar, J. and Ayache, N. (1994). Rigid and affine registration of smooth surfaces using differential properties. In *ECCV*.

[34] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2009). Object detection with discriminatively trained part based models. In *PAMI*.

[35] Felzenszwalb, P. F. and Huttenlocher, D. P. (2000). Efficient matching of pictorial structures. In *CVPR*.

[36] Ferrante, E. and Paragios, N. (2013). Non-rigid 2d-3d medical image registration using markov random fields. In *MICCAI*.

[37] Finley, T. and Joachims, T. (2008). Training structural SVMs when exact inference is intractable. In *ICML*.

[38] Fitzgibbon, A. W. (2001). Robust registration of 2d and 3d point sets. In *BMVC*.

[39] Fix, A., Joachims, T., Park, S. M., and Zabih, R. (2013). Structured learning of sum-of-submodular higher order energy functions. In *ICCV*.

[40] Fix, A., Wang, C., and Zabih, R. (2014). A primal-dual algorithm for higher-order multilabel markov random fields. In *CVPR*.

[41] Ford, L. R. and Fulkerson, D. R. (1962). *Flows in Networks*. Princeton University Press.

[42] Franc, V. and Savchynskyy, B. (2008). Discriminative learning of max-sum classifiers. In *JMLR*.

[43] Frank, M. and Wolfe, P. (1956). An algorithm for quadratic programming. In *Naval Research Logistics Quarterly*.

[44] Friedman, J., Hastie, T., and Tibshirani, R. (2007). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*.

[45] Friedman, J., Hastie, T., and Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*.

[46] Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*.

[47] Fujishige, S. (1990). Submodular functions and optimization. *Annals of Discrete Math*, 47.

[48] Glocker, B., Komodakis, N., Tziritas, G., Navab, N., and Paragios, N. (2008). Dense image registration through mrfs and efficient linear programming. *Medical Image Analysis*, 12(6).

[49] Goldberg, A. V., Hed, S., Kaplan, H., Tarjan, R. E., and Werneck, R. F. (2011). Maximum flows by incremental breadth-first search. In *European Symposium on Algorithms*.

[50] Gould, S., Amat, F., and Koller, D. (2009). Alphabet soup: A framework for approximate energy minimization. In *CVPR*.

[51] Greig, D., Porteous, B., and Seheult, A. (1989). Exact maximum a posteriori estimation for binary images. In *Journal of the Royal Statistical Society*.

[52] Hastie, T., Rosset, S., Tibshirani, R., and Zhu, J. (2004). The entire regularization path for the support vector machine. *JMLR*.

[53] Hilton, A., Stoddart, A. J., Illingworth, J., and Windeatt, T. (1996). Marching triangles: Range image fusion from complex object modelling. In *ICPR*.

[54] Hochbaum, D. S. and Hong, S. P. (1995). About strongly polynomial time algorithms for quadratic optimization over submodular constraints. In *Mathematical Programming*.

[55] Horst, R. and Thoai, N. (1999). DC programming overview. *Journal of Optimization Theory and Applications*.

[56] Ishikawa, H. and Geiger, D. (1998a). Occlusions, discontinuities, and epipolar lines in stereo. In *ECCV*.

[57] Ishikawa, H. and Geiger, D. (1998b). Segmentation by grouping junctions. In *CVPR*.

[58] J., T. (2005). A support vector method for multivariate performance measures. In *ICML*.

[59] Jaggi, M. (2011). *Sparse convex optimization methods for machine learning*. PhD thesis, ETH Zurich.

[60] Jaggi, M. (2013). Revisiting frank-wolfe: Projection-free sparse convex optimization. In *ICML*.

[61] Joachims, T., Finley, T., and Yu, C. (2009). Cutting-plane training of structural SVMs. *Machine Learning*.

[62] Jung, H. Y., Lee, K. M., and Lee, S. U. (2011). Stereo reconstruction using high order likelihood. In *ICCV*.

[63] Kassel, P. (1995). *A Comparison of Approaches to On-line Handwritten Character Recognition*. PhD thesis, MIT Spoken Language Systems Group.

[64] Kawahara, Y., Nagano, K., Tsuda, K., and Bilmes, J. A. (2009). Submodularity cuts and applications. In *NIPS*.

[65] Kleinberg, J. and Tardos, E. (1999). Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. In *FOCS*.

[66] Kohli, P., Kumar, M., and Torr, P. (2007). P3 & beyond: Solving energies with higher order cliques. In *CVPR*.

[67] Kohli, P., Ladicky, L., and Torr, P. (2008). Robust higher order potentials for enforcing label consistency. In *CVPR*.

[68] Kohli, P. and Torr, P. (2006). Measuring uncertainty in graph cut solutions – efficiently computing min-marginal energies using dynamic graph cuts. In *ECCV*.

