

Cross-Domain Video Concept Detection Using Adaptive SVMs

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

Rong Yan
IBM T.J.Watson Research
19 Skyline Drive
Hawthorne, NY 10532
yanr@us.ibm.com

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

ABSTRACT

Many multimedia applications can benefit from techniques for adapting existing classifiers to data with different distributions. One example is cross-domain video concept detection which aims to adapt concept classifiers across various video domains. In this paper, we explore two key problems for classifier adaptation: (1) how to transform existing classifier(s) into an effective classifier for a new dataset that only has a limited number of labeled examples, and (2) how to select the best existing classifier(s) for adaptation. For the first problem, we propose *Adaptive Support Vector Machines* (A-SVMs) as a general method to adapt one or more existing classifiers of any type to the new dataset. It aims to learn the “delta function” between the original and adapted classifier using an objective function similar to SVMs. For the second problem, we estimate the performance of each existing classifier on the sparsely-labeled new dataset by analyzing its score distribution and other meta features, and select the classifiers with the best estimated performance. The proposed method outperforms several baseline and competing methods in terms of classification accuracy and efficiency in cross-domain concept detection in the TRECVID corpus.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation, Performance

Keywords

Adaptive SVMs, Classifier Adaptation, Cross-domain Video Concept Detection

1. INTRODUCTION

We consider the problem of **cross-domain video concept detection**, which aims to generalize models built for

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MM’07, September 23–28, 2007, Augsburg, Bavaria, Germany.
Copyright 2007 ACM 978-1-59593-701-8/07/0009 ...\$5.00.



Figure 1: “Anchor” shots in LBC and NTDTV



Figure 2: “Weather” shots in CNN and CCTV

detecting semantic concepts in one domain of video data to other domains. Here a domain can be a video genre, a content provider, or a video program. The data distributions of different domains are likely to be different. For instance, the TRECVID collection [13] is such a multi-domain video corpus which has news video from different TV channels and in different languages. Conventional concept detection methods [11, 13, 14, 19] build concept classifiers from labeled examples in one or more domains and assume that the test data come from the *same* domain(s). In real applications, however, there is often a need to apply classifiers to data from a *different* domain. Cross-domain concept detection is challenging because, as shown in Figure 1 and 2, video shots of easily recognizable concepts such as “anchor” and “weather” exhibit dissimilar visual features when they are from two domains. As a result, concept classifiers trained from one domain may perform poorly on the other domains, and it is too costly to manually collect a large number of examples and build separate classifiers for every domain. While the mismatch between training and testing distribution is a fundamental problem in machine learning [5, 17, 18], it is particularly severe in the multimedia area due to the large data variance and the “semantic gap” between content and semantics. In this paper, we investigate general methods for adapting existing classifier(s) to data with different distributions and their application in cross-domain video concept detection.

We consider the general problem of a binary classification task with respect to a given topic in a **primary dataset** \mathcal{D}^p , where only a limited number of instances \mathcal{D}_l^p are labeled and the majority \mathcal{D}_u^p are unlabeled (thus, $\mathcal{D}^p = \mathcal{D}_l^p \cup \mathcal{D}_u^p$).

In addition, there are one or multiple **auxiliary datasets** as $\mathcal{D}_1^a, \dots, \mathcal{D}_M^a$, each of which is drawn from a distribution that might be different from the distribution of the primary dataset. The auxiliary datasets are fully labeled, and an **auxiliary classifier** f_k^a has been trained from each of these datasets \mathcal{D}_k^a . In order to classify \mathcal{D}^p , simply applying any f_k^a may not achieve good performance due to the mismatched distribution. On the other hand, using a new classifier learned only from the limited labeled examples in \mathcal{D}_l^p may suffer from the high variance problem. For better bias-variance tradeoff, one should leverage both the knowledge in the auxiliary data and the labeled primary examples to build better classifiers for the primary data.

In this paper, we explore *classifier adaptation* techniques that aim to adapt auxiliary classifier(s) to the primary dataset based on limited labeled examples in an efficient and principled manner. Two subproblems are studied:

1. Given auxiliary classifier(s) of any type, how to migrate it (them) into an effective classifier for the primary dataset using limited labeled examples?
2. How to choose the best auxiliary classifier for adaptation if there are multiple existing classifiers?

For problem 1, we propose Adaptive Support Vector Machines (A-SVMs) for adapting an auxiliary classifier to the primary dataset by learning a “delta function” between the decision functions of the auxiliary and new classifier using an objective function extended from standard SVMs. We then propose a simple extension of A-SVMs to deal with multiple auxiliary classifiers, and an efficient learning algorithm for A-SVMs based on quadratic programming. For problem 2, we base our selection of auxiliary classifiers on the estimation of their performance on the primary dataset which is basically unlabeled. Our performance estimation method exploits features from the score distribution (of the classifier) and its consistency with other classifiers.

A-SVMs has several unique properties that distinguish it from related methods on transfer learning, incremental learning, and drifting concept detection. First, it is a “black-box” method in that sense that it can handle auxiliary classifiers of *arbitrary* types, and it does not require the availability of auxiliary data. This makes it applicable to domains where the classifiers are unconventional and/or the old data are inaccessible for some reasons. Second, its learning process does not involve auxiliary data and is therefore as fast as learning a classifier from the new (primary) data. Last, instead of explicitly modeling the distribution difference, it takes a discriminative approach and measures the distribution difference by estimating a classifier’s performance on another distribution. This is because modeling distribution change is extremely hard in high-dimensional spaces and not necessarily useful to classification, for which the distribution of positive data is more critical (but unknown).

We evaluate the proposed methods to cross-domain video concept detection in the TRECVID collection [13], which consists of news video from multiple sources. In the experiments of adapting concept classifiers across data sources, the adapted classifiers trained by A-SVMs significantly outperform both auxiliary classifiers and new classifiers trained exclusively from labeled examples. We also compare A-SVMs to an *aggregate* approach that re-trains a classifier using all the (both primary and auxiliary) data, and an *ensemble* ap-

proach that combines the outputs of classifiers trained separately from the primary and auxiliary data. Experiments show that our method achieves better performance than the *ensemble* approach and comparable performance to the *aggregate* approach while using 1/10 of its training time.

The remainder of this paper is organized as follows. Section 2 reviews the related work. The proposed adaptive SVM model is described in Section 3, and the method for selecting auxiliary classifier is discussed in Section 4. Two alternative methods for adaptation are introduced in Section 5. We describe the application of video concept detection in Section 6 and present the experiment results in Section 7. The conclusions and future work are discussed in Section 8.

2. RELATED WORK

Classifier adaptation can be seen as an effort to solve the fundamental problem of mismatched distributions between the training and testing data. This problem occurs in concept detection in a video corpus such as TRECVID [13], which contains data from different sources (programs). In existing approaches [11, 14, 19], concept classifiers are built from and applied to data collected from *all* the programs without considering their difference on distribution. In this paper, we consider a different scenario where classifiers trained from one or several programs are adapted to a different program. The proposed classifier adaptation method is related to the work on drifting concept detection in the data mining community and transfer learning and incremental learning in the machine learning community.

