A General Framework for Classifier Adaptation and its Applications in Multimedia

Jun Yang

CMU-LTI-09-007

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:

Alexander G. Hauptmann (Chair)
Christos Faloutsos
Jie Yang
Shih-Fu Chang (Columbia University)

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
In Language and Information Technologies

Abstract

For the analysis and retrieval of multimedia data, machine learning techniques have been extensively applied to build models that map various feature vectors of the data into semantic labels. As multimedia data come from a wide variety of domains (e.g., genres, sources), each having its distinctive data characteristics, models trained from one domain do not usually generalize well to other domains. For example, the performance of semantic concept detectors trained from news video drops 60-70% when they are applied on documentary video. Meanwhile, it is prohibitively expensive to build new models for each and every domain due to the high cost for labeling training examples. Therefore, techniques for adapting models across different domains are desirable for better performance and reduced human cost.

In this thesis, we investigate a generic adaptation problem in multimedia and other areas, which is to adapt supervised classifiers trained from one or more *source domains* to a new classifier for a *target domain* that has only limited labeled examples. The foundation of our work is a general framework for *function-level classifier adaptation* based on the regularized loss minimization principle. Fundamentally different from existing adaptation techniques, this framework adapts a classifier by directly modifying its decision function rather than re-training over the data in source domains, making it highly efficient and applicable to any type of classifier. Under this framework, one can derive concrete adaptation algorithms by plugging-in any loss and regularization functions, among which we elaborate on adaptive support vector machines (a-SVM) and adaptive kernel logistic regression (a-KLR). We further extend this framework for multi-classifier adaptation, namely adapting multiple existing classifiers into a classifier for the target domain, in a way that the contributions of these existing classifiers are automatically determined. We evaluate the proposed approaches in cross-domain

semantic concept detection based on TRECVID corpora. The results show that our approaches outperform existing (adaptation and non-adaptation) methods in terms of accuracy and/or efficiency, and adaptation from multiple classifiers offers further benefits. We also demonstrate the effectiveness of our approaches in adapting classifiers of text documents and of EEG data.

We then focus on improving the *cost-efficiency* of adaptation by selecting and prioritizing adaptation tasks involving multiple classifiers. We approach this problem by first conducting a comprehensive analysis of the generalizability of concept classifiers, which is related to the cost-efficiency of adapting a classifier. This analysis reveals strong correlations between generalizability and various meta-features of a classifier, ranging from model structure to the distribution of its output. We show that generalizability can be predicted quantitatively from these model meta-features using regression models. Based on the predictions of generalizability, we propose several *selective adaptation* methods for selecting the classifiers to be adapted and allocating their training examples such that they achieve higher overall post-adaptation performance than equally adapting every classifier.

Acknowledgement

As a Ph.D. student at Carnegie Mellon University, I have the privilege to work with some of the finest researchers on computer science, which makes this thesis a reality. I owe a great debt to these people for their guidance, support, and kindness.

First of all, I would like to thank my advisor, Alex Hauptmann, for his great guidance and support over the past five years. I have learned not only cutting-edge research from a true expert in the area of multimedia, but also his down-to-earth working ethic. I am especially thankful for the freedom I was given to explore various research topics and to collaborate with different people outside the group, which cultivates my independent thinking and broaden my understanding of related areas. It has been a true pleasure to spend my Ph.D. under Alex's supervision.

I would also like to thank my other thesis committee members, Christos Faloutsos, Jie Yang, and Shih-fu Chang, for their assistance and feedbacks at various stages of this thesis. It is their comments and suggestions that make this thesis more accurate, more complete and certainly more pleasant to read.

I am greatly indebted to my outstanding colleagues and collaborators. I have worked closely with colleagues in Informedia group, Mike Christel, Rong Yan, Weihao Lin, Robert Chen, Robert Baron, Bryan Maher, Datong Chen, with whom I had many insightful discussions and joint publications. I also benefit greatly from collaborations with people outside the group, including Yan Liu and Eric Xing from CMU, and Xiao Wu, Yu-Gang Jiang, and Chong-Wah Ngo from City University of Hong Kong. My gratitude also goes to IBM T.J. Watson Research Center, which supported my study financially through its renowned Ph.D. Graduate Fellowship program and invited me for a wonderful internship, during which I had the honor to work with experts in the area, including my mentor Rong Yan, my manager Murray Campbell and John Smith,

as well as Apostol Natsev, Lexing Xie, and Jelena Tesic.

