Adapting SVM Classifiers to Data with Shifted Distributions

Jun Yang
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 10523
juny@cs.cmu.edu

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

Alexander G. Hauptmann
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 10523
alex@cs.cmu.edu

Abstract

Many data mining applications can benefit from adapting existing classifiers to new data with shifted distributions. In this paper, we present Adaptive Support Vector Machine (Adapt-SVM) as an efficient model for adapting a SVM classifier trained from one dataset to a new dataset where only limited labeled examples are available. By introducing a new regularizer into SVM's objective function, Adapt-SVM aims to minimize both the classification error over the training examples, and the discrepancy between the adapted and original classifier. We also propose a selective sampling strategy based on the loss minimization principle to seed the most informative examples for classifier adaptation. Experiments on an artificial classification task and on a benchmark video classification task shows that Adapt-SVM outperforms several baseline methods in terms of accuracy and/or efficiency.

1 Introduction

Many real-world applications face a common problem that the data distribution associated with a specific topic or category is likely to shift, especially in streaming data generated over a long period of time or heterogenous data gathered from multiple sources. This creates a fundamental difficulty for learning methods such as supervised classifiers that assume the training and testing data are drawn from the same distribution. This problem has been frequently seen in data mining (as concept drift), adaptive filtering (as users' changing preferences), and text/multimedia document classification (as large in-class variances). Applications in these areas can largely benefit from an efficient method for adapting existing classifiers to a new set of data with a different distribution.

We formulate this general problem into the following scenario. Consider a binary classification task with respect to a given topic in a **primary dataset**, where only a limited number of examples are labeled. Besides, there is a fully-labeled **auxiliary dataset**, and an **auxiliary classifier** has been trained from it. The primary data is drawn from a distribution that is related to, but different from, the distribution of the auxiliary data in a way unknown to the learner. We call this kind of distribution as a **shifted distribution**.

To classify the primary data, the auxiliary classifier may not perform well since it is biased to the training distribution. On the other hand, a new classifier trained only from the limited examples in the primary data, although unbiased, may suffer from a high variance. To achieve a better bias-variance tradeoff, various methods have been proposed in the data mining and machine learning community to leverage the knowledge of the auxiliary (old) data to build better classifiers for the primary (new) data. Two main approaches are to construct an "ensemble" combining the output of classifiers trained independently from the two datasets [4, 10], or to train an "aggregated" classifier on the labeled examples combined from both datasets [3, 5, 11]. In this paper, we seek a rather different approach that *adapts* the auxiliary classifier (more precisely, its decision function) to the primary data using its limited labeled examples (which results in an **adapted classifier**), with three goals:

Efficiency: Adapting an existing classifier must be less expensive than training a classifier on the labeled examples combined from the primary and auxiliary set.

Accuracy: On average, the adapted classifier is expected to outperform existing approaches based on ensemble classifiers and aggregated classifiers.

Minimum human effort: Adapting a classifier should require significantly fewer labeled examples than what is needed to train a new classifier of the same accuracy.

We propose *Adaptive Support Vector Machines* (Adapt-SVM) as an efficient and principled method for adapting the auxiliary classifier to the primary data with a shifted distribution using its limited labeled examples. The key idea is

to modify the regularizer in the objective function of SVMs so that both the classification error over the training examples and the discrepancy between the adapted and the auxiliary classifier are minimized. A fundamental difference between this method and existing methods in drifting concept detection [4, 3, 10] and transfer learning [5, 11] is that it directly manipulates the auxiliary classifier and involves *no* auxiliary data in its training process. This makes it applicable to domains where old data are unaccessible, and also more efficient than models trained over old data. Moreover, based on the loss minimization principle, a *selective sampling* strategy is proposed to identify the most useful examples to help the adaptation of the auxiliary classifier. Experiments on a synthetic dataset and a benchmark video classification task show that our approach outperform other methods in terms of accuracy and/or efficiency, and using selective sampling offers further improvement.

2 Adaptive Support Vector Machines

We use $D = \{x_i, y_i\}_{i=1}^N$ to denote a set of N labeled instances in the *primary* dataset, where x_i is the i^{th} data vector and $y_i \in \{-1, 1\}$ is its binary label. In addition to D, there is a large number of *unlabeled* data in the primary dataset to be classified. Moreover, $D' = \{x'_i, y'_i\}_{i=1}^{N'}$ denotes a set of N' labeled instances in the *auxiliary* dataset.

