Journal of Machine Learning Research 9 (2008) 67-104 Submitted 3/07; Revised 9/07; Published 1/08

Discriminative Learning of Max-Sum Classifiers

Vojtěch Franc[*] XFRANCV@CMP.FELK.CVUT.CZ
Fraunhofer-FIRST.IDA
Kekuléstrasse 7
12489 Berlin, Germany

Bogdan Savchynskyy BOGDAN@IMAGE.KIEV.UA
The International Research and Training Centre of Information Technologies and Systems
Prospect Akademika Glushkova, 40
Kiev, Ukraine, 03680

Editor: Marina Meilă

Abstract

The max-sum classifier predicts *n*-tuple of labels from *n*-tuple of observable variables by maximizing a sum of quality functions defined over neighbouring pairs of labels and observable variables. Predicting labels as MAP assignments of a Random Markov Field is a particular example of the max-sum classifier. Learning parameters of the max-sum classifier is a challenging problem because even computing the response of such classifier is NP-complete in general. Estimating parameters using the Maximum Likelihood approach is feasible only for a subclass of max-sum classifiers with an acyclic structure of neighbouring pairs. Recently, the discriminative methods represented by the perceptron and the Support Vector Machines, originally designed for binary linear classifiers, have been extended for learning some subclasses of the max-sum classifier. Besides the max-sum classifiers with the acyclic neighbouring structure, it has been shown that the discriminative learning is possible even with arbitrary neighbouring structure provided the quality functions fulfill some additional constraints. In this article, we extend the discriminative approach to other three classes of max-sum classifiers with an arbitrary neighbourhood structure. We derive learning algorithms for two subclasses of max-sum classifiers whose response can be computed in polynomial time: (i) the max-sum classifiers with supermodular quality functions and (ii) the max-sum classifiers whose response can be computed exactly by a linear programming relaxation. Moreover, we show that the learning problem can be approximately solved even for a general max-sum classifier.

Keywords: max-xum classifier, hidden Markov networks, support vector machines

1. Introduction

Let $(\mathcal{T},\mathcal{E})$ be an undirected graph, where $\mathcal{T}$ is a finite *set of objects* and $\mathcal{E} \subseteq \binom{\mathcal{T}}{2}$ is a set of object pairs defining *a neighborhood structure*. A pair $\{t,t'\}$ of objects belonging to $\mathcal{E}$ will be called *neighbors* or *neighboring objects*. Let each object $t \in \mathcal{T}$ be characterized by an observation x_t and a label y_t which take values from a finite set $\mathcal{X}$ and $\mathcal{Y}$, respectively. We denote an ordered $|\mathcal{T}|$-tuple of observations by $\mathbf{x} = (x_t \in \mathcal{X} \mid t \in \mathcal{T})$ and an ordered $|\mathcal{T}|$-tuple of labels by $\mathbf{y} = (y_t \in \mathcal{Y} \mid t \in \mathcal{T})$. Each object $t \in \mathcal{T}$ is assigned a function $q_t \colon \mathcal{Y} \times \mathcal{X} \to \mathbb{R}$ which determines a *quality* of a label y_t given an observation x_t. Each object pair $\{t,t'\} \in \mathcal{E}$ is assigned a function $g_{tt'} \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$

∗. Secondary address for the first author is Center for Machine Perception, Czech Technical University, Faculty of Electrical Engineering, Technicka 2, 166 27 Prague 6, Czech Republic.

which determines the *quality* of labels y_t and $y_{t'}$. We adopt the convention $g_{tt'}(y,y') = g_{t't}(y',y)$. Let $\mathbf{q} \in \mathbb{R}^{|\mathcal{T}||\mathcal{X}||\mathcal{Y}|}$ and $\mathbf{g} \in \mathbb{R}^{|\mathcal{E}||\mathcal{Y}|^2}$ be ordered tuples which contain elements $q_t(y,x)$, $t \in \mathcal{T}$, $x \in \mathcal{X}$, $y \in \mathcal{Y}$ and $g_{tt'}(y,y')$, $\{t,t'\} \in \mathcal{E}$, $y,y' \in \mathcal{Y}$, respectively. We consider a class of *structured classifiers* $f\colon \mathcal{X}^{\mathcal{T}} \to \mathcal{Y}^{\mathcal{T}}$ parametrized by $(\mathbf{q},\mathbf{g})$ that predict labeling $\mathbf{y}$ from observations $\mathbf{x}$ by selecting the labeling with the maximal quality, that is,

$$f(\mathbf{x};\mathbf{q},\mathbf{g}) = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}^{\mathcal{T}}} F(\mathbf{x},\mathbf{y};\mathbf{q},\mathbf{g}) = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}^{\mathcal{T}}} \left[\sum_{t\in\mathcal{T}} q_t(y_t,x_t) + \sum_{\{t,t'\}\in\mathcal{E}} g_{tt'}(y_t,y_{t'}) \right]. \tag{1}$$

We will call the classification rule of the form (1) a *max-sum classifier*. The problem of computing the output of the max-sum classifier, that is, the evaluation of the right-hand side of (1) is called the *max-sum labeling problem* or shortly the *max-sum problem* (it is also known as the weighted constraint satisfaction problem). The max-sum problem is known to be NP-complete in general. We will use the six-tuple $(\mathcal{T},\mathcal{E},\mathcal{Y},\mathbf{q},\mathbf{g},\mathbf{x})$ to denote a max-sum problem instance which must be solved to classify $\mathbf{x}$.

This article deals with a problem of learning the parameters $(\mathbf{q},\mathbf{g})$ of a max-sum classifier from a finite training set $\mathcal{L} = \left\{(\mathbf{x}^j,\mathbf{y}^j) \in \mathcal{X}^{\mathcal{T}} \times \mathcal{Y}^{\mathcal{T}} \mid j \in \{1,\dots,m\}\right\}$. Henceforth we will use the shortcut $\mathcal{J} = \{1,\dots,m\}$.

A typical example of a max-sum classifier is the maximum aposteriori (MAP) estimation in Markov models. In this case, the observations $\mathbf{x}$ and the labels $\mathbf{y}$ are assumed to be realizations of random variables $\mathbf{X} = (X_t \mid t \in \mathcal{T})$ and $\mathbf{Y} = (Y_t \mid t \in \mathcal{T})$. It is assumed that only the pairs of variables (X_t,Y_t), $t \in \mathcal{T}$ and $(Y_t,Y_{t'})$, $\{t,t'\} \in \mathcal{E}$ are directly statistically dependent. Then the joint probability of $\mathbf{X}$ and $\mathbf{Y}$ is given by the Gibbs distribution

$$P(\mathbf{x},\mathbf{y};\mathbf{q},\mathbf{g}) = \frac{1}{Z} \exp F(\mathbf{x},\mathbf{y};\mathbf{q},\mathbf{g}), \tag{2}$$

where Z is the *partition function* which normalizes the distribution. The optimal Bayesian classifier which minimizes the probability of misclassification $f(\mathbf{x}) \neq \mathbf{y}$ assigns labels according to $\mathbf{y}^* = \operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}^{\mathcal{T}}} P(\mathbf{y}\,|\,\mathbf{x})$. It is easy to see that the classifier (1) coincides with the optimal one which minimizes the misclassifications.

Applications of the classifier (1) are image de-noising (Besag, 1986), image labeling (Chou and Brown, 1990) stereo matching (Boykov et al., 2001), natural language processing (Collins, 2002), 3D image segmentation (Anguelov et al., 2005), etc.

1.1 Existing Approaches to Learning Max-Sum Classifiers

A *generative approach* to learning a max-sum classifier is based on an estimation of parameters $(\mathbf{q},\mathbf{g})$ of the Gibbs distribution (2). Having the distribution estimated, a classifier minimizing an expected risk (Bayesian risk) for a given loss function can be inferred using the Bayesian decision making framework. Maximum-Likelihood (ML) estimation methods are well known for Markov models with an acyclic graph $(\mathcal{T},\mathcal{E})$ (see, for example, Schlesinger and Hlaváč, 2002). In the general case, however, the ML estimation is not tractable because no polynomial time algorithm is known for computing a partition function exactly. Approximate methods to compute the partition function are based on a Gibbs sampler (Hinton and Sejnowski, 1986; Jerrum and Sinclair, 1993). Another disadvantage of the generative approach is the fact that little is known about an expected risk of the classifiers inferred from an imprecisely estimated statistical model.

A *discriminative approach* is an alternative method which does not require explicit modeling of the underlying probability distribution. It is based on a direct optimization of classifier parameters $(\mathbf{q},\mathbf{g})$ in order to minimize an error estimate of a classifier performance computed on a finite training set. It is easy to see that the score function $F(\mathbf{x},\mathbf{y};\mathbf{q},\mathbf{g})$ of the max-sum classifier (1) is linear in its parameters $(\mathbf{q},\mathbf{g})$ which allows to exploit methods for learning linear classifiers. In the case of a consistent training set, that is, if there exists a classifier with zero empirical error, the problem of learning a linear classifier can be expressed as a problem of satisfying a set of linear inequalities. This problem is efficiently solvable by the perceptron algorithm. Variants of the perceptron algorithm for learning parameters of the max-sum classifier (1) with a chain and tree neighborhood structure were published in Schlesinger and Hlaváč (2002) and Collins (2002). These algorithms exploit the fact that a perceptron can be used whenever the response of the learned classifier can be computed efficiently. This applies for the acyclic neighbourhood structure since the response of the max-sum classifier can be computed by the *dynamic programming* (DP).

The *Support Vector Machines* (SVM) (Vapnik, 1998) are another representative of a discriminative approach for learning linear classifiers which has proved to be successful in numerous applications. Unlike the perceptron algorithm, the SVMs allows learning also from an inconsistent training set. Learning is formulated as minimization of a regularized risk functional which can be further transformed to a convex quadratic programming (QP) task suitable for optimization. The original SVMs are designed for learning the classifiers which estimate a single label. In the recent years, the SVMs have been extended for learning linear classifiers which can estimate a set of interdependent labels. In particular, the Hidden Markov Support Vector Machines (Altun and Hofmann, 2003; Altun et al., 2003) and the Max-Margin Markov Networks (Taskar et al., 2004b) were proposed for learning max-sum classifiers with an acyclic neighboring structure. In this case, learning requires solving a QP task with a huge number of linear constraints proportional to the cardinality of the output space of the classifier. In analogy to the perceptron, this task is tractable if there exists an efficient algorithm that solves the *loss-augmented classification* (LAC) task which involves optimizing (1) with the loss function added to the objective function $F(\mathbf{x},\mathbf{y};\mathbf{q},\mathbf{g})$ (c.f. Section 3.2 for details). For additively decomposable loss functions, the LAC task becomes an instance of the max-sum problem easily solvable by the DP provided the neighbourhood structure is acyclic.

Learning of the Associative Markov Networks (AMN) with an arbitrary neighbourhood structure $\mathcal{E}$ was proposed by Taskar et al. (2004a). The AMN is the max-sum classifier with the quality functions $\mathbf{g}$ restricted in a way similar to the Potts model. The LAC task was approximated by a linear programming (LP) relaxation specially derived for the AMN model. Incorporating an LP relaxation into the SVM QP task, Taskar et al. (2004a) constructed a new compact QP task which has only a polynomial number of constraints. Even though the resulting QP task is polynomially solvable, general purpose QP solvers do not provide a practical solution since they scale poorly with the problem and training set size. Recently, Taskar et al. (2006) proposed a reformulation of the structured learning problem as a convex-concave saddle-point problem which is efficiently solvable by the dual extragradient algorithm. This new framework is applicable for structured classifiers for which a certain projection step can be solved. Taskar et al. (2006) showed that the projection step is tractable for the max-sum classifiers with supermodular functions $\mathbf{g}$ and binary labels $|\mathcal{Y}| = 2$.

Tsochantaridis et al. (2005) proposed a general framework for learning linear classifiers with an interdependent labels. Their approach is based on solving the underlying SVM QP task by a cutting plane algorithm which requires as a subroutine an algorithm solving the LAC task for the particular classifier. The approach cannot be directly applied for the general max-sum classifiers since the

LAC task is not tractable. An alternative approach to the cutting plane algorithm was proposed by Ratliff and Bagnell (2006) who used subgradient methods for optimizing the SVM QP task. This approach also relies on an efficient solution to the LAC task.

An approximated cutting plane algorithm was proposed in Finley and Joachims (2005) to learn a specific structured model for correlation clustering. The inference as well as the corresponding LAC task of this clustering model are NP-complete. The authors suggested to replace an exact solution of the LAC task required in the cutting plane algorithm by its polynomially solvable LP relaxation.

1.2 Contributions

In this article, we build on the previously published approaches which use the perceptron and the SVMs for learning the max-sum classifiers. We propose learning algorithms for three classes of max-sum classifiers for which, up to our knowledge, the discriminative approaches have not been applied yet. Namely, the contributions of the article are as follows:

- We formulate and solve the problem of learning the supermodular max-sum classifier with an arbitrary neighbourhood structure $\mathcal{E}$ and without any restriction on the number of labels $|\mathcal{Y}|$ (Taskar et al., 2006, consider only two labels). We propose a variant of the perceptron algorithm for learning from a consistent training set. For an inconsistent training set, we extend the cutting plane algorithm of Tsochantaridis et al. (2005) such that it maintains the quality functions $\mathbf{g}$ supermodular during the course of the algorithm thus making the LAC task efficiently solvable.

- We formulate and solve the problem of learning the *max-sum classifier with a strictly trivial equivalent* which is a subclass of *max-sum classifiers with a trivial equivalent*. We will show in Section 2 that the latter class can be equivalently characterized by the fact that the LP relaxation (Schlesinger, 1976; Koster et al., 1998; Chekuri et al., 2001; Wainwright et al., 2002) is tight for it. It is known, that the class of problems with a trivial equivalent contains acyclic problems (Schlesinger, 1976) and supermodular problems (Schlesinger and Flach, 2000). We will extend this result to show that problems with a strictly trivial equivalent which we can learn contain the acyclic and the supermodular max-sum problems with a unique solution.

  Learning of the max-sum classifiers with a strictly trivial equivalent leads to an optimization problem with a polynomial number of linear constraints. We propose variants of the perceptron and of the cutting plane algorithm for learning from a consistent and an inconsistent training set, respectively. In this case, the LAC task does not require any specialized max-sum solver since it can be solved exhaustively. Moreover, the QP task to which we transform the learning problem is of the same form as the QP task required for ordinary multi-class SVMs (Crammer and Singer, 2001) which allows using existing optimization packages.

- We formulate the problem of learning the general max-sum classifier, that is, without any restriction on the neighborhood structure $\mathcal{E}$ and the quality functions $\mathbf{g}$. We show that the learning problem can be solved approximately by a variant of the cutting plane algorithm proposed by Finley and Joachims (2005) which uses the LP relaxation to solve the LAC task approximately. In contrast to Taskar et al. (2004a), we do not incorporate the LP relaxation

to the SVM QP task but we keep it separately which allows to exploit existing solvers for LP relaxation of the max-sum problem.

For a simplicity, we will concentrate on the max-sum classifier (1). All the proposed methods, however, are applicable whenever the set of labels $\mathcal{Y}$ is finite and the quality functions are linear in parameters, that is, they are of the form $q_t(x_t,y_t) = \langle \mathbf{w}, \Psi_t(x_t,y_t)\rangle$ and $g_{tt'}(y_t,y_t) = \langle \mathbf{w}, \Psi_{tt'}(y_t,y_{t'})\rangle$ where $\mathbf{w} \in \mathbb{R}^d$ is a parameter vector and $\Psi_t \colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$, $\Psi_{tt'} \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^d$ are arbitrary fixed mappings. Furthermore, all the proposed algorithms can be introduced in the form containing only dot products between $\Psi_t(x,y)$ and $\Psi_{tt'}(y,y')$ which allows to use the kernel functions (Vapnik, 1998).

1.3 Structure of the Article

The article is organized as follows. In Section 2, we describe three classes of the max-sum classifiers for which we will later derive learning algorithms. Perceptron algorithm and the SVM framework for learning linear classifiers with an interdependent labels is reviewed in Section 3. In Section 4, we formulate problems of learning the max-sum classifiers from a consistent training set and we propose variants of the perceptron algorithm for their solution. In Section 5, we formulate problems of learning the max-sum classifiers from inconsistent training set using the SVM framework. In Section 6, we propose an extended cutting plane algorithm to solve the learning problem defined in Section 5. Section 7 describes experiments. Finally, in Section 8 we give conclusions.

2. Classes of Max-Sum Problems

In the rest of the article, we consider the max-sum problems $P = (\mathcal{T}, \mathcal{E}, \mathcal{Y}, \mathbf{q}, \mathbf{g}, \mathbf{x})$ in which $(\mathcal{T}, \mathcal{E})$ is an arbitrary undirected connected graph and $\mathbf{q}$ are arbitrary quality functions. The three classes of the max-sum problems described below differ in the structure of the finite set of labels $\mathcal{Y}$ and the quality functions $\mathbf{g}$.

2.1 General Max-Sum Problem and LP Relaxation

The first class which we consider is the general max-sum problem with no restrictions imposed on $\mathcal{Y}$ and $\mathbf{g}$. Solving (1) when P is a general max-sum problem is known to be NP-complete. An approximate solution can be found by the linear programming (LP) relaxation proposed independently by Schlesinger (1976), Koster et al. (1998), Chekuri et al. (2001), and Wainwright et al. (2002). We introduce only the basic concepts of the LP relaxation necessary for this article. For more details on the topic we refer to the mentioned publications or to a comprehensive survey in Werner (2007) from which we adopted notation and terminology.

Let $(\mathcal{T} \times \mathcal{Y}, \mathcal{E}_{\mathcal{Y}})$ denote an undirected graph with edges $\mathcal{E}_{\mathcal{Y}} = \{\{(t,y),(t',y')\} \mid \{t,t'\} \in \mathcal{E}, y,y' \in \mathcal{Y}\}$. This graph corresponds to the *trellis diagram* used to visualize Markov chains. Each *node* $(t,y) \in \mathcal{T} \times \mathcal{Y}$ and each *edge* $\{(t,y),(t',y')\} \in \mathcal{E}_{\mathcal{Y}}$ is assigned the numbers $\beta_t(y)$ and $\beta_{tt'}(y,y')$, respectively. Let $\boldsymbol{\beta} \in \mathbb{R}^{|\mathcal{T}||\mathcal{Y}|+|\mathcal{E}||\mathcal{Y}|^2}$ be an ordered tuple which contains elements $\beta_t(y)$, $(t,y) \in \mathcal{T} \times \mathcal{Y}$ and $\beta_{tt'}(y,y')$, $\{(t,y),(t',y')\} \in \mathcal{E}_{\mathcal{Y}}$. Let Λ_P denote a set of *relaxed labelings* which contains vectors $\boldsymbol{\beta}$ satisfying

$$\sum_{y' \in \mathcal{Y}} \beta_{tt'}(y,y') = \beta_t(y)\,, \;\; \{t,t'\} \in \mathcal{E}\,, y \in \mathcal{Y}\,, \qquad \sum_{y \in \mathcal{Y}} \beta_t(y) = 1\,, \;\; t \in \mathcal{T}\,, \qquad \boldsymbol{\beta} \geq \mathbf{0}\,.$$

The LP relaxation of (1) reads

$$\boldsymbol{\beta}^* = \operatorname*{argmax}_{\boldsymbol{\beta}\in\Lambda_P} \left[\sum_{t\in\mathcal{T}} \sum_{y\in\mathcal{Y}} \beta_t(y) q_t(y,x_t) + \sum_{(t,t')\in\mathcal{E}} \sum_{(y,y')\in\mathcal{Y}^2} \beta_{tt'}(y,y') g_{tt'}(y,y') \right]. \tag{3}$$

It can be seen that solving (3) with an additional constraint $\boldsymbol{\beta} \in \{0,1\}^{\mathcal{T}}$ is an integer programming problem equivalent to (1). Further, we will introduce concepts of *equivalent problems*, *equivalent transformations* and *trivial problems* which are tightly connected to the LP relaxation.

A representation of the max-sum problem $P = (\mathcal{T}, \mathcal{E}, \mathcal{Y}, \mathbf{q}, \mathbf{g}, \mathbf{x})$ is not minimal since there exists an infinite number of equivalent max-sum problems $P' = (\mathcal{T}, \mathcal{E}, \mathcal{Y}, \mathbf{q}', \mathbf{g}', \mathbf{x})$ with different quality functions $(\mathbf{q}', \mathbf{g}')$ but the same quality for all labelings: the max-sum problems P and P' are called *equivalent* if $F(\mathbf{x},\mathbf{y};\mathbf{q},\mathbf{g}) = F(\mathbf{x},\mathbf{y};\mathbf{q}',\mathbf{g}')$ for all $\mathbf{y} \in \mathcal{Y}^{\mathcal{T}}$ (Schlesinger, 1976; Wainwright et al., 2002).

Next, we introduce a transformation of a max-sum problem to its arbitrary equivalent. These *equivalent transformations* are originally due to Schlesinger (1976) and recently are also known as *reparametrization* in Wainwright et al. (2002) and Kolmogorov (2006). Let $\varphi_{tt'} \colon \mathcal{Y} \to \mathbb{R}$ and $\varphi_{t't} \colon \mathcal{Y} \to \mathbb{R}$ be a pair of functions introduced for each pair of neighbouring objects $\{t,t'\} \in \mathcal{E}$, that is, we have $2|\mathcal{E}|$ functions in total. The value $\varphi_{tt'}(y)$ is called *potential* at label y of an object t in the direction t'. Note that the potentials correspond to messages in belief propagation (see, for example, Pearl, 1988; Yedidia et al., 2005). We will use $\boldsymbol{\varphi} \in \mathbb{R}^{2|\mathcal{E}||\mathcal{Y}|}$ to denote an ordered tuple which contains elements $\varphi_{tt'}(y)$, $\{t,t'\} \in \mathcal{E}$, $y \in \mathcal{Y}$ and $\varphi_{t't}(y')$, $\{t,t'\} \in \mathcal{E}$, $y' \in \mathcal{Y}$. Let $\mathcal{N}(t) = \{t' \in \mathcal{T} \mid \{t,t'\} \in \mathcal{E}\}$ denote the set of objects neighbouring with the object $t \in \mathcal{T}$. Finally, let $P^{\boldsymbol{\varphi}} = (\mathcal{T}, \mathcal{E}, \mathcal{Y}, \mathbf{q}^{\boldsymbol{\varphi}}, \mathbf{g}^{\boldsymbol{\varphi}}, \mathbf{x})$ denote the max-sum problem constructed from the max-sum problem $P = (\mathcal{T}, \mathcal{E}, \mathcal{Y}, \mathbf{q}, \mathbf{g}, \mathbf{x})$ by the following transformation

$$g^{\boldsymbol{\varphi}}_{tt'}(y,y') = g_{tt'}(y,y') + \varphi_{tt'}(y) + \varphi_{t't}(y'), \quad \{t,t'\} \in \mathcal{E}, y, y' \in \mathcal{Y}, \tag{4a}$$

$$q^{\boldsymbol{\varphi}}_{t}(y,x_t) = q_t(y,x_t) - \sum_{t'\in\mathcal{N}(t)} \varphi_{tt'}(y), \quad t \in \mathcal{T}, y \in \mathcal{Y}. \tag{4b}$$

By substituting (4) to $F(\mathbf{x},\mathbf{y};\mathbf{q},\mathbf{g}) = F(\mathbf{x},\mathbf{y};\mathbf{q}',\mathbf{g}')$ it is easy to show that the problems P and $P^{\boldsymbol{\varphi}}$ are equivalent for arbitrary potentials $\boldsymbol{\varphi}$. Moreover, it has been shown (Schlesinger, 1976; Kolmogorov, 2006) that the converse is also true, that is, if any two max-sum problems are equivalent then they are related by (4) for some potentials $\boldsymbol{\varphi}$.

Now, we can define the class of *trivial max-sum problems*. Node (t,y) is called *maximal* if $q_t(y,x_t) = \max_{y\in\mathcal{Y}} q_t(y,x_t)$ and edge $\{(t,y),(t',y')\}$ is called maximal if $g_{tt'}(y,y') = \max_{y,y'\in\mathcal{Y}} g_{tt'}(y,y')$. The max-sum problem P is called *trivial* if there exists a labeling $\mathbf{y}$ which can be formed only from the maximal nodes and edges. Checking whether a given P is trivial leads to an instance of a *constraint satisfaction problem* CSP (also called *consistent labeling problem*) (Rosenfeld et al., 1976; Haralick and Shapiro, 1979). The CSP is NP-complete in general and it is equivalent to solving a max-sum problem when the quality functions $(\mathbf{q},\mathbf{g})$ take only two values $\{-\infty, 0\}$. Let $U(\mathbf{x};\mathbf{q},\mathbf{g})$ be a *height* (also called *energy*) of the max-sum problem P defined as a sum of qualities of the maximal nodes and the maximal edges, that is,

$$U(\mathbf{x};\mathbf{q},\mathbf{g}) = \sum_{t\in\mathcal{T}} \max_{y\in\mathcal{Y}} q_t(y,x_t) + \sum_{\{t,t'\}\in\mathcal{E}} \max_{y,y'\in\mathcal{Y}} g_{tt'}(y,y'). \tag{5}$$

Comparing (1) and (5) shows that $U(\mathbf{x};\mathbf{q},\mathbf{g})$ is an upper bound on the quality of the optimal labeling, that is, $U(\mathbf{x};\mathbf{q},\mathbf{g}) \geq \max_{\mathbf{y}\in\mathcal{Y}^{\mathcal{T}}} F(\mathbf{x},\mathbf{y};\mathbf{q},\mathbf{g})$. It is easy to see that the upper bound is tight if and only if the max-sum problem is trivial. The following result is central to the LP relaxation:

Theorem 1 *Schlesinger (1976); Werner (2005) Let $\mathcal{C}$ be a class of equivalent max-sum problems. Let C contain at least one trivial problem. Then any problem in C is trivial if and only if it has minimal height.*

This suggests to solve the max-sum problem P by searching for an equivalent $P^{\boldsymbol{\varphi}}$ with the minimal height, which leads to

$$\boldsymbol{\varphi}^* = \operatorname*{argmin}_{\boldsymbol{\varphi}} U(\mathbf{x}; \mathbf{q}^{\boldsymbol{\varphi}}, \mathbf{g}^{\boldsymbol{\varphi}}) \,. \tag{6}$$

The problem (6) can be expressed as an LP task which is known to be a *dual to the LP relaxation* (3). Having $\boldsymbol{\varphi}^*$ one can try to verify whether $P^{\boldsymbol{\varphi}^*}$ is trivial, that is, whether there exists a labeling composed of the maximal nodes and edges of $P^{\boldsymbol{\varphi}^*}$. If such labeling is found then it is the optimal solution of the max-sum problem P. Otherwise, one can use heuristics to find an approximate solution by searching for such labeling which contains as much maximal nodes and edges as possible.

Though (6) is solvable in polynomial time, a general purpose LP solvers are applicable only for small instances. A specialized solvers for LP relaxation were published in Koval and Schlesinger (1976). Recently, Wainwright et al. (2002) and Kolmogorov (2006) proposed a tree-reweighted algorithm based on minimizing a more general upper bound composed of trees spanning the graph $(\mathcal{T}, \mathcal{E})$ of which (5) is a special case.

We propose a learning algorithm for a general max-sum classifier which requires an arbitrary LP relaxation solver as a subroutine. We only require that the LP solver returns an approximate solution $\hat{\mathbf{y}}$ and an upper bound on $\max_{\mathbf{y} \in \mathcal{Y}^{\mathcal{T}}} F(\mathbf{x}, \mathbf{y}; \mathbf{q}, \mathbf{g})$.

2.2 Supermodular Max-Sum Problems

Definition 1 *A function $g_{tt'} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is* supermodular *if*

1. *The set of labels $\mathcal{Y}$ is totally ordered; w.l.o.g. we consider $\mathcal{Y} = \{1, 2, \ldots, |\mathcal{Y}|\}$ endowed with the natural order.*
2. *For each four-tuple $(y_t, y'_t, y_{t'}, y'_{t'}) \in \mathcal{Y}^4$ of labels such that $y_t > y'_t$ and $y_{t'} > y'_{t'}$ the following inequality holds:*

$$g_{tt'}(y_t, y_{t'}) + g_{tt'}(y'_t, y'_{t'}) \geq g_{tt'}(y_t, y'_{t'}) + g_{tt'}(y'_t, y_{t'}). \tag{7}$$

A max-sum problem in which the label set $\mathcal{Y}$ is totally ordered and all the functions $\mathbf{g}$ are supermodular is called a *supermodular max-sum problem*. In addition, we will also consider the max-sum problem in which the inequalities (7) are fulfilled strictly. In this case the corresponding max-sum problem will be called *strictly supermodular*.

A naive approach to check whether a given max-sum problem is supermodular amounts to verifying all $|\mathcal{E}| \cdot |\mathcal{Y}|^4$ inequalities in (7). We will use a more effective way which requires to check only $|\mathcal{E}| \cdot (|\mathcal{Y}| - 1)^2$ inequalities thanks to the following well-known theorem:

Theorem 2 *The function $g_{tt'} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is* supermodular *if for each each pair of labels $(y, y') \in \mathcal{Y}^2$ such that $y + 1 \in \mathcal{Y}$, $y' + 1 \in \mathcal{Y}$ the following inequality holds*

$$g_{tt'}(y, y') + g_{tt'}(y + 1, y' + 1) \geq g_{tt'}(y, y' + 1) + g_{tt'}(y + 1, y'). \tag{8}$$

Proof The proof follows trivially from the equality

$$g_{tt'}(y_t,y_{t'})+g_{tt'}(y'_t,y'_{t'})-g_{tt'}(y_t,y'_{t'})-g_{tt'}(y'_t,y_{t'}) \\ = \sum_{\substack{y_t>z\geq y'_t \\ y_{t'}>z'\geq y'_{t'}}} \Big[g_{tt'}(z,z')+g_{tt'}(z+1,z'+1)-g_{tt'}(z,z'+1)-g_{tt'}(z+1,z') \Big],$$

and the fact that all the summands are non-negative by the condition (8). ■

A similar proposition holds for strictly supermodular problems: a max-sum problem is *strictly supermodular* if for each pair of neighboring objects $\{t,t'\} \in \mathcal{E}$ and for each pair of labels $(y,y') \in \mathcal{Y}^2$ such that $y+1 \in \mathcal{Y}$, $y'+1 \in \mathcal{Y}$ the following inequality holds:

$$g_{tt'}(y,y')+g_{tt'}(y+1,y'+1) > g_{tt'}(y,y'+1)+g_{tt'}(y+1,y'). \tag{9}$$

In this paper, we exploit the fact that the optimal solution of the supermodular problems can be found in a polynomial time (Schlesinger and Flach, 2000). Kolmogorov and Zabih (2002) proposed an efficient algorithm for the binary case $|\mathcal{Y}|=2$ which is based on transforming the supermodular max-sum problem to the max-flow problem from the graph theory. A not widely known extension for a general case $|\mathcal{Y}|>2$ was proposed in Kovtun (2004) and Schlesinger (2005). We will propose learning algorithms which require an arbitrary solver for the supermodular max-sum problem as a subroutine.

2.3 Max-Sum Problems with Strictly Trivial Equivalent

In this section we consider *max-sum problems with a strictly trivial equivalent* which is a subclass of *problems with a trivial equivalent* described in Section (2.1). The main reason for defining this subclass is that these problems are more suitable for learning than problems with a trivial equivalent. It was shown (Schlesinger, 1976; Schlesinger and Flach, 2000) that problems with a trivial equivalent contain two well-known polynomially solvable subclasses of max-sum problems: (i) the problems with an acyclic neighborhood structure $\mathcal{E}$ and (ii) the supermodular problems. We will give a similar result which applies for the problems with a strictly trivial equivalent. In particular, we will show that the class of problems with a strictly trivial equivalent contains all acyclic and supermodular problems which have a unique solution. This shows that learning algorithms for the max-sum classifiers with a strictly trivial equivalent introduced below are applicable for a wide range of polynomially solvable problems.

Max-sum problems with a trivial equivalent are those for which LP relaxation (6) is tight, that is, $U(\mathbf{x};\mathbf{q}^{\boldsymbol{\varphi}^*},\mathbf{g}^{\boldsymbol{\varphi}^*}) = \max_{\mathbf{y}\in\mathcal{Y}^{\mathcal{T}}} F(\mathbf{x},\mathbf{y};\mathbf{q},\mathbf{g})$, and thus $\max_{\mathbf{y}\in\mathcal{Y}^{\mathcal{T}}} F(\mathbf{x},\mathbf{y};\mathbf{q},\mathbf{g})$ can be computed in a polynomial time. The tight LP relaxation, however, does not imply that the optimal solution $\mathbf{y}^* = \operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}^{\mathcal{T}}} F(\mathbf{x},\mathbf{y};\mathbf{q},\mathbf{g})$ can be found in a polynomial time. As we mentioned, to find $\mathbf{y}^*$ from $(\mathbf{q}^{\boldsymbol{\varphi}^*},\mathbf{g}^{\boldsymbol{\varphi}^*})$ requires searching for a labeling formed only by the maximal nodes and edges which need not be unique. Finding such labeling leads to the CSP which is NP-complete. We exploit the fact that the labeling can be found trivially if all the maximal nodes and edges are unique. As a result, the optimal solution $\mathbf{y}^*$ can be found in a polynomial time by solving the LP relaxation and finding the maximal nodes and edges.

Definition 2 *Max-sum problem P is* strictly trivial *if:*

1. *There exists a unique maximal node (t,y) for each object $t \in \mathcal{T}$.*
2. *For each maximal edge $\{(t,y),(t',y')\} \in \mathcal{E}_{\mathcal{Y}}$ the nodes (t,y) and (t',y') are maximal.*

It is clear that a strictly trivial problem has a unique solution which is composed of all maximal nodes and edges. Checking whether a given problem is strictly trivial requires only $O(|\mathcal{T}||\mathcal{Y}| + |\mathcal{E}||\mathcal{Y}|^2)$ operations, that is, finding the maximal nodes and edges and checking if they are unique and if they form a labeling.

Recall that the max-sum problem P has a *strictly trivial equivalent* if there exists a strictly trivial problem which is equivalent to P. Finally, we give a theorem which asserts that the class of problems with a strictly trivial equivalent includes at least the acyclic problems and the supermodular problems which have a unique solution.

Theorem 3 *Let $P = (\mathcal{T},\mathcal{E},\mathcal{Y},\mathbf{q},\mathbf{g},\mathbf{x})$ be a max-sum problem and let P have a unique solution. If $(\mathcal{T},\mathcal{E})$ is an acyclic graph or quality functions $\mathbf{g}$ are supermodular then P is equivalent to some strictly trivial problem.*

Proof is given in Appendix A.

3. Discriminative Approach to Learning Structured Linear Classifiers

In this section, we review the discriminative approach to learning linear classifiers which we will later apply to the max-sum classifiers.

Let $\mathbf{x} \in \boldsymbol{\mathcal{X}}$ and $\mathbf{y} \in \boldsymbol{\mathcal{Y}}$ be observations and labels generated according to some fixed unknown distribution $P(\mathbf{x},\mathbf{y})$. We are interested in designing a classifier $f\colon \boldsymbol{\mathcal{X}} \to \boldsymbol{\mathcal{Y}}$ which estimates labels from observations. Let $L\colon \boldsymbol{\mathcal{Y}} \times \boldsymbol{\mathcal{Y}} \to \mathbb{R}$ be a loss function penalizing a prediction $f(\mathbf{x})$ by a penalty $L(\mathbf{y}, f(\mathbf{x}))$ provided the true output is $\mathbf{y}$. The goal is to find a classifier which minimizes the *expected (Bayesian) risk*

$$R[f] = \int_{\boldsymbol{\mathcal{X}}\times\boldsymbol{\mathcal{Y}}} L(\mathbf{y}, f(\mathbf{x}))\, dP(\mathbf{x},\mathbf{y})\,.$$

The risk $R[f]$ cannot be directly minimized because the distribution $P(\mathbf{x},\mathbf{y})$ is unknown. Instead, we are given a finite training set $\mathcal{L} = \{(\mathbf{x}^j,\mathbf{y}^j) \in \boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}} \mid j \in \mathcal{J}\}$ i.i.d. sampled from $P(\mathbf{x},\mathbf{y})$.

The discriminative approach to learning classifiers does not require estimation of a probability distribution. It is based on direct optimization of parameters of a classifier in order to minimize a substitutional risk functional which approximates the desired risk $R[f]$ and can be computed from a finite training set. Let us consider a class of linear classifiers

$$f(\mathbf{x};\mathbf{w}) = \operatorname*{argmax}_{\mathbf{y}\in\boldsymbol{\mathcal{Y}}} \langle \mathbf{w}, \Psi(\mathbf{x},\mathbf{y})\rangle\,. \tag{10}$$

The classifier is determined by a parameter vector $\mathbf{w} \in \mathbb{R}^d$ and some fixed mapping $\Psi\colon \boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}} \to \mathbb{R}^d$. A simple proxy for $R[f(\bullet;\mathbf{w})]$ is the *empirical risk*

$$R_{\text{emp}}[f(\bullet;\mathbf{w})] = \frac{1}{m}\sum_{j\in\mathcal{J}} L(\mathbf{y}^j, f(\mathbf{x}^j;\mathbf{w}))\,.$$

Let us assume a class of loss functions which satisfy $L(\mathbf{y},\mathbf{y}')=0$ if $\mathbf{y}=\mathbf{y}'$ and $L(\mathbf{y},\mathbf{y}')>0$ if $\mathbf{y}\neq\mathbf{y}'$. Further, we will distinguish two learning scenarios: (i) learning from a *consistent training set* and (ii) learning from an *inconsistent training set*. The training set $\mathcal{L}$ is *consistent* if there exists a parameter vector $\mathbf{w}^*$ such that the empirical risk of the linear classifier is zero, that is, $R_{\text{emp}}[f(\bullet;\mathbf{w}^*)]=0$. In the opposite case, the training set is *inconsistent*.

In Section 3.1, we review the perceptron algorithm suitable for learning linear classifiers from a consistent training set. The Support Vector Machine (SVM) approach to learning from an inconsistent training set is described in Section 3.2.

3.1 Learning from a Consistent Training Set Using the Perceptron Algorithm

In the case of a consistent training set $\mathcal{L}$, the learning problem is to find parameters $\mathbf{w}^*$ of the linear classifier (10) such that the empirical risk $R_{\text{emp}}[f(\bullet;\mathbf{w}^*)]=0$. This amounts to finding parameters $\mathbf{w}^*$ which satisfy the set of non-linear equations

$$\mathbf{y}^j = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} \langle \mathbf{w}, \Psi(\mathbf{x},\mathbf{y})\rangle\,, \quad j\in\mathcal{J}\,. \tag{11}$$

In the rest of the article, we will assume that the training set does not contain examples with the same observations $\mathbf{x}^j=\mathbf{x}^{j'}$ but different labelings $\mathbf{y}^j\neq\mathbf{y}^{j'}$, that is, we will search for a classifier which returns a unique labeling for each example from the training set. It is convenient to transform (11) to an equivalent problem of solving a set of linear strict inequalities

$$\langle \mathbf{w}, \Psi(\mathbf{x}^j,\mathbf{y}^j)\rangle > \langle \mathbf{w}, \Psi(\mathbf{x}^j,\mathbf{y})\rangle\,, \quad j\in\mathcal{J}, \mathbf{y}\in\mathcal{Y}\setminus\{\mathbf{y}^j\}\,,$$

which, using $\hat{\Psi}(\mathbf{x}^j,\mathbf{y}^j,\mathbf{y}) = \Psi(\mathbf{x}^j,\mathbf{y}^j) - \Psi(\mathbf{x}^j,\mathbf{y})$, can be written in a compact form

$$\langle \mathbf{w}, \hat{\Psi}(\mathbf{x}^j,\mathbf{y}^j,\mathbf{y})\rangle > 0\,, \quad j\in\mathcal{J}, \mathbf{y}\in\mathcal{Y}\setminus\{\mathbf{y}^j\}\,. \tag{12}$$

An efficient method to solve the problem (12) is the *perceptron* algorithm:

Algorithm 1 Perceptron algorithm

1: Set $\mathbf{w} := 0$.

2: Find a violated inequality in (12), that is, find indices $j^*\in\mathcal{J}$, $\mathbf{y}^*\in\mathcal{Y}\setminus\{\mathbf{y}^{j^*}\}$ such that

$$\langle \mathbf{w}, \hat{\Psi}(\mathbf{x}^{j^*},\mathbf{y}^{j^*},\mathbf{y}^*)\rangle \leq 0\,.$$

3: If there is no violated inequality then $\mathbf{w}$ solves (12) and the algorithm halts. Otherwise update the current solution

$$\mathbf{w} := \mathbf{w} + \hat{\Psi}(\mathbf{x}^{j^*},\mathbf{y}^{j^*},\mathbf{y}^*)\,,$$

and go to Step 2.

Provided the training set $\mathcal{L}$ is consistent, that is, the inequalities (12) are satisfiable, the perceptron terminates after a finite number of iterations (Novikoff, 1962). The number of iterations of the perceptron algorithm does not depend on the number of inequalities of (12). This property is crucial for the problems with a very large number of inequalities, which are of interest in our article.

In Step 2, the perceptron algorithm needs to find a violated inequality in (12). This subtask can be solved by computing the classifier responses $\hat{\mathbf{y}}^j = f(\mathbf{x}^j;\mathbf{w})$, $j \in \mathcal{J}$ by evaluating (10). If for some $j \in \mathcal{J}$ the response $\hat{\mathbf{y}}^j \neq \mathbf{y}^j$, that is, the classifier commits an error on the j-th example, then the indices $(j,\hat{\mathbf{y}}^j)$ identify the violated inequality. As a result, the perceptron algorithm is applicable for learning of an arbitrary linear classifier for which the classification task (10) can be evaluated efficiently.

In case of a general max-sum classifier, the classification task is NP-complete so that using the perceptron algorithm is not feasible. In Section 4 we will show, however, how to use the perceptron for learning the strictly supermodular max-sum classifiers and the max-sum classifiers with a strictly trivial equivalent.

3.2 Learning from Inconsistent Training Set Using SVM Approach

In practical applications, the training set is often inconsistent. SVMs (Vapnik, 1998) constitute a popular approach to learning linear classifiers applicable to both consistent and inconsistent training sets. The SVM learning is based on minimizing *regularized risk* which is a sum of the empirical risk $R_{\text{emp}}[f(\bullet;\mathbf{w})]$ and a regularization term. Minimization of the regularization term corresponds to the notion of large margin introduced to prevent over-fitting. Since the empirical risk is not convex for most loss functions used in classification it is replaced by a convex piece-wise linear upper bound which is more suitable for optimization. Originally, SVMs were designed for a binary case $|\boldsymbol{\mathcal{Y}}| = 2$ and $0/1$-loss function $L_{0/1}(\mathbf{y},\mathbf{y}') = [\![\mathbf{y} \neq \mathbf{y}']\!]$ where $[\![\bullet]\!]$ equals 1 if the term inside the brackets is satisfied and it is 0 otherwise. The multi-class variant of SVMs ($|\boldsymbol{\mathcal{Y}}| > 2$) were introduced in Vapnik (1998) and Crammer and Singer (2001) but it still assumes the $0/1$-loss only function.

Taskar et al. (2004b) extended SVMs for learning the max-sum classifiers (1) with an acyclic neighbourhood structure and they proposed using the additive loss function (14) suitable for these problems. Recently, the approach was generalized by Tsochantaridis et al. (2005) who consider an arbitrary structured linear classifier (10) and a general loss function. Learning of structured SVMs classifiers leads to the following convex QP task

$$(\mathbf{w}^*,\boldsymbol{\xi}^*) = \operatorname*{argmin}_{\mathbf{w},\boldsymbol{\xi}} \left[\frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m}\sum_{j\in\mathcal{J}} \xi_j \right], \tag{13a}$$

subject to

$$\langle \mathbf{w}, \Psi(\mathbf{x}^j,\mathbf{y}^j) - \Psi(\mathbf{x}^j,\mathbf{y}) \rangle \geq L(\mathbf{y}^j,\mathbf{y}) - \xi_j, \qquad j \in \mathcal{J}, \mathbf{y} \in \boldsymbol{\mathcal{Y}}. \tag{13b}$$

The objective function of (13) comprises the regularization term $\frac{1}{2}\|\mathbf{w}\|^2$ and the sum $\frac{1}{m}\sum_{j\in\mathcal{J}}\xi_j$ weighted by the regularization constant $C > 0$. It can be shown (Tsochantaridis et al., 2005) that the sum $\frac{1}{m}\sum_{j\in\mathcal{J}}\xi_j$ is an upper bound on the empirical risk $R_{\text{emp}}[f(\bullet;\mathbf{w})]$ of the linear classifier (10). Thus the learning objective is to minimize an upper bound on the empirical risk and to penalize parameters with high $\|\mathbf{w}\|^2$. The loss function can be an arbitrary function $L\colon \boldsymbol{\mathcal{Y}} \times \boldsymbol{\mathcal{Y}} \to \mathbb{R}$ which satisfies $L(\mathbf{y},\mathbf{y}') = 0$ if $\mathbf{y} = \mathbf{y}'$ and $\infty > L(\mathbf{y},\mathbf{y}') > 0$ if $\mathbf{y} \neq \mathbf{y}'$. In cases which are of interest in our article, the set $\boldsymbol{\mathcal{Y}} = \mathcal{Y}^{\mathcal{T}}$ and a reasonable choice is the *additive loss function*, also known as the Hamming distance, which is defined as

$$L_\Delta(\mathbf{y},\mathbf{y}') = \sum_{t\in\mathcal{T}} [\![y_t \neq y'_t]\!], \tag{14}$$

that is, it counts the number of misclassified objects. The trade-off between the regularization term and the upper bound is controlled by the constant C. A suitable setting is usually found by tuning C on an independent data set or using the cross-validation.

The number of constraints (13b) equals to $m|\mathcal{Y}|$. In the structured learning $|\mathcal{Y}|$ is huge, for example, in the case of the max-sum classifiers it grows exponentially because $|\mathcal{Y}| = |\mathcal{Y}|^{|T|}$. As a result, the QP task (13) is not tractable for general purpose QP solvers. Tsochantaridis et al. (2005) proposed a specialized cutting plane algorithm to approximate (13) by a reduced QP task which has the same objective function (13a) but uses only a small subset of the constraints (13b). The efficiency of the approximation relies on the sparseness of the solution (13) which means that majority of the linear inequality constraints (13b) are inactive and they can be excluded without affecting the solution. To select the constraints, the cutting plane algorithm requires solving the LAC task

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \left[L(\mathbf{y}^j, \mathbf{y}) + \left\langle \mathbf{w}, \Psi(\mathbf{x}^j, \mathbf{y}) \right\rangle \right]. \tag{15}$$

Similarly to the perceptron algorithm, the cutting plane algorithm is applicable for learning of an arbitrary linear classifier for which the LAC task (15) can be solved efficiently.

In case of a general max-sum classifier and an additive loss function the LAC task becomes an instance of a general max-sum problem. Even though a general max-sum problem can be solved only approximately, we will show that it suffices to solve the learning problem (13) with a good precision. Moreover, we will be able to determine how the found solution differs from the optimal one. In case of a supermodular max-sum classifier we will augment the learning problem (13) in such a way that the obtained quality functions will be supermodular and thus the task (15) becomes solvable precisely in a polynomial time. Finally, in case of a max-sum problem with a strictly trivial equivalent, we will give a new formulation of the learning problem which does not require solving the task (15) at all since the number of constraints will be sufficiently small to use an exhaustive search.

4. Learning a Max-Sum Classifier from a Consistent Training Set

In this section, we assume that the training set is consistent with respect to a given max-sum classifier. This means that there exists at least one max-sum classifier in a given class which classifies all training examples correctly. We will introduce variants of the perceptron algorithm which, provided the assumption holds, find a classifier in a finite number of iterations. In practice, however, there is no general way to verify whether the assumption holds unless the perceptron halts. Consequently, if the perceptron algorithm does not halt after a reasonable number of iterations we cannot draw any conclusion about the solution. In Section 5, we will avoid this drawback using the SVM approach able to deal with inconsistent training sets.

To use the perceptron algorithm, we will reformulate the learning problem as an equivalent task of satisfying a set of linear strict inequalities. A keystone of the perceptron algorithm is an efficient procedure to find a violated inequality in the underlying set which amounts to classifying the examples from the training set. Therefore, using the perceptron algorithm is not feasible for the general max-sum problem whose evaluation is NP-complete. Nevertheless, we will formulate the learning problem even for this case because it will serve as a basis for constructing the algorithm for an inconsistent training set. Next, we will derive learning algorithms for strictly supermodular max-sum classifiers and max-sum classifiers with a strictly trivial equivalent. For these two classes, max-

sum classifiers can be evaluated efficiently by polynomial-time algorithms which makes possible to use the perceptron. Moreover, in the case of a max-sum classifier with a strictly trivial equivalent, learning will require finding a violated inequality from only a polynomially-sized set which can be accomplished exhaustively without any max-sum solver.

4.1 General Max-Sum Classifier

Problem 1 *(Learning a general max-sum classifier from a consistent training set). For a given training set* $\mathcal{L} = \{(\mathbf{x}^j, \mathbf{y}^j) \in \mathcal{X}^{\mathcal{T}} \times \mathcal{Y}^{\mathcal{T}} \mid j \in \mathcal{J}\}$ *find quality functions* $\mathbf{q}$ *and* $\mathbf{g}$ *such that*

$$\mathbf{y}^j = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}^{\mathcal{T}}} F(\mathbf{x}^j, \mathbf{y}; \mathbf{q}, \mathbf{g}), \qquad j \in \mathcal{J}. \tag{16}$$

It can be seen, that the general max-sum classifier (1) can be represented as the linear classifier (10). In particular, the quality functions $\mathbf{q}$ and $\mathbf{g}$ are merged to a single parameter vector $\mathbf{w} = (\mathbf{q}; \mathbf{g}) \in \mathbb{R}^d$ of dimension $d = |\mathcal{T}||\mathcal{Y}||\mathcal{X}| + |\mathcal{E}||\mathcal{Y}|^2$. Further, we have $\boldsymbol{\mathcal{X}} = \mathcal{X}^{\mathcal{T}}$ and $\boldsymbol{\mathcal{Y}} = \mathcal{Y}^{\mathcal{T}}$. The mapping $\Psi\colon \mathcal{X}^{\mathcal{T}} \times \mathcal{Y}^{\mathcal{T}} \to \mathbb{R}^d$ can be constructed of indicator functions in such a way that $F(\mathbf{x}, \mathbf{y}; \mathbf{q}, \mathbf{g}) = \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}) \rangle$.

Following the approach of learning linear classifiers described in Section 3.1, we can reformulate Problem 1 to an equivalent problem of solving a huge set of linear inequalities

$$F(\mathbf{x}^j, \mathbf{y}^j; \mathbf{q}, \mathbf{g}) > F(\mathbf{x}^j, \mathbf{y}; \mathbf{q}, \mathbf{g}), \quad j \in \mathcal{J}, \mathbf{y} \in \mathcal{Y}^{\mathcal{T}} \setminus \{\mathbf{y}^j\}. \tag{17}$$

To find a solution of (17) by the perceptron, we would need an efficient algorithm to compute a response of a general max-sum classifier. Because solving a max-sum problem is NP-complete in general, using the perceptron algorithm is not tractable. In Section 5.1, we will use the formulation (17) as a basis for constructing an algorithm for learning from an inconsistent training set. In this case, it will be possible to use the LP relaxation to approximate the response of a max-sum classifier.

4.2 Strictly Supermodular Max-Sum Classifier

Problem 2 *(Learning strictly supermodular max-sum classifier from a consistent training set). For a given training set* $\mathcal{L} = \{(\mathbf{x}^j, \mathbf{y}^j) \in \mathcal{X}^{\mathcal{T}} \times \mathcal{Y}^{\mathcal{T}} \mid j \in \mathcal{J}\}$ *find quality functions* $\mathbf{q}$ *and* $\mathbf{g}$ *such that equations (16) are satisfied and the quality functions* $\mathbf{g}$ *are strictly supermodular, that is,* $\mathbf{g}$ *satisfy the condition (9).*

Similarly to the general case, we reformulate Problem 2 as an equivalent problem of solving a set of strict linear inequalities

$$F(\mathbf{x}^j, \mathbf{y}^j; \mathbf{q}, \mathbf{g}) > F(\mathbf{x}^j, \mathbf{y}; \mathbf{q}, \mathbf{g}), \quad j \in \mathcal{J}, \mathbf{y} \in \mathcal{Y}^{\mathcal{T}} \setminus \{\mathbf{y}^j\}, \tag{18a}$$

$$\left.\begin{aligned} g_{tt'}(y, y') + g_{tt'}(y+1, y'+1) > g_{tt'}(y, y'+1) + g_{tt'}(y+1, y'), \\ \{t, t'\} \in \mathcal{E}, (y, y') \in \{1, \ldots, |\mathcal{Y}| - 1\}^2. \end{aligned}\right\} \tag{18b}$$

The system (18) comprises two sets of linear inequalities (18a) and (18b). The inequalities (18a) enforce an error-less response of a max-sum classifier on the training set. The inequalities (18b), if satisfied, guarantee that the quality functions $\mathbf{g}$ are strictly supermodular (c.f. Section 2.2).

To apply the perceptron algorithm we need an efficient method to find a violated inequality in (18). In the case of the inequalities (18b), it can be accomplished by an exhaustive search since there are only $|\mathcal{E}|(|\mathcal{Y}|-1)^2$ inequalities. To find a violated inequality in (18a), where the number $m(|\mathcal{Y}|^{|\mathcal{T}|}-1)$ grows exponentially, we need to compute the response of a max-sum classifier on the training examples. This can be done efficiently provided the inequalities (18b) are already satisfied as it guarantees that the qualities $\mathbf{g}$ are supermodular. Thus we use the perceptron algorithm to repeatedly solve the inequalities (18b) and, as soon as (18b) are satisfied, we find a violated inequality in (18a) by solving a supermodular max-sum problem. The proposed variant of the perceptron to solve (18) is as follows:

Algorithm 2 Learning strictly supermodular max-sum classifier by perceptron

1: Set $\mathbf{q} := \mathbf{0}$ and $\mathbf{g} := \mathbf{0}$.

2: Find a violated inequality in (18b), that is, find a quadruple $\{t,t'\} \in \mathcal{E}$, $y,y' \in \mathcal{Y}$ such that

$$g_{tt'}(y,y') + g_{tt'}(y+1,y'+1) - g_{tt'}(y,y'+1) - g_{tt'}(y+1,y') \leq 0\,.$$

3: If no such quadruple (t,t',y,y') exists then ($\mathbf{g}$ are already supermodular) go to Step 4. Otherwise use the found (t,t',y,y') to update current $\mathbf{g}$ by

$$\begin{array}{rclrcl} g_{tt'}(y,y') & := & g_{tt'}(y,y')+1\,, & g_{tt'}(y+1,y'+1) & := & g_{tt'}(y+1,y'+1)+1\,, \\ g_{tt'}(y,y'+1) & := & g_{tt'}(y,y'+1)-1\,, & g_{tt'}(y+1,y') & := & g_{tt'}(y+1,y')-1\,, \end{array}$$

and go to Step 2.

4: Find a violated inequality in (18a), that is, find an index $j \in \mathcal{J}$ and a labeling $\mathbf{y}$ such that

$$\mathbf{y}^j \neq \mathbf{y} := \operatorname*{argmax}_{\mathbf{y}' \in \mathcal{Y}^{\mathcal{T}}} F(\mathbf{x}^j, \mathbf{y}'; \mathbf{q}, \mathbf{g})\,.$$

5: If no such $(j, \mathbf{y})$ exist than $(\mathbf{q}, \mathbf{g})$ solve the task (18) and the algorithm halts. Otherwise use the found $(j, \mathbf{y})$ to update current $\mathbf{q}$ and $\mathbf{g}$ by

$$\begin{array}{rclrcll} g_{tt'}(y_t^j, y_{t'}^j) & := & g_{tt'}(y_t^j, y_{t'}^j)+1, & g_{tt'}(y_t, y_{t'}) & := & g_{tt'}(y_t, y_{t'})-1\,, & \{t,t'\} \in \mathcal{E}\,, \\ q_t(y_t^j, x_t^j) & := & q_t(y_t^j, x_t^j)+1, & q_t(y_t, x_t^j) & := & q_t(y_t, x_t^j)-1\,, & t \in \mathcal{T}\,. \end{array}$$

and go to Step 2.

Since Algorithm 2 is nothing but the perceptron algorithm applied to the particular set of linear constraints (18), the Novikoff's theorem (Novikoff, 1962) readily applies. Thus Algorithm 2 terminates in a finite number of iterations provided (18) is satisfiable, that is, if there exists a supermodular max-sum classifier $f(\bullet; \mathbf{q}, \mathbf{g})$ with zero empirical error risk on the training set $\mathcal{L}$.

4.3 Max-Sum Classifier with a Strictly Trivial Equivalent

Problem 3 *(Learning max-sum classifier with a strictly trivial equivalent from a consistent training set). For a given training set* $\mathcal{L} = \{(\mathbf{x}^j, \mathbf{y}^j) \in \mathcal{X}^{\mathcal{T}} \times \mathcal{Y}^{\mathcal{T}} \mid j \in \mathcal{J}\}$ *find quality functions* $\mathbf{q}$ *and* $\mathbf{g}$ *such that equations (16) are satisfied and the max-sum problems* $P^j = (\mathcal{T}, \mathcal{E}, \mathcal{Y}, \mathbf{q}, \mathbf{g}, \mathbf{x}^j)$, $j \in \mathcal{J}$ *have a strictly trivial equivalent.*

In Problem 3, we again search for a max-sum classifier which classifies all training examples correctly but, in addition, all max-sum problems P^j, $j \in \mathcal{J}$ should have a strictly trivial equivalent. This means that if we compute a response of a max-sum classifier which solves Problem 3 for each training example using LP relaxation we get a unique labeling equal to that in the training set.

Because the equivalent transformations (4) cover the whole class of equivalent problems, the problem $P = (\mathcal{T}, \mathcal{E}, \mathcal{Y}, \mathbf{q}, \mathbf{g}, \mathbf{x})$ has a strictly trivial equivalent if and only if there exist potentials $\boldsymbol{\varphi}$ such that $P^{\boldsymbol{\varphi}}$ is strictly trivial. Recall that the strictly trivial problem has unique maximal nodes and edges (cf. Definition 2). Consequently, if the problem P has a strictly trivial equivalent and its optimal solution is $\mathbf{y}^*$ then there must exist potentials $\boldsymbol{\varphi}$ such that the following set of linear inequalities holds

$$q_t^{\boldsymbol{\varphi}}(y_t^*, x_t) > q_t^{\boldsymbol{\varphi}}(y, x_t), \quad t \in \mathcal{T}, y \in \mathcal{Y} \setminus \{y_t^*\}, \tag{19a}$$

$$g_{tt'}^{\boldsymbol{\varphi}}(y_t^*, y_{t'}^*) > g_{tt'}^{\boldsymbol{\varphi}}(y, y'), \quad \{t, t'\} \in \mathcal{E}, (y, y') \in \mathcal{Y}^2 \setminus \{(y_t^*, y_{t'}^*)\}. \tag{19b}$$

Note, that (19) is a set of $|\mathcal{T}||\mathcal{Y}| + |\mathcal{E}||\mathcal{Y}|^2$ strict inequalities which are linear in $(\mathbf{q}, \mathbf{g})$ and $\boldsymbol{\varphi}$ as can be seen after substituting (4) to (19). By replicating (19) for max-sum problems $P^j = (\mathcal{T}, \mathcal{E}, \mathcal{Y}, \mathbf{q}, \mathbf{g}, \mathbf{x}^j)$, $j \in \mathcal{J}$ whose unique solutions are required to be $\mathbf{y}^j$, $j \in \mathcal{J}$ we obtain an equivalent formulation of Problem 3 which requires satisfaction of a set of strict linear inequalities

$$q_t^{\boldsymbol{\varphi}^j}(y_t^j, x_t^j) > q_t^{\boldsymbol{\varphi}^j}(y, x_t^j), \quad j \in \mathcal{J}, t \in \mathcal{T}, y \in \mathcal{Y} \setminus \{y_t^j\}, \tag{20a}$$

$$g_{tt'}^{\boldsymbol{\varphi}^j}(y_t^j, y_{t'}^j) > g_{tt'}^{\boldsymbol{\varphi}^j}(y, y'), \quad j \in \mathcal{J}, \{t, t'\} \in \mathcal{E}, (y, y') \in \mathcal{Y}^2 \setminus \{(y_t^j, y_{t'}^j)\}. \tag{20b}$$

The system (20) is to be solved with respect to the quality functions $(\mathbf{q}, \mathbf{g})$ and the potentials $\boldsymbol{\varphi}^j$, $j \in \mathcal{J}$ introduced for each P^j, $j \in \mathcal{J}$. Note that the number of inequalities in the problem (20) is substantially smaller compared to the learning of a general max-sum classifier. In particular, (20) contains only $m|\mathcal{T}|(|\mathcal{Y}|-1) + m|\mathcal{E}|(|\mathcal{Y}|^2-1)$ inequalities compared to $m(|\mathcal{Y}|^{|\mathcal{T}|}-1)$ for a general max-sum classifier. Therefore a violated inequality in (20) can be easily selected by an exhaustive search. The proposed variant of the perceptron to solve (20) is as follows:

Algorithm 3 Learning the max-sum classifier with a strictly equivalent by perceptron

1: Set $\mathbf{g} := 0$, $\mathbf{q} := 0$, $\boldsymbol{\varphi}^j := 0$, $j \in \mathcal{J}$.

2: Find a violated inequality in (20a), that is, find a triplet $j \in \mathcal{J}$, $t \in \mathcal{T}$, $y \in \mathcal{Y} \setminus \{y_t^j\}$ such that

$$q_t(y_t^j, x_t^j) - \sum_{t' \in \mathcal{N}(t)} \varphi_{tt'}^j(y_t^j) \le q_t(y, x_t^j) - \sum_{t' \in \mathcal{N}(t)} \varphi_{tt'}^j(y).$$

3: If no such triplet (j, t, y) exists then go to Step 4. Otherwise update $\mathbf{q}$ and $\boldsymbol{\varphi}^j$ by

$$\begin{aligned} \varphi_{tt'}^j(y_t^j) &:= \varphi_{tt'}^j(y_t^j) - 1, \quad \varphi_{tt'}^j(y) := \varphi_{tt'}^j(y) + 1, \quad t' \in \mathcal{N}(t), \\ q_t(y_t^j, x_t^j) &:= q_t(y_t^j, x_t^j) + 1, \quad q_t(y, x_t^j) := q_t(y, x_t^j) - 1. \end{aligned}$$

4: Find a violated inequality in (20b), that is, find a quintuple $j \in \mathcal{J}$, $\{t,t'\} \in \mathcal{E}$, $(y,y') \in \mathcal{Y}^2 \setminus \{(y^j_t, y^j_{t'})\}$ such that

$$g_{tt'}(y^j_t, y^j_{t'}) + \varphi^j_{tt'}(y^j_t) + \varphi^j_{t't}(y^j_{t'}) \leq g_{tt'}(y,y') + \varphi^j_{tt'}(y) + \varphi^j_{t't}(y') \,.$$

5: If no such quintuple (j,t,t',y,y') exists and no update was made in Step 3 then the current $(\mathbf{q}, \mathbf{g}, \boldsymbol{\varphi}^j, j \in \mathcal{J})$ solves the task (20) and the algorithm halts. Otherwise update $\mathbf{g}$ and $\boldsymbol{\varphi}^j$ by

$$\begin{array}{rcrrcr} \varphi^j_{tt'}(y^j_t) & := & \varphi^j_{tt'}(y^j_t) + 1 \,, & \varphi^j_{t't}(y^j_{t'}) & := & \varphi^j_{t't}(y^j_{t'}) + 1 \,, \\ \varphi^j_{tt'}(y) & := & \varphi^j_{tt'}(y) - 1 \,, & \varphi^j_{t't}(y') & := & \varphi^j_{t't}(y') - 1 \,, \\ g_{tt'}(y^j_t, y^j_{t'}) & := & g_{tt'}(y^j_t, y^j_{t'}) + 1 \,, & g_{tt'}(y,y') & := & g_{tt'}(y,y') - 1 \,. \end{array}$$

and go to Step 2.

By the Novikoff's theorem, Algorithm 3 terminates in a finite number of iterations provided (20) is satisfiable, that is, if there exists a max-sum classifier $f(\bullet; \mathbf{q}, \mathbf{g})$ with zero empirical risk on $\mathcal{L}$ and all max-sum problems $P^j = (\mathcal{T}, \mathcal{E}, \mathcal{Y}, \mathbf{q}, \mathbf{g}, \mathbf{x}^j)$, $j \in \mathcal{J}$ have a strictly trivial equivalent.

5. Learning a Max-Sum Classifier from an Inconsistent Training Set

In this section, we formulate problems of learning max-sum classifiers from an inconsistent training set using the SVM framework described in Section 3.2. We will formulate the learning problems for a general max-sum classifier, a supermodular max-sum classifier, and a max-sum classifier with a strictly trivial equivalent. In all cases, learning will amount to solving an instance of a convex QP task. In Section 6, we will propose an extended version of the cutting plane algorithm (Tsochantaridis et al., 2005) which can solve these QP tasks efficiently.

In Section 4, we showed that learning of max-sum classifiers can be expressed as satisfying a set of strict linear inequalities. In the case of an inconsistent training set, however, these linear inequalities become unsatisfiable. Therefore we augment linear inequalities with non-negative slack variables, which relaxes the problem and makes it always satisfiable. In analogy to the SVM, we will minimize the Euclidean norm of an optimized parameters and the sum of slack variables. The problems are formulated in such a way that the sum of slack variables is a piece-wise linear upper bound on the empirical risk for a certain loss function. We will consider two different loss functions. In case of a general max-sum classifier and a supermodular max-sum classifier, we will use the *additive loss function* $L_\Delta(\mathbf{y}, \mathbf{y}') = \sum_{t \in \mathcal{T}} L_t(y_t, y'_t)$ where $L_t \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is any function which satisfies $L_t(y,y') = 0$ for $y = y'$ and $L_t(y,y') > 0$ otherwise. The Hamming distance $L_\Delta(\mathbf{y}, \mathbf{y}') = \sum_{t \in \mathcal{T}} [\![\mathbf{y} \neq \mathbf{y}']\!]$ is the particular case of the additive loss which seems to be a reasonable choice in many (not necessarily all) applications. In case of a max-sum classifier with a strictly trivial equivalent, we will consider the *0/1-loss function* $L_{0/1}(\mathbf{y}, \mathbf{y}') = [\![\mathbf{y} \neq \mathbf{y}']\!]$. The 0/1-loss function penalizes equally all incorrect predictions regardless of how many labels are misclassified. The additive loss is preferable to the 0/1-loss function in most structured classification problems. On the other hand, there are applications for which the 0/1-loss function is a natural choice, for example, problems with small number of objects like the one presented in Section 7.2.

5.1 General Max-Sum Classifier

Problem 4 *(Learning a general max-sum classifier from an inconsistent training set). For a given training set* $\mathcal{L} = \left\{(\mathbf{x}^j, \mathbf{y}^j) \in \mathcal{X}^{\mathcal{T}} \times \mathcal{Y}^{\mathcal{T}} \,\middle|\, j \in \mathcal{J}\right\}$ *and a regularization constant* C *find quality functions* $\mathbf{q}$ *and* $\mathbf{g}$ *solving the following QP task*

$$(\mathbf{g}^*, \mathbf{q}^*, \boldsymbol{\xi}^*) = \operatorname*{argmin}_{\mathbf{g},\mathbf{q},\boldsymbol{\xi}} \left[\frac{1}{2}\left(\|\mathbf{g}\|^2 + \|\mathbf{q}\|^2\right) + \frac{C}{m}\sum_{j\in\mathcal{J}} \xi_j\right], \tag{21a}$$

subject to

$$F(\mathbf{x}^j, \mathbf{y}^j; \mathbf{q}, \mathbf{g}) - F(\mathbf{x}^j, \mathbf{y}; \mathbf{q}, \mathbf{g}) \geq L_\Delta(\mathbf{y}^j, \mathbf{y}) - \xi_j\,, \quad j \in \mathcal{J}, \mathbf{y} \in \mathcal{Y}^{\mathcal{T}}\,. \tag{21b}$$

The QP task (21) fits to the formulation (13) of structured SVM learning with the max-sum classifier (1) plugged in. The number of optimized parameters $(\mathbf{q};\mathbf{g}) \in \mathbb{R}^d$ equals to $d = |\mathcal{X}||\mathcal{Y}||\mathcal{T}| + |\mathcal{Y}|^2|\mathcal{E}|$. The main difficulty in solving (21) stems from the huge number $n = m|\mathcal{Y}|^{|\mathcal{T}|}$ of the linear inequalities (21b) which define the feasible set.

For a later use, it is convenient to rewrite the QP task (21) using a compact notation

$$\begin{aligned} (\mathbf{w}^*, \boldsymbol{\xi}^*) &= \operatorname*{argmin}_{\mathbf{w},\boldsymbol{\xi}} Q_P(\mathbf{w}, \boldsymbol{\xi}) = \operatorname*{argmin}_{\mathbf{w},\boldsymbol{\xi}} \left[\frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m}\sum_{j\in\mathcal{J}} \xi_j\right], \\ \text{s.t.} \quad & \langle \mathbf{w}, \mathbf{z}_i \rangle \geq b_i - \xi_j\,, \quad j \in \mathcal{J}, i \in I_j\,, \end{aligned} \tag{22}$$

where $I_1 \cup \cdots \cup I_m = I = \{1, \ldots, n\}$, $I_u \cap I_v = \{\emptyset\}$, $u \neq v$, denote disjoint sets of indices such that each $i \in I$ has assigned an unique pair $(j, \mathbf{y})$, $j \in \mathcal{J}$, $\mathbf{y} \in \mathcal{Y}^{\mathcal{T}}$; the vector $\mathbf{w} = (\mathbf{q};\mathbf{g}) \in \mathbb{R}^d$ comprises both the optimized quality functions and $(\mathbf{z}_i \in \mathbb{R}^d, b_i \in \mathbb{R})$, $i \in I$, are constructed correspondingly to inscribe the inequalities (21b).

In Section 6, we will introduce a variant of the cutting plane algorithm to solve the QP task (21) (or (22), respectively). The cutting plane algorithm requires a subroutine which solves the LAC task (15). Using the compact notation, the LAC task reads $u_j = \operatorname{argmax}_{i\in I_j}(b_i - \langle \mathbf{w}, \mathbf{z}_i \rangle)$. Since this is NP-complete in general, we will use an LP relaxation which solves the task approximately. We will show that it is possible to assess the quality of the found solution $(\mathbf{w}, \boldsymbol{\xi})$ in terms of the objective of the learning task, that is, it is possible to bound the difference $Q_P(\mathbf{w}, \boldsymbol{\xi}) - Q_P(\mathbf{w}^*, \boldsymbol{\xi}^*)$. Further, we will prove that when the LAC task is solved exactly then the cutting plane algorithm finds an arbitrary precise solution $Q_P(\mathbf{w}, \boldsymbol{\xi}) - Q_P(\mathbf{w}^*, \boldsymbol{\xi}^*) \leq \varepsilon$, $\varepsilon > 0$, after a finite number of iterations. This guarantee does not apply for learning of the general max-sum classifier when an LP relaxation providing only an approximate solution is used. Though there is no theoretical guarantee, we will experimentally show that using an LP relaxation is sufficient to find a practically useful solution.

5.2 Supermodular Max-Sum Classifier

Problem 5 *(Learning a supermodular max-sum classifier from an inconsistent training set). For a given training set* $\mathcal{L} = \left\{(\mathbf{x}^j, \mathbf{y}^j) \in \mathcal{X}^{\mathcal{T}} \times \mathcal{Y}^{\mathcal{T}} \,\middle|\, j \in \mathcal{J}\right\}$ *and a regularization constant* C *find quality functions* $\mathbf{q}$ *and* $\mathbf{g}$ *solving the following QP task*

$$(\mathbf{g}^*, \mathbf{q}^*, \boldsymbol{\xi}^*) = \operatorname*{argmin}_{\mathbf{g},\mathbf{q},\boldsymbol{\xi}} \left[\frac{1}{2}\left(\|\mathbf{g}\|^2 + \|\mathbf{q}\|^2\right) + \frac{C}{m}\sum_{j\in\mathcal{J}} \xi_j\right], \tag{23a}$$

subject to

$$F(\mathbf{x}^j,\mathbf{y}^j;\mathbf{q},\mathbf{g}) - F(\mathbf{x}^j,\mathbf{y};\mathbf{q},\mathbf{g}) \geq L_\Delta(\mathbf{y}^j,\mathbf{y}) - \xi_j\,, \quad j \in \mathcal{J}, \mathbf{y} \in \mathcal{Y}^\mathcal{T}\,, \tag{23b}$$

$$\left.\begin{aligned} g_{tt'}(y,y') + g_{tt'}(y+1,y'+1) \geq g_{tt'}(y,y'+1) + g_{tt'}(y+1,y')\,, \\ \{t,t'\} \in \mathcal{E}\,, (y,y') \in \{1,\ldots,|\mathcal{Y}|-1\}^2\,. \end{aligned}\right\} \tag{23c}$$

Compared to the task (21) of learning a general max-sum classifier, the task (23) defined for the supermodular max-sum classifier contains additional linear constraints (23c). The added constraints (23c), when satisfied, guarantee that the found quality function $\mathbf{g}$ is supermodular. The total number of constraints increases to $n = m|\mathcal{Y}|^{|\mathcal{T}|} + |\mathcal{E}|(|\mathcal{Y}|-1)^2$. A compact form of the QP task (23) reads

$$\begin{aligned} (\mathbf{w}^*,\boldsymbol{\xi}^*) = \operatorname*{argmin}_{\mathbf{w},\boldsymbol{\xi}} & \left[\frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m}\sum_{j\in\mathcal{J}}\xi_j\right], \\ \text{s.t.} \quad \langle \mathbf{w},\mathbf{z}_i\rangle &\geq b_i\,, \qquad i \in I_0\,, \\ \langle \mathbf{w},\mathbf{z}_i\rangle &\geq b_i - \xi_j\,, \quad j \in \mathcal{J}\,, i \in I_j\,, \end{aligned} \tag{24}$$

where $b_i = 0$, $i \in I_0$, and $\mathbf{z}_i$, $i \in I_0$, account for the added supermodular constraints (23c).

In Section 6, we introduce a variant of the cutting plane algorithm which maintains the constraints (23c) satisfied. Thus the quality functions are always supermodular and the LAC task can be solved precisely by efficient polynomial-time algorithms. As a result, we can guarantee that the cutting plane algorithm finds a solution with an arbitrary finite precision in a finite number of iterations.

5.3 Max-sum Classifier with Strictly Trivial Equivalent

Problem 6 *(Learning a max-sum classifier with a strictly trivial equivalent from an inconsistent training set). For a given training set* $\mathcal{L} = \{(\mathbf{x}^j,\mathbf{y}^j) \in \mathcal{X}^\mathcal{T} \times \mathcal{Y}^\mathcal{T} \,\big|\, j \in \mathcal{J}\}$ *and regularization constant* C *find quality functions* $\mathbf{q}$ *and* $\mathbf{g}$ *that solve the following QP task*

$$(\mathbf{g}^*,\mathbf{q}^*,\boldsymbol{\xi}^*,\boldsymbol{\varphi}^{j*}, j\in\mathcal{J}) = \operatorname*{argmin}_{\mathbf{g},\mathbf{q},\boldsymbol{\xi},\boldsymbol{\varphi}^j, j\in\mathcal{J}} \left[\frac{1}{2}\big(\|\mathbf{g}\|^2 + \|\mathbf{q}\|^2 + \sum_{j\in\mathcal{J}}\|\boldsymbol{\varphi}^j\|^2\big) + \frac{C}{m}\sum_{j\in\mathcal{J}}\xi_j\right], \tag{25a}$$

subject to

$$\begin{aligned} q_t^{\boldsymbol{\varphi}^j}(y_t^j,x_t^j) - q_t^{\boldsymbol{\varphi}^j}(y,x_t^j) &\geq 1-\xi_j\,, \quad j\in\mathcal{J}\,, t\in\mathcal{T}\,, y\in\mathcal{Y}\setminus\{y_t^j\}\,, \\ g_{tt'}^{\boldsymbol{\varphi}^j}(y_t^j,y_{t'}^j) - g_{tt'}^{\boldsymbol{\varphi}^j}(y,y') &\geq 1-\xi_j\,, \quad j\in\mathcal{J}\,, \{t,t'\}\in\mathcal{E}\,, (y,y')\in\mathcal{Y}^2\setminus\{(y_t^j,y_{t'}^j)\}\,, \\ \xi_j &\geq 0\,, \qquad j\in\mathcal{J}\,. \end{aligned} \tag{25b}$$

The problem (25) is derived from the problem (20) which was formulated for a consistent training set. In the consistent case, it is required that each max-sum problem P^j has a strictly trivial equivalent whose unique solution equals to the desired labeling $\mathbf{y}^j$ given in the training set. In the case of an inconsistent training set, we allow some max-sum problems to violate this requirement. To this end, each subset of linear inequalities in (25b) which corresponds to the given max-sum problem P^j is relaxed by a single non-negative slack variable ξ_j. If a slack variable $\xi_j \geq 1$ then either the solution of the max-sum problem P^j differs from $\mathbf{y}^j$ or P^j has no trivial equivalent. The number of

such max-sum problems is upper bounded by $\sum_{j\in\mathcal{J}}^{m}\xi_j$ which is included into the objective function of (25a). Thus it corresponds to minimization of an upper bound on the empirical risk with the $0/1$-loss functions. The objective function also contains the Euclidean norm of the quality functions $(\mathbf{q},\,\mathbf{g})$ and the potentials $\boldsymbol{\varphi}^j$. Including the potentials $\boldsymbol{\varphi}^j$ into the objective function is somewhat arbitrary since it penalizes the transformation of max-sum problems to their trivial equivalents. An advantage of including the potentials is the fact that the dual representation of the task (25) has a simpler form which corresponds to the QP task of an ordinary multi-class SVM. Another variant is to remove the potentials from the objective function which corresponds to including a set of linear constraints to the dual task of (25) making the problem more difficult.

Compared to the previous learning problems, the number of variables $d=|\mathcal{X}||\mathcal{Y}||\mathcal{T}|+|\mathcal{Y}|^2|\mathcal{E}|+2m|\mathcal{E}||\mathcal{Y}|$ in (25) is increased by $2m|\mathcal{E}||\mathcal{Y}|$, however, the number of linear constraints $n=m|\mathcal{E}|(|\mathcal{Y}|^2-1)+m|\mathcal{T}|(|\mathcal{Y}|-1)+m$ is drastically smaller. In particular, n grows only polynomially compared to the exponential growth in the previous cases.

The QP task (25) can be rewritten into the compact form (22), that is, the same QP task as required when learning the general max-sum classifier. Unlike the general case, the optimized parameters $\mathbf{w}$ now comprise $\mathbf{w}=(\mathbf{q};\mathbf{g};\boldsymbol{\varphi}^1;\ldots;\boldsymbol{\varphi}^m)\in\mathbb{R}^d$ and the vectors $\mathbf{z}_i$, $i\in I$, are constructed correspondingly. However, as mentioned above, the main difference is much smaller $n=|I|$. As a result, the LAC task required by the cutting plane algorithm can be easily accomplished by an exhaustive search.

6. Algorithm for Quadratic Programming Tasks

In this section, we propose an algorithm to solve the QP tasks (21), (23) and (25) required for learning the max-sum classifiers from inconsistent training sets. We extend the cutting plane algorithm by Tsochantaridis et al. (2005) and its approximate version by Finley and Joachims (2005) in two directions. First, we will consider a more general QP task (23) which contains linear inequalities both with and without slack variables. Moreover, the inequalities without slack variables are required to be satisfied during the whole course of the algorithm. This extension is necessary for learning a supermodular max-sum classifier. Second, we propose to use a different stopping conditions to halt the algorithm. The original algorithm by Tsochantaridis et al. (2005) halts the optimization as soon as the linear constraints of a QP task are violated by a prescribed constant $\bar{\varepsilon}>0$ at most. We will use a stopping condition which is based on the duality gap. This allows us to control the precision of the found solution directly in terms of the optimized objective function. Moreover, the stopping condition can be easily evaluated even when the LAC task is solved only approximately by an LP relaxation which is useful for learning general max-sum classifiers. Finally, we will prove that the proposed algorithm converges in a finite number of iterations even in the case of a general max-sum classifier where the LAC task is NP-complete. We point out that the proof is similar to that of Tsochantaridis et al. (2005) but is technically simpler and it applies for the extended cutting plane algorithm proposed here. A general QP task which covers all the QP tasks (21), (23) and (25) reads

$$
\begin{aligned}
&(\mathbf{w}^*,\boldsymbol{\xi}^*)=\operatorname*{argmin}_{\mathbf{w},\boldsymbol{\xi}} Q_P(\mathbf{w},\boldsymbol{\xi})=\operatorname*{argmin}_{\mathbf{w},\boldsymbol{\xi}}\left[\frac{1}{2}\|\mathbf{w}\|^2+\frac{C}{m}\sum_{j\in\mathcal{J}}\xi_j\right],\\
&\text{s.t.}\qquad \begin{array}{lll}\langle\mathbf{w},\mathbf{z}_i\rangle & \geq & b_i\,, \qquad i\in I_0\,,\\ \langle\mathbf{w},\mathbf{z}_i\rangle & \geq & b_i-\xi_j\,, \quad j\in\mathcal{J}, i\in I_j\,,\end{array}
\end{aligned}
\tag{26}
$$

where $I = I_0 \cup I_1 \cup \cdots \cup I_m = \{1,\ldots,n\}$, $I_i \cap I_j = \{\emptyset\}$, $i \neq j$ are index sets. Note that we obtain the QP task (21) and (25) by setting $I_0 = \{\emptyset\}$. The Wolf dual of (26) reads

$$\begin{aligned} \boldsymbol{\alpha}^* &= \operatorname*{argmax}_{\boldsymbol{\alpha}} Q_D(\boldsymbol{\alpha}) = \operatorname*{argmax}_{\boldsymbol{\alpha}} \left[\langle \mathbf{b}, \boldsymbol{\alpha} \rangle - \frac{1}{2} \langle \boldsymbol{\alpha}, \mathbf{H}\boldsymbol{\alpha} \rangle \right], \\ \text{s.t.} \quad \sum_{i \in I_j} \alpha_i &= \frac{C}{m}, \quad j \in \mathcal{J}, \\ \alpha_i &\geq 0, \quad i \in I, \end{aligned} \tag{27}$$

where $\mathbf{H}$ is a positive semi-definite matrix $[n \times n]$ which contains the dot products $H_{i,j} = \langle \mathbf{z}_i, \mathbf{z}_j \rangle$. The primal variables $(\mathbf{w}^*, \boldsymbol{\xi}^*)$ can be computed from the dual variables $\boldsymbol{\alpha}^*$ by

$$\mathbf{w}^* = \sum_{i \in I} \alpha_i^* \mathbf{z}_i, \quad \xi_j^* = \max_{i \in I_j} (b_i - \langle \mathbf{w}^*, \mathbf{z}_i \rangle), \ j \in \mathcal{J}.$$

Note that the vectors $\mathbf{z}_i$, $i \in I$ appear in (27) only as dot products and thus the kernel functions can be potentially used.

It is not tractable to solve the QP task directly neither in the primal form nor in the dual form due to the exponential number of constraints or variables, respectively. The cutting plane algorithm alleviates the problem by exploiting the sparseness of maximal margin classifiers. Sparseness means that only a small portion of constraints of the primal task are active in the optimal solution, or equivalently, a small portion of dual variables are non-zero.

We denote $\overline{I} = I_0 \cup \overline{I}_1 \cup \cdots \cup \overline{I}_m$ disjoint subsets of selected indices from $I = I_0 \cup I_1 \cup \cdots \cup I_m$, that is, $\overline{I}_j \subseteq I_j$, $j \in \mathcal{J}$, while I_0 is always the same. The reduced task is obtained from (27) by considering only the selected variables $\{\alpha_i \mid i \in \overline{I}\}$ while the remaining variables $\{\alpha_i \mid i \in I \setminus \overline{I}\}$ are fixed to zero, that is, the reduced dual QP task reads

$$\overline{\boldsymbol{\alpha}} = \operatorname*{argmax}_{\boldsymbol{\alpha}} \left[\sum_{i \in \overline{I}} \alpha_i b_i - \frac{1}{2} \sum_{i \in \overline{I}} \sum_{j \in \overline{I}} \alpha_i \alpha_j H_{ij} \right], \tag{28a}$$

$$\begin{aligned} \text{s.t.} \quad \sum_{i \in \overline{I}_j} \alpha_i &= \frac{C}{m}, \quad j \in \mathcal{J}, \\ \alpha_i &\geq 0, \quad i \in \overline{I}, \\ \alpha_i &= 0, \quad i \in I \setminus \overline{I}. \end{aligned} \tag{28b}$$

To solve the reduced task (28), it is enough to maintain explicitly just the selected variables and thus its size scales only with $|\overline{I}|$. The cutting plane algorithm tries to select the subsets $\overline{I} = I_0 \cup \overline{I}_1 \cup \cdots \cup \overline{I}_m$ such that (i) $|\overline{I}|$ is sufficiently small to make the reduced task (28) solvable by standard optimization packages and (ii) the obtained solution well approximates the original task. The proposed extension of the cutting plane algorithm is as follows:

Algorithm 4 A cutting plane algorithm for QP task (26)

1: Select arbitrarily $\left(\overline{I}_j := \{u_j\}, u_j \in I_j\right)$, $j \in \mathcal{J}$. Set the desired precision $\varepsilon > 0$.

2: For selected indices $\overline{I} = I_0 \cup \overline{I}_1 \cup \cdots \cup \overline{I}_m$ solve the reduced task (28) to obtain $\overline{\boldsymbol{\alpha}}$ and compute the primal variable $\overline{\mathbf{w}} := \sum_{i \in \overline{I}} \overline{\alpha}_i \mathbf{z}_i$.

3: For all $j \in \mathcal{J}$ do:

3a: Solve the LAC task

$$u_j := \operatorname*{argmax}_{i \in I_j} \left(b_i - \langle \overline{\mathbf{w}}, \mathbf{z}_i \rangle \right) , \tag{29}$$

and set $\overline{\xi}_j := b_{u_j} - \langle \overline{\mathbf{w}}, \mathbf{z}_{u_j} \rangle$. If (29) is solved only approximately (when using LP relaxation) then set $\overline{\xi}_j$ to the smallest available upper bound on $b_{u_j} - \langle \overline{\mathbf{w}}, \mathbf{z}_{u_j} \rangle$.

3b: Set $\overline{I}_j := \overline{I}_j \cup \{u_j\}$ provided the following condition holds

$$\frac{C}{m}(b_{u_j} - \langle \overline{\mathbf{w}}, \mathbf{z}_{u_j} \rangle) - \sum_{i \in \overline{I}_j} \overline{\alpha}_i (b_i - \langle \overline{\mathbf{w}}, \mathbf{z}_i \rangle) > \frac{\varepsilon}{m} . \tag{30}$$

4: If the current solution $(\overline{\mathbf{w}}, \overline{\boldsymbol{\xi}}, \overline{\boldsymbol{\alpha}})$ satisfies the stopping condition

$$Q_P(\overline{\mathbf{w}}, \overline{\boldsymbol{\xi}}) - Q_D(\overline{\boldsymbol{\alpha}}) \leq \varepsilon , \tag{31}$$

or if the condition (30) was violated for all $j \in \mathcal{J}$ then halt. Otherwise go to Step 2.

Note that Algorithm (4) explicitly maintains only the selected variables $\{\alpha_i \mid i \in \overline{I}\}$ and the corresponding pairs $\{(b_i, \mathbf{z}_i) \mid i \in \overline{I}\}$. In Step 3a, the algorithm requires a subroutine to solve the LAC task (29). We consider two different variants of the algorithm. First, the task (29) can be solved precisely which applies to learning of supermodular max-sum classifiers and max-sum classifiers with a strictly trivial equivalent. Second, the task (29) can be solved only approximately, which applies to the general case when the max-sum problem is solved by an LP relaxation. The key property of an LP relaxation which we exploit here is that it provides an upper bound on the optimal solution of a max-sum problem, that is, an upper bound on the quantity $b_{u_j} - \langle \overline{\mathbf{w}}, \mathbf{z}_{u_j} \rangle$.

In Step 3b, the condition (30) is evaluated to decide whether adding the constraint $\langle \mathbf{w}, \mathbf{z}_{u_j} \rangle \geq b_{u_j} - \xi_j$ to the reduced QP task (28) brings a sufficient improvement or not. In Lemma 1 introduced below, we show that adding at least one constraint guarantees a minimal improvement regardless of whether the LAC problem is solved precisely or approximately. The algorithm halts when no constraint is added in Step 3b. This situation occurs only if the algorithm has converged to a solution which satisfies the stopping condition (31) provided the LAC task is solved exactly (cf. Theorem 4 and its proof).

Let us discuss the stopping condition (31). Note that the primal $(\overline{\mathbf{w}}, \overline{\boldsymbol{\xi}})$ and the dual variables $\overline{\boldsymbol{\alpha}}$ are feasible during entire progress of the algorithm. This holds even in the case when an LP relaxation is used to solve (29) since the $\overline{\xi}_j$, $j \in \mathcal{J}$ are set to upper bounds on their optimal values. Having feasible primal and dual variables, we can use the weak duality theorem to write

$$Q_P(\overline{\mathbf{w}}, \overline{\boldsymbol{\xi}}) - Q_P(\mathbf{w}^*, \boldsymbol{\xi}^*) \leq Q_P(\overline{\mathbf{w}}, \overline{\boldsymbol{\xi}}) - Q_D(\overline{\boldsymbol{\alpha}}) ,$$

which implies that any solution satisfying the stopping condition (31) differs from the optimal solution by at most ε. Note that the precision parameter $\bar{\varepsilon}$ used in the original stopping condition of Tsochantaridis et al. (2005) can also be related to the duality gap. Namely, it can be shown that $\bar{\varepsilon}$ approximate solution satisfies $Q_P(\overline{\mathbf{w}}, \overline{\boldsymbol{\xi}}) - Q_D(\overline{\boldsymbol{\alpha}}) \leq C\bar{\varepsilon}$.

Finally, we show that the algorithm converges in a finite number of iterations. The proof is based on Lemma 1 which asserts that a minimal improvement $\Delta_{\min} > 0$ of the dual objective function is guaranteed provided at least one new variable was added in Step 3(b) to the reduced task, that is, the condition (30) was satisfied for at least one $j \in \mathcal{J}$.

Lemma 1 *Provided at least one new variable was added in Step 3(b) of Algorithm 4 the improvement in the dual objective function $Q_D(\boldsymbol{\alpha})$ obtained after solving the reduced task is not less that*

$$\Delta_{\min} = \min\left\{\frac{\varepsilon}{2m}, \frac{\varepsilon^2}{8C^2R^2}\right\}, \quad \textit{where} \quad R = \max_{j\in\mathcal{J}} \max_{i\in I_j} \|\mathbf{z}_i\|\,.$$

Proof of Lemma 1 is given in Appendix B.

Using Lemma 1, it is easy to show that Algorithm 4 halts after a finite number of iterations. Note that the proof applies even for the case when the LAC task (29) is solved only approximately.

Theorem 4 *Let us assume that the primal QP task (26) is bounded and feasible. Algorithm 4 halts for arbitrary $\varepsilon > 0$ after at most T iterations, where*

$$T = \left(Q_D(\boldsymbol{\alpha}^*) - Q_D(\overline{\boldsymbol{\alpha}}^1)\right) \max\left\{\frac{2m}{\varepsilon}, \frac{8C^2R^2}{\varepsilon^2}\right\}, \qquad R = \max_{j\in\mathcal{J}} \max_{i\in I_j} \|\mathbf{z}_i\|\,,$$

and $\overline{\boldsymbol{\alpha}}^1$ is the solution of the reduced task obtained after the first iteration of Algorithm 4.

Proof Algorithm 4 halts if either the stopping condition (31) holds or no new variable was added in Step 3(b). Provided the algorithm does not halt in Step 4, the dual objective functions $Q_D(\boldsymbol{\alpha})$ is improved by at least $\Delta_{\min} > 0$ which is given in Lemma 1. Since the difference $Q_D(\boldsymbol{\alpha}^*) - Q_D(\overline{\boldsymbol{\alpha}}^1)$ is bounded from above the algorithm cannot pass through the Step 4 infinite number times and we can write

$$T \leq \frac{Q_D(\boldsymbol{\alpha}^*) - Q_D(\overline{\boldsymbol{\alpha}}^1)}{\Delta_{\min}} = \left(Q_D(\boldsymbol{\alpha}^*) - Q_D(\overline{\boldsymbol{\alpha}}^1)\right) \max\left\{\frac{2m}{\varepsilon}, \frac{8C^2R^2}{\varepsilon^2}\right\} < \infty\,.$$

■

The bound on a maximal number of iterations (or the bound on $\Delta_{\min}$, respectively) given in Theorem 4 is obtained based on the worst case analysis. As a result, the bound is an over-pessimistic estimate of the number of iterations usually required in practice. Because the bound is not useful for computing practical estimates of the number of iterations, we do not derive its particular variants for the QP tasks (21), (23) and (25).

Finally, we give a theorem which asserts that the stopping condition (31) is always satisfied after Algorithm 4 halts provided the LAC task (29) is solved exactly, that is, the found solution achieves the desired precision $Q_P(\overline{\mathbf{w}}, \overline{\boldsymbol{\xi}}) - Q_P(\mathbf{w}^*, \boldsymbol{\xi}^*) \leq \varepsilon$.

Theorem 5 *Let us assume that the LAC task (29) is solved exactly and the assumptions of Theorem 4 hold. In this case the condition (31) holds after Algorithm 4 halts.*

Proof We show that the assumption that no new variable was added in Step 3(b) and the condition (31) is violated leads to a contradiction. If no variable was added then the condition (30) is violated for all $j \in \mathcal{J}$. Summing up the violated conditions (30) yields

$$\sum_{j\in\mathcal{J}} \left[\frac{C}{m}\left(b_{u_j} - \langle\overline{\mathbf{w}}, \mathbf{z}_{u_j}\rangle\right) - \sum_{i\in\overline{I}_j} \overline{\alpha}_i \left(b_i - \langle\overline{\mathbf{w}}, \mathbf{z}_i\rangle\right)\right] \leq \sum_{j\in\mathcal{J}} \frac{\varepsilon}{m}\,. \tag{32}$$

The vector $\overline{\boldsymbol{\alpha}}$ is the optimal solution of the reduced task which includes all the variables $\{\alpha_i \,|\, i \in I_0\}$. Therefore by the complementary slackness (Karush-Kuhn-Tucker conditions) we have

$$\sum_{i \in I_0} \overline{\alpha}_i (b_i - \langle \overline{\mathbf{w}}, \mathbf{z}_i \rangle) = 0 \,. \tag{33}$$

By adding (33) to (32) and using $\overline{\mathbf{w}} = \sum_{i \in \overline{I}} \overline{\alpha}_i \mathbf{z}_i$, $\overline{\xi}_j = b_{u_j} - \langle \overline{\mathbf{w}}, \mathbf{z}_{u_j} \rangle$ we get

$$\begin{aligned}
\sum_{j \in \mathcal{J}} \left[\frac{C}{m} \left(b_{u_j} - \langle \overline{\mathbf{w}}, \mathbf{z}_{u_j} \rangle \right) - \sum_{i \in \overline{I}_j} \overline{\alpha}_i \left(b_i - \langle \overline{\mathbf{w}}, \mathbf{z}_i \rangle \right) \right] - \sum_{i \in \overline{I}_0} \overline{\alpha}_i \left(b_i - \langle \overline{\mathbf{w}}, \mathbf{z}_i \rangle \right) &\leq \varepsilon \\
\frac{C}{m} \sum_{j \in \mathcal{J}} \left(b_{u_j} - \langle \overline{\mathbf{w}}, \mathbf{z}_{u_j} \rangle \right) - \sum_{i \in \overline{I}} \overline{\alpha}_i \left(b_i - \langle \overline{\mathbf{w}}, \mathbf{z}_i \rangle \right) &\leq \varepsilon \\
\frac{C}{m} \sum_{j \in \mathcal{J}} \overline{\xi}_j - \sum_{i \in \overline{I}} \overline{\alpha}_i b_i + \langle \overline{\mathbf{w}}, \overline{\mathbf{w}} \rangle &\leq \varepsilon \,.
\end{aligned}$$

The last inequality can be equivalently written as $Q_P(\overline{\mathbf{w}}, \overline{\boldsymbol{\xi}}) - Q_D(\overline{\boldsymbol{\alpha}}) \leq \varepsilon$ which is in contradiction to the assumption that (31) is violated. ■

To summarize, in case when the LAC task (29) is solved exactly Algorithm 4 finds a solution with an arbitrary finite precision $\varepsilon > 0$ in a finite number of iterations. In case when the task (29) is solved only approximately by an LP relaxation Algorithm 4 always halts after a finite number of iterations but there is no guarantee that the desired precision was attained. The attained precision, however, can be determined by evaluating the duality gap $Q_P(\overline{\mathbf{w}}, \overline{\boldsymbol{\xi}}) - Q_D(\overline{\boldsymbol{\alpha}}) \geq Q_P(\overline{\mathbf{w}}, \overline{\boldsymbol{\xi}}) - Q_P(\mathbf{w}^*, \boldsymbol{\xi}^*)$.

In next sections, we will apply Algorithm 4 to the three particular instances of the QP tasks (21), (23) and (25).

6.1 General Max-Sum Classifier

In this section, we discuss optimization of the QP task (21) (or its compact form (22), respectively) using Algorithm 4.

In Step 1 of Algorithm 4, we have to select the initial subsets $(\overline{I}_j = \{u_j\}, u_j \in I_j), j \in \mathcal{J}$. A simple way is to use $u_j = (j, \mathbf{y}^j)$, which is equivalent to selecting the primal constraints $\langle \mathbf{w}, \mathbf{z}_i \rangle \geq b_i - \xi_j$ with $\mathbf{z}_i = \mathbf{0}$ and $b_i = 0$. Note that the task (21) does not contain the primal constraints without slack variables which implies that the subset $I_0 = \{\emptyset\}$.

The LAC task (29) required in Step 2a of Algorithm 4 amounts to solving an instance of a general max-sum problem

$$\begin{aligned}
\hat{\mathbf{y}}^j &= \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}^{\mathcal{T}}} \left[L_\Delta(\mathbf{y}, \mathbf{y}^j) - F(\mathbf{x}^j, \mathbf{y}^j; \mathbf{q}, \mathbf{g}) + F(\mathbf{x}^j, \mathbf{y}; \mathbf{q}, \mathbf{g}) \right] \\
&= \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}^{\mathcal{T}}} \left[\sum_{t \in \mathcal{T}} \left(q_t(y_t, x_t^j) + [\![y_t^j \neq y_t]\!] \right) + \sum_{\{t,t'\} \in \mathcal{E}} g_{tt'}(y_t, y_{t'}) \right] .
\end{aligned} \tag{34}$$

The obtained labeling $\hat{\mathbf{y}}^j$ is then used to construct the vector $\mathbf{z}_{(j,\hat{\mathbf{y}}^j)} = \Psi(\mathbf{x}^j, \mathbf{y}^j) - \Psi(\mathbf{x}^j, \hat{\mathbf{y}}^j)$. Because the task (34) is NP-complete in general, we use an LP relaxation discussed in Section 2.1. This

means that the found $\hat{\mathbf{y}}^j$ is not optimal but it is usually good enough to satisfy the condition (30) which then leads to a guaranteed improvement of the optimized dual objective (cf. Lemma 1). LP relaxation algorithms also provides an upper bound UB_j on the optimal solution of (34) which is essential to compute feasible $\overline{\xi}_j$, $j \in \mathcal{J}$ in Step 3(a) of Algorithm 4. In particular, $\overline{\xi}_j$ is computed as $\overline{\xi}_j = \mathrm{UB}_j - F(\mathbf{x}^j, \mathbf{y}^j; \mathbf{q}, \mathbf{g})$.

6.2 Supermodular Max-Sum Classifier

In this section, we discuss optimization of the QP task (23) (or its compact form (24), respectively) using Algorithm 4.

The initialization in Step 1 of Algorithm 4 can be carried out in the same way as described in Section 6.1. Since the QP task (24) involves the primal constraints $\langle \mathbf{w}, \mathbf{z}_i \rangle \geq b_i$, $i \in I_0$, without slack variables the corresponding $\mathbf{z}_i$, $i \in I_0$, must be included in the reduced QP task during the entire progress of Algorithm 4. The size $|I_0| = |\mathcal{E}|(|\mathcal{Y}|-1)^2$ grows only quadratically which allows to use standard optimization packages.

The LAC task (29) is of the same form as that for a general max-sum problem, that is, it requires solving (34). Unlike the general case, the task (34) can be solved exactly by a polynomial-time algorithms provided the quality functions $\mathbf{g}$ obey the supermodularity condition. The requirement of supermodularity is expressed by the primal conditions $\langle \overline{\mathbf{w}}, \mathbf{z}_i \rangle \geq 0$, $i \in I_0$ which are always included in the reduced task (28) and thus they should be always satisfied. There is one numerical issue, however, concerning a finite precision of QP solvers used to optimize the reduced task. It is possible that the primal constraints on supermodularity are slightly violated even if the reduced task is solved with a high precision. This might potentially (even though we have not observed the problem in practice) cause problems when solving the LAC task which relies on the supermodularity of $\mathbf{g}$. An easy solution to avoid the problem is to add the primal constraints $\langle \overline{\mathbf{w}}, \mathbf{z}_i \rangle \geq 0$, $i \in I_0$ with $\overline{\mathbf{w}} = \sum_{i \in \overline{I}} \alpha_i \mathbf{z}_i$ to the dual reduced task (28). In particular, we can solve the reduced task (28) subject to the constraints (28b) augmented by additional constraints $\sum_{i \in \overline{I}} \alpha_i H_{ij} \geq 0\,, \quad j \in I_0$. Note that adding of the additional constraints changes just the feasible set but it does not change the solution. Solving the reduced task with the added constraints by using any QP solver which produces a feasible solution with a finite precision (e.g., *interior point methods*) guarantees that the supermodularity constraints are satisfied.

6.3 Max-Sum Classifier with Strictly Trivial Equivalent

Unlike the previous two cases, the QP task (25) (or its compact form (22) respectively) contains only $n = m|\mathcal{E}|(|\mathcal{Y}|^2-1) + m|\mathcal{T}|(|\mathcal{Y}|-1) + m$ linear constraints. The form of the QP task (25) is the same as required for learning of an ordinary multi-class SVM (Crammer and Singer, 2001) which makes it possible to use existing optimization packages. The problem can be also solved by the proposed Algorithm 4. In this case, an initialization of the subset $(\overline{I}_j = \{u_j\}, u_j \in I_j), j \in \mathcal{J}$ performed in Step 1, can simply select the indices which correspond to the constraints $\xi_j \geq 0$, $j \in \mathcal{J}$. The LAC task (29) required in Step 3a can be solved exhaustively which means that we do not need any max-sum solver during the learning stage.

7. Experiments

Even though this article focuses primarily on theory, we present some examples to demonstrate functionality of the proposed learning algorithms. The examples are not meant to be a comprehensive evaluation.

7.1 Image Segmentation

We consider the problem of learning a classifier for segmentation of color images. We compare a general max-sum classifier, a supermodular max-sum classifier and a standard multi-class SVMs classifier (Crammer and Singer, 2001) as a baseline approach. In particular, we used the formulations given in Problem 4 and Problem 5 for learning the general and supermodular classifiers from the inconsistent training sets. To solve the LAC task required by Algorithm 4, we used an LP relaxation solver based on the Augmented Directed Acyclic Graph (ADAG) algorithm (Schlesinger 1976; see also the tutorial by Werner 2007).

We used the following three sets of color images (see Figure 1):

Microsoft Research (MSR) database: The original database [1] contains 240 images of 9 objects along with their manual segmentation. We selected a subset of 32 images each of which contains a combination of the following four objects: *cow*, *grass*, *water* and *void*. All images are of size 213×320 pixels. We created 3 random splits of the images into 10 training, 9 validation and 13 testing images. Reported results are averages taken over these three splits.

Shape-Sorter: We collected snapshots of a toy shape-sorter puzzle placed on a carpet. The snapshots are taken under varying view-angle and lighting conditions. The images contain 6 objects of different colors which we manually segmented. We split the images into a training set containing 14 images of size 100×100 pixels, validation and testing sets each containing 4 images of size 200×200 pixels.

Landscape: The data set is created from a single landscape image of size 280×600 which was manually segmented into *sky*, *border line* and *ground* areas. We divided the landscape image into 20 non-overlapping sub-windows of size 70×120 pixels. Finally, we split the 20 images into 5 training, 4 validation and 11 testing images.

In this experiment we define the max-sum classifier as follows. The set of objects $\mathcal{T} = \{(i,j) \mid i \in \{1,\dots,H\}, j \in \{1,\dots,W\}\}$ corresponds to the pixels of the input image of size $H \times W$. We consider a 4-neighborhood structure of image pixels, that is, $\mathcal{E}$ contains undirected edges between all pixels $(i,j) \in \mathcal{T}$ and $(i',j') \in \mathcal{T}$ which satisfy $|i-i'|+|j-j'|=1$. Each pixel $t \in \mathcal{T}$ is characterized by its observable state x_t and a label y_t. The observable state $x_t \in \mathcal{X} = \mathbb{R}^3$ is a vector containing RGB color values of t-th pixel. The label $y_t \in \mathcal{Y} = \{1,\dots,N\}$ assigns t-th pixel to one of N segments. We assume that the quality functions $q\colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ and $g\colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ do not depend on pixel $t \in \mathcal{T}$ (so called homogeneous model). The quality function $q(x,y) = \langle \mathbf{w}_y, x \rangle$ is linear in parameter $\mathbf{w}_y \in \mathbb{R}^3$, $y \in \mathcal{Y}$, as well as the function $g(y,y')$ represented by a vector $\mathbf{g} \in \mathbb{R}^{N^2}$. Hence the learned parameter vector $\mathbf{w} = (\mathbf{w}_1;\dots;\mathbf{w}_N;\mathbf{g}) \in \mathbb{R}^d$ is of dimension $d = 3N + N^2$. For the baseline approach we used a linear multi-class SVM which is equivalent to a max-sum classifier with a constant quality function $g(y,y')$, that is, the interrelation between labels is neglected.

1. Database B1 can be found at https://research.microsoft.com/vision/cambridge/recognition/default.htm.

	Multi-class SVM	General max-sum	Supermodular max-sum
MSR-database	21.74%	14.16%	14.03%
Shape-Sorter	1.31%	0.91%	0.92%
Landscape	6.22%	0.69%	0.65%

Table 1: Performance of max-sum classifiers compared to an ordinary SVM classifier on three segmentation data sets. The table contains average percentages of misclassified pixels per image.

We stopped the cutting plane Algorithm 4 when either the precision $(Q_P(\mathbf{w},\boldsymbol{\xi}) - Q_D(\boldsymbol{\alpha}))/Q_P(\mathbf{w},\boldsymbol{\xi}) \leq 10^{-2}$ was attained or no improving constraint was found after solving the LAC task. For the supermodular max-sum classifier the required precision was always attained since the LAC can be solved exactly. For the general max-sum classifier the algorithm failed to converge only in a few cases when the regularization constant C was set too high. The average time required for learning on an ordinary desktop PC was around 11 hours for the MSR-database, 3 hours for the Shape-Sorter data set and 6 minutes for the Landscape data set. The bottleneck is solving the LAC task by the ADAG max-sum solver which takes approximately 95% of the training time. Though we have not fully exploited this possibility, the training time for the supermodular max-sum classifier can be considerably reduced by using min-cut/max-flow algorithms to solve the LAC task. The min-cut/max-flow algorithms optimized for a grid neighbourhood structure can achieve near real-time performance (Boykov and Kolmogorov, 2004). For this reason the supermodular max-sum classifier is favourable when the training time or the classification time is of importance.

The only free parameter of the learning algorithm is the regularization constant C which we tuned on the validation images. The classifier with the best performance on the validation set was then assessed on independent testing images. Due to a different size of images it is convenient to present errors in terms of the normalized additive loss function $L(\mathbf{y},\mathbf{y}') = \frac{100}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}[\![y_t \neq y'_t]\!]$ corresponding to percentage of misclassified pixels in the image. Obtained results are summarized in Table 1. It is seen that the max-sum classifiers significantly outperformed the multi-class SVM on all data sets. Performances of the general and the supermodular max-sum classifiers are almost identical. This shows a good potential of the learning algorithm to extend applicability of the polynomially solvable class of supermodular max-sum problems. Note that selecting a proper supermodular function by hand is difficult due to an unintuitive form of the supermodularity conditions (7). Figure 1 shows examples of testing images and their labeling estimated by the max-sum classifier.

7.2 Learning the Rules of Sudoku

In this section, we demonstrate the algorithm for learning the max-sum classifier with a strictly trivial equivalent. We will consider a problem of learning the game rules of a logic-based number placement puzzle Sudoku[2]. Figures 2(a) and 2(b) show an example of the Sudoku puzzle and its solution, respectively. The aim of the puzzle is to fill in a 9×9 grid such that each column, each row and each of 9 non-overlapping 3×3 boxes contains the numbers from 1 to 9. A player starts

2. For more details and references see http://en.wikipedia.org/wiki/Sudoku .

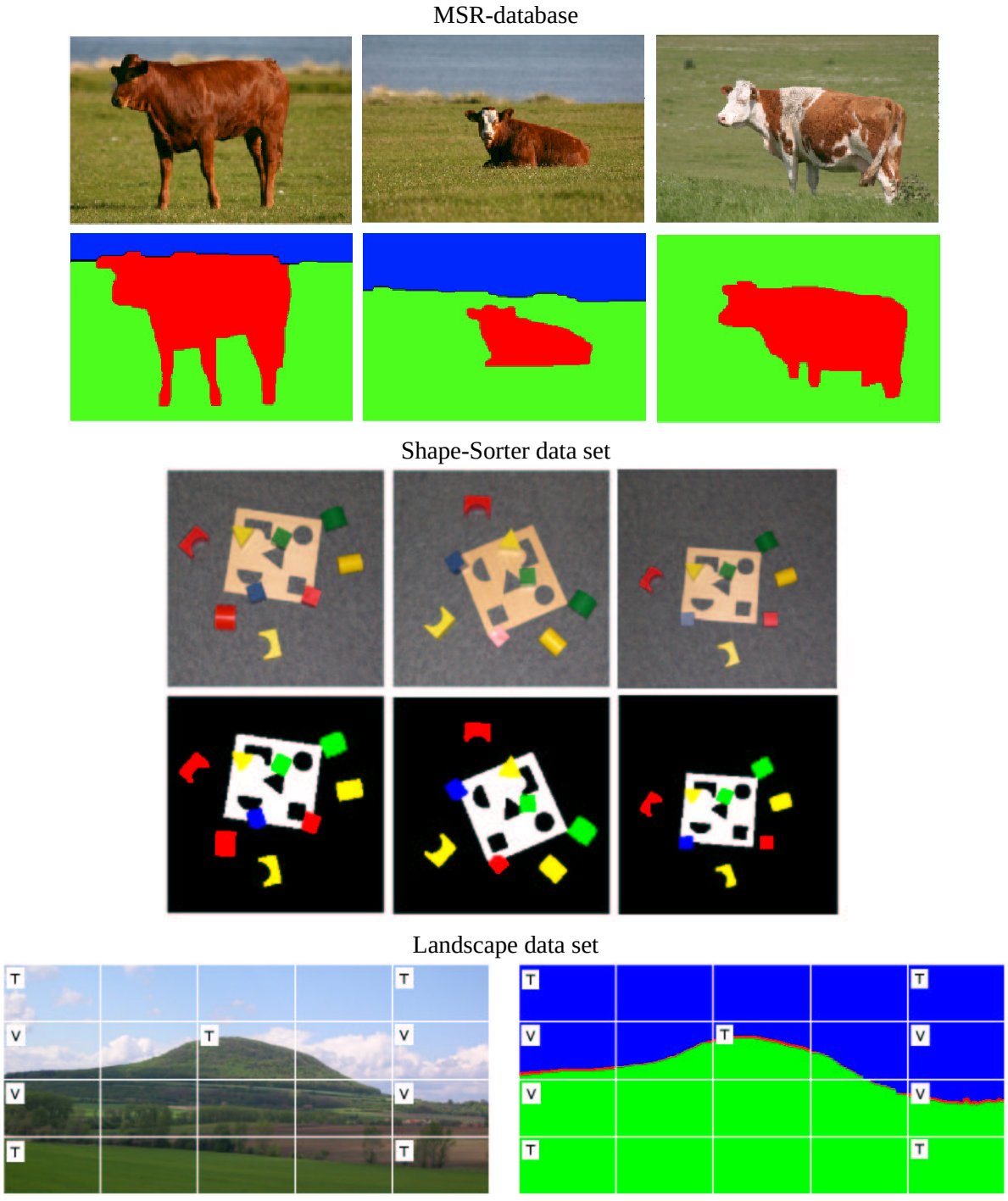

Figure 1: A sample from three data sets used in experiments. The figures show testing images and their labelings estimated by the max-sum classifier. For the Landscape data set the training and the validation sub-images are marked.

					8			
	1	9	5	6		2		
2	5			1		3	6	
9					2		8	1
	8	2	6		9			
5	7		1					2
	2	1		9			4	3
		5		7	6	8		
8	9		3					

(a)

7	6	3	4	2	8	1	9	5
4	1	9	5	6	3	2	7	8
2	5	8	9	1	7	3	6	4
9	3	4	7	5	2	6	8	1
1	8	2	6	3	9	4	5	7
5	7	6	1	8	4	9	3	2
6	2	1	8	9	5	7	4	3
3	4	5	2	7	6	8	1	9
8	9	7	3	4	1	5	2	6

(b)

(c)

Figure 2: Example of Sudoku puzzle: (a) input puzzle; (b) solution; (c) handwritten Sudoku grid created from the USPS database.

from an incompletely filled grid which is constructed such a way that the proper puzzle has a unique solution.

Our goal is to learn a max-sum classifier (artificial player) able to solve arbitrary Sudoku puzzle. A part of the rules is a priori known to the system while the rest is learned from examples of incomplete puzzle and its correct solution. In a standard scenario, the learned classifier is validated on a testing set. In this particular case, however, it is possible to show that the found max-sum classifier solves correctly all the Sudoku puzzles (the number of Sudoku solutions is approximately 6.6×10^{27}, that is, the number of puzzles is much larger).

Note, that the Sudoku puzzle can be naturally expressed as solving the *constraint satisfaction problem* CSP which is a subclass of a max-sum labeling problem when the quality functions attain only two values $\{0, -\infty\}$. From this viewpoint, a max-sum classifier is a richer model than necessarily needed for Sudoku puzzle.

Let us define the learning problem using the notation of this paper. Solving the puzzle is equivalent to solving the max-sum problem $(\mathcal{T}, \mathcal{E}, \mathcal{Y}, \mathbf{q}, \mathbf{g}, \mathbf{x})$. The set of objects $\mathcal{T} = \{(i, j) \mid i \in \{1, \ldots, 9\}, j \in \{1, \ldots, 9\}\}$ corresponds to the cells of a 9×9 grid. The neighbourhood structure

$$\mathcal{E} = \{\{(i, j), (i', j')\} \mid i = i' \vee j = j' \vee (\lceil i/3 \rceil = \lceil i'/3 \rceil \wedge \lceil j/3 \rceil = \lceil j'/3 \rceil)\},$$

contains pairs of cells whose relation plays a role in the game rules, that is, the cells in rows, columns, and the 3×3 boxes. The set of observations $\mathcal{X} = \{\square, 1, \ldots, 9\}$ represents the input filling of cells where the special symbol $\square$ means an empty cell. The set of labels $\mathcal{Y} = \{1, \ldots, 9\}$ corresponds to numbers which a player can fill in. The quality functions $\mathbf{q}$ and $\mathbf{g}$ are assumed to be homogeneous, that is, $q_t(x, y) = q(x, y)$ and $g_{tt'}(y, y') = g(y, y')$. Moreover, the structure of quality function $q(y, x)$ is assumed to be $q(y, x) = q(y)$ for $y = x$ and $q(y, x) = q_\square$ for $x = \square$ or $x \neq y$. The particular values of quality functions $(\mathbf{q}; \mathbf{g}) \in \mathbb{R}^{9 \times 9 + 10}$, which specify the rules of the game, are unknown to the system.

Using the setting introduced above, it is easy to see that the solution of a max-sum problem $(\mathcal{T}, \mathcal{E}, \mathcal{Y}, \mathbf{q}, \mathbf{g}, \mathbf{x})$ is a correct solution of the Sudoku puzzles if for any triplet $(y, y', y'') \in \mathcal{Y}^3$ such

$q(y,x)$

$q_\square$	y	1	2	3	4	5	6	7	8	9
-1870	$q(y)$	867	295	696	234	224	339	228	455	378

$g(y,y')$

y/y'	1	2	3	4	5	6	7	8	9
1	-552	19	47	66	65	73	61	66	73
2	84	-500	67	57	71	56	79	60	65
3	78	30	-437	48	48	47	68	50	54
4	73	-28	63	-266	55	53	76	56	60
5	23	67	57	47	-397	46	68	49	52
6	74	66	58	49	49	-448	69	49	54
7	-26	9	28	49	48	47	-342	48	54
8	36	77	60	49	49	46	67	-441	52
9	79	20	57	48	46	48	65	51	-441

Table 2: The quality functions **q** and **g** learned from examples. The learned functions **q** and **g** satisfy the conditions (35) which guarantee the optimal solution.

that $y' \neq y''$ the quality functions satisfy the following conditions

$$q_\square < q(y) \qquad \text{and} \qquad g(y,y) < g(y',y'')\,. \tag{35}$$

The first condition just forces the max-sum classifier to use the numbers from filled in cells as labels. The second condition, in compliance with to the Sudoku rules, ensures that neighbouring cells will not be assigned the same label. The conditions (35) allows us to verify whether the found classifier is an optimal one.

Because a proper Sudoku puzzle have a unique solution it is natural to require that the max-sum classifier also guarantees this condition. The conditions (35) show that the ground truth quality function $g(y,y')$ is not supermodular. For this reasoning, Algorithm 3 for learning of the max-sum classifier with a strictly trivial solution seems to be a reasonable choice.

We created 18 distinct training sets each containing a single example of an incomplete puzzle and its correct solution. In all cases, the proposed Algorithm 3 converged to a solution after the number of iterations ranging 12×10^3 to 95×10^3 approximately, which took less than one minute on an ordinary desktop PC. The conditions (35) were satisfied for all the solutions found by an algorithm, that is, an optimal max-sum classifier was found just from a single example (using more general **q** and **g** would probably require more examples). Table 2 shows an example of the found quality functions **q** and **g**. In this case, it is even possible to precisely compute the output of the max-sum classifier by branch-and-bound algorithm since the depth searching tree is limited due the requirement on a unique solution of the Sudoku and modest complexity tractable for human players. Note, however, that no max-sum solver was required during the course of the learning algorithm.

	Multi-class SVM	Max-Sum classifier
Linear	8.81%	7.20%
Gaussian kernel	5.80%	0.04%

Table 3: Recognition of Sudoku grids created from the USPS database.

7.3 Handwritten Character Recognition with Structured Outputs

In this section, we compare the max-sum classifier with a strictly trivial equivalent learned by the Perceptron Algorithm 3 against an ordinary multi-class SVM classifier. To this end, we modified the Sudoku problem from Section 7.2 in two ways. Firstly, the input observations are handwritten digits taken from the USPS database. In particular, the input observation $x \in \mathcal{X} = \mathbb{R}^{256}$ is a vector containing pixels of a grey-scale image 16×16 which depicts a digit from 1 to 9. Secondly, the input of the classifier is a fully solved Sudoku grid, that is, $\mathcal{Y} = \{1,\dots,9\}$, as the multi-class SVM cannot solve an incomplete puzzle unlike the max-sum classifier. The neighbourhood structure $(\mathcal{T},\mathcal{E})$ is the same as in the experiment from Section 7.2. Figure 2(c) shows and example of input observations and Figure2(b) shows a corresponding labeling to be estimated by the max-sum classifier.

Similarly to the experiment in Section 7.2, we considered a quality function $g(y,y')$ of a general form $\mathbf{g} \in \mathbb{R}^{|\mathcal{Y}|^2}$. However, we used two different forms of the quality functions $q(x,y)$. First, a linear function $q(x,y) = \langle \mathbf{w}_y, x\rangle$ which lead to learning a parameter vector $\mathbf{w} = (\mathbf{w}_1;\dots;\mathbf{w}_{|\mathcal{Y}|};\mathbf{g}) \in \mathbb{R}^d$ of dimension $d = 256|\mathcal{Y}| + |\mathcal{Y}|^2 = 2385$. Second, we applied the Gaussian kernel function defined as $k(x,x') = \exp(-\sigma\|x-x'\|^2)$ for some $\sigma > 0$. In this case, the quality function is $q(x,y) = \langle \mathbf{v}_y, \mathbf{k}(x)\rangle$ where $\mathbf{k}(x) = (k(x,x_t^j) \mid t \in \mathcal{T}, j = 1,\dots,m)$ denotes a vector of kernel functions centered in all training observations. The corresponding parameter vector $\mathbf{w} = (\mathbf{v}_1;\dots;\mathbf{v}_{|\mathcal{Y}|};\mathbf{g}) \in \mathbb{R}^d$ was of dimension $d = m|\mathcal{T}||\mathcal{Y}| + |\mathcal{Y}|^2 = 7371$.

We created a set of 30 examples of Sudoku grids $\mathbf{x} = (x_t \in \mathcal{X} \mid t \in \mathcal{T})$ and their solutions $\mathbf{y} = (y_t \in \mathcal{Y} \mid t \in \mathcal{T})$. Note that a single grid $\mathbf{x}$ contains $9 \times 9 = 81$ digits from the USPS database. We generated 3 random splits of the 30 examples into 10 training, 10 validation and 10 testing examples. The Perceptron Algorithm 3 required a few seconds to converge to a solution with zero training error when the linear kernel was used and around 3 minutes for the Gaussian kernel. The optimal kernel width σ was tuned on the validation data. In addition, we also tuned the regularization constant C for the multi-class SVM. The model with the best performance on the validation set was then assessed on the testing data. Table 3 shows the average classification performance computed over the 3 random splits.

It is seen that the max-sum classifier significantly outperforms the multi-class SVM regardless of the used kernel functions. The classification error 5.80% achieved for the multi-class SVM with the Gaussian kernel is slightly higher than 4.00% reported in Schölkopf et al. (1995). The reason is that we trained on a smaller number of examples (namely, 810 compared to 7291). On the other hand, the smaller training set is sufficient to achieve nearly error-less performance when the structure in the output space is considered. In particular, the error of the max-sum classifier with the Gaussian kernel is 0.04%. Note that without considering the structure a human recognition rate is 2.5% and the best published machine performance is 2.6% (Schölkopf et al., 1995).

8. Conclusions

In this article we have examined discriminative learning of max-sum classifiers. Learning of a max-sum classifier leads to satisfying a set of linear inequalities or solving a QP task with linear inequality constraints. A characteristic feature of these tasks is a huge number of linear constraints which is proportional to the number of possible responses (labelings) of a max-sum classifier. Efficient optimization methods for solving these tasks are the perceptron and the cutting plane algorithm, respectively. These methods manage to solve large problems provided the response of a max-sum classifier can be evaluated efficiently. Direct application of these methods is not tractable because computing a response of a general max-sum classifier is NP-complete.

We have proposed variants of the perceptron and the cutting plane algorithm for learning supermodular max-sum classifiers whose response can be computed efficiently in polynomial time. We have augmented the optimization tasks by additional linear constraints which guarantee that a max-sum classifier is supermodular. The perceptron and the cutting plane algorithm are modified such that added constraints on supermodularity are maintained satisfied during the course of optimization. This modification allows to compute the response of a max-sum classifier efficiently thus making the learning problem tractable.

We have defined a class of max-sum classifiers with a strictly trivial equivalent which are solvable exactly in polynomial time by an LP relaxation. We have showed that this class covers at least acyclic and supermodular max-sum classifiers with a unique solution. Another favorable property of this class is that the learning problems contain only polynomially-sized sets of linear constraints. As a result, the perceptron and the cutting plane algorithms do not require to call any max-sum solver during the course of optimization.

We have proposed a variant of the cutting plane algorithm which can approximately solve the learning problem formulated for the general max-sum classifier. The response of the max-sum classifier is approximated by an LP relaxation for which specialized optimization algorithms exist. Using an approximate response prohibits a guarantee that the cutting plane algorithm finds a solution with an arbitrary precision. This is not an obstacle, however, for using the algorithm in practice as we demonstrated experimentally.

Acknowledgments

Authors are deeply grateful to Prof. Michail I. Schlesinger for inspiring this work by many important ideas which he delivered in personal communication and during his lectures on structural pattern recognition. Our special thanks go to Tomáš Werner for reading this article and giving us many valuable comments.

The authors would like to acknowledged the support from the EU INTAS project PRINCESS 04-77-7347. The main part of this work has been done while the first author was with the Center for Machine Perception. The first author was also supported by Marie Curie Intra-European Fellowship grant SCOLES (MEIF-CT-2006-042107) during his current fellowship in Fraunhofer-FIRST.IDA institute.

Appendix A.

In this appendix we prove Theorem 3 introduced in Section 2.3. Prior to proving the theorem, we will introduce the concept of *local consistency* (Schlesinger, 1976) and the theorem asserting that the problems with a trivial equivalent contain the acyclic and supermodular problems (Schlesinger, 1976; Schlesinger and Flach, 2000).

Definition 3 *The maximal node* (t,y) *is called* locally consistent *if for each neighboring object* $t' \in \mathcal{N}(t)$ *there exists a maximal edge* $\{(t,y),(t',y')\}$. *The maximal edge* $\{(t,y),(t',y')\}$ *is called* locally consistent *if the nodes* (t,y) *and* (t',y') *are maximal.*

The maximal nodes and maximal edges which do not satisfy Definition 3 will be called *locally inconsistent*.

Theorem 6 *Schlesinger (1976); Schlesinger and Flach (2000) Let* $P = (\mathcal{T},\mathcal{E},\mathcal{Y},\mathbf{q},\mathbf{g},\mathbf{x})$ *be a max-sum problem. If* $(\mathcal{T},\mathcal{E})$ *is an acyclic graph or quality functions* $\mathbf{g}$ *are supermodular then* P *is equivalent to a trivial problem.*

Recall, that P with the optimal solution $\mathbf{y}^* \in \mathcal{Y}^{\mathcal{T}}$ has a trivial equivalent $P^{\boldsymbol{\varphi}}$ if there exist potentials $\boldsymbol{\varphi}$ such that the following set of linear inequalities holds

$$\begin{array}{rcll} q_t^{\boldsymbol{\varphi}}(y_t^*,x_t) & \geq & q_t^{\boldsymbol{\varphi}}(y,x_t)\,, & t \in \mathcal{T}\,, y \in \mathcal{Y} \setminus \{y_t^*\}\,, \\ g_{tt'}^{\boldsymbol{\varphi}}(y_t^*,y_{t'}^*) & \geq & g_{tt'}^{\boldsymbol{\varphi}}(y_t,y_{t'})\,, & \{t,t'\} \in \mathcal{E}\,, (y,y') \in \mathcal{Y}^2 \setminus \{(y_t^*,y_{t'}^*)\}\,. \end{array} \tag{36}$$

The system (36) differs from the definition of problems with a strictly trivial equivalent (19) just by using the non-strict inequalities. Finally, we will prove two auxiliary lemmas:

Lemma 2 *Let* P *be an acyclic problem which has a unique solution* $\mathbf{y}^*$ *and let* $P^{\boldsymbol{\varphi}}$ *be a trivial equivalent of* P. *Then only two cases can occur: (i)* $P^{\boldsymbol{\varphi}}$ *is strictly trivial or (ii) there is at least one maximal locally inconsistent node or edge.*

Proof We will show that violating both the assertions (i) and (ii) contradicts the assumption that $\mathbf{y}^*$ is unique. Assuming $P^{\boldsymbol{\varphi}}$ is not strictly trivial implies that there exists a maximal node (t,y_t^0) such that $y_t^0 \neq y_t^*$. Let us construct a labeling $\mathbf{y}^0$ such that y_t^0 belongs to $\mathbf{y}^0$. The remaining labels are determined by repeating the following procedure $(|\mathcal{T}|-1)$ times:

- Let $t' \in \mathcal{T}$ be an object whose label $y_{t'}^0$ has been already determined and let $t'' \in \mathcal{N}(t')$ be an object whose label $y_{t''}^0$ has not been determined yet. Then set up the label $y_{t''}^0$ such that $\{(t',y_{t'}^0),(t'',y_{t''}^0)\}$ is a maximal edge.

The constructed labeling $\mathbf{y}^0$ is the optimal solution of P because it is composed of maximal nodes and edges. Note that this simple construction of the optimal labeling is possible because the graph $(\mathcal{T},\mathcal{E})$ is acyclic. Thus we have $\mathbf{y}^0 \neq \mathbf{y}^*$ because $y_t^0 \neq y_t^*$ which implies that $\mathbf{y}^*$ is not unique. ■

Lemma 3 *Let* P *be a supermodular problem with an unique solution* $\mathbf{y}^*$ *and let* $P^{\boldsymbol{\varphi}}$ *be a trivial equivalent of* P. *Then only two cases can occur: (i)* $P^{\boldsymbol{\varphi}}$ *is strictly trivial or (ii) there is at least one maximal locally inconsistent node or edge.*

Proof We will show that violating both the assertions (i) and (ii) contradicts the assumption that $\mathbf{y}^*$ is unique. Let $\mathcal{Y}_t^0 = \{(t,y) \mid q_t^{\boldsymbol{\varphi}}(y,x_t) = \max_{y' \in \mathcal{Y}} q_t^{\boldsymbol{\varphi}}(y',x_t)\}$ denote a set of all maximal nodes corresponding to the object $t \in \mathcal{T}$. Let us construct the labeling $\mathbf{y}^h = (y_t^h \mid t \in \mathcal{T})$ composed of the highest maximal nodes; (t,y_t^h) is the highest maximal node if $(t,y_t^h) \in \mathcal{Y}_t^0$ and $y_t^h > y$ for all $(t,y) \in \mathcal{Y}_t^0 \setminus \{t,y_t^0\}$. Recall, that the labels are fully ordered for the supermodular problems.

Now, we show that the labeling $\mathbf{y}^h$ is optimal since all its edges $\{(t,y_t^h),(t',y_{t'}^h)\} \in \mathcal{E}_{\mathcal{Y}}$ are also maximal. Let us assume that there exists an edge $\{(t,y_t^h),(t',y_{t'}^h)\}$ which is not maximal. Then, by assumption of local consistency, there exist edges $\{(t,y_t^h),(t',y_{t'})\}$ and $\{(t,y_t),(t',y_{t'}^h)\}$ which are maximal. Note that $y_t < y_t^h$ and $y_{t'} < y_{t'}^h$ because (t,y_t^h) and $(t',y_{t'}^h)$ are the highest nodes. From the condition of supermodularity (7) and (4a) we have

$$\begin{aligned} 0 \leq g_{tt'}(y_t^h,y_{t'}^h) + g_{tt'}(y_t,y_{t'}) - g_{tt'}(y_t^h,y_{t'}) - g_{tt'}(y_t,y_{t'}^h) \\ = g_{tt'}^{\boldsymbol{\varphi}}(y_t^h,y_{t'}^h) + g_{tt'}^{\boldsymbol{\varphi}}(y_t,y_{t'}) - g_{tt'}^{\boldsymbol{\varphi}}(y_t^h,y_{t'}) - g_{tt'}^{\boldsymbol{\varphi}}(y_t,y_{t'}^h)\,. \end{aligned}$$

This condition, however, cannot be satisfied if the edge $\{(t,y_t^h),(t',y_{t'}^h)\}$ is not maximal which is a contradiction. Similarly, we can show that the labeling $\mathbf{y}^l$ composed of the lowest maximal nodes (defined analogically) is also optimal.

Finally, the assumption that $P^{\boldsymbol{\varphi}}$ is not strictly trivial implies that for some object $t \in \mathcal{T}$ there exists a maximal node (t,y_t^0) such that $y_t^0 \neq y_t^*$. W.l.o.g. we can select (t,y_t^0) which is either the highest maximal or the lowest maximal node. Thus $y_{t^*}^0$ belongs either to the labeling $\mathbf{y}^h$ or $\mathbf{y}^l$ which are optimal and differ from $\mathbf{y}^*$. This contradicts the assumption that $\mathbf{y}^*$ is unique. ■

Proof of Theorem 3: Let P be an acyclic or supermodular max-sum problem with the unique solution $\mathbf{y}^* \in \mathcal{Y}^{\mathcal{T}}$. Let $\boldsymbol{\varphi}$ be the potentials such that the max-sum problem $P^{\boldsymbol{\varphi}}$ is a trivial equivalent of P. The existence of such $P^{\boldsymbol{\varphi}}$ is guaranteed by Theorem 6. Then, by Lemma 2 and Lemma 3, $P^{\boldsymbol{\varphi}}$ is either strictly trivial or there exists a maximal node (t,y_t^0) or a maximal edge $\{(t,y_t^0),(t',y_{t'}^0)\}$ which are locally inconsistent, that is, (t,y_t^0) and $\{(t,y_t^0),(t',y_{t'}^0)\}$ do not belong to $\mathbf{y}^*$. We will introduce a procedure which changes the potentials $\boldsymbol{\varphi}$ in such a way that the inconsistent maximal node (t,y_t^0) or inconsistent maximal edge $\{(t,y_t^0),(t',y_{t'}^0)\}$, respectively, become non-maximal while other maximal (non-maximal) nodes or edges remain maximal (non-maximal). Repeating this procedure for all inconsistent maximal nodes and edges makes the inequalities (36) satisfied strictly, that is, the problem $P^{\boldsymbol{\varphi}}$ becomes strictly trivial which is to be proven. The procedures for elimination of inconsistent nodes and edges read:

Elimination of the inconsistent maximal node (t,y_t^0)**:** Let $t' \in \mathcal{N}(t)$ be such a neighbor of t that the set of edges $\mathcal{E}_{tt'}(y_t^0) = \{\{(t,y_t^0),(t',y_{t'})\} \mid y_{t'} \in \mathcal{Y}\}$ does not contain any maximal edge. Let ε be a number computed as

$$\varepsilon = \frac{1}{2}\left[\max_{(y,y') \in \mathcal{Y}^2} g_{tt'}^{\boldsymbol{\varphi}}(y,y') - \max_{y' \in \mathcal{Y}} g_{tt'}^{\boldsymbol{\varphi}}(y_t^0,y')\right].$$

Since $\mathcal{E}_{tt'}(y_t^0)$ does not contain maximal edges this implies $\varepsilon > 0$. Adding ε to the potential $\varphi_{tt'}(y_t^0) := \varphi_{tt'}(y_t^0) + \varepsilon$ decreases the quality $q_t^{\boldsymbol{\varphi}}(y_t^0,x_t) = q_t(y_t^0,x_t) - \sum_{t'' \in \mathcal{N}(t)} \varphi_{tt''}(y_t^0)$ by ε and increases the qualities $g_{tt'}^{\boldsymbol{\varphi}}(y,y') = g_{tt'}(y,y') + \varphi_{tt'}(y) + \varphi_{t't}(y')$, $\{(t,y),(t',y')\} \in \mathcal{E}_{tt'}(y_t^0)$ by ε. This change makes the node (t,y_t^0) non-maximal while the edges $\mathcal{E}_{tt'}(y_t^0)$ remain non-maximal as before. The qualities of other nodes and edges remain unchanged.

Elimination of the inconsistent maximal edge $\{(t,y_t^0),(t',y_{t'}^0)\}$**:** W.l.o.g. let $(t',y_{t'}^0)$ be a non-maximal node. Notice, that all edges from $\mathcal{E}_{t't}(y_{t'}^0) = \{(t,y_t),(t',y_{t'}^0)\} \mid y_t \in \mathcal{Y}\}$ are locally inconsistent and they cannot be a part of the optimal solution. Let ε be a number computed as

$$\varepsilon = \frac{1}{2}\left[\max_{y\in\mathcal{Y}} q_{t'}^{\boldsymbol{\varphi}}(y,x_{t'}) - q_{t'}^{\boldsymbol{\varphi}}(y_{t'}^0,x_{t'})\right].$$

Because $(t',y_{t'}^0)$ is non-maximal $\varepsilon > 0$. Subtracting ε from the potential $\varphi_{t't}(y_{t'}^0) := \varphi_{t't}(y_{t'}^0) - \varepsilon$ decreases the qualities $g_{tt'}^{\boldsymbol{\varphi}}(y,y') = g_{tt'}(y,y') + \varphi_{tt'}(y) + \varphi_{t't}(y')$, $\{(t,y),(t',y')\} \in \mathcal{E}_{t't}(y_{t'}^0)$ by ε and increases the quality $q_{t'}^{\boldsymbol{\varphi}}(y_{t'}^0,x_{t'}) = q_{t'}(y_{t'}^0,x_{t'}) - \sum_{t''\in\mathcal{N}(t')}\varphi_{tt''}(y_{t'}^0)$ by ε. This change makes all edges from $\mathcal{E}_{t't}(y_{t'}^0)$ non-maximal while the node $(t',y_{t'}^0)$ remains non-maximal as before. The qualities of other nodes and edges remain unchanged. ■

Appendix B.

In this appendix we prove Lemma 1 given in Section 6.

Proof of Lemma 1: We show that if a new variable was added then we can construct a vector $\overline{\boldsymbol{\beta}}$ such that optimizing the dual objective $Q_D(\boldsymbol{\alpha})$ over a line segment between the current solution $\overline{\boldsymbol{\alpha}}$ and the vector $\overline{\boldsymbol{\beta}}$ yields a guaranteed improvement. Since the reduced QP task solved in Step 2 optimizes in the space of all selected variables which contains the line segment between $\overline{\boldsymbol{\alpha}}$ and $\overline{\boldsymbol{\beta}}$, the obtained improvement cannot be smaller.

Let us assume an optimization of the dual objective $Q_D(\boldsymbol{\alpha})$ of the QP task (27) w.r.t. a line segment between the current solution $\overline{\boldsymbol{\alpha}}$ and an arbitrary feasible vector $\overline{\boldsymbol{\beta}}$. The problem is equivalent to searching for the maximum of an univariate quadratic function

$$\begin{aligned} Q_L(\tau) &= Q_D\Big((1-\tau)\overline{\boldsymbol{\alpha}} + \tau\overline{\boldsymbol{\beta}}\Big) \\ &= (1-\tau)\langle \mathbf{b}, \overline{\boldsymbol{\alpha}}\rangle + \tau\langle \mathbf{b}, \overline{\boldsymbol{\beta}}\rangle - \frac{1}{2}(1-\tau)^2\langle \overline{\boldsymbol{\alpha}}, \mathbf{H}\overline{\boldsymbol{\alpha}}\rangle - \tau(1-\tau)\langle \overline{\boldsymbol{\beta}}, \mathbf{H}\overline{\boldsymbol{\alpha}}\rangle - \frac{1}{2}\tau^2\langle \overline{\boldsymbol{\beta}}, \mathbf{H}\overline{\boldsymbol{\beta}}\rangle\,, \end{aligned}$$

over the closed interval $0 \le \tau \le 1$. The maximum is attained at the vector

$$\overline{\boldsymbol{\alpha}}_{\text{new}} = (1-\overline{\tau})\overline{\boldsymbol{\alpha}} + \overline{\tau}\overline{\boldsymbol{\beta}} \tag{37}$$

where

$$\overline{\tau} = \operatorname*{argmax}_{0\le\tau\le 1} Q_L(\tau)\,. \tag{38}$$

The derivative of $Q_L(\tau)$ reads

$$\frac{\partial Q_L(\tau)}{\partial \tau} = \langle \mathbf{b}, \overline{\boldsymbol{\beta}} - \overline{\boldsymbol{\alpha}}\rangle + (1-\tau)\langle \overline{\boldsymbol{\alpha}}, \mathbf{H}\overline{\boldsymbol{\alpha}}\rangle - (1-2\tau)\langle \overline{\boldsymbol{\beta}}, \mathbf{H}\overline{\boldsymbol{\alpha}}\rangle - \tau\langle \overline{\boldsymbol{\beta}}, \mathbf{H}\overline{\boldsymbol{\beta}}\rangle\,.$$

An objective function is improved, that is, $Q_D(\overline{\boldsymbol{\alpha}}_{\text{new}}) - Q_D(\overline{\boldsymbol{\alpha}}) > 0$, iff the vector $\overline{\boldsymbol{\beta}}$ satisfies

$$\left.\frac{\partial Q_L(\tau)}{\partial \tau}\right|_{\tau=0} = \langle \overline{\boldsymbol{\beta}} - \overline{\boldsymbol{\alpha}}, \mathbf{b} - \mathbf{H}\overline{\boldsymbol{\alpha}}\rangle > 0\,. \tag{39}$$

Provided (39) holds, the maximum of $Q_L(\tau)$ w.r.t. $0 \le \tau \le 1$ is attained within the open interval $0 < \tau < 1$ or on its boundary $\tau = 1$. The maximum of $Q_L(\tau)$ w.r.t. unbounded τ can be found by solving $\frac{\partial Q_L}{\partial \tau} = 0$ for τ. If the resulting τ exceeds 1 then the optimum of the line segment optimization is attained at $\overline{\tau} = 1$. Thus we can write the solution of (38) in a closed form

$$\overline{\tau} = \min \left\{ 1, \frac{\langle \overline{\boldsymbol{\beta}} - \overline{\boldsymbol{\alpha}}, \mathbf{b} - \mathbf{H}\overline{\boldsymbol{\alpha}} \rangle}{\langle \overline{\boldsymbol{\alpha}} - \overline{\boldsymbol{\beta}}, \mathbf{H}(\overline{\boldsymbol{\alpha}} - \overline{\boldsymbol{\beta}}) \rangle} \right\} . \tag{40}$$

Analytical formulas for improvement can be derived substituting (37) and (40) to $\Delta = Q_D(\overline{\boldsymbol{\alpha}}_{\text{new}}) - Q_D(\overline{\boldsymbol{\alpha}})$. For $\overline{\tau} < 1$ we get

$$\Delta = \frac{\langle \overline{\boldsymbol{\beta}} - \overline{\boldsymbol{\alpha}}, \mathbf{b} - \mathbf{H}\overline{\boldsymbol{\alpha}} \rangle^2}{2\langle \overline{\boldsymbol{\alpha}} - \overline{\boldsymbol{\beta}}, \mathbf{H}(\overline{\boldsymbol{\alpha}} - \overline{\boldsymbol{\beta}}) \rangle} , \tag{41}$$

and for $\overline{\tau} = 1$ we get

$$\Delta \quad = \quad \langle \overline{\boldsymbol{\beta}} - \overline{\boldsymbol{\alpha}}, \mathbf{b} - \mathbf{H}\overline{\boldsymbol{\alpha}} \rangle - \frac{1}{2}\langle \overline{\boldsymbol{\alpha}} - \overline{\boldsymbol{\beta}}, \mathbf{H}(\overline{\boldsymbol{\alpha}} - \overline{\boldsymbol{\beta}}) \rangle \ge \frac{1}{2}\langle \overline{\boldsymbol{\beta}} - \overline{\boldsymbol{\alpha}}, \mathbf{b} - \mathbf{H}\overline{\boldsymbol{\alpha}} \rangle . \tag{42}$$

The last inequality in (42) follows from $\langle \overline{\boldsymbol{\beta}} - \overline{\boldsymbol{\alpha}}, \mathbf{b} - \mathbf{H}\overline{\boldsymbol{\alpha}} \rangle \ge \langle \overline{\boldsymbol{\alpha}} - \overline{\boldsymbol{\beta}}, \mathbf{H}(\overline{\boldsymbol{\alpha}} - \overline{\boldsymbol{\beta}}) \rangle$ which holds for $\overline{\tau} = 1$ as seen from (40).

Let us consider that a new variable with index u_j was added in Step 3(b), that is, the condition (30) was satisfied. Using $\langle \overline{\mathbf{w}}, \mathbf{z}_{u_j} \rangle = [\mathbf{H}\boldsymbol{\alpha}]_{u_j}$ we can rewrite the condition (30) as

$$\frac{C}{m}[\mathbf{b} - \mathbf{H}\overline{\boldsymbol{\alpha}}]_{u_j} - \sum_{i \in I_j} \overline{\alpha}_i [\mathbf{b} - \mathbf{H}\overline{\boldsymbol{\alpha}}]_i > \frac{\varepsilon}{m} . \tag{43}$$

Let us construct the feasible vector $\overline{\boldsymbol{\beta}} = (\overline{\beta}_1, \ldots, \overline{\beta}_n)^T$ as follows

$$\overline{\beta}_i = \begin{cases} \frac{C}{m} & \text{if} \quad i = u_j , \\ 0 & \text{if} \quad i \in I_j \setminus \{u_j\} , \\ \overline{\alpha}_i & \text{if} \quad i \in I \setminus I_j . \end{cases} \tag{44}$$

As was shown above, optimization over the line segment yields an improvement provided (39) holds. Substituting $\overline{\boldsymbol{\beta}}$ constructed by (44) into the formula (39) we get

$$\frac{dQ_L}{d\tau}\Big|_{\tau=0} = \langle \overline{\boldsymbol{\beta}} - \overline{\boldsymbol{\alpha}}, \mathbf{b} - \mathbf{H}\overline{\boldsymbol{\alpha}} \rangle = \frac{C}{m}[\mathbf{b} - \mathbf{H}\overline{\boldsymbol{\alpha}}]_{u_j} - \sum_{i \in I_j} \overline{\alpha}_i [\mathbf{b} - \mathbf{H}\overline{\boldsymbol{\alpha}}]_i > \frac{\varepsilon}{m} , \tag{45}$$

where the last inequality follows from (43). This implies that optimizing w.r.t. line segment between $\overline{\boldsymbol{\alpha}}$ and the vector $\overline{\boldsymbol{\beta}}$ yields positive improvement. Now, we derive a lower bound on this improvement. Combining (45) with (42) we immediately get

$$\Delta \ge \frac{\varepsilon}{2m} \quad \text{for} \quad \overline{\tau} = 1 . \tag{46}$$

Before deriving the bound for $\bar{\tau} < 1$, we must find an upper bound on the denominator of (41). Let us define $\mathcal{A}_j = \left\{\boldsymbol{\alpha} \,|\, \sum_{i \in I_j} \alpha_i = \frac{C}{m}\,, \alpha_i \geq 0\,, \forall i \in I_j\right\}$. Then we can write

$$\begin{aligned}
\langle \overline{\boldsymbol{\alpha}} - \overline{\boldsymbol{\beta}}, \mathbf{H}(\overline{\boldsymbol{\alpha}} - \overline{\boldsymbol{\beta}}) \rangle &\leq \max_{j \in \mathcal{J}} \left\| \sum_{i \in I_j} \overline{\alpha}_i \mathbf{z}_i - \sum_{i \in I_j} \overline{\beta}_i \mathbf{z}_i \right\|^2 \\
&\leq \left(2 \max_{j \in \mathcal{J}} \max_{\boldsymbol{\alpha} \in \mathcal{A}_j} \left\| \sum_{i \in I_j} \alpha_i \mathbf{z}_i \right\| \right)^2 \qquad (47) \\
&= \frac{4C^2}{m^2} \max_{j \in \mathcal{J}} \max_{i \in I_j} \|\mathbf{z}_i\|^2 \,.
\end{aligned}$$

Combining (45) and (47) with (41) yields

$$\Delta \geq \frac{\varepsilon^2}{8C^2R^2} \quad \text{for} \quad \bar{\tau} = 1 \quad \text{and} \quad R = \max_{j \in \mathcal{J}} \max_{i \in I_j} \|\mathbf{z}_i\| \,. \qquad (48)$$

Taking the minimum of improvements (46) and (48) gives the bound on the minimal improvement. ■

References

Y. Altun and T. Hofmann. Large margin methods for label sequence learning. In *European Conference on Speech Communication and Technology (EuroSpeech)*, 2003.

Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden markov support vector machines. In *International Conference on Machine Learning*. ACM Press, New York, 2003.

D. Anguelov, B. Taskar, V. Chatabashev, D. Koller, D. Gupta, G. Heitz, and A. Ng. Discriminative learning of markov random fields for segmentation of 3D scan data. In *International Conference on Computer Vision and Pattern Recognition*, pages 167–176. IEEE Computer Society, Washington, DC, 2005.

J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society*, 48: 259–302, 1986.

Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, Sept. 2004.

Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, Nov. 2001.

C. Chekuri, S. Khanna, J. Naor, and L. Zosin. Approximation algorithms for the metric labeling problem via a new linear programming formulation. In *Symposium on Discrete Algorithms*, pages 109–118, 2001.

P.B. Chou and C.M. Brown. The theory and practice of bayesian image labeling. *International Journal on Computer Vision*, 4(3):185–210, June 1990.

M. Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Conference on Empirical Methods in Natural Language Processing*, pages 1–8, 2002.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, Dec. 2001.

T. Finley and T. Joachims. Supervised clustering with support vector machines. In *International Conference on Machine Learning*, pages 217–224. ACM Press, New York, 2005.

R. M. Haralick and L. G. Shapiro. The consistent labeling problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):173–184, 1979.

G. E. Hinton and T. J. Sejnowski. Learning and relearning in boltzmann machines. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 282–317. MIT Press, Cambridge, 1986.

M. Jerrum and A. Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM Journal of Computing*, 22:1087–1116, 1993.

V. Kolmogorov. Convergent tree-reweighted message passing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1568–1583, 2006.

V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? In *European Conference on Computer Vision*, pages 65–81. Springer-Verlag, 2002.

A. Koster, C. P. M. Hoesel, and A. W. J. Kolen. The partial constraint satisfaction problem: Facets and lifting theorems. *Operations Research Letters*, 23(3–5):89–97, 1998.

V.A. Koval and M.I. Schlesinger. Dvumernoe programmirovanie v zadachakh analiza izobrazheniy (two-dimensional programming in image analysis problems). *USSR Academy of Science, Automatics and Telemechanics*, 8:149–168, 1976. In Russian.

I. Kovtun. *Segmentaciya Zobrazhen na Usnovi Dostatnikh Umov Optimalnosti v NP-povnikh Klasakh Zadach Strukturnoi Rozmitki (Image Segmentation Based on Sufficient Conditions Of Optimality in NP-complete Classes of Structural Labeling Problems)*. PhD thesis, IRTC ITS Nat. Academy of Science Ukraine, Kiev, 2004. In Ukrainian.

A. Novikoff. On convergence proofs of perceptrons. In *Symposium on Mathematical Theory of Automata*, volume 12, pages 615–622. Polytechnic Institute of Brooklyn, 1962.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA, USA, 1988.

N.D. Ratliff and J.A. Bagnell. Subgradient methods for maximum margin structured learning. In *ICML Workshop on Learning in Structured Output Spaces*, 2006.

A. Rosenfeld, R. A. Hummel, and S. W. Zucker. Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man, and Cybernetics*, 6(6):420–433, June 1976.

D. Schlesinger. *Structurelle Ansätze für die Stereoreconstruction*. PhD thesis, Technische Universität Dresden, Fakultät Informatik, Institut für Künstliche Intelligenz, July 2005. In German.

M.I. Schlesinger. Sintaksicheskiy analiz dvumernykh zritelnikh signalov v usloviyakh pomekh (Syntactic analysis of two-dimensional visual signals in noisy conditions). *Kibernetika*, 4:113–130, 1976. In Russian.

M.I. Schlesinger and B. Flach. Some solvable sublcasses of structural recognition problems. In *Czech Pattern Recognition Workshop*. Czech Pattern Recognition Society, 2000.

M.I. Schlesinger and V. Hlaváč. *Ten Lectures on Statistical and Structural Pattern Recognition*. Kluwer Academic Publishers, 2002.

B. Schölkopf, C. Burges, and V. Vapnik. Extracting support data for a given task. In U.M. Fayyad and R. Uthurusamy, editors, *International Conference on Knowledge Discovery & Data Mining*. AAAI Press, Menlo Park, 1995.

B. Taskar, V. Chatalbashev, and D. Koller. Learning associative markov networks. In *International Conference on Machine Learning*. ACM Press, New York, 2004a.

B. Taskar, C. Guestrin, and D. Koller. Maximum-margin markov networks. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2004b.

B. Taskar, S. Lacoste-Jullien, and M.I. Jordan. Structured prediction, dual extragradient and bregman projections. *Journal of Machine Learning Research*, 7:1627–1653, Jul. 2006.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, Sep. 2005.

V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, Inc., 1998.

M. Wainwright, T. Jaakkola, and A. Willsky. MAP estimation via agreement on hypertrees: message passing and linear programming approaches. In *Conference on Communication, Control and Computing*, 2002.

T. Werner. A linear programming approach to max-sum problem: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1165–1179, July 2007.

T. Werner. A linear programming approach to max-sum problem: A review. Research Report CTU–CMP–2005–25, Center for Machine Perception, K13133 FEE Czech Technical University, Prague, Czech Republic, December 2005.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.