Moreover, I have been fortunate to know many good friends and fellow students at CMU. Their names include Wen Wu, Jie Lu, Fan Li, Luo Si, Joy Zhang, Jerry Zhu, Yanjun Qi, Hua Zhong, Juchang Hua, Yan Li, Hong Yan, Qifa Ke, Zhenzhen Kou, Chenyu Wu, Bin Zhao, Jimeng Sun, and many more. It is their support and friendship that makes my Ph.D. life more pleasant and enjoyable.

Last but not the least, I would like to thank my wife Ying and my parents for their love and support, without which I would not have survived the long journey of Ph.D.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The explosive growth of multimedia data makes their analysis, classification, and retrieval a critical problem in both research and industry. Machine learning techniques are playing an increasingly important role in this area, where models are built for tasks varying from classifying images into categories, detecting semantic concepts in video shots, to matching image and video data with user queries. Since multimedia data come from a variety of data domains, there is a need to generalize and adapt models trained from one domain to other domains. Compared with building new models for every domain, adapting existing models is beneficial in terms of reduced manual effort for labeling training data and better performance, but also challenging given that data distribution changes arbitrarily across domains. This thesis is dedicated to developing efficient and principled approaches to adapting models across multimedia domains. We will discuss the motivation, goal, and challenges of this research, and briefly overview our approaches and the key contributions.

1.1 Motivation and Task Definition

Adapting models for multimedia data is necessary because such models in general have *poor generalizability* across different domains, i.e., a model trained from one domain performs much worse on the data from the other domains than on the data from the same domain. This is mainly due to the change of data distribution across different domains. Multimedia data are represented by feature vectors, or data points in feature

space, and the distribution of these data points may change from one domain to another. For example, if color histogram is the feature, the data distribution of C-Span footage on political debates is very different from that of the footage in Discovery channel. An assumption critical to the success of machine learning techniques is that the training and test data come from the same distribution. Therefore, the shift of data distribution between two domains inevitably causes models trained from one domain to fail to generalize to another.

An example in this case is semantic concept detection, where supervised classifiers are built to distinguish whether a video shot is relevant or irrelevant to certain semantic concepts. For example, a *Studio* classifier in news video distinguishes video shots containing studio scene from those that do not contain studio scene. Such a classifier is trained from labeled video shots from a certain news video channel. As shown in Figure 1.1, the studio scenes from NBC and NTDTV channel look different in terms of background, room setting, and the number of people. Therefore, the performance of the *Studio* classifier trained from NBC drops from 0.83 AP (average precision) on NBC to only 0.24 on NTDTV, while the performance of NTDTV's classifier drops from 0.98 AP on NTDTV to 0.25 on NBC.

A straightforward way to address the generalizability problem is to build new models for every domain. This is however a costly approach given that labeling multimedia data is time-consuming and the size of training data needed to build reliable models is large. According to the statistics on the TRECVID 2007 collaborative annotation [78], roughly 215 intense man-hours were spent on labeling 50-hour video w.r.t 36 concepts in order to build models for another 50-hour video. In reality, the footage in many video archives (e.g., YouTube.com) can easily exceed thousands of hours, and a reasonably comprehensive semantic ontology contains hundreds of semantic concepts [2]. Building models for every domain in real-world video archives is prohibitively expensive. On the other hand, existing models trained from other domains provide valuable information to similar tasks in a new domain, and should not be thrown away. Exploiting the knowledge in these out-of-domain models reduces the number of labeled examples otherwise needed for building new models from scratch, and consequently, the human effort for labeling them and the cost for training models over them. So compared with building news models, adapting existing models may require *fewer*