2.1 The Linear Model

In standard linear SVM, the label of a data vector x is determined by the sign of a linear decision function, i.e., $\hat{y} = sgn(f(x)) = sgn(x^T\beta)$, where $\beta = \{\beta_i\}_{i=1}^M$ are the model parameters[1]. Training a linear SVM classifier involves the following optimization problem:

$$\min_{\beta} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i \qquad (1)$$
$$\text{subject to} \quad \xi_i \geq 0, \;\; y_i x_i^T\beta \geq 1 - \xi_i \;\; \forall i$$

where $\sum_i \xi_i$ measures the total classification error, and $\|\beta\|^2$ is a regularization term that is inversely related to margin between training examples of two classes.

Adapt-SVM extends linear SVM to incorporate the knowledge of the auxiliary data D' through a modified regularization term. We assume the auxiliary classifier trained from D' to be a linear SVM as $f'(x) = x^T\beta'$, where β' can be represented as $\beta' = \sum_{i=1}^{N'} \alpha'_i y'_i x'_i$ according to the representer's theorem. Since the distribution of D is relevant to that of D', it is reasonable to treat the auxiliary classifier $f'(x)$ as a "prior model" of the adapted classifier $f(x)$ to be learned. This is achieved by modifying the regularizer to penalize the discrepancy between the new parameters β and the parameters β' of the auxiliary classifier. Specifically, we replace $\|\beta'\|^2$ in Eq.(1) with a new regularizer $\|\beta - \beta'\|^2$:

$$\min_{\beta} \frac{1}{2}\|\beta - \beta'\|^2 + C\sum_{i=1}^{N}\xi_i \qquad (2)$$
$$\text{subject to} \quad \xi_i \geq 0, \;\; y_i x_i^T\beta \geq 1 - \xi_i \;\; \forall i$$

This modified objective function seeks to reduce the distance between β and β' while minimizing the classification error. Based on the triangular inequality, we have $\|\beta'\| + \|\beta - \beta'\| \geq \|\beta\|$. Because $\|\beta\|^2$ is inversely related to the margin, and $\|\beta'\|$ is a constant, minimizing $\|\beta - \beta'\|^2$ is equivalent to maximizing the lower bound of the margin. Therefore, Eq.(2) also ensures a decision boundary with a large margin. The objective function in Eq.(2) can be rewritten as the following (primal) Lagrangian:

$$L_P = \frac{1}{2}\|\beta - \beta'\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\{y_i x_i^T\beta - (1-\xi_i)\} - \sum_{i=1}^{N}\mu_i\xi_i \qquad (3)$$

where $\alpha_i \geq 0, \mu_i \geq 0$ are Lagrange multipliers. We minimize L_P by setting its derivative with respect to β and ξ to zero, which gives $\beta = \sum_{i=1}^{N}\alpha_i y_i x_i + \beta'$ and $\alpha_i = C - \mu_i$ for every i. Substituting β and α_i into Eq.(3), we get the Lagrange dual objective function:

$$L_D = \sum_{i=1}^{N}(1 - \lambda_i)\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j x_i^T x_j \qquad (4)$$

where $\lambda_i = y_i \sum_{j=1}^{N'} \alpha'_j y'_j x_i^T x'_j = y_i f'(x_i)$. The model is now parameterized by $\{\alpha_i\}_{i=1}^N$, which can be estimated by maximizing L_D under the constraints $0 \leq \alpha_i \leq C$. This is a quadratic programming (QP) problem solved using Platt's sequential minimal optimization algorithm [6]. Given the solutions $\hat{\alpha}$, the decision function of linear Adapt-SVM is:

$$f(x) = f'(x) + \sum_{i=1}^{N}\hat{\alpha}_i y_i x^T x_i \qquad (5)$$

The resulted classifier can be seen as adapted from the auxiliary classifier $f'(x)$ with additional support vectors from the primary set as $D_{SV} = \{(x_i, y_i) \in D | \alpha_i > 0\}$.

A key difference between the dual form of SVM [1] and that of Adapt-SVM in Eq.(4) is that that latter contains the extra term λ. A larger α_i is preferred in order to maximize L_D if $\lambda_i < 0$, i.e., if the auxiliary classifier f' misclassifies x_i because $\lambda_i = y_i f'(x_i)$, and vice versa. This is intuitive because α_i is the weight of each support vector $x_i \in D_{SV}$ in the adapted classifier $f(x)$ in Eq.(5). If the auxiliary classifier misclassifies x_i (or $sgn(f'(x_i)) \neq y_i$),

[1] This linear function is equivalent to $f(x) = x^T\beta + b$ if we add a constant term as an additional dimension of every data vector x.

the output of the adapted classifier $f(x_i)$ needs to be made different from $f'(x_i)$ such that it can classify x_i correctly (or $sgn(f(x_i)) = y_i$), which is realized by adding a support vector x_i with a large weight α_i.

2.2 The Kernel Model

To achieve non-linear decision boundaries, we project each data vector x into a feature vector $\phi(x)$ in a space of a higher or even infinite dimension, and learn $f(x) = \phi(x)^T\beta$ in the projected feature space. Based on the "kernel trick", we can replace $x_i^T x_j$ in objective function Eq.(4) by the kernel function $K(x_i, x_j) = \langle\phi(x_i), \phi(x_j)\rangle$ in order to learn $f(x)$, even if the feature map $\phi(\cdot)$ is unknown. The dual form of kernel Adapt-SVM can be written as:

$$L_D = \sum_{i=1}^{N}(1-\lambda_i)\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(x_i, x_j) \quad (6)$$

which is maximized under constraint $0 \le \alpha_i \le C$. The decision function of kernel Adapt-SVM is $f(x) = f'(x) + \sum_{i=1}^{N}\hat{\alpha}_i y_i K(x, x_i)$.

2.3 Discussion

Learning cost. Given the dual form in Eq.(4) and (6), Adapt-SVM does not introduce any extra variables or constraints into the optimization problem of SVM. The only extra term is λ_i. Based on its definition, computing all the λ_i involves computing $x_i^T x'_j$ (or $K(x_i, x'_j)$ in kernel model) for any $x_i \in D$ and $x_j \in D'$, so this is a one-time computational cost of $N \times N'$ evaluations of the dot product or kernel function. Also, since $\lambda_i = y_i f'(x_i)$, λ_i can be computed even when auxiliary data $\{x'_i\}$ are unavailable.

Cost factor C. Since α_i is the weight of each support vector $x_i \in D$ and $0 \le \alpha_i \le C$, the cost factor C is an upper bound on the contribution of x_i to the adapted classifier $f(x)$. Similarly, the cost factor of the auxiliary classifier C' is the upper bound for every $x'_i \in D'$. Thus, C and C' controls the relative contribution of the auxiliary and primary data on a *per instance* basis. In general, we expect $C > C'$ because the classifier is intended for data drawn from the same distribution as D. The appropriate ratio of C/C' should largely depend on the similarity between the two distributions. If the distribution of D and D' is similar, a smaller C/C' is preferred so that the model relies more on the auxiliary classifier, and vice versa.

Comparison with other models. The decision function $f(x)$ defined by Eq.(5) has the same form as an ensemble classifier: the first term is the auxiliary classifier $f'(x)$ trained from D', and the second term resembles a classifier learned from D. However, it is different from a genuine ensemble classifier that combines two base classifiers trained *independently* from D and D' [4, 10], because $\{\alpha_i\}$ are estimated under the influence of $f'(x)$ and their values are different from those estimated *entirely* from D. Moreover, our model is also different from an "aggregated" SVM classifier trained from $D \cup D'$ [3, 5, 11], because the latter treats both $\{\alpha_i\}$ and $\{\alpha'_i\}$ as parameters, because we treat $\{\alpha'_i\}$ as constants. With N' fewer parameters to estimate, our model is more efficient as N' is typically larger than N.

3 Selective Sampling for Adaptation

An question related to classifier adaptation is to identify which examples in the primary set we should choose to label in order to efficiently adapt existing classifiers. Intuitively, using *informative samples* would generate a better classifier than the one learned from *random samples*. While sample selection strategy has been studied in active learning [2, 9], the existing methods are mainly for building new classifiers. Here, we are interested in finding samples that provide *complementary* information to the auxiliary classifier and help its adaptation to the primary data in the *most efficient way*. In this section, we propose a selective sampling strategy for classifier adaptation with a theoretical justification based on the loss minimization principle.