Classifying data with mismatched distribution is similar to the problem of drifting concept detection if we treat the target class as a concept. In the data mining community, two types of approaches have been proposed to detect drifting concepts from data streams. The first type of approaches selects training instances using a fixed/adaptive window that moves over the data stream, weights them by their age or utility to the target concept, and uses them to build and update a single target model. Examples in this category are the approach proposed by Klinkenberg and Joachims [6] and by Delany et al. [2]. The second type of approaches maintains an (weighted) ensemble classifier that combines a set of base classifiers learned from different chunks of the data stream. The work of Wang et al. [16], Kolter and Maloof [7] belong to this category. The A-SVMs model is different from them as it directly adapts an existing model to the new data, and avoids the overhead of training over the existing data or maintaining multiple base models.

Transfer learning aims to apply knowledge learned from one or more tasks to improve the performance on other related tasks. Similar to our approach, some transfer learning methods leverage the knowledge of auxiliary data or other form of prior knowledge to build better models for the current data. The approaches of Liao et al. [8], Wu and Dietterich [17], and Wu and Srihari [18] incorporate the (weighted) auxiliary data into the current training data to train a more reliable SVM or logistic regression classifier. But, because their training process involves all the auxiliary data, they are more expensive than our model which directly manipulates the classifier(s) learned from auxiliary data.

Incremental learning methods, such as incremental SVMs [15, 1], continuously update a model with new examples without re-training over all the examples. When the training and test distribution is identical, A-SVMs can be treated as

a generic incremental method which can handle classifiers of any type. It is also more efficient than existing methods [15, 1] whose training involves at least part of the previous examples (e.g., support vectors).

3. ADAPTIVE SUPPORT VECTOR MACHINES

Let $\mathcal{D}^p = \mathcal{D}_l^p \cup \mathcal{D}_u^p$ denote a partially-labeled *primary dataset*, where \mathcal{D}_l^p is the labeled subset and \mathcal{D}_u^p is the unlabeled subset. The size of labeled subset \mathcal{D}_l^p is very small compared with that of the whole dataset \mathcal{D}^p . We have $\mathcal{D}_l^p = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ where \mathbf{x}_i is the i th data vector and $y_i \in \{-1, +1\}$ is its binary label. For notational simplicity, we let each data vector \mathbf{x} always 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. In addition to the primary dataset \mathcal{D}^p , there exists a fully-labeled *auxiliary dataset* $\mathcal{D}^a = \{(\mathbf{x}_i^a, y_i^a)\}_{i=1}^{N^a}$ or multiple auxiliary datasets as $\mathcal{D}_1^a, \dots, \mathcal{D}_M^a$ with $\mathcal{D}_k^a = \{(\mathbf{x}_i^k, y_i^k)\}_{i=1}^{N_k^a}$, where $\mathbf{x}_i^a, \mathbf{x}_i^k \in \mathbb{R}^{d+1}$ and $y_i^a, y_i^k \in \{-1, +1\}$. The distribution of auxiliary data is likely to be different from the primary data. We assume that an *auxiliary classifier* $f^a(\mathbf{x})$ (or $f_k^a(\mathbf{x})$) has been trained from each auxiliary dataset \mathcal{D}^a (or \mathcal{D}_k^a), which predicts the data label through the sign of its decision function, i.e., $\hat{y} = f^a(\mathbf{x})$. These auxiliary classifiers can be trained using *any* classification algorithm, such as SVMs, Naive Bayes, decision tree, etc.

Our goal is to learn a classifier $f(\mathbf{x})$ that can accurately classify the primary dataset. We first briefly review the standard SVMs which train $f(\mathbf{x})$ from the labeled examples \mathcal{D}_l^p . Inspired by SVMs, adaptive support vector machines (A-SVMs) are able to adapt an auxiliary classifier $f^a(\mathbf{x})$ to a new classifier $f(\mathbf{x})$ based on the labeled examples in \mathcal{D}_l^p . The basic A-SVMs are further extended to a more general form which can adapt a combination of multiple existing candidates $f_1^a(\mathbf{x}), \dots, f_M^a(\mathbf{x})$ to the new classifier.

3.1 Standard SVMs

In SVMs, the label of a data vector \mathbf{x} is determined by the sign of a linear decision function as $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$, where $\mathbf{w} \in \mathbb{R}^{d+1}$ are the model parameters. The non-linear classification boundaries preferred by many classification tasks are achieved through the “kernel trick”: we use a feature map ϕ to project each data vector \mathbf{x} into a feature vector $\phi(\mathbf{x})$ in a space of a higher or even infinite dimension, and rewrite f as a linear function in that projected feature space, i.e., $f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$. Linear decision boundaries in the projected feature space corresponds to non-linear boundaries in the original input space. The form of the decision boundary is determined by the kernel function $K(x, x') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$, which defines the inner product of two projected feature vectors. Our following discussion is based on kernel SVMs, since linear SVMs are special cases of the kernel model when $\phi(\mathbf{x}) = \mathbf{x}$ and $K(x, x') = \langle \mathbf{x}, \mathbf{x}' \rangle$.

Training a standard SVM classifier from $\mathcal{D}_l^p = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ involves the following optimization problem:

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

where $\sum_i \xi_i$ measures the total classification error, and $\|\mathbf{w}\|^2$ is a regularization term that is inversely related to margin

between training examples of two classes. This objective function seeks a decision boundary that achieves a small classification error and meanwhile creates a large margin, with two goals balanced by a scalar cost factor C .

3.2 One-to-one Adaptation

In this setting, we consider adapting the auxiliary classifier $f^a(\mathbf{x})$ trained from the auxiliary data \mathcal{D}^a to a new classifier $f(\mathbf{x})$ for the primary data \mathcal{D}^p . In A-SVMs, this is implemented by adding a “delta function” in the form of $\Delta f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$ on the basis of $f^a(\mathbf{x})$:

$$f(\mathbf{x}) = f^a(\mathbf{x}) + \Delta f(\mathbf{x}) = f^a(\mathbf{x}) + \mathbf{w}^T \phi(\mathbf{x}) \quad (2)$$

where \mathbf{w} are the parameters to be estimated from the labeled examples in \mathcal{D}_l^p .

To learn the parameter \mathbf{w} of delta function Δf , we propose an objective function inspired by SVMs:

$$\begin{aligned} \min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i \quad (3) \\ \text{s.t. } \xi_i \geq 0 \\ y_i f^a(\mathbf{x}_i) + y_i \mathbf{w}^T \phi(\mathbf{x}_i) \geq 1 - \xi_i, \quad \forall (\mathbf{x}_i, y_i) \in \mathcal{D}_l^p \end{aligned}$$

Similar to Eq.(1), $\sum_i \xi_i$ measures the total classification error of the adapted classifier $f(\mathbf{x})$. The regularization term $\|\mathbf{w}\|^2$, although having exactly the same form as in Eq.(1), has a different meaning because \mathbf{w} are the linear parameters of $\Delta f(\mathbf{x})$ rather than $f(\mathbf{x})$. This regularizer favors a “minimal” delta function that is close to $\Delta f(\mathbf{x}) = 0$, and consequently, a decision function $f(\mathbf{x})$ that is close to the auxiliary classifier $f^a(\mathbf{x})$. Therefore, the objective function in Eq.(3) seeks a new decision boundary that is close to the boundary of the auxiliary classifier (in the feature space) and meanwhile separates the labeled examples in \mathcal{D}_l^p well. The cost factor C in A-SVMs balances the contribution between the auxiliary classifier (through the regularizer) and the training examples. The larger C is, the smaller the influence of the auxiliary classifier is. Thus, one should use a small C for a “good” auxiliary classifier and vice versa.