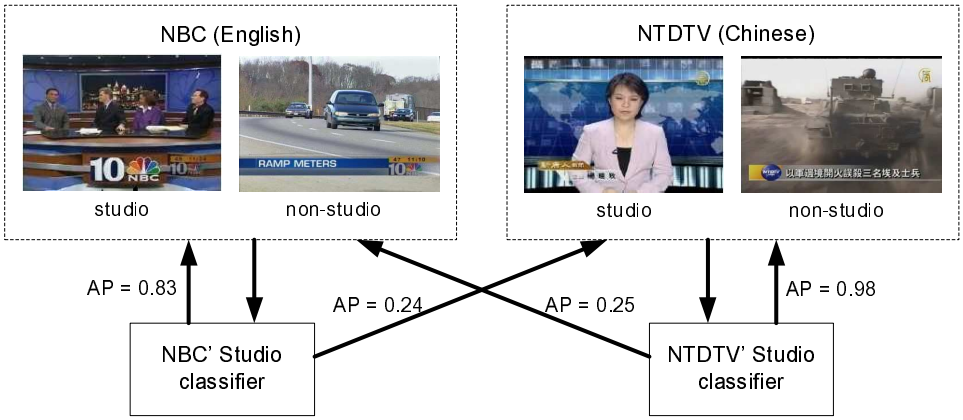


Figure 1.1: The performance as average precision (AP) of "Studio" classifiers trained from video in NBC or NTDTV news channel. There is a significant decline in performance when a classifier trained from one channel is applied to another channel.

labeled examples to achieve the same performance, or achieve *higher performance* using the same number of labeled examples.

Given the poor generalizability of existing models and the cost for building new ones, adaptation of existing models becomes a solution providing a good tradeoff between performance and efficiency. However, this problem has been *overlooked* and *understudied* in the multimedia research community. While there has been related work on adapting concept detectors across correlated concepts [69, 112, 77] and adapting retrieval models across query classes [111, 60], very little effort [55] has been devoted to the problem of adaptation across data domains. A systematic approach to this cross-domain model adaptation problem is an important and challenging research topic.

Many learning tasks in multimedia can be formulated as *a classification problem* of mapping feature vectors $\mathbf{x}$ of some type to semantic, categorical labels y. For example, semantic concept detection is about mapping low-level features of video shots into a binary label indicating the presence (or absence) of a semantic concept. A multimedia retrieval model maps a set of similarity scores between an image (or video shot) and a query into a label indicating relevance or irrelevance. Other tasks can be formulated similarly. Therefore, *supervised classifiers* which learn such feature-label mapping $f : \mathbf{x} \rightarrow y$ are the most frequently used type of models for multimedia. The goal of this thesis is to develop efficient and principled approaches to adapting supervised classifiers across different domains, and use them to solve adaptation problems in multimedia and other areas. Formally, we define domain and classifier adaptation as:

Definition 1. *A **domain** is a set of multimedia data generated from the same data distribution $p(\mathbf{x})$ and class-conditional distribution $p(y|\mathbf{x})$. Concretely, a domain is described by image or video data belonging to a certain genre, or created by a specific producer, etc. For example, cartoon images and photographs are two image domains, news video and documentaries are two video domains, and news video from different channels can be also viewed as from different domains.*

Definition 2. ***Cross-domain classifier adaptation** is to adapt supervised classifiers for a given task trained from one or more **source domains** into a new classifier that works well on a different **target domain**. We call the existing classifiers **source classifiers** and the new classifier the **target classifier**. We further assume that (1) the source domains are related to the target domain in the sense that source classifiers have better-than-random performance on the target domain; (2) only a limited number of labeled data are available in the target domain, while the labeled data in the source domains are relatively sufficient.*

The definition of domain implies that different domains may have *different distributions*. In some cases, the data distribution $p(\mathbf{x})$ changes across domains, while the class-conditional $p(y|\mathbf{x})$ stays the same. Even in this case, classifiers trained from one domain are unlikely to capture the true $p(y|\mathbf{x})$ due to the bias in the distribution of its training data. For example, we train an *Anchor* classifier from CCVT news video, where anchors always appear in studio setting, and apply it to CNN news video, where anchors often appear outdoors. The standard of judging whether a video shot has an anchor is always the same, which means $p(y|\mathbf{x})$ remains the same, but the classifier is probably unable recognize outdoor anchors as it has never "seen" them before. In other cases, both $p(\mathbf{x})$ and $p(y|\mathbf{x})$ change across domains. For example, in a retrieval model that maps various similarity scores into a relevance label, the influence of a face similarity score (as a component of $\mathbf{x}$) to the label y changes from people-related queries to object-related queries. Since both $p(\mathbf{x})$ and $p(y|\mathbf{x})$ may change, we make no assumption as to *whether* and *how* the distribution changes between the source and target domain. This does not mean the two domains are totally irrelevant. Instead, we assume that the source classifier is at least somewhat helpful to the task in the target domain.

1.2 Research Challenges

Model adaptation has been studied in different areas and a number of techniques have been proposed. In machine learning, inductive transfer and multi-task learning methods apply knowledge learned from one or more tasks to help solve related tasks [33, 20, 65, 67, 105, 119]. In data mining, techniques have been developed to recognize drifting concepts from streaming data [61, 92, 102]. Specialized methods are also available for adapting language models and parsers in natural language processing (NLP) [9, 53, 82] and adapting acoustic models in speech recognition [45, 64]. Nevertheless, the state-of-the-art techniques for adaptation are unable to meet all the challenges of adapting models of multimedia data due to their unique characteristics. The challenges include:

- *Modeling distribution changes of multimedia data is technically infeasible.* As mentioned, the data distribution $p(\mathbf{x})$ almost always changes across multimedia domains, and the class-conditional $p(y|\mathbf{x})$ often changes too. Changes of $p(\mathbf{x})$ are difficult to capture because there are no generic and accurate (generative) models for the distribution of multimedia data, which can be represented by many different features. Modeling changes of $p(y|\mathbf{x})$ is even harder because, given our problem setting, most of the data labels y are unavailable on the target domain. Indeed, modeling distribution change is a harder problem than classifying the data, and solving it would make the solution to classification problems trivial. Therefore, existing methods [13, 14, 93, 118] that rely on the knowledge or assumptions about whether and how distribution changes are not suitable for adaptation problems in multimedia. We need adaptation approaches requiring no such knowledge or assumptions.

- *Training on raw data from source domains is costly.* Models for multimedia data can be expensive to build because the data features are typically high-dimensional and real-valued vectors, and the learning algorithms (e.g, SVM) have super-linear training cost with respect to the data size. Training the target classifier over data aggregated from all domains, a widely used approach in previous work [32, 61, 65, 105], is inefficient given the large size of source data. Moreover, the raw data in source domains can be unavailable or inaccessible due to copyright or privacy issues, a typical example of which is surveillance video. An efficient

approach is desired to exploit more compact representations of domain knowledge than the raw data.

- *There exist multiple source domains (classifiers) with different utility.* Due to the diversity of multimedia data, there are sometimes more than just one source domain (e.g., video from different news channels), and therefore more than one source classifier available for adaptation. These source classifiers are of different levels of utility to the classification task in the target domain. There is a need for an adaptation approach that takes advantage of all the source classifiers. In this approach, the contribution of these source classifiers needs to be weighted to reflect their utility to the target domain. To our knowledge, there has not been a general method in the literature that can adapt multiple classifiers into one classifier.

- *The cost-efficiency of adaptation varies between classifiers.* Some classifiers generalize better across domains than other classifiers, and therefore, the improvement of their performance as the result of adaptation is not as great as those that are less generalizable. This shows the cost-efficiency of adaptation, i.e., the performance improvement against the number of training examples, is different cross classifiers. To maximize the overall cost-efficiency of adaptation, one needs to carefully select the classifiers to be adapted and prioritize them in terms of the training examples each receives, rather than adapting every classifier equally. Modeling and predicting classifiers' generalizability to new domains, a challenging research problem by itself, is important for determining the cost-efficiency in adapting each classifier.

The challenges discussed above are not specific to only multimedia data. The challenges on efficiency, on accommodating multiple source domains, and on analyzing classifiers' generalizability, are general ones faced by any systematic approach to classifier adaptation. We hope that by addressing these challenges, the adaptation techniques developed in this thesis are not only applicable to problems in multimedia but general enough to be used for problems in other areas.

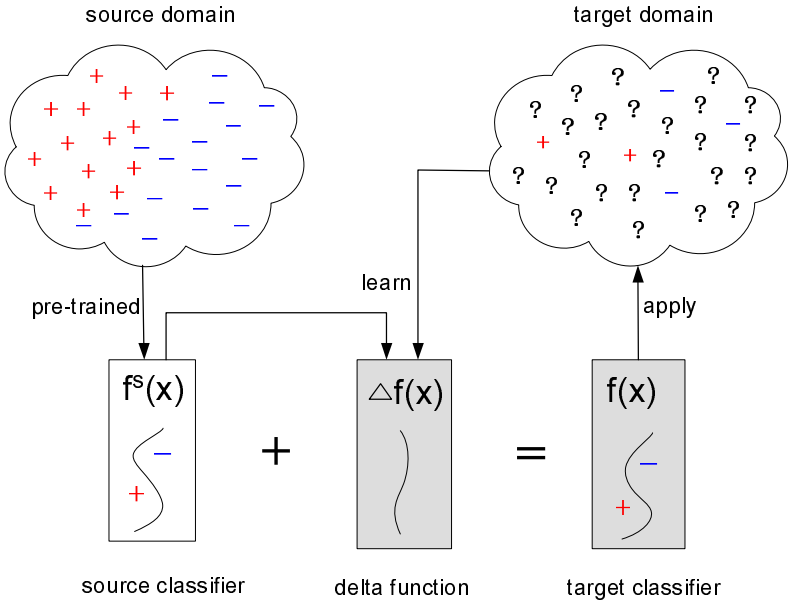


Figure 1.2: The basic framework for function-level classifier adaptation. Red '+' denotes positive instances, blue '-' denotes negative instances, and '?' denotes unlabeled instances. The components in shaded rectangles need to be learned.

1.3 An Overview of Our Approach

To address the challenges raised above, we organize our research in this thesis around three correlated problems:

1. *How to adapt a classifier from a source domain to a target domain in an efficient and principled approach?*

2. *How to adapt multiple source classifiers into a target classifier in a way that reflects their utility?*

3. *How to select and prioritize adaptation tasks based on predictions of classifiers' generalizability, with the goal of maximizing the cost-efficiency of adaptation?*

The foundation of our work is a general framework for *function-level classifier adaptation* based on loss minimization principle, which is illustrated in Figure 1.2. In this framework, a source classifier represented by its decision function $f^s(\mathbf{x})$ is adapted to a new classifier $f(\mathbf{x})$ for the target domain by adding a "delta function" $\Delta f(\mathbf{x})$ to $f^s(\mathbf{x})$.

The delta function $\Delta f(\mathbf{x})$ is learned based on the labeled examples in the target domain and the source classifier $f^s(\mathbf{x})$, using a general objective function that attempts to achieve two goals: (1) minimizing the classification loss (error) of the target classifier $f(\mathbf{x})$ on the labeled examples, and (2) minimizing a regularizer that penalizes the distance between $f(\mathbf{x})$ and $f^s(\mathbf{x})$ in the function space. In other words, it makes only the smallest changes to $f^s(\mathbf{x})$ that are necessary for $f(\mathbf{x})$ to correctly classify the labeled examples. In practice, this general framework can be "instantiated" into concrete adaptation algorithms by plugging in specific loss and regularization functions into its objective function. We elaborate on two of such algorithms, namely *adaptive support vector machines (a-SVM)* and *adaptive kernel logistic regression (a-KLR)*, where the latter can be also derived from a probabilistic perspective. This addresses the problem of adapting a classifier from a source domain to a target domain.

As a fundamental difference from existing approaches which utilize raw data from the source domain, our framework utilizes the source classifier as a *summary* of the knowledge distilled from the raw data. Using such a compact representation of domain knowledge has important implications on the efficiency and applicability of our approach. First, the adaptation process is very efficient due to the freedom from training over typically a large amount of labeled source data. The cost of adapting a classifier based on limited target data is substantially lower than training over all the data, because the training cost of algorithms like SVM is super-linearly related to the data size. Second, avoiding using raw data in source domains makes our approach applicable to tasks where such data are unavailable or inaccessible, typical in applications involving copyright-protected or privacy-related data such as surveillance video. Last but not the least, our approach can be used to adapt a classifier of *any type* as long as it can be represented by a decision function $f(\mathbf{x})$, or more precisely, a "black-box" that outputs a value for any input data point $\mathbf{x}$.

While the basic framework can be used for adapting one classifier into another, in practice there is often a need for adapting multiple classifiers into one target classifier. To accommodate such need, we extend the basic framework into a more sophisticated one for multi-classifier adaptation based on (still) regularized loss minimization principle. As shown in Figure 1.3, in this extended framework the target classifier $f(\mathbf{x})$ is the sum of the delta function $\Delta f(\mathbf{x})$ and a *weighted ensemble* of multiple source classifiers

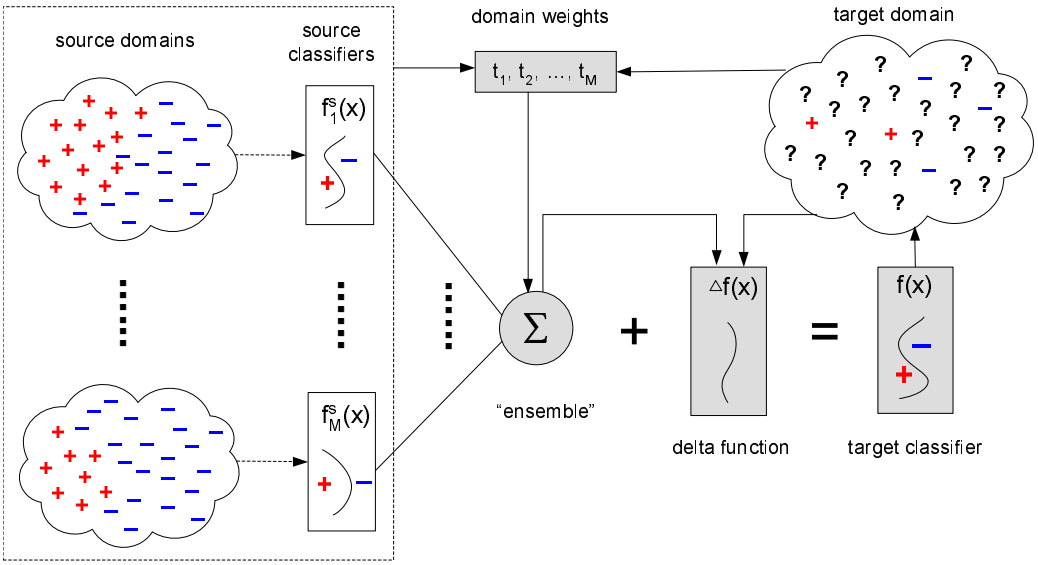


Figure 1.3: The extended adaptation framework with multiple source domains and domain analysis. The components in shaded rectangles need to be learned.

$f_1^s(\mathbf{x}), ..., f_M^s(\mathbf{x})$, where the weights $\mathbf{t} = \{t_1, .., t_M\}$ reflect the utility of these source classifiers w.r.t the target domain. Both $\Delta f(\mathbf{x})$ and $\mathbf{t}$ are learned simultaneously using the original objective function augmented with a new regularizer on $\|\mathbf{t}\|^2$. From this extended framework, we derive *multi-adaptive support vector machines* (ma-SVM) as a counterpart of a-SVM for multi-classifier adaptation. We show that the weights $\mathbf{t}$ learned automatically in ma-SVM indeed reflect the utility of different source classifiers. This addresses the second problem on adapting multiple classifiers into one classifier.

While the first two problems focus on the methods for classifier adaptation, the third one focuses on improving the *cost-efficiency* of adaptation. Specifically, we investigate the problem of selecting and prioritizing adaptation tasks involving multiple classifiers such that their overall performance is maximized after adaptation using a fixed number of training examples. We approach this problem by conducting an empirical study of concept classifiers' generalizability, because generalizability affects the potential room of improvement from adapting a classifier. The study reveals strong correlations between the generalizability of a classifier and its various meta-features, including the ratio of positive training data, model complexity, and the distribution of its output. We build a regression-based *generalizability model* capable of predicting

how well a classifier generalizes to (unlabeled) new domains from these meta-features. Based on this generalizability model, we propose several *selective adaptation* methods, which select and prioritize adaptation tasks such that the less-generalizable classifiers are adapted with higher priority than the more-generalizable ones.

Throughout the thesis, the proposed approaches are evaluated on cross-domain semantic concept detection, the task of adapting concept classifiers trained from one domain to another domain, based on TRECVID video corpora [89]. The experiments show that the classifiers adapted using our approaches consistently outperform by significant margins both the original classifiers and new classifiers trained from scratch using exclusively the target data, which demonstrate the benefits of classifier adaptation. It is also shown that the selective adaptation methods achieve higher overall performance than adapting every classifier equally using the same amount of training examples. We also apply our approaches to adaptation problems in other areas, such as adapting classifiers of text documents and classifiers of EEG (brain signal) data, which further confirms the effectiveness of our approaches.

1.4 Major Contributions

We summarize the major contributions of this thesis below:

- This thesis provides **a comprehensive study on the issue of generalizability in semantic concept detection**, a key learning task by itself and the foundation of many other tasks in multimedia. While previous work evaluates the performance of concept classifiers on its training domain, we focus on how well their performance *generalizes* to new domains. Based on experiments on TRECVID video corpora, this study reveals that in general concept classifiers generalize poorly across different domains, whether the evaluated domains are different news video channels or different video types, with an average relative performance decline of 60% to 70% in our datasets. The poor generalizability, as the study further reveals, is because the majority of concept classifiers (trained by SVM) are unable to summarize the data and resort to a nearest-neighbor type of approach in classification. This is, to our knowledge, the first comprehensive study on the generalizability of semantic concept classifiers, and it emphasizes

the need for classifier adaptation in multimedia.

- This thesis presents **a survey of state-of-the-art techniques related to model adaptation in several disciplines**, including data mining, machine learning, and multimedia. The survey shows that classifier adaptation across domains is an important but overlooked topic in multimedia, although there has been previous work on concept detection using the correlation between semantic concepts and the adaptation of retrieval models. It also reviews techniques in transfer learning, multi-task learning, incremental learning, sample-bias correction, and data mining of drifting concept, and discusses their connections and differences with the adaptation techniques proposed in this thesis. Compared with existing techniques which carry out adaptation at the data level, parameter level, and/or representation level, our approach is unique and different in that it is the only one we know that directly modifies the decision function of supervised classifiers.

- This thesis proposes **a general framework for function-level classifier adaptation**. It differs from existing adaptation methods in that it directly modifies a classifier's decision function based on the regularized loss minimization principle. The rationale underlying this framework can be interpreted as to make only *minimal but necessary* modifications to a classifier to allow it to correctly classify labeled training examples in the target domain. Because the adaptation is achieved through modifications of the decision function instead of re-training over the "old data" (i.e., raw data in source domains), this framework is highly efficient, capable of adapting any type of classifier that can be represented with decision functions, and also applicable to tasks where the old data are not available. From this framework one can derive virtually an infinite number of concrete adaptation algorithms by plugging in different loss and regularization functions into its objective function. We elaborate on two of such algorithms, adaptive SVM and adaptive KLR, the latter of which provides a probabilistic perspective to this framework. Experiments show that the proposed approach outperforms other alternