A Framework for Classifier Adaptation and its Applications in Concept Detection

Jun Yang
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
juny@cs.cmu.edu

Alexander G. Hauptmann
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
alex@cs.cmu.edu

ABSTRACT

There is often a need to adapt supervised classifiers such as semantic concept detectors across different domains of data. This paper describes a generic framework for *function-level classifier adaptation* based on regularized loss minimization. It directly modifies the decision function of an existing classifier of any type into a classifier for a new domain, based on limited labeled data in the new domain and no "old data", which makes it an efficient and flexible framework. We then extend this framework to adapt multiple classifiers into one classifier, with the weights of existing classifiers learned automatically to reflect their utility. We elaborate on two concrete adaptation algorithms derived from the framework, namely *adaptive SVM* and *multi-adaptive SVM*, for one-to-one and many-to-one adaptation respectively. In the experiments of adapting semantic concept detectors across video channels/types, our adaptation approach is proven to be superior to using original (unadapted) classifiers or building new ones in terms of accuracy and labeling effort.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing

General Terms

Algorithms, Experimentation, Performance

Keywords

Classifier Adaptation, Semantic Content Detection, Adaptive SVM, Multi-adaptive SVM

1. INTRODUCTION

It is known that the performance of supervised classifiers suffers when training and test data follow different distributions. Nevertheless, we hope classifiers generalize well beyond their training data to avoid the cost of building new classifiers. Such dilemma is frequently seen in the area of multimedia, as images/videos come from a variety of domains with distinctive characteristics. Here a *domain* refers to data of a certain type, from a certain source, or generated in a certain period of time, etc. A typical example is seen in semantic concept detection, where binary classifiers are built to determine whether an image or video shot is relevant to a given semantic concept. When such concept detectors are built from one domain (e.g., news video) and applied to another (e.g., documentary), the performance is usually very low.

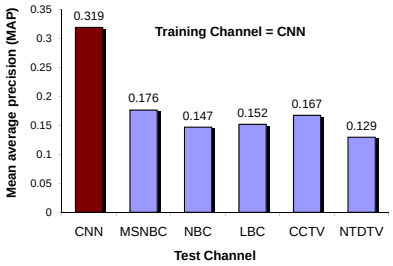


Figure 1: Average performance of 39 CNN-based concept detectors applied to 6 channels

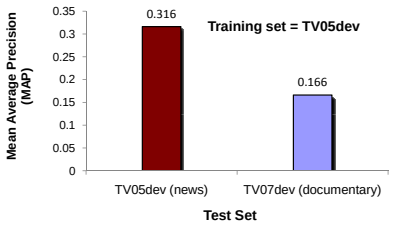


Figure 2: Average performance of 36 TV05dev-based concept detectors on TV05dev and TV07dev

To measure the impact of domain change on semantic concept detection, we perform this experiment on the development set of TRECVID 2005 data (TV05Dev), which contains news video from 6 channels, and the development set of TRECVID 2007 data (TV07Dev), which contains documentary video. We build classifiers for 39 concepts from CNN data in TV05Dev and evaluate their performance on data from each of the 6 channels (including CNN) in TV05Dev. Figure 1 shows that the performance drops as much as 50% when the training and test data are from different channels, which is surprising given that all data are news video. We repeat this experiment between the entire TV05Dev and TV07Dev collection, and observe the same trend as shown

MIR'08, October 30–31, 2008, Vancouver, British Columbia, Canada.

in Figure 2. A survey of semantic concept detection performance in such cross-domain settings is presented in [17].

While existing classifiers do not generalize, building new ones for every domain can be impractical given the cost for labeling training data. From TRECVID 2003 to 2006, every year news video of 67 to 176 hours from different sources were labeled w.r.t 10 to 39 concepts as the training data for concept detection. Based the "rule of thumb" that a person on average needs one second to label one video shot for one concept, the labeling effort is measured at 190.7 man-hours in 2003, 260 in 2006, 807 in 2005, and 1652 in 2006. Clearly, this does not scale up to most real-world video archives.

To overcome the poor generalizability of classifiers, we choose to adapt existing classifiers to new domains based on limited labeled examples. The adapted classifiers are expected to outperform existing classifiers, and require less labeled data compared with building new classifiers to reach the same accuracy. The state-of-the-art model adaptation techniques, however, are either inefficient as they rely on the "old data" for training [3, 7, 8, 16, 19], or restricted to certain classification algorithms (e.g., [13]) or specific applications ([2, 5, 15]). They are seriously limited in multimedia area where the data size is typically large and the classification algorithms are diversified. Many techniques also require the knowledge of data distribution change between domains, while modeling the distribution of image/video features is more difficult than classifying them. Thus, there is a need for a more efficient, flexible, and broadly applicable approach for adaptation problems in multimedia.

In this paper, we propose a generic framework for *function-level classifier adaptation* based on regularized loss minimization. Different from existing techniques, it directly modifies the decision function of a *source classifier* (of any type) trained from a source domain into a *target classifier* for a target domain. The adaptation requires only limited labeled examples from the target domain, and no raw data from the source domain, making this framework very efficient and flexible. We then extend this framework to handle multi-classifier adaptation, namely adapting multiple source classifiers into a target classifier with the weights of source classifier learned automatically to reflect their utility. We describe two concrete adaptation algorithms derived from the proposed frameworks, namely *adaptive SVM* (a-SVM) and *multi-adaptive SVM* (ma-SVM), for single-classifier and multi-classifier adaptation. Finally, we apply our approaches to adapt semantic concept detectors across different data channels and/or types, and the results show that adaptation is superior to either using the source classifiers or building new ones in terms of accuracy and labeling effort.

This paper bears significant improvements over our earlier work [18] on classifier adaptation. First, while the focus of the earlier work is on a-SVM as a *specific* adaptation algorithm, here we put forward a *generic* framework for adaptation based on loss minimization principle. We show that a-SVM is merely a special case of this framework, and we can derive many more algorithms from this framework by plugging in various loss functions. Moreover, we further extend this framework for multi-classifier adaptation, where the weights of source classifiers are learned automatically to reflect their utility, and derive ma-SVM as a novel algorithm from the extended framework. This is fundamentally different from, and superior to, the many-to-one adaptation approach in our previous work [18], which was mapped to one-to-one adaptation by manually setting the weights of source classifiers.

The rest of the paper is organized as follows. Section 2 reviews the related work on model adaptation. We describe the function-level adaptation framework and a-SVM algorithm for single-classifier adaptation in Section 3, as well as their counterparts for multi-classifier adaptation in Section 4. Section 5 presents the experiment results on cross-domain semantic concept detection. The conclusions and future works are discussed in Section 6.

2. RELATED WORK

Model adaptation has been studied as transfer learning in the machine learning community and as drifting concept detection in the data mining community. We review the existing adaptation techniques of the following types.

Data-level adaptation combines the labeled "old data" in the source domains with those in the target domain in order to build a better model. This is the idea behind transfer learning method for k-nearest neighbor [16], for support vector machines by Wu and Dietterich [19], and for logistic regression by Liao et al. [8]. Some data mining methods for detecting drifting concepts in streaming data [3, 7] adopt the same approach. Training is typically expensive for these methods because the size of old data is very large. Selecting and weighting the old data is another difficult issue.

Representation-level adaptation learns effective feature representations and/or distance metrics from related tasks and use them for a new task. Thrun [16] suggested to learn both a feature representation and a distance function from the labeled data of related tasks. Moreover, Raina [13] used unlabeled images collected from various sources to learn high-level feature representations that can make image classification tasks easier in general. This approach is general and efficient because the new representation is learned once and applicable to other tasks. However, the new representations may not provide enough leverage to the new task.

Parameter-level adaptation uses the parameters of existing models as the "prior" for the parameters of the new model to be learned. Several methods in this category focus on Bayesian logistic regression with a Gaussian prior on parameters. Marx et al. [9] computed the mean and variance of the Gaussian prior from the parameters of the models for related tasks. Raina et al. [14] constructed a Gaussian prior with its covariance matrix encoding the word correlations derived from text classification tasks and applied it to similar tasks. These methods are restrictive in the sense that the new models must belong to the same type as the old ones in order to have correspondence between their parameters.

There are also specialized adaptation techniques for certain problem domains, such as adapting context-free grammar [15] and language models [2] in NLP, and adapting acoustic models in speech recognition [5]. These domain-specific techniques are not generally applicable.

In multimedia, the problem of model adaptation is in general overlooked, although there is a great need for adaptation techniques. The most relevant work has been semantic concept detection based on correlated concepts, which can be thought as adapting concept detectors between correlated concepts. For example, Amir et al. [1] concatenated the prediction scores of various correlated concepts into a long feature vector called "model vector", based on which a SVM classifier was built for each concept. Qi et al. [12] proposed

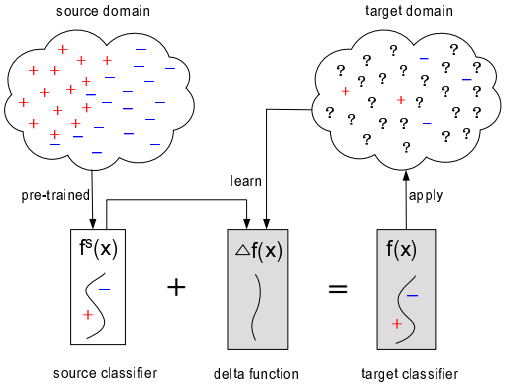


Figure 3: The framework for function-level classifier adaptation.

correlative multi-label (CML) framework to automatically identify concept correlations and learn concept detectors simultaneously. However, there is no prior work on adapting semantic concept detectors across data domains.

3. SINGLE-CLASSIFIER ADAPTATION

We propose a novel approach called *function-level classifier adaptation* which overcomes some limitations of existing methods. The idea is to directly modify a classifier's decision function, without (re-)training over the "old data" or knowing how data distributions changes. It is a generic framework in that it can adapt classifiers of any type and accommodate any loss function.

The notations are described as follows. The *target domain* has a small set of labeled data and many unlabeled data. We denote the labeled data as $\mathcal{D}^t = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where N is the number of instances, $\mathbf{x}_i$ is the feature vector of the i_{th} instance, and $y_i \in \{-1, +1\}$ is its binary label. For simplicity, we let each data vector include a constant 1 as its first element such that $\mathbf{x}_i \in \mathbb{R}^{d+1}$, where d is the number of features. The *source domain* contains a large set of labeled data $\mathcal{D}^s$, whose distribution is related to but different from the distribution of $\mathcal{D}^t$ in an unknown way. A *source classifier* has been trained from the source data $\mathcal{D}^a$ using *any* classification algorithm, denoted by its decision function $f^s(\mathbf{x})$. (In this paper, we treat a classifier and its decision function interchangebly.) Our goal is to adapt $f^s(\mathbf{x})$ into a *target classifier* $f(\mathbf{x})$ based on the labeled examples $\mathcal{D}^t$ such that $f(\mathbf{x})$ works well on the target domain.

3.1 The Basic Adaptation Framework

The function-level adaptation framework is based on regularized loss minimization principle. As illustrated in Figure 3, the target classifier $f(\mathbf{x})$ is defined as the sum of the source classifier $f^s(\mathbf{x})$ and a delta function $\Delta f(\mathbf{x})$:

$$f(\mathbf{x}) = f^s(\mathbf{x}) + \Delta f(\mathbf{x}) \tag{1}$$

This means that $f^s(\mathbf{x})$ is adapted to $f(\mathbf{x})$ simply by adding $\Delta f(\mathbf{x})$ to it. The delta function $\Delta f(\mathbf{x})$ is learned from the labeled examples $\mathcal{D}^t = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ in the target domain and also the source classifier $f^s(\mathbf{x})$, by minimizing the following regularized empirical risk:

$$\min_{\Delta f} \sum_{i=1}^{N} L(y_i, f(\mathbf{x}_i)) + \lambda \Omega(\|\Delta f\|_{\mathcal{H}}) \tag{2}$$

where $L(\cdot)$ is a loss function, $\Omega(\cdot)$ is some monotonically increasing regularization function, $\|\cdot\|_{\mathcal{H}}$ is the norm of a function in a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ as a function space, and λ is a scalar.

In Eq.(2), the first term measures the classification error (loss) of the target classifier $f(\mathbf{x})$ on the training examples D^t. The second term is a regularizer that controls the complexity of the hypothesis space. Because $\|\Delta f\|_{\mathcal{H}} = \|f - f^s\|_{\mathcal{H}}$, this regularizer is equal to the distance between the source and target classifier in the function space. Hence, the target classifier $f(\mathbf{x})$ learned under this framework must achieve two goals simultaneously, namely 1) *minimal classification error on the training examples,* and 2) *minimal deviation from the source classifier* $f^s(\cdot)$. The two goals are balanced by λ.

The goal of minimal deviation from $f^s(\cdot)$ is crucial. If minimal classification error is the only goal, one may find a large or infinite number of classifiers achieving the same classification error (even zero error when the training size is small), although many of them do not generalize well beyond the training examples. The regularizer serves as the second criterion for choosing candidate classifiers by penalizing those far away from the source classifier $f^s(\cdot)$. In other words, this adaptation framework embraces *"minimum necessary changes"* principle, i.e., it makes *minimum* changes to the source classifier that are *necessary* to correctly classify the labeled examples. The source classifier can be conceived as the *"prior model"* of the target classifier.

From bias-variance perspective, this framework reduces the high variance due to the limited training examples by relying on the source classifier trained from sufficient out-of-domain data. It represents a middle way between two extremes, namely using a unbiased, high-variance classifier trained only from limited examples, or the low-variance but biased source classifier. We expect the adapted classifier to achieve better bias-variance tradeoff.

This function-level adaptation framework offers great flexibility and efficiency. Because it directly manipulates the source classifier as an *abstraction* of the source domain, it is more efficient than existing methods that train models over (a large number of) raw data in source domains [4, 8, 19]. This also makes it applicable even when the raw data are not accessible due to copyright or privacy issue, such as surveillance video. Last but not the least, this framework treats source classifiers as functions $f^s(\cdot)$, so it can adapt classifiers of *any type* (SVM, logistic regression, etc) as long as they are represented as decision functions that can be evaluated on the training examples. In other words, it adapts a classifier as if it is a "black box".

From this generic framework, one can derive concrete algorithms for classifier adaptation by choosing certain loss functions $L(\cdot)$, regularization functions $\Omega(\cdot)$, and the form of the delta function $\Delta f(\cdot)$. While the choices are virtually infinite, we focus on a specific algorithm called *adaptive SVM*, which is derived by using the loss function of SVM.

3.2 Adaptive SVM (a-SVM)

In adaptive SVM or a-SVM, the delta function takes either a linear form $\Delta f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ or a non-linear form $\Delta f(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x})$, which is essentially a linear function in a transformed feature space with $\phi(\cdot)$ as the feature map. We use the non-liner form in this paper, of which the linear form is a special case. Δf is fully specified by its parameters $\mathbf{w}$.

We adopt the hinge loss function of SVM as $L(y, f(\mathbf{x})) = \max(1-yf(\mathbf{x}), 0)$, and a trivial regularization function $\Omega(x) = x$, which leads to $\Omega(\|\Delta f\|_{\mathcal{H}}) = \|\Delta f\|_{\mathcal{H}} = \|\mathbf{w}\|^2$. The objective function of a-SVM is given by plugging this loss function and regularizer into Eq.(2):

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \max(1 - y_i f(\mathbf{x}_i), 0) \quad (3)$$

This is equivalent to the following function:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i \quad (4)$$
$$\text{s.t. } \xi_i \geq 0,\ y_i f^s(\mathbf{x}_i) + y_i \mathbf{w}^T \phi(\mathbf{x}_i) \geq 1 - \xi_i\ \forall (\mathbf{x}_i, y_i) \in \mathcal{D}^t$$

While this objective function has the same form as that of SVM (see Eq.(12.8) in [6]), there is a fundamental difference: here $\mathbf{w}$ is the parameter of $\Delta f(\mathbf{x})$, not $f(\mathbf{x})$. Also, we will show that the regularizer $\|\mathbf{w}\|^2 = \|f - f^s\|_{\mathcal{H}}$ is equal to the distance between the source and target classifier in the function space, instead of the margin in SVM. Since $\sum_i \xi_i$ measures the classification error, Eq.(4) seeks a new classification boundary *around* the source classifier that can correctly classify the labeled examples in $\mathcal{D}^t$. The cost factor C balances the two goals, with smaller C indicating larger influence of the source classifier, and vice versa.

By integrating the constraints in Eq.(4) using Lagrange multipliers, we can rewrite the objective function as the following (primal) Lagrangian function:

$$\begin{aligned} L_P \quad = \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \mu_i \xi_i \quad (5) \\ & - \sum_{i=1}^{N} \alpha_i (y_i f^s(\mathbf{x}_i) + y_i \mathbf{w}^T \phi(\mathbf{x}_i) - (1 - \xi_i)) \end{aligned}$$

where $\alpha_i \geq 0, \mu_i \geq 0$ are Lagrange multipliers. We minimize L_P by setting its derivative with respect to $\mathbf{w}$ and ξ to zero, which results in:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \phi(\mathbf{x}_i), \qquad \alpha_i = C - \mu_i, \quad \forall i \quad (6)$$

From Eq.(6) we see that $\Delta f(\cdot) = \sum_{i=1}^{N} \alpha_i y_i K(\cdot, \mathbf{x}_i)$ as a function in RKHS, where $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$ is called kernel function. Given the definition of inner product in RKHS, we can prove the regularizer $\|\mathbf{w}\|^2$ is equal to the distance between the target classifier $f(\mathbf{x})$ and the source classifier $f^s(\mathbf{x})$ in RKHS.

$$\|f - f^s\|^2 = \|\Delta f\|^2 = \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{w}\|^2 \quad (7)$$

Substituting Eq.(6) into Eq.(5), we get the Lagrange dual objective function:

$$L_D = \sum_{i=1}^{N} (1 - y_i f^s(\mathbf{x}_i))\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \quad (8)$$

The model parameters $\alpha = \{\alpha_i\}_{i=1}^{N}$ can be estimated by maximizing L_D under the constraint $0 \leq \alpha_i \leq C, \forall i$. This would give a solution equivalent to that obtained by minimizing the primal function L_P. Maximizing L_D over α is a quadratic programming (QP) problem solved using a variation of the standard Sequential Minimal Optimization (SMO) algorithm for SVM, which is described in details in our previous work [18].

Given the solutions $\hat{\alpha}$, the target classifier is written as:

$$f(\mathbf{x}) = f^s(\mathbf{x}) + \sum_{i=1}^{N} \hat{\alpha}_i y_i K(\mathbf{x}, \mathbf{x}_i) \quad (9)$$

where $(\mathbf{x}_i, y_i) \in \mathcal{D}^t$. Note that the second term, which is the delta function as $\Delta f(\mathbf{x}) = \sum_i \hat{\alpha}_i y_i K(\mathbf{x}, \mathbf{x}_i)$, has the form of a SVM model. Thus, the target classifier $f(\mathbf{x})$ can be seen as the source classifier $f^s(\mathbf{x})$ augmented with support vectors from the labeled examples of the target domain.

4. MULTI-CLASSIFIER ADAPTATION

We have described a framework and an algorithm for adapting a single classifier. In practice, there are often multiple existing classifiers that are helpful to the classification task in the target domain. To take advantage of them, we extend our framework to perform multi-classifier adaptation, or adapting multiple source classifiers into one target classifier. It is fundamentally different from our previous many-to-one adaptation method [18] in that the weights of the source classifiers are learned automatically. From this framework we derive *multi-adaptive SVM* as an algorithm for multi-classifier adaptation.

4.1 The Extended Adaptation Framework

Suppose there are M source domains and there is a source classifier trained from each domain, denoted as $\{f_k^s\}_{k=1}^{M}$. We combine these source classifiers by a weighted sum as $\sum_k t_k f_k^s(\mathbf{x})$, where $\mathbf{t} = \{t_k\}_{k=1}^{M}$ are the weights. We treat this combination as a single classifier to be adapted to the target classifier $f(\mathbf{x})$:

$$f(\mathbf{x}) = \sum_{k=1}^{M} t_k f_k^s(\mathbf{x}) + \Delta f(\mathbf{x}) \quad (10)$$

The weights $\{t_k\}_{k=1}^{M}$ of source classifiers can be set manually based on their utility to the target domain, as we did in our earlier work [18]. In this case, the combination $\sum_k t_k f_k^s(\mathbf{x})$ is *fixed* and this becomes a single-classifier adaptation problem. However, manually defining the weights is rarely practical or desirable given the difficulty of knowing a classifier's utility to the (unlabeled) target domain.

To overcome this problem, we extend our framework to allow the weights $\{t_k\}_{k=1}^{M}$ to be learned *automatically* and *simultaneously* with the target classifier $f(\mathbf{x})$ in one learning process. This is realized by adding another regularizer $\Psi(\|\mathbf{t}\|)$ to the regularized loss expression:

$$\min_{\Delta f, \mathbf{t}} \sum_{i=1}^{N} L(y_i, f(\mathbf{x}_i)) + \lambda\Omega(\|\Delta f\|_{\mathcal{H}}) + \beta\Psi(\|\mathbf{t}\|^2) \quad (11)$$

where $\Psi(\cdot)$ is a monotonically increasing regularization function, $\|\mathbf{t}\|^2$ is the L-2 norm of the weights, and β is a scalar.

This new regularizer $\Psi(\|\mathbf{t}\|^2)$ penalizes large weights on source classifiers, so it seeks to minimize the overall contribution of source classifiers. While this appears to be counter-intuitive, it can be understood from the structure of the target classifier $f(\mathbf{x})$. Since $f(\mathbf{x})$ is the sum of the delta function $\Delta f(\cdot)$ and the source classifiers $\{f_k^s(\cdot)\}_k$, there is a competition between the two terms. The two regularizers

balance the two terms by penalizing their cost, with the old regularizer $\Omega(\|\Delta f\|_{\mathcal{H}})$ preventing over-complex $\Delta f(\cdot)$, and the new regularizer $\Psi(\|\mathbf{t}\|^2)$ preventing too much reliance on the source classifiers, both of which tend to be over-fitting.

4.2 Multi-Adaptive SVM (ma-SVM)

From the above framework, we derive *multi-adaptive SVM* or *ma-SVM* as the counterpart of a-SVM for multi-classifier adaptation. This is done by plugging SVM's hinge loss function and trivial regularization functions (i.e., $\Omega(x) = x$ and $\Psi(x) = x$) into Eq.(11). It is easy to show the objective function is equivalent to:

$$\min_{\mathbf{w},\mathbf{t}} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{2}B\|\mathbf{t}\|^2 + C\sum_{i=1}^{N}\xi_i$$
$$\text{s.t. } \xi_i \geq 0, \quad y_i \sum_{k=1}^{M} t_k f_t^s(\mathbf{x}_i) + y_i \mathbf{w}^T \phi(\mathbf{x}_i) \geq 1 - \xi_i$$

By integrating the constraints as Lagrange multipliers, we can rewrite this objective function as a minimization problem of the following Lagrange (primal) function:

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{2}B\|\mathbf{t}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}u_i\xi_i - \sum_{i=1}^{N}\alpha_i(y_i\sum_k t_k f_k^s(\mathbf{x}_i) + y_i\mathbf{w}^T\phi(\mathbf{x}_i) - (1-\xi_i)) \quad (12)$$

where $\alpha_i > 0$ and $u_i > 0$ are Lagrange multipliers. We set the derivative of L_P against $\mathbf{w}$, $\mathbf{t}$, and ξ to zero, which gives:

$$\mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i \phi(\mathbf{x}_i), \quad t_k = \frac{1}{B}\sum_{i=1}^{N}\alpha_i y_i f_k^s(\mathbf{x}_i), \quad \alpha_i = C - \mu_i \quad (13)$$

The expression of t_k shows a connection between the weight of a source classifier f_k^s and its performance on the target domain. t_k is a weighted sum of terms $y_i f_k^s(\mathbf{x}_i)$, which is a "margin" indicating how well f_k^s classifies training example $\mathbf{x}_i$. The margin is larger if f_k^s correctly predicts the label of $\mathbf{x}_i$, and vice versa. Thus, source classifiers that classify the labeled examples better are assigned a larger weight, and vice versa. This intuitive weighting comes naturally from the loss minimization framework. This justifies the introduction of regularizer $\|\mathbf{t}\|^2$ in the objective function.

By plugging Eq.(13) into the primal Lagrangian Eq.(12), we obtain the dual Lagrange function as:

$$L_D = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \left(K(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{B}\sum_k f_k^s(\mathbf{x}_i) f_k^s(\mathbf{x}_j) \right) \quad (14)$$

The parameters $\alpha = \{\alpha_i\}_{i=1}^N$ are estimated by maximizing L_D using an variation of the standard SMO algorithm. The target classifier can be expressed using the estimated $\hat{\alpha}$ as:

$$f(\mathbf{x}) = \sum_{i=1}^{N}\hat{\alpha}_i y_i \left(K(\mathbf{x}_i, \mathbf{x}) + \frac{1}{B}\sum_{k=1}^{M} f_k^s(\mathbf{x}_i) f_k^s(\mathbf{x}) \right) \quad (15)$$

From Eq.(15), we can interpret ma-SVM as a standard SVM that treats *the outputs of source classifiers as additional features*. Suppose the output of source classifiers on $\mathbf{x}$ is an extra feature vector $\mathbf{f} = [f_1^a(\mathbf{x}), ..., f_M^a(\mathbf{x})]$. Similarly, $\mathbf{f}_i = [f_1^a(\mathbf{x}_i), ..., f_M^a(\mathbf{x}_i)]$ is the extra feature for $\mathbf{x}_i$. We can rewrite the target classifier as $f(\mathbf{x}) = \sum_i \alpha_i y_i (K(\mathbf{x}_i, \mathbf{x}) + \frac{1}{B}\mathbf{f}_i \cdot \mathbf{f})$. While $K(\mathbf{x}_i, \mathbf{x})$ is the similarity between $\mathbf{x}_i$ and $\mathbf{x}$ in the (transformed) feature space, and $\mathbf{f}_i \cdot \mathbf{f}$ measures their similarity in terms of the output of source classifiers. If the classifiers' outputs on $\mathbf{x}_i$ and $\mathbf{x}$ are close, $\mathbf{f}_i \cdot \mathbf{f}$ is large and their similarity is high, and vice versa. Compared with a SVM model $f(\mathbf{x}) = \sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x})$, ma-SVM extends similarity measure between $\mathbf{x}$ and $\mathbf{x}_i$ to include the similarity in the classifier-output space. The idea of treating model (classifier) outputs as additional features is not new. For example, the "model-vector" approach by Natsev et al. [11] constructed the semantic feature of a video shot from the scores of semantic concept detectors and used it in retrieval. Our analysis provides a formal interpretation of this ad-hoc technique: using classifier outputs as features is equivalent to adapting these classifiers under regularized loss minimization framework.

5. EXPERIMENTS IN CROSS-DOMAIN CONCEPT DETECTION

5.1 Data Collections

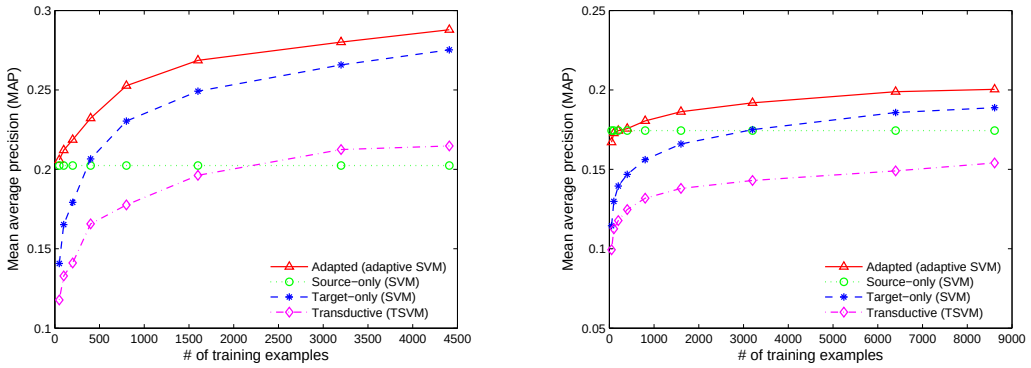
We use the development set of the TRECVID benchmark video collection in 2005 and 2007, referred to as TV05Dev and TV07Dev. The TV05Dev collection contains 86 hours of broadcast news video from 6 TV channels, including CNN, NBC, MSNBC, CCTV (Mandarin), NTDTV (Mandarin), and LBC (Arabic). Due to the difference on editing styles, target audience, and other factors, the data from different channels exhibit different characteristics. The 86-hour video has been manually partitioned into 61,901 video shots, which are relatively evenly distributed among the channels. The TV07Dev collection has 50 hours of documentary video, partitioned into 21,532 shots. In both collections, each video shot is represented by a single video frame as its "keyframe". Each keyframe is depicted by a 273-d feature vector, which consists of a 225-d color moment feature computed from 5×5 grids and a 48-d Gabor texture feature. Since there is not much change of content within a shot, we use this 273-d keyframe feature as the feature of the whole shot.

Labels of 39 semantic concepts are provided on the entire TV05Dev collection as part of the LSCOM-Lite project [10]. The TV07Dev collection are annotated w.r.t the same set of concepts expect 3 of them (thus totally 36 concepts). These concepts cover a wide variety of types, including objects (e.g., *Car*), visual scenes (e.g., *Sky*), semantic topics (e.g., *Military*), human activities (e.g., *Meeting*), etc.

5.2 One-to-one Adaptation

We first evaluate the performance of single-classifier adaptation based on adaptive SVM (a-SVM). Among the 6 channels in TV05Dev, we choose NBC as the source domain and CNN as the target domain. The NBC data (9,322 shots) are fully labeled w.r.t the 39 concepts. The CNN data (11,025 shots) are split along the timeline into a *training portion* containing 40% of the shots (4,410 shots) and a *test portion* containing all the remaining shots.

For each concept, we train the source classifier using SVM from all the NBC data. Then we *randomly* label a certain number of shots in the training portion of CNN, and



(a) CNN to NBC

(b) TV05Dev (news) to TV07Dev (documentary)

Figure 4: Performance of adapted classifiers and source-only, target-only, and tranducstive classifiers on LSCOM-lite concepts: (a) NBC to CNN setting, (b) TV05Dev to TV07Dev setting.

use them to adapt the source classifier to a target classifier based on a-SVM. The target classifier is evaluated on the test portion of the CNN data using average precision (AP) as performance metric. To reduce randomness, we repeat the experiment on 4 random labeled samples of the same size, and compute the average performance. We gradually increase the number of labeled shots available for adaptation and repeat the same experiment, until all the 4,410 shots in the training portion are used. Since the average ratio of positive data of these concepts is 7%, the positive examples are quite scarce and insufficient for training reliable classifiers.

For comparison purpose, we also evaluate three alternative methods for classifying data in target domain (CNN). To ensure comparability, all the classifiers are trained using SVM with RBF kernel with $\gamma = 0.1$ and cost factor $C = 1$, which has reasonable performance in cross-validation.

- **Source-only:** This approach applies the original source classifier to the target domain without any adaptation. It uses no labeled data from the target domain.
- **Target-only:** This approach builds a new classifier from scratch using only the labeled examples in the target domain. It uses neither the source classifier nor data in the source domain.
- **Transductive:** The limited labeled data and abundant unlabeled data in the target domain makes semi-supervised learning (SSL) an appealing approach. We use transductive SVMs (TSVM), a widely used SSL method, to build classifiers based on *all* the video shots (labeled and unlabeled) in the target domain. It uses no source classifier or data in the source domain.

Figure 4(a) compares the average performance of 39 concepts in terms of mean average precision (MAP) between a-SVM and the other three methods. On average, the `adapted` classifiers trained by a-SVM outperform other methods by a large margin. When the training examples are scarce, the new classifiers trained from CNN (`target-only`) have very low performance due to the lack of training data, while the the source classifiers (`source-only`) have more decent performance. In this stage, the adapted classifiers perform slightly better than the source classifiers, and avoid the "cold start" problem with little training data. As more labeled examples become available, the performance of adapted classifiers and of target-only classifiers climbs together, with the adapted classifiers keeping their lead even when the training size reaches its maximum. From another perspective, adapted classifiers need much less training data to reach the same performance than building new classifiers from scratch. For example, target-only classifiers need 400 labeled examples to achieve the same MAP of adapted classifiers at only 50 examples. The `transductive` approach has the lowest performance, showing that unlabeled data hurt the performance. This implies that in this problem data points close to each other in the feature space may not share the same label, which violate the assumption of semi-supervised learning.

We repeat the experiment using the entire TV05Dev as the source domain and TV07Dev as the target domain (with the same training/test split). The results are shown in Figure 4. The relationships between these methods remain the same as in the NBC-to-CNN setting, except that the source classifiers provide more leverage to the target domain.

Besides the average performance, it is also important to see whether our adaptation method excel *consistently* on every single concept. In Figure 5, we show the performance of the `source-only`, `target-only`, and `adapted` classifier on a per-concept basis in NBC-to-CNN setting. The adapted classifier is the best performer for most concepts, and a close second for the remaining concepts. Also note that for some concepts (e.g., *Person*, *Sky*), the source classifier trained from NBC performs so well on CNN that adaptation can hardly improve its performance. This issue is further discussed in Section 6.

5.3 Many-to-one adaptation

We then evaluate the performance of our multi-classifier adaptation method. In this experiment, we treat CNN as the target domain and all the other 5 channels in TV05Dev as source domains. For each concept, we use multi-adaptive SVM (ma-SVM) to adapt the 5 classifiers (of this concept) trained separately from these 5 fully-labeled channels into a new classifier for CNN. As before, the adapted classifier is evaluated against the test portion of CNN.

Figure 6 compares the MAP of 39 concepts between a multi-classifier adaptation run, which uses all the 5 channels

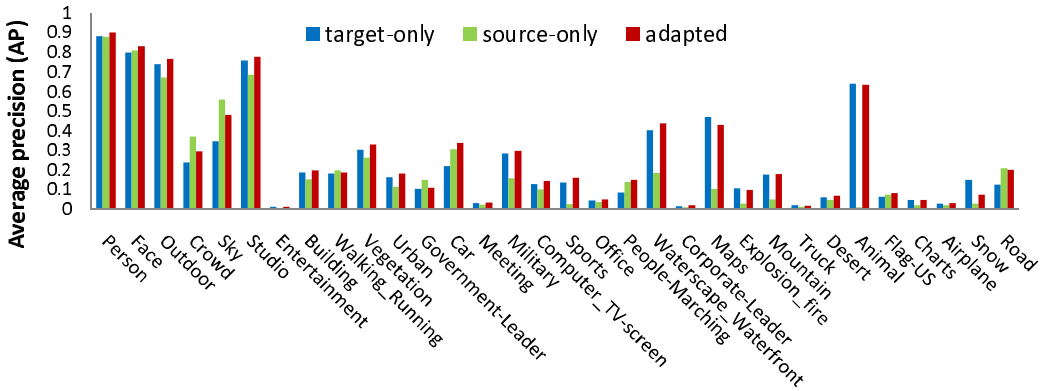


Figure 5: Per-concept performance comparison between source-only, target-only, and adapted classifiers in NBC-to-CNN setting, with 1,600 labeled examples per concept

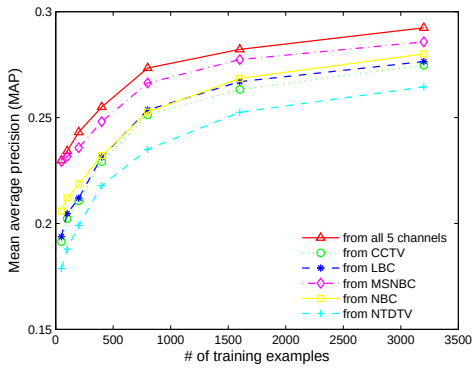


Figure 6: Performance comparison between multi-classifier adaptation (ma-SVM) and single-classifier adaptation (a-SVM)

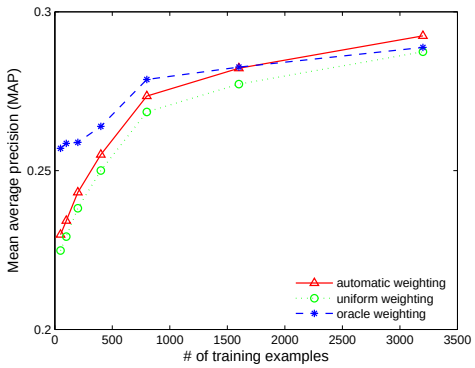


Figure 7: Performance comparison between different weighting schemes in multi-classifier adaptation

as source domains, and 5 single-classifier adaptation runs, each using a specific channel as source domain. The benefit of using multiple source domains (classifiers) is clear, as it outperforms even the best run that uses a single source domain (classifier). We also notice that the performance gap between different single-classifier adaptation runs is quite large, which means that different channels are not equally useful. Thus, it is important in multi-classifier adaptation that classifiers trained from different channels are weighted in a way to reflect their utility.

In order to see whether the weights automatically learned by ma-SVM are effective, we compare it with two alternative weighting methods. The *uniform weighting* approach assigns equal weights to all source classifiers and ensure the weights sum up to 1. The *oracle weighting* approach assigns weights proportional to the actual performance of source classifiers on the target domain and ensures the weights sum up to 1. It is called "oracle" because such performance is supposed to be *unknown* without completely labeling the target domain. While this performance-based weighting is not guaranteed to be optimal, it provides a *quasi* upper bound on the performance for multi-classifier adaptation. For both uniform and oracle weighting, since the weights are fixed before adaptation, we combine the source classifiers using the assigned weights and adapt this combination as a single classifier to the new classifier using a-SVM.

Figure 7 compares these weighting schemes in multi-classifier adaptation. The automatic weighting by ma-SVM has better performance than uniform weighting, suggesting that ma-SVM is able to weight source classifiers properly. However, it is well below the oracle weighting when the training size is small. This is due to the fact that, as revealed in Section 4.2, the weights learned by ma-SVM are related to the performance of source classifiers on the labeled examples. Since the performance on limited examples may not reflect the performance on the entire target domain, the learned weights are far from optimal. When there are more training examples, the automatically learned weights are as good and even better than the oracle weights.

6. CONCLUSIONS

We have described a generic framework for adapting one or more existing classifiers to a classifier for a new domain through modifications of their decision functions. This framework is efficient and flexible as it relies only on limited labeled data in the new domain and needs no "old data". We have elaborated on two algorithms derived from this framework, namely adaptive SVM and multi-adaptive SVM, for one-to-one and many-to-one adaptation respectively. In cross-domain semantic concept detection, we have shown that our adaptation approach is superior to either using the original (unadapted) classifiers or building new classifiers from scratch in terms of accuracy and labeling effort.

While this paper has mainly focused on *how* to adapt a classifier, another important question is *whether* to adapt a classifier. This question deserves more future work. Figure 5 shows that some classifiers generalize to new domains better than the other classifiers, and the more generalizable classifiers do not need adaptation as much as the others. A classifier with very poor generalizability may not worth adaptation at all; it is better to discard it and build a new classifier from scratch. The key problem here is to determine a classifier generalizability, i.e., how do we estimate a classifier's performance on a new domain *without* labeling it? Our preliminary study on the generalizability of classifiers for semantic concept detection [17] suggested several heuristics to identify generalizable classifiers. If classifiers' generalizability is measurable, we will be able to prioritize classifiers such that the most needed ones are adapted first and/or receive more labeled examples. Adapting classifiers selectively is more cost-efficient than blindly adapting all classifiers.

Another future direction on adaptation focuses on data transofrmation/normalizaiton. Instead of adapting models to a new domain, one might as well transform the data in the new domain such that the distribution is similar to the distribution of data in the old domain. This requires knowledge on the distributions of the two domains, which is available or obtainable in some applications (e.g., speech recognition). The original models need no adaptation if the data can be transformed. A related idea is to identify features whose distribution is more consistent across different domains. Domain-invariant features will also minimize the need for adapting models.

7. ACKNOWLEDGEMENT

This work was supported in part by the National Science Foundation (NSF) under Grants No. IIS-0535056 and CNS-0751185.

8. REFERENCES

[1] A. Amir and et al. IBM research TRECVID-2003 video retrieval system. In *Worshop of TRECVID 2003*, 2003.

[2] M. Bacchiani and B. Roark. Unsupervised language model adaptation. In *IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing*, 2003.

[3] P. Cunningham, N. Nowlan, S. Delany, and M. Haahr. A case-based approach to spam filtering that can track concept drift. In *Proc. of ICCBR Workshop on Long-lived CBR Systems*, 2003.

[4] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu. Boosting for transfer learning. In *Proc. of the 24th Int'l conference on Machine learning*, pages 193–200, 2007.

[5] J. Gauvain and C. Lee. Maximum a posteriori estimation for multivariate Gaussian mixtureobservations of Markov chains. *IEEE Trans. on Speech and Audio Processing*, 2(2):291–298, 1994.

[6] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: Data mining, inference, and prediction.* Springer, 2001.

[7] R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. In *Proc. of Int'l Conf. on Machine Learning*, pages 487–494, 2000.

[8] X. Liao, Y. Xue, and L. Carin. Logistic regression with an auxiliary data source. In *Proc. of Int'l Conf. on Machine Learning*, pages 505–512, 2005.

[9] Z. Marx, M. Rosenstein, L. Kaelbling, and T. Dietterich. Transfer learning with an ensemble of background tasks. In *NIPS 2005 Workshop on Inductive Transfer: 10 Years Later*, 2005.

[10] M. R. Naphade, L. Kennedy, J. R. Kender, S. F. Chang, J. Smith, P. Over, and A. Hauptmann. A light scale concept ontology for multimedia understanding for TRECVID 2005. In *IBM Research Technical Report*, 2005.

[11] A. P. Natsev, M. R. Naphade, and J. R. Smith. Semantic representation: search and mining of multimedia content. In *Proc. of the tenth ACM SIGKDD Int'l conference on Knowledge discovery and data mining*, pages 641–646, 2004.

[12] G.-J. Qi, X.-S. Hua, Y. Rui, J. Tang, T. Mei, and H.-J. Zhang. Correlative multi-label video annotation. In *ACM Multimedia 2007*, pages 17–26, 2007.

[13] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *Proc. of the Twenty-Fourth Int'l Conf. on Machine Learning*, 2007.

[14] R. Raina, A. Y. Ng, and D. Koller. Constructing informative priors using transfer learning. In *Proc. of Int'l Conf. on Machine Learning*, pages 713–720, 2006.

[15] B. Roark and M. Bacchiani. Supervised and unsupervised pcfg adaptation to novel domains. In *Proc. of the 2003 Conf. of NAACL-HLT*, pages 126–133, 2003.

[16] S. Thrun. Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems*, volume 8, pages 640–646, 1996.

[17] J. Yang and A. Hauptmann. (Un)Reliability of Video Concept Detection. In *ACM Int'l Conf. on Image and Video Retrieval (CIVR)*, 2007.

[18] J. Yang, R. Yan, and A. Hauptmann. Cross-domain Video Concept Detection using Adaptive SVMs. In *ACM Multimedia*, 2007.

[19] P. Wu and T. G. Dietterich. Improving SVM accuracy by training on auxiliary data sources. In *Proc. of Int'l Conf. on Machine Learning*, page 110, 2004.