The objective function in Eq.(3) can be rewritten as the following (primal) Lagrangian:

$$\begin{aligned} L_P = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \mu_i \xi_i \\ - \sum_{i=1}^N \alpha_i (y_i f^a(\mathbf{x}_i) + y_i \mathbf{w}^T \phi(\mathbf{x}_i) - (1 - \xi_i)) \quad (4) \end{aligned}$$

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:

$$\begin{aligned} \mathbf{w} &= \sum_{i=1}^N \alpha_i y_i \phi(\mathbf{x}_i) \\ \alpha_i &= C - \mu_i, \quad \forall i \quad (5) \end{aligned}$$

From the above, it is easy to show that $\Delta f(\cdot) = \sum_{i=1}^N \alpha_i y_i K(\cdot, \mathbf{x}_i)$ as a function in the Reproducing Kernel Hilbert Space (RKHS). Given the definition of inner product in RKHS, we have

$$\begin{aligned} \|f - f^a\|^2 &= \|\Delta f\|^2 = \langle \Delta f, \Delta f \rangle \\ &= \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{w}\|^2 \quad (6) \end{aligned}$$

This offers a more intuitive interpretation of the regularizer $\|\mathbf{w}\|^2$ in Eq.(3). Since $\|\mathbf{w}\|^2 = \|f - f^a\|^2$, the regularizer aims to minimize the distance between the adapted decision function $f(\mathbf{x})$ and the auxiliary one $f^a(\mathbf{x})$ in the RKHS.

In addition to Eq.(5), the Karush-Kuhn-Tucker (KKT) conditions, which the optimal solution of Eq.(4) must satisfy, also include:

$$\begin{aligned} \alpha_i \{y_i f^a(\mathbf{x}_i) + y_i \mathbf{w}^T \mathbf{x}_i - (1 - \xi_i)\} &= 0 \\ \alpha_i &\geq 0 \\ y_i f^a(\mathbf{x}_i) + y_i \mathbf{w}^T \mathbf{x}_i - (1 - \xi_i) &\geq 0 \\ \mu_i \xi_i &= 0 \\ \mu_i &\geq 0 \\ \xi_i &\geq 0 \end{aligned} \quad (7)$$

Substituting Eq.(5) into Eq.(4), we get the Lagrange dual objective function:

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

where $\lambda_i = y_i f^a(\mathbf{x}_i)$. 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 the algorithm in Section 3.4. Given the solutions $\hat{\alpha}$, the adapted decision function is written as:

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

where $(\mathbf{x}_i, y_i) \in \mathcal{D}_l^p$. The adapted classifier $f(\mathbf{x})$ can be seen as augmented from the auxiliary classifier $f^a(\mathbf{x})$ with support vectors from the labeled subset of the primary data.

The only difference between the dual form of standard SVMs [5] and the dual form of A-SVMs in Eq.(8) is that the latter contains the extra term λ . We discuss how λ affect the estimation of α . In Eq.(8), if $\lambda_i = y_i f^a(\mathbf{x}_i) < 0$, which indicates the auxiliary classifier f^a misclassifies \mathbf{x}_i , a larger α_i is preferred in order to maximize the dual form L_D , and vice versa. This is intuitive because the resulting classifier $f(\mathbf{x})$ is adapted from $f^a(\mathbf{x})$ with the additional support vectors $\mathbf{x}_i \in \mathcal{D}_l^p$ and α_i can be seen as their weight. If the auxiliary classifier $f^a(\mathbf{x})$ misclassifies \mathbf{x}_i , the output of the adapted classifier $f(\mathbf{x}_i)$ needs to deviate from $f^a(\mathbf{x}_i)$ such that it can predict y_i correctly. This is realized by introducing a support vector \mathbf{x}_i with a large weight α_i . On the other hand, if the auxiliary classifier correctly classifies \mathbf{x}_i , the value $f(\mathbf{x}_i)$ does not need to be different from $f^a(\mathbf{x}_i)$, so the weight α_i can be small or even zero.

As is obvious from its dual form in Eq.(8), an A-SVM classifier does not introduce any extra variables or constraints other than those associated with the labeled examples in \mathcal{D}_l^p . Therefore, adapting $f^a(\mathbf{x})$ to $f(\mathbf{x})$ is no more expensive than training a standard SVM model entirely from \mathcal{D}_l^p , except the cost associated with computing every λ_i for $i = 1, \dots, N$. Since $\lambda_i = y_i f^a(\mathbf{x}_i)$ is a constant throughout the optimization process, there is only a one-time cost for evaluating $f^a(\mathbf{x})$ at N data points in \mathcal{D}_l^p . Although the actual cost varies depending on the complexity of the auxiliary classifier $f^a(\mathbf{x})$, it is typically small compared with the cost of iteratively optimizing the model parameters $\{\alpha_i\}_{i=1}^N$.

3.3 Many-to-one Adaptation

In many real-world applications, there are multiple auxiliary datasets $\mathcal{D}_1^a, \dots, \mathcal{D}_M^a$ with similar distributions to the primary (current) dataset \mathcal{D}^p . While our previous model allows only one auxiliary dataset, we remove this restriction by proposing an extended A-SVMs model which is able to leverage multiple auxiliary classifiers $f_1^a(\mathbf{x}), \dots, f_M^a(\mathbf{x})$. The key idea is to first construct an “ensemble” of auxiliary classifiers, and then adapt this ensemble to a new classifier $f(\mathbf{x})$ using the same method in Section 3.2. The adapted classifier has the following form:

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

where $t_k \in (0, 1)$ is the weight of each auxiliary classifier $f_k^a(\mathbf{x})$ which sums to one as $\sum_{k=1}^M t_k = 1$. The objective function is given by replacing $f^a(\mathbf{x})$ with $\sum_{k=1}^M t_k f_k^a(\mathbf{x}_i)$ in Eq.(3):

$$\begin{aligned} \min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i \\ \text{s.t. } \xi_i \geq 0, \quad y_i \sum_{k=1}^M t_k f_k^a(\mathbf{x}_i) + y_i \mathbf{w}^T \phi(\mathbf{x}_i) \geq 1 - \xi_i \end{aligned} \quad (11)$$

and the Lagrange dual form is rewritten as:

$$L_D = \sum_{i=1}^N (1 - \lambda_i) \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \quad (12)$$

where λ_i is now defined as $\lambda_i = y_i \sum_{k=1}^M t_k f_k^a(\mathbf{x}_i)$. Similarly, maximizing L_D under the KKT conditions (which need to be modified accordingly) gives the estimation of the model parameters $\hat{\alpha}$, and the adapted classifier is expressed as:

$$f(\mathbf{x}) = \sum_{k=1}^M t_k f_k^a(\mathbf{x}) + \sum_{i=1}^N \hat{\alpha}_i y_i K(\mathbf{x}, \mathbf{x}_i) \quad (13)$$

3.4 An Efficient Learning Algorithm

We describe an efficient algorithm for A-SVMs. The parameters α in A-SVMs are estimated by maximizing the dual objective function defined in Eq.(8) or Eq.(12). This is a large-scale quadratic programming (QP) problem, where the number of variables $\{\alpha_i\}_{i=1}^N$ is equal to the number of labeled examples. The sequential minimal optimization (SMO) algorithm proposed by Platt [12] can efficiently solve a large QP problem by decomposing it into a series of QP subproblems and optimizing them iteratively. We present a modified SMO algorithm to A-SVMs as follows.

Specifically, α is the optimal solution to the QP problem in Eq.(8) if and only if the KKT conditions in Eq.(7) are fulfilled. This QP problem is solved when, for all i :

$$\begin{aligned} \alpha_i = 0 &\Rightarrow \mu_i = C, \xi_i = 0 \Rightarrow y_i f(\mathbf{x}_i) \geq 1 \\ 0 < \alpha_i < C &\Rightarrow \mu_i > 0, \xi_i = 0 \Rightarrow y_i f(\mathbf{x}_i) = 1 \\ \alpha_i = C &\Rightarrow \mu_i = 0, \xi_i \geq 0 \Rightarrow y_i f(\mathbf{x}_i) \leq 1 \end{aligned} \quad (14)$$

Eq.(14) provides a way to check the optimality condition of the problem and to find variables that violate the optimality condition. This SMO algorithm chooses one variable to optimize in each iteration. While there are many ways to select

working variables, we adopt an intuitive heuristic of selecting the variable α_{i^*} that violates the optimality condition the most:

$$i^* = \operatorname{argmax}_{i \in \{i_{up}, i_{low}\}} |y_i f(\mathbf{x}_i) - 1| \quad (15)$$

where $i_{up} = \operatorname{argmin}_{i \in \{t | \alpha_t < C\}} y_i f(\mathbf{x}_i)$, $i_{low} = \operatorname{argmax}_{i \in \{t | \alpha_t > 0\}} y_i f(\mathbf{x}_i)$

Without loss of generality, let us suppose α_1 is the working variable to optimize. We update it by setting the derivative of the dual objective function L_D against α_1 to zero:

$$\frac{\partial L_D}{\partial \alpha_1} = 1 - y_1 f^{old}(x_1) - y_1 (\alpha_1^{new} - \alpha_1^{old}) K(x_1, x_1) \triangleq 0$$

where $f^{old}(x)$ is the decision function Eq.(9) or (13) evaluated using the existing value of α . This leads to an analytical solution of α_1 :

$$\alpha_1^{new} = \alpha_1^{old} + \frac{1 - y_1 f^{old}(x_1)}{K(x_1, x_1)} \quad (16)$$

which may need to be clipped to satisfy the constraint $0 \leq \alpha_1 \leq C$. To learn an A-SVM classifier, we iteratively choose working variables by Eq.(15) and optimize them using Eq.(16). This process continues until the optimality condition in Eq.(14) is satisfied up to a certain accuracy.

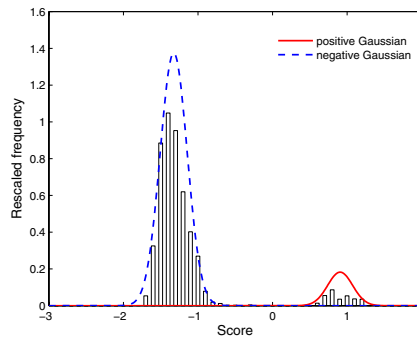
4. AUXILIARY CLASSIFIERS SELECTION

An important problem related to the success of classifier adaptation is the selection of auxiliary classifiers from a set of existing classifiers. Ideally, one should select the “best” classifier such that the classifier adapted from it outperforms those adapted from other existing classifiers on the primary dataset. This metric, however, is difficult to compute, and in practice we select the classifiers that have the highest *estimated performance* on the primary dataset to be the auxiliary classifiers. This is because intuitively, a classifier with the high performance on the primary data must be trained from a similar distribution and thus provides a good starting point for adaptation.

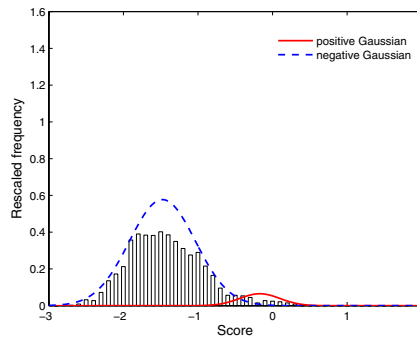
A classifier’s performance on the limited labeled examples \mathcal{D}_l^p may not be a reliable approximation of its performance on the whole data \mathcal{D}^p , because \mathcal{D}_l^p is too small to faithfully represent the distribution of \mathcal{D}^p . Labeling more examples in \mathcal{D}^p improves the quality of the estimation at the cost of extra human labor, which offsets the benefit of classifier adaptation on minimizing labeling effort. In order to improve the performance estimation without having to label more examples, we explore several meta-level features that indicate a classifier’s performance on a given dataset and can be computed without data labels. Some of these features come from the score distribution of the classifier, while the other features are related to its consistency with other classifiers.

4.1 Features on Score Distribution

A classifier produces a score for each instance to indicate how likely it is a positive (or negative) instance. The distribution of the scores from a “good” classifier exhibit certain properties. For example, the scores of positive instances should be well separated from the scores of negative ones, and we can use such property to judge the “goodness” of a classifier. However, it is difficult to examine the between-class score separation on the primary data where most instance labels are unknown. To overcome this problem, we



(a) Classifier A (AP=0.972)



(b) Classifier B (AP=0.124)

Figure 3: The score distribution of two classifiers on the same dataset. The histograms show the actual score distributions, and the Gaussian curves are fit by EM. Classifier A performs better than B.

propose a model-based approach: assuming that the scores of positive and negative data follow distributions of a certain family, we recover their respective distributions from the overall score distribution using the Expectation Maximization (EM) algorithm [5], and then examine the score separation from the estimated distributions.

Modeling score distribution have been discussed in the context of combining multiple search engine outputs [9] and rank aggregation [3]. Given the diversity of classification algorithms, it is impossible to find a distribution family that fits the scores produced by all the classifiers. Therefore, we made several simplifying assumptions on the scores generated by a classifier in our problem: (1) the scores are in the range of $(-\infty, +\infty)$; (2) the boundary between two classes is zero, with scores above zero indicating positive instances, and vice versa; (3) the absolute value of a score indicates the degree of confidence on the predicted label. While seemingly restrictive, these assumptions are actually general enough to include popular methods such as SVMs and logistic regression, which have been the dominant algorithms in our target application on video concept detection [14, 19].

Similar to [3], we use two Gaussian distributions to fit the scores of positive and negative instances respectively. This is because Gaussian distributions fit the actual scores well and their parameters are easy to estimate. Suppose $z = f^a(\mathbf{x})$ is the score of instance \mathbf{x} produced by classifier f^a . The scores of positive instances follow the distribution $p(z|y = 1) = \mathcal{N}(u_p, \sigma_p^2)$, where u_p and σ_p^2 are the mean and variance. Similarly, $p(z|y = -1) = \mathcal{N}(u_n, \sigma_n^2)$ is the distribution of

Feature	Description
<i>sample_ap</i>	average precision of f^a on the labeled subset \mathcal{D}_l^p of the primary data
<i>max_score</i>	the maximum score produced by f^a on the primary data
<i>pos_neg_dist</i>	distance b/w the estimated mean scores of positive data and of negative data
<i>pos_mid_dist</i>	distance b/w the mean score of positive data to the point where the positive and negative score distribution intersects
<i>aggregation_ap</i>	average precision of f^a computed on pseudo labels derived from score aggregation

Table 1: Meta-level features for predicting the performance of a classifier $f^a(\mathbf{x})$ on the primary dataset \mathcal{D}^p .

the scores of negative instances. We also assume the prior of labels to be $P(y = 1) = \pi$ and $P(Y = -1) = 1 - \pi$. The overall score distribution is therefore a Gaussian mixture model with two components:

$$p(z) = \pi \mathcal{N}(u_p, \sigma_p) + (1 - \pi) \mathcal{N}(u_n, \sigma_n) \quad (17)$$

The parameters $(\pi, u_p, \sigma_p, u_n, \sigma_n)$ of this score model can be easily estimated if both the scores $\{z_i\}$ and labels $\{y_i\}$ are available. If the labels are unknown, which is the case in our problem, we can estimate these parameters from only the scores $\{z_i\}$ using EM algorithm. The EM algorithm iteratively improves the model parameters starting from their initial values until it finds two Gaussian components that best fit the scores. Figure 3 shows the estimated score distributions of two “studio” classifiers, one trained from NTDTV-1 program and the other from NBC-1 program in TRECVID collection [13], both applied to the NTDTV-2 program (see Section 6.1 for the details of these datasets). We find that the Gaussian mixture models estimated by EM fit the score distributions well. Moreover, it shows a strong relationship between the score distribution and the classifier performance, because the classifier with larger between-class score separation has much higher performance.

Suppose $(\hat{\pi}, \hat{u}_p, \hat{\sigma}_p, \hat{u}_n, \hat{\sigma}_n)$ are the parameters estimated by the EM algorithm. An intuitive indicator of the score separation is the distance between the mean of the positive and negative data as $\hat{u}_p - \hat{u}_n$. A slightly different feature, which is proven to be effective in previous work [9], is the distance between the mean of the positive \hat{u}_p to the middle point z^{mid} at which the two Gaussian densities intersect (i.e., $p(z^{mid}|y = 1) = p(z^{mid}|y = -1)$).

4.2 Features on Score Aggregation

Another way to evaluate classifiers is based on the notion that the “average” of multiple classifiers is usually better than any individual one. Specifically, we aggregate the outputs of multiple classifiers to predict the labels of the primary data, and then use these “pseudo” labels to evaluate each classifier. Assuming z_i^1, \dots, z_i^M are the scores on instance \mathbf{x}_i produced by a set of M existing classifiers, a principled way to aggregate these scores is to compute the posterior distribution as $P(y_i = 1 | z_i^1, \dots, z_i^M)$. If the outputs of different classifiers are independent given an instance and its label, we have the following based on Baye’s rule:

$$P(y_i = 1 | z_i^1, \dots, z_i^M) = \frac{P(y_i = 1) \prod_{k=1}^M p(z_i^k | y_i = 1)}{\sum_{y_i = -1, 1} P(y_i) \prod_{k=1}^M p(z_i^k | y_i)} \quad (18)$$

where $p(z_i^k | y_i = 1)$ and $p(z_i^k | y_i = -1)$ are Gaussian distributions fit by the EM algorithm described above, and the prior $P(y_i)$ is set to the ratio of positive (or negative) instances in the training data of these classifiers.

An individual classifier is evaluated by measuring the agreement between its output and the estimated posterior probability. One way is to convert the posteriors into the pseudo labels as $\hat{y}_i = \text{sgn}(P(y_i = 1 | z_i^1, \dots, z_i^M) - 0.5)$, and compute a certain performance metric (e.g., average precision [13]) of the classifier based on the pseudo labels. Another way is to convert the posteriors into ranks and measure the consistency between this aggregated rank and the rank generated by a classifier using the Kendall tau distance. Empirically we find the measure based on pseudo labels is better.

4.3 Predicting Classifier Performance

We build a regression model to predict a classifier’s performance in terms of average precision (AP) on the primary dataset. As summarized in Table 1, the input includes the meta-level features collected from score distribution and score aggregation, as well as the performance of the classifier on the labeled examples \mathcal{D}_l^p of the primary set. The output is the classifier’s performance on the whole primary set \mathcal{D}^p . The regression model is trained using support vector regression (SVR) [4]. The existing classifier(s) with the highest (estimated) AP on the primary data are selected as auxiliary classifiers. Note that we choose AP as the performance metric because it is widely used in video concept detection and retrieval [11, 13, 14, 19]. The features used here are general enough for predicting other metrics such as classification accuracy.

5. ALTERNATIVE ADAPTATION METHODS

We describe two alternative approaches based on SVMs that leverage auxiliary data or classifiers for classifying the primary data, and discuss their relations to A-SVMs.

Aggregate Approach: This is a computationally intensive approach which learns a single SVM classifier using *all* the labeled examples in the auxiliary datasets $\{\mathcal{D}_k^a\}_k$ and in the primary dataset \mathcal{D}_l^p . The decision function is in the form of $f^{agg}(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$ with \mathbf{w} are estimated from the following objective function:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i + \sum_{k=1}^M C_k^a \sum_{i=1}^N \xi_i^k \quad (19) \\ \text{s.t.} \quad & \xi_i \geq 0, \quad \xi_i^k \geq 0 \\ & y_i \mathbf{w}^T \phi(\mathbf{x}_i) \geq 1 - \xi_i, \quad \forall (\mathbf{x}_i, y_i) \in \mathcal{D}_l^p \\ & y_i^k \mathbf{w}^T \phi(\mathbf{x}_i^k) \geq 1 - \xi_i^k, \quad \forall (\mathbf{x}_i^k, y_i^k) \in \mathcal{D}_k^a, \forall k \end{aligned}$$

The cost factor C and C_k^a can be seen as the weights of the instances in the primary set \mathcal{D}_l^p and the auxiliary set \mathcal{D}_k^a . This allows the instances from different datasets to be weighted according to their relative importance. Usually, we have $C > C_k^a$ because the examples from the primary set are more valuable.

Many existing works [2, 6, 8, 17] in data mining and transfer learning can be seen as variants of this aggregate approach. This approach is fundamentally different from A-SVMs in that it involves the auxiliary data in the training process while A-SVMs directly manipulate the auxiliary classifiers. The number of parameters to be optimized is N in A-SVMs and $N + \sum_{k=1}^M N_k^a$ in this aggregate approach, so training A-SVMs is significantly more efficient. However, it is hard to predict which one performs better in terms of classification accuracy and we leave it to empirical study. Moreover, the A-SVMs model is more flexible because it is applicable even when the auxiliary data are unavailable or inaccessible for some reasons.

Ensemble Approach: While the aggregate approach combines the training examples, the ensemble approach combines the output of classifiers trained separately on different datasets, i.e., the auxiliary classifiers and a SVM classifier trained from the labeled examples in \mathcal{D}_l^p (denoted as the primary classifier $f^p(\mathbf{x})$). The final score of an instance \mathbf{x} in this ensemble classifier is computed as:

$$f^{ens}(\mathbf{x}) = C f^p(\mathbf{x}) + \sum_{k=1}^M C_k^a f_k^a(\mathbf{x}) \quad (20)$$

where C and C_k^a here are weights for the primary and auxiliary classifiers. This has been the approach used in [7, 16]. The ensemble form in Eq.(20) is similar to that of A-SVMs in Eq.(10) except one important difference. Here, the primary classifier $f^p(\mathbf{x})$ is trained independently from the auxiliary classifiers $\{f_k^a\}_k$, while in A-SVMs the delta function $\Delta f(\mathbf{x})$ is trained under the influence of $\{f_k^a\}_k$, which can provide valuable prior information beyond the limited labeled examples.

6. CROSS-DOMAIN VIDEO CONCEPT DETECTION

6.1 The Task and Collection

Video concept detection is to automatically classify video shots by the presence or absence of certain semantic concepts, such as *Studio*, *Outdoor*, and *Sports*. Cross-domain video concept detection is to do that in a corpus with multiple domains (data sources) by extending concept classifiers across different domains. We investigate the problem of cross-domain concept detection in the benchmark dataset used in TREC Video Retrieval Evaluation 2005 (TRECVID) [13]. It contains 86-hour video footage, which is partitioned into 74,523 video shots. Each shot is represented by one video frame as its “keyframe”, and 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. As part of the LSCOM-Lite project [10], all the shots have been manually annotated with (binary) labels indicating the presence or absence of 39 semantic concepts. These concepts cover a wide variety of types, varying from outdoor scene (*Building*, *Road*), indoor setting (*Studio*, *Meeting*), to news genre (*Sports*, *Entertainment*), and general objects (*Airplane*, *Animal*).

The video footage in TRECVID belongs to 13 news programs from 6 channels, including CNN, NBC, MSNBC, CCTV (Chinese), NTDTV (Chinese), and LBC (Arabic), making it suitable for evaluating cross-domain concept detection. Each channel except LBC has two news programs, while

LBC has three. We name each program by their channel, such as CNN-1, CNN-2, and so on. The video shots are relatively evenly distributed among the news programs. As exemplified in Figure 1 and 2, due to editing styles and other factors, shots belonging to the same concept but from different programs exhibit large visual difference. Based on our observation, programs from the same channel (a.k.a sibling programs) are usually more similar compared with programs from different channels in terms of data distribution.

In each setting of our experiments, we pick one of the 39 concepts as the *target concept* and one of the 13 programs as the *target program*. The target program is the primary dataset and the other 12 programs are the candidates of auxiliary datasets. A limited number of randomly selected shots are labeled in the target program. Meanwhile, the other 12 programs are fully labeled, and a classifier for the target concept has been trained from each of these programs. The goal is to classify the video shots in the target program by the presence or absence of the target concept. Our approach is to select one or more existing classifiers as auxiliary classifiers and adapt them into a new classifier for the target program based on the limited labeled shots in it. By varying the target program and target concept, we have 13×39 settings. We remove the settings where the number of relevant (positive) shots in the target program is less than 10, which results in 384 qualified settings for evaluation.

The adapted classifier is evaluated on the *unlabeled* shots in the target program, as the labeled ones have been polluted in the training. We convert the classifier output into a shot list ranked in descending order by their scores, and measure its quality using average precision (AP), which is a standard metric in TRECVID. AP is the average of the precisions of the ranking list truncated at each of the relevant shots, and its value approximates the area under the precision-recall curve. We also use mean average precision (MAP) to average the APs in multiple settings.

6.2 Adaptation Strategies

We examine a variety of adaptation strategies for building concept classifiers for a target program where only limited labeled examples are available. In each strategy, we first adopts a criterion to rank all the classifiers trained on the other programs by their utility to the target program, selects the top-ranked ones as auxiliary classifiers (programs), and picks an adaptation method to train the classifier for the target program. We list the design choices on these aspects below:

Selection Criterion: We examine five utility criteria to rank the candidate classifiers.

- **Oracle** ranks the candidate classifiers by their *actual* performance in AP on the target program. It guarantees to be optimal, but it is also unrealistic as it requires *all* the labels of the target program.
- **Random** ranks the candidate classifiers randomly.
- **Prior** places the classifier trained on the sibling program of the target program (i.e., the one from the same channel as the target) at the top, and ranks other classifiers randomly. It uses the prior knowledge that sibling programs are likely to have similar data distribution. Such prior knowledge is unavailable to the other criteria discussed below.

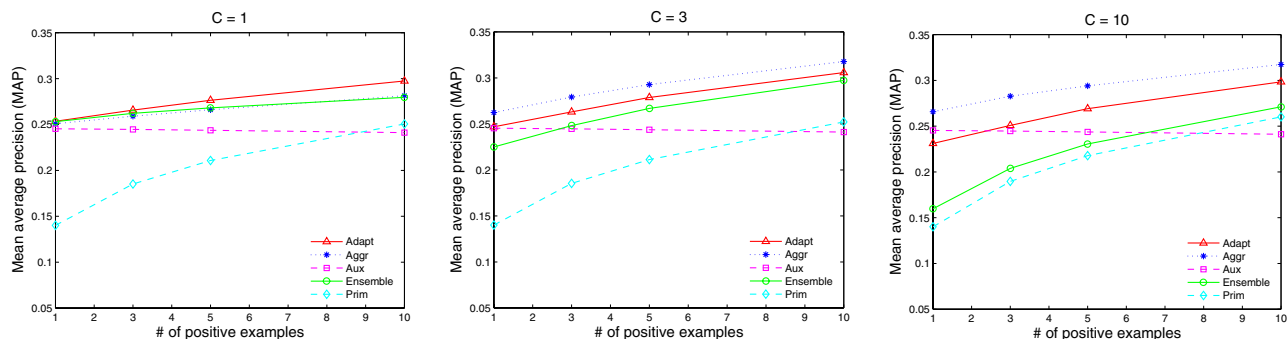


Figure 4: The performance (MAP) of 5 methods on the target program with $C = 1, 3, 10$ and Prior selection criterion of auxiliary classifier.

- **Sample** ranks the candidate classifiers by their performance in AP on the small set of labeled shots in the target program.
- **Meta** ranks the candidate classifiers by their predicted performance (AP) on the target program. It uses the regression model in Section 4 to predict a classifier’s AP based on meta-level features listed in Table 1.

Number of Auxiliary Classifiers: Given a ranking criterion, the top-ranked candidate classifiers are selected as auxiliary classifiers. We vary of the number of selected classifiers from 1 to 5 in order to study its impact on the classification performance.

Methods: We compare five types of classifiers, among which three make use of the auxiliary classifiers or data:

- **Prim:** a SVM classifier trained entirely from the labeled examples in the target program, denoted as primary classifier. This is a baseline method.
- **Aux:** the auxiliary classifier, or an ensemble of multiple auxiliary classifiers. This is another baseline method.
- **Adapt:** the classifier adapted from the auxiliary classifier(s) **Aux** using the labeled examples in the target program, based on the proposed A-SVMs model.
- **Aggr:** an aggregate SVM classifier trained from all the labeled examples in the auxiliary programs and in the target program (i.e., Eq.(19)).
- **Ensemble:** a weighted ensemble classifier combining the scores of the auxiliary classifiers **Aux** and the primary classifier **Prim** as a weighted sum (i.e., Eq.(20)).

Great cares are taken to ensure these methods are comparable. We use the same RBF kernel function $K(x_i, x_j) = e^{-\rho \|x_i - x_j\|^2}$ with $\rho = 0.1$ in all the methods. For the cost factor, we use a fixed $C^a = 1$ for training of **Aux** and the same C for the training of **Prim**, **Adapt**, **Aggr** and **Ensemble**, i.e., the C in Eq.(3), (11), (19) and (20) are set to the same value. This ensures **Adapt** and **Aggr** are comparable as both weight auxiliary and primary data in the same way, and because C and C^a are weights for classifiers in **Ensemble**, this also makes **Ensemble** comparable to **Adapt** and **Aggr**. When multiple auxiliary classifiers are available, we set C_k^a in Eq.(19) and (20) to $C_k^a = t_k = 1/M$, where t_k is the weight for auxiliary classifiers in Eq.(10) and M is the number of auxiliary classifiers. This means we treat multiple auxiliary classifiers (or datasets) equally.

# of positive / all examples	Total training time (minutes)		
	Prim/Ensemble	Adapt	Aggr
1 / 16.5	10.3	13.5	191.0
3 / 49.4	10.5	13.8	209.9
5 / 82.4	12.1	14.9	229.0
10 / 164.7	17.4	20.1	271.9

Table 2: The total training time of each method summed from all the settings.

7. EXPERIMENTAL RESULTS

We present the experimental results on adaptation methods applied to cross-domain video concept detection in the TRECVID collection. The results are presented in two parts. The first part focuses on the comparison between A-SVMs and other adaptation and baseline methods, while the second part is on the selection of auxiliary classifiers.

7.1 Results on Adaptation Methods

As mentioned in Section 6.1, there are a total of 384 settings depending on the choice of target concept and target program. In each setting, we build 5 classifiers as **Prim**, **Aux**, **Adapt**, **Aggr**, and **Ensemble** to classify the data in the target program. We use the **Prior** criterion to select the sibling program as the auxiliary for the target program. Because the labeled examples are sampled randomly, we repeat the experiment in each setting for 4 times with different random samples and average the performance. In TRECVID, positive examples are usually more valuable than negative ones since most of the concepts are rare. Therefore, we sample positive and negative examples separately according to their ratio, and use the number of positive ones as an indicator of the amount of information available to the classification methods. In the experiment, we set the number of positive examples to 1, 3, 5, and 10. The corresponding number of negative examples (averaged from different concepts) concepts are 15.5, 46.4, 77.4, and 154.7.

Figure 4 shows the performance of the 5 classifiers in terms of MAP across the 384 settings against the number of positive training examples. On average, the three adaptation methods as **Adapt**, **Aggr**, **Ensemble** perform significantly better than the two baseline methods as **Prim** and **Aux**, indicating the benefit of leveraging the knowledge of auxiliary data or classifiers. Depending on the choice of the cost factor C , either **Adapt** or **Aggr** is the top-performer, while **Ensemble** is worse. Since **Adapt** has comparable performance with **Aggr**

Metric	MAP of the top-ranked classifier				Ratio of the optimal classifier ranked in top 3			
	1	3	5	10	1	3	5	10
# of positive examples								
Oracle	0.247	0.247	0.247	0.247	100%	100%	100%	100%
Prior	0.211	0.211	0.211	0.211	68.2%	68.2%	68.2%	68.2%
Meta	0.188	0.201	0.208	0.216	56.6%	60.7%	68.0%	76.9%
Sample	0.153	0.186	0.201	0.218	44.6%	57.8%	64.9%	76.2%
Random	0.132	0.132	0.132	0.132	30.1%	30.1%	30.1%	30.1%

Table 3: Comparison of 5 auxiliary classifier selection criteria.

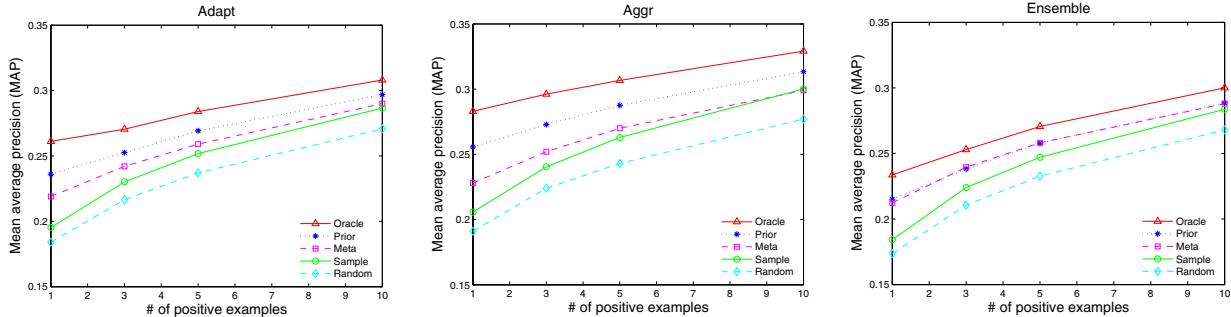


Figure 5: Performance of Adapt, Aggr, and Ensemble using 5 auxiliary classifier selection criteria. Only the top-ranked classifier is used as the auxiliary classifier; $C = 3$.

and is much more efficient to train (see Table 2), we conclude that A-SVMs provide a good tradeoff between effectiveness and efficiency for classifier adaptation. Another advantage of Adapt (or A-SVMs) is that it is relatively less sensitive to the choice of C compared with Aggr and Ensemble.

Efficiency in terms of training time is another aspect of the performance. Table 2 summarizes the total training time for all the settings of the above methods except Aux which has already been trained. Among them, the training time for Prim and for Ensemble are basically identical since the latter involves the training of the former. The training cost of Adapt is only slightly higher than Prim, while the cost of Aggr is an order of magnitude higher due to the incorporation of auxiliary training data (over 1,000 extra examples in average). This shows the efficiency advantage of A-SVMs as the result of directly manipulation of the classifier functions.

7.2 Results on Auxiliary Classifier Selection

In each setting, we need to select auxiliary classifier(s) for the target program from a pool of classifiers trained on the other 12 programs. This is done by the 5 selection criteria discussed in Section 6.2. The candidate classifiers are ranked using each of the criteria, and the top-ranked ones are selected as auxiliary classifiers. We evaluate these selection criteria using two metrics: (a) the actual performance (MAP) of the top-ranked classifier on the target program; (b) the ratio that the optimal classifier (the one with the highest performance on the target program) is ranked within top 3. The results are summarized in Table 3. Among these criteria, Oracle is the optimal criterion, Random is the baseline, and the performance of the other three criteria are in between. The prior knowledge on whether two programs are from the same channel turns out to be very useful, since the Prior is best criterion except Oracle. Sample is a relatively poor criterion when the labeled examples are rare, but gets better as more examples are labeled. Meta is much better than Sample when the labeled examples are scarce, showing that the meta-level features from score distribution are useful in terms of predicting the classifier’s performance.

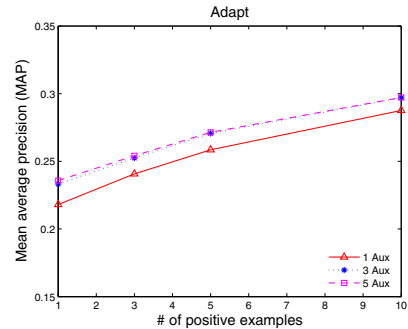


Figure 6: Performance of Adapt with different number of auxiliary classifiers selected by Meta ($C = 3$)

We then evaluate the performance of the adaptation methods (Adapt, Aggr, and Ensemble) using the auxiliary classifier selected by these criteria. Only the classifier ranked first is chosen as the auxiliary classifier, and the cost factor is fixed to $C = 3$. The performance in terms of MAP computed from all the settings are shown in Figure 5. We find that the choice of auxiliary classifier has a large impact to the performance despite the adaptation method used. In fact, this choice is more important for good adaptation methods, because the gaps between different criteria are larger for Aggr and Adapt than for Ensemble. Moreover, the order in which these selection criteria performs is consistent with Table 3. Prior is our first choice if such prior knowledge is available. If not, we should use Meta which exploits the meta-level features of a classifier in addition to its performance on the labeled samples, especially when they are scarce.

Does more auxiliary classifiers imply better performance? Using the Meta selection criterion, we evaluate the Adapt method with 1, 3, and 5 auxiliary classifiers. The auxiliary classifiers are equally weighted, i.e., $t_k = 1/M$ in Eq.(10). As shown in Figure 6, increasing the number of auxiliary clas-

sifiers from one to three results in a moderate performance improvement, but only minimal improvement is seen when the number increases to 5. Comparing this result to Figure 5, we conclude that increasing the number of auxiliary classifiers is not as effective as choosing the right auxiliary classifier.

8. CONCLUSION

We have proposed adaptive support vector machines (A-SVMs) for adapting auxiliary classifiers to a new dataset which contains only limited labeled examples, and a method for selecting the most effective auxiliary classifiers for adaptation. This general method has been evaluated on cross-domain video concept detection in the TRECVID 2005 collection. We have the following observations from the experiments. First, adapted classifiers trained by A-SVMs significantly outperform auxiliary classifiers and new classifiers trained from the labeled examples; Second, compared with other adaptation techniques, our approach achieves better performance than the ensemble approach and comparable performance to the aggregate approach while requiring 1/10 of the latter’s training time; Third, selecting good auxiliary classifiers for adaptation is critical to the performance, and our selection method has proved to be effective.

There are many open questions for the framework of classifier adaptation. For example, currently the cost factor in our model, which balances the contribution between the auxiliary classifiers and training examples, is set manually or empirically. Since it has shown to have an impact on the performance (see Figure 4), a mechanism for automatically choosing the “right” cost factor is desirable. One possibility is to associate it with the performance of the auxiliary classifier, which can be predicted using the method in Section 4, such that a good auxiliary classifier has a larger contribution and vice versa. This is related to other important questions such as whether to perform classifier adaptation and how many auxiliary classifiers are needed. In extreme cases, one can directly use the existing classifiers without adaptation if they perform well enough, or discard existing classifiers and build new ones if they turn out to be useless. We plan to explore these questions in our future work.

9. REFERENCES

- [1] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Proc. of Neural Information Processing Systems*.
- [2] S. J. Delany, P. Cunningham, A. Tsymbal, and L. Coyle. A case-based technique for tracking concept drift in spam filtering. *Knowledge-Based Systems*, 18(4–5):187–195, 2005.
- [3] D. Ding and B. Zhang. Probabilistic model supported rank aggregation for the semantic concept detection in video. In *Proc. of ACM Int’l Conf. on Image and Video Retrieval*, 2007.
- [4] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. pages 155–161, 1996.
- [5] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: Data mining, inference, and prediction*.
- [6] 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.
- [7] J. Z. Kolter and M. A. Maloof. Dynamic weighted majority: A new ensemble method for tracking concept drift. In *Proc. of 3rd IEEE Int’l Conf. on Data Mining*, page 123, 2003.
- [8] X. Liao, Y. Xue, and L. Carin. Logistic regression with an auxiliary data source. In *Proc. of the 22nd Int’l Conf. on Machine Learning*, pages 505–512, New York, NY, USA, 2005. ACM Press.
- [9] R. Manmatha, T. Rath, and F. Feng. Modeling score distributions for combining the outputs of search engines. In *Proc. of the 24th Annual Int’l ACM SIGIR conf. on Research and Development in Information Retrieval*, pages 267–275, New York, NY, USA, 2001.
- [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. Tesic. Learning the semantics of multimedia queries and concepts from a small number of examples. In *Proc. of 13th Annual ACM Int’l Conf. on Multimedia*, pages 598–607, 2005.
- [12] J. Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods: support vector learning*, pages 185–208, 1999.
- [13] A. Smeaton and P. Over. Trecvid: Benchmarking the effectiveness of information retrieval tasks on digital video. In *Proc. of the Intl. Conf. on Image and Video Retrieval*, 2003.
- [14] C. G. M. Snoek, M. Worring, J. C. van Gemert, J.-M. Geusebroek, and A. W. M. Smeulders. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *Proc. of 14th Annual ACM Int’l Conf. Multimedia*, pages 421–430, 2006.
- [15] N. Syed, H. Liu, and K. Sung. Incremental learning with support vector machines. In *In Proc. of the Workshop on Support Vector Machines, at the Int’l Joint Conf. on Artificial Intelligence*, 1999.
- [16] H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proc. of 9th ACM SIGKDD Int’l Conf. on Knowledge Discovery and Data Mining*, pages 226–235, 2003.
- [17] 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.
- [18] X. Wu and R. Srihari. Incorporating prior knowledge with weighted margin support vector machines. In *Proc. of the 10th ACM SIGKDD Int’l Conf. on Knowledge Discovery and Data Mining*, pages 326–333, 2004.
- [19] Y. Wu, E. Y. Chang, K. C.-C. Chang, and J. R. Smith. Optimal multimodal fusion for multimedia data analysis. In *Proc. of the 12th annual ACM Int’l Conf. on Multimedia*, pages 572–579, 2004.