Formally, let $\mathcal{P}$ be a pool of instances in the primary dataset, and D be a set of instances sampled from $\mathcal{P}$ for user labeling. Also, let $P(y|x)$ be the conditional probability of an example x in $\mathcal{P}$, and $P(x)$ be its marginal probability. Given a margin-based classifier f, such as SVM, the estimated loss of x is written as $L(yf(x))$. Thus, the expected risk of the decision function f is defined as $R(f) = E_x E_{y|x}(L(yf(x))$.

Suppose f^D is the classifier adapted from an existing classifier f' using examples in D. f^D is equivalent to f' when $D = \varnothing$, and it gradually deviates from f' when more examples are included in D. In this case, the optimal sample set is the one that can be used to minimize the expected risk of the adapted classifier f^D, or equivalently, the one that achieves the largest risk reduction from the auxiliary classifier f', i.e., $D_{opt} = \text{argmax}_D(R(f') - R(f^D))$.

Given the difficulty of computing the expected risk over the full distribution $P(x)$, it is more feasible to measure this risk over the data in the pool $\mathcal{P}$, i.e., $R(f) = \frac{1}{|\mathcal{P}|}\sum_{x\in\mathcal{P}} E_{y|x}(L(yf(x))$. Therefore, D_{opt} is given by:

$$D_{opt} = \underset{D}{\text{argmax}} \sum_{x\in\mathcal{P}} E_{y|x}\Big(L(yf'(x)) - L(yf^D(x))\Big) \quad (7)$$

Theoretically, maximizing Eq.(7) leads to the optimal sample set. In practice, however, this is prohibitively expensive because there are $2^{|\mathcal{P}|}$ possible choices for D and for each choice f^D needs to be re-trained to update the estimate of the expected loss. To provide a tractable solution, we use

the risk reduction over the sample set D to approximate the risk reduction over the whole collection $\mathcal{P}$. Moreover, we assume that the updated classifier f^D can always correctly predict the labels of any $x \in D$. Therefore, the risk of f^D as $\sum_D E_{y|x} L(yf^D(x))$ is so small that it can be dropped. With these two assumptions, Eq.(7) can be reduced to

$$D_{opt} = \operatorname*{argmax}_D \sum_{x \in D} E_{y|x} L(yf'(x)). \quad (8)$$

Eq.(8) eliminates the computationally intensive step of retraining f^D for every choice of D. To further simplify, we assume the samples are collected in a "greedy" manner. That is, we repeatedly choose x using the following criterion and add it to the existing sample set, i.e., $D = D \vee \{x\}$:

$$\operatorname*{argmax}_{x \in \mathcal{P} \setminus D} \sum_{y=\{-1,1\}} P(y|x) L(yf'(x)). \quad (9)$$

Finally, we need to estimate the conditional probability $P(y|x)$. Although the classifier outputs a confidence score $f(x)$ indicating the relevance of each x, due to the limited training examples, it is unlikely a reliable estimate of $P(y|x)$ even after it is normalized to $[0, 1]$. We develop two probability estimation models based on the hinge loss function of SVM, i.e., $L(yf(x)) = \max(0, 1 - yf(x))$.

Prior Model: We assume $P(y|x)$ is unrelated to the prediction made by f'; instead, we set it to the prior distribution of y in the auxiliary data D', i.e., $P(y = \pm 1|x) \approx P_{D'}(y = \pm 1)$, where $P_{D'}(y = \pm 1)$ is the ratio of positive (or negative) instances in D'. In this case, the sample selection criterion in Eq.(9) is written as:

$$\operatorname*{argmax}_{x \in \mathcal{P} \setminus D} \Big(P_{D'}(y = 1) \max(0, 1 - f'(x)) + P_{D'}(y = -1) \max(0, 1 + f'(x)) \Big) \quad (10)$$