[69] Kohli, P. and Torr, P. (2007). Dynamic graph cuts for efficient inference in Markov random fields. In *PAMI*.

[70] Kolmogorov, V. (2006). Convergent tree-reweighted message passing for energy minimization. In *PAMI*.

[71] Kolmogorov, V. (2012). Minimizing a sum of submodular functions. In *Discrete Applied Mathematics*.

[72] Kolmogorov, V. and Zabih, R. (2001). Visual correspondence with occlusions using graph cuts. In *ICCV*.

[73] Kolmogorov, V. and Zabih, R. (2002). Multi-camera scene reconstruction via graph cuts. In *ECCV*.

[74] Kolmogorov, V. and Zabih, R. (2004). What energy functions can be minimized via graph cuts. In *PAMI*.

[75] Komodakis, N. (2011). Efficient training for pairwise or higher order crfs via dual decomposition. In *CVPR*.

[76] Komodakis, N., Paragios, N., and Tziritas, G. (2011). MRF energy minimization and beyond via dual decomposition. In *PAMI*.

[77] Komodakis, N., Tziritas, G., and Paragios, N. (2007). Fast, approximately optimal solutions for single and dynamic mrfs. In *CVPR*.

[78] Komodakis, N., Xiang, B., and Paragios, N. (2015). A framework for efficient structured max-margin learning of high-order mrf models. In *PAMI*.

[79] Krause, A. and Cevher, V. (2010). Submodular dictionary selection for sparse representation. In *ICML*.

[80] Krause, A. and Golovin, D. (2014). *Submodular function maximization*. Cambridge University Press.

[81] Krause, A. and Guestrin, C. (2005). Near-optimal nonmyopic value of information in graphical models. In *UAI*.

[82] Kulesza, A. and Pereira, F. (2007). Structured learning with approximate inference. In *NIPS*.

[83] Kumar, M. P. (2014). Rounding-based moves for metric labeling. In *NIPS*.

[84] Kumar, M. P. and Koller, D. (2009). MAP estimation of semi-metric MRFs via hierarchical graph cuts. In *UAI*.

[85] Kumar, M. P., Turki, H., Preston, D., and Koller, D. (2015). Parameter estimation and energy minimization for region-based segmentation. *PAMI*.

[86] Lacoste-julien, S., Jaggi, M., Schmidt, M., and Pletscher, P. (2013). Stochastic block-coordinate Frank-wolfe optimization for structural SVMs. In *ICML*.

[87] Ladicky, L., Russell, C., Kohli, P., and Torr, P. (2010). Graph cut based inference with co-occurrence statistics. In *ECCV*.

[88] Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

[89] Lan, X., Roth, S., Huttenlocher, D., and Black, M. (2006). Efficient belief propagation with learned higher-order markov random fields. In *ECCV*.

[90] Lee, D., Hofmann, M., Steinke, F., Altun, Y., Cahill, N. D., and Scholkopf, B. (2009). Learning similarity measure for multi-modal 3D image registration. In *CVPR*.

[91] Long, J. L., Zhang, N., and Darrell, T. (2014). Do Convnets Learn Correspondence? In *NIPS*.

[92] Lukasewitz, I. and Lacoste-Julien, S. (2013). Block-coordinate frank-wolfe optimization: A study on randomized sampling methods. Technical report, Ecole Polytechnique.

[93] Maji, S., Bourdev, L., and Malik, J. (2011). Action recognition from a distributed representation of pose and appearance. In *CVPR*.

[94] Meier, L., Van De Geer, S., and Bühlmann, P. (2008). The group lasso for logistic regression. *Journal of the Royal Statistical Society*.

[95] Michel, F., Bronstein, M., Bronstein, A., and Paragios, N. (2011). Boosted metric learning for 3D multi-modal deformable registration. In *ISBI*.

[96] Mohapatra, P., Jawahar, C. V., and Kumar, M. P. (2014). Efficient optimization for average precision svm. In *NIPS*.

[97] Murphy, K. P., Weiss, Y., and Jordan, M. I. (1999). Loopy belief propagation for approximate inference: An empirical study. In *UAI*.

[98] Muu, L. and Oettli, W. (1991). Method for minimizing a convex-concave function over a convex set. *Journal of Optimization Theory and Applications*.

[99] Needell, D., Srebro, N., and Ward, R. (2014). Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. In *arXiv preprint arXiv:1310.5715v3*.

[100] Nesterov, Y. (2012). Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*.

[101] Oliva, A. and Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. In *IJCV*.

[102] Ong, C. J., Shao, S. Y., and Yang, J. B. (2010). An improved algorithm for the solution of the regularization path of svm. *IEEE Transactions on Neural Networks*.

[103] Ou, Y., Sotiras, A., Paragios, N., and Davatzikos, C. (2010). Dramms: Deformable registration via attribute matching and mutual-saliency weighting. In *Medical Image Analysis*.