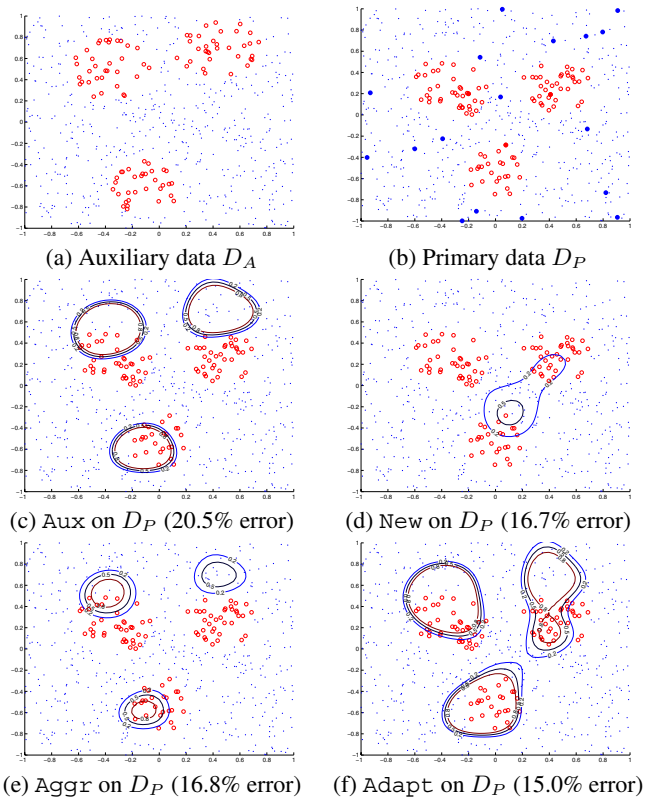
This is related to the "Random Labels" model in [2] which dissociates $p(y|x)$ with $f(x)$ and set it uniformly. Our model is more accurate when the two classes are unbalanced, typical in applications like concept or event detection.

Best Worst Model We approximate the expected loss with the smallest loss among all the possible labels, which implicitly assume that f' correctly predicts the label of x, i.e., $y = sgn(f'(x))$. Thus Eq.(9) can be written as:

$$\operatorname*{argmax}_{x \in \mathcal{P} \setminus D} \min_{y=-1,1} \max(0, 1 - yf'(x)) \quad (11)$$

$$= \operatorname*{argmax}_{x \in \mathcal{P} \setminus D} \max[0, 1 - sgn(f'(x))f'(x)] = \operatorname*{argmax}_{x \in \mathcal{P} \setminus D} |f'(x)|$$

This model chooses the *most ambiguous* examples, which are also the examples closest to the decision boundary of f'. This is similar to the uncertainty sampling strategy in active learning [2, 9].



(a) Auxiliary data D_A (b) Primary data D_P

(c) `Aux` on D_P (20.5% error) (d) `New` on D_P (16.7% error)

(e) `Aggr` on D_P (16.8% error) (f) `Adapt` on D_P (15.0% error)

Figure 1. The decision boundary of various classifiers on a synthetic primary dataset

4 Experiments

4.1 A Synthetic Dataset

To illustrate our model, we generate a synthetic auxiliary set D_A and a primary set D_P, from different distributions in a 2-d feature space. Each set has 100 positive and 500 negative data. The positive data in each set are generated from a Gaussian mixture model with 3 components, and the negative data are sampled uniformly outside the area of the positive data. For D_A, the 3 Gaussian components are centered at $(-0.4, 0.5)$, $(0.5, 0.7)$, and $(-0.1, -0.6)$, while for D_P their means shift to $(-0.4, 0.3)$, $(0.5, 0.3)$, and $(0, -0.65)$.

Figure 1 (a) and (b) shows the distribution of D_A and D_P, where small circles denote positive instances and dots denote negative instances. We assume *all* the instances in D_A are labeled, while only 20 instances are labeled in D_P, including 3 positive and 17 negative instances, shown as large dots in Figure 1 (b). To classify D_P, we use four classifiers trained on SVM using a RBF kernel $K(x_i, x_j) = e^{-\rho \|x_i - x_j\|^2}$ with $\rho = 5$. They are (1) an `Aux` classifier trained from the 600 labeled instances in D_A, (2) a `New` classifier trained from the 20 labeled instances in D_P, (3) an `Aggr` classifier trained from all the 620 labeled instances in

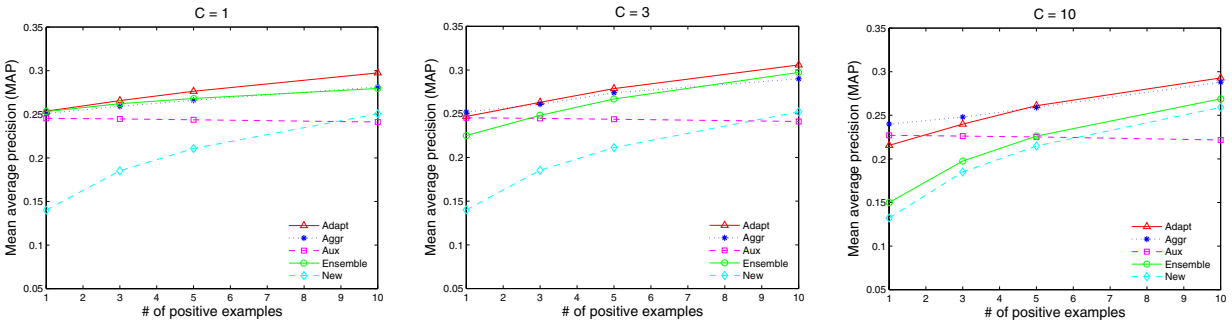


Figure 2. Comparison of five classifiers on video classification performance averaged across 360 concept-program settings (random sampling).

$D_A \cup D_P$; (4) an `Adapt` classifier adapted from `Aux` using the 20 instances in D_P based on Adapt-SVM. In the training of these classifier, we set $C' = 1$ for the instances in D_A and $C = 10$ for the instances in D_P to reflect the relative importance of the two datasets. We plot the decision boundary of the four classifiers on D_P in Figure 1 (c) to (f). The error rate of each classifier is shown below each figure. Not surprisingly, `Aux` and `Aggr` are biased towards the distribution of D_A, and unable to accurately classify D_P. `New` is unbiased, but has a large variance due to the limited training data. `Adapt` achieves the lowest error rate, and its decision boundary captures the distribution of the positive data in D_P more precisely than the other classifiers.

4.2 News Video Classification

We evaluate Adapt-SVM based on a benchmark video classification task TRECVID 2005 [7]. It contains 86-hour video footage of 13 TV news programs from 6 channels. All but one channels have two news programs, and the remaining one has two programs. The video in the collection has been segmented into 74,523 shots. Each shot is represented by one video frame as its "keyframe", and a keyframe is associated with a 273-d feature vector describing its color and texture properties. This collection comes with manually assigned labels with respect to 34 semantic concepts.

In each setting of the experiment, we pick one of the 34 concepts as the target concept, and one of the 13 programs as the primary program. The other program in the *same* channel as the primary program is treated as the auxiliary program. We train a SVM classifier for the target concept using *all* the shots in the target program (based on their feature vectors), and adapt it to the target program using a limited number of shots sampled from it. The adapted classifier is evaluated on the shots in the target program *except* those used as training examples. We convert the classifier output into a shot list ranked by their relevance scores, and measure the quality of this list using average precision (AP). We also use mean average precision (MAP) to average the AP scores in multiple settings. By varying the target concept and primary program, we have 34×13 concept-program settings. We remove the settings where the number of relevant shots is less than 10, which results in 360 settings.

We compare our adapted classifier `Adapt` with another 3 SVM classifiers: `Aux` trained from all the data in the auxiliary program, `New` trained from the labeled instances in the primary program, and `Aggr` trained from the labeled instances in both programs with different weights. We also include an `Ensemble` classifier which computes the relevance score of each instance as a weighted sum of the scores of `New` and `Aux`. All these classifiers are trained with RBF kernel with $\rho = 0.1$. To make `Adapt` comparable to `Aggr` and `Ensemble`, we ensure that the weight C' for auxiliary instance/classifier and the weight C for primary instance/classifier in these models are the same. We use $C' = 1$ in all the experiments while vary C from 1 to 10 in order to learn its impact on the performance.

Classification Accuracy: Figure 2 shows the performance of the 5 classifiers in terms of MAP across 360 concept-program settings against the number of positive training examples [2]. Random sampling is used in this experiment. On average, `Adapt` outperforms the other four classifiers in most of the cases. Only when $C = 10$ and training examples are scarce that `Adapt` performs slightly worse than `Aggr` and `Aux`. Knowing that `Adapt` is better than `Aggr` and `Ensemble` is especially encouraging, since they represent two widely used approaches to exploiting the knowledge of auxiliary data.

We find the performance of `Adapt` closely related to the choice of the cost factor C. When $C = 1$, `Adapt` relies more on the prior model and it behaves similar to `Aux`. This gives `Adapt` a "warm start" with very limited examples, but also makes it too conservative to fully utilize the incoming examples. When $C = 10$, `Adapt` relies more on the labeled examples. As a result, its performance suffers a bit

[2] Since most concepts are infrequent (the ratio of positive instances of a concept is 3.7% on average), the positive examples are more valuable for training a classifier and thus its number is a better indicator of the amount of information available. The number of negative examples used depends on the positive-negative ratio in each program.

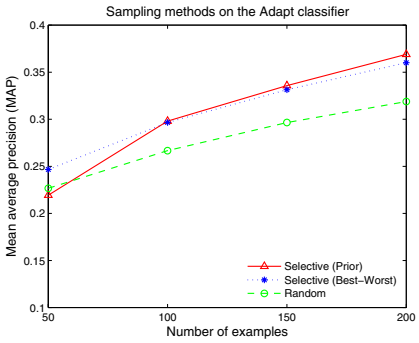


Figure 3. Sampling methods on `Adapt`

initially due to a high variance, but ends up higher when more examples become available. Since it is not legitimate to choose C based on the performance on the test set, in practice we need to choose C by cross-validation.

Efficiency: We compare the training time of `Adapt` with that of `New` and `Aggr`. `Aux` is trained "offline" and thus its cost does not matter, and the cost of `Ensemble` is equal to that of `New`. The total training time in minutes for all the settings, with $C = 3$ and 10 positive examples (approx. 164 total examples), is 17.4 for `New` or `Ensemble`, 20.1 for `Adapt`, and 271.9 for `Aggr`. This shows that adapting a classifier is only slightly more expensive than training one from the new examples, but an order of magnitude less expensive than training a "big" classifier using aggregated data. This makes the proposed method applicable to interactive scenarios and/or large-scale problems.

Selective Sampling: We compare two selective sampling strategies described in Section 3, `Prior` and `Best-Worst`, with random sampling. Since they select different samples from the data which cause the remaining to be different, we evaluate them based on *all* the data in the target program. Figure 3 shows the MAP of all the settings achieved by `Adapt` with the three sampling methods. Both selective sampling methods are considerably better than random sampling, showing that using more informative samples does help classifier adaptation. Between them, `Prior` is slightly better than `Best-Worst` except when there are only 50 samples.

5 Related Work and Discussion

The problem of classifier adaptation has been studied in many related areas. It is closely related to the problem of drifting concept detection in mining of streaming data, which is solved either by constructing an ensemble classifier combining a set of base classifiers trained from different chunks of the data stream (e.g., [10], [4]), or by training a single classifier using aggregated (and weighted) instances sampled from the data stream (e.g., [3]). Our work also belongs to transfer learning, which aims to apply knowledge learned in one or more tasks to improve the performance on a related task. Many methods for transfer learning [5, 11] take the "aggregation" approach, which incorporate the labeled data of related tasks into the current training set in order to build better classifiers. Our method is fundamentally different since it directly modifies an existing model to fit the new data, which avoids the cost of training over aggregated data. Our method can be also used as an incremental learning algorithm, and it is more efficient than existing incremental algorithms (e.g., [8]) that involve training over (at least) part of the previous data.

References

[1] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[2] C. Campbell, N. Cristianini, and A. Smola. Query learning with large margin classifiers. In *Proc. of ICML*, 2000.

[3] R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. In *Proc. of ICML*, pages 487–494, 2000.

[4] J. Z. Kolter and M. A. Maloof. Dynamic weighted majority: A new ensemble method for tracking concept drift. In *Proc. of ICDM*, page 123, 2003.

[5] X. Liao, Y. Xue, and L. Carin. Logistic regression with an auxiliary data source. In *Proc. of ICML*, pages 505–512, 2005.

[6] J. Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods: support vector learning*, 1999.

[7] A. Smeaton and P. Over. Trecvid: Benchmarking the effectiveness of infomration retrieval tasks on digital video. In *Proc. of CIVR*, 2003.

[8] N. Syed, H. Liu, and K. Sung. Incremental learning with support vector machines. In *In Workshop on Support Vector Machines, at the IJCAI*, 1999.

[9] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In *Proc. of ICML*, pages 999–1006, 2000.

[10] H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proc. of SIGKDD*, pages 226–235, 2003.

[11] P. Wu and T. G. Dietterich. Improving SVM accuracy by training on auxiliary data sources. In *Proc. of ICML*, page 110, 2004.