[104] Park, M. Y. and Hastie, T. (2007). L1 regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society*.

[105] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

[106] Potts, R. B. (1952). Some generalized order-disorder transformations. In *Cambridge Philosophical Society*.

[107] Ramalingam, S., Taguchi, Y., Marks, T., and Tuzel, O. (2010). P2⊓: A minimal solution for registration of 3d points to 3d plane. In *ECCV*.

[108] Ratliff, N. D., Bagnell, J. A., and Zinkevich, M. A. (2007). (online) subgradient methods for structured prediction. In *AISTATS*.

[109] Rosenfeld, N., Meshi, O., Globerson, A., and Tarlow, D. (2014). Learning structured models with the AUC loss and its generalizations. In *AISTAT*.

[110] Rueckert, D., Sonoda, L. I., Hayes, C., Hill, D. L. G., Leach, M. O., and Hawkes, D. J. (1999). Nonrigid registration using free-form deformations: Application to breast mr images. In *IEEE TMI*.

[111] Scharstein, D. and Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *IJCV*.

[112] Schlesinger, D. and Flach, B. (2006). Transforming an arbitrary minsum problem into a binary one. Technical report, TU Dresden.

[113] Shalev-Shwartz, S., Singer, Y., Srebro, N., and Cotter, A. (2007). Pegasos: Primal estimated sub-gradient solver for svm. In *ICML*.

[114] Shor, N. Z., Kiwiel, K. C., and Ruszcayǹski, A. (1985). *Minimization Methods for Non-differentiable Functions*. Springer-Verlag New York.

[115] Sotiras, A., Davatzikos, C., and Paragios, N. (2014). Deformable medical image registration: A survey. *IEEE TMI*.

[116] Studholme, C., Hill, D., and Hawkes, D. (1995). Automated 3d registration of truncated mr and ct images of the head. In *BMVC*.

[117] Szeliski, R. (2006). *Image alignment and stitching: A tutorial*. Foundations and Trends in Computer Graphics and Vision.

[118] Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., and Rother, C. (2008). A comparative study of energy minimization methods for markov random fields with smoothness-based priors. In *PAMI*.

[119] Szummer, M., Kohli, P., and Hoiem, D. (2008). Learning crfs using graph cuts. In *ECCV*.

[120] Tappen, M. and Freeman, W. (2003). Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters. In *ICCV*.

[121] Tarlow, D., Givoni, I., and Zemel, R. (2010). Hop-map: Efficient message passing with high order potentials. In *AISTATS*.

[122] Taskar, B., Guestrin, C., and Koller, D. (2003). Max-margin Markov networks. In *NIPS*.

[123] Taskar, B., Lacoste-Julien, S., and Jordan, M. I. (2006). Structured prediction, dual extragradient and bregman projections. In *JMLR*.

[124] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*.

[125] Toga, A. W. (2008). Learning based coarse-to-fine image registration. *CVPR*.

[126] Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In *ICML*.

[127] Vapnik, V. (1998). *Statistical learning theory*. Wiley.

[128] Veksler, O. (1999). *Efficient Graph-based Energy Minimization Methods in Computer Vision*. PhD thesis, Cornell University.

[129] Vicente, S., Kolmogorov, V., and Rother, C. (2009). Joint optimization of segmentation and appearance models. In *ICCV*.

[130] Wachinger, C. and Navab, N. (2010). Manifold Learning for Multi-Modal Image Registration. In *BMVC*.

[131] Woodford, O., Torr, P., Reid, I., and Fitzgibbon, A. (2008). Global stereo reconstruction under second order smoothness priors. In *CVPR*.

[132] Wu, G., Kim, M., Wang, Q., Gao, Y., Liao, S., and Shen, D. (2013). Unsupervised Deep Feature Learning for Deformable Registration of MR Brain Images. In *MICCAI*.

[133] Yang, L. and Jin, R. (2006). Distance metric learning: A comprehensive survey. *Department of Computer Science, Michigan State University*.

[134] Yang, Y. and Ramanan, D. (2011). Articulated pose estimation using flexible mixtures of parts. In *CVPR*.

[135] Yu, C. N. and Joachims, T. (2009). Learning structural svms with latent variables. In *ICML*.

[136] Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society*.

[137] Yue, Y., Finley, T., Radlinski, F., and Joachims, T. (2007). A support vector method for optimizing average precision. In *SIGIR*.

[138] Yuille, A. and Rangarajan, A. (2003). The concave-convex procedure. *Neural Computation*.

[139] Zhao, P. and Zhang, T. (2015). Stochastic optimization with importance sampling for regularized loss minimization. In *ICML*.

[140] Zhu, J., Rosset, S., Hastie, T., and Tibshirani, R. (2004). 1-norm support vector machines. In *NIPS*.

[141] Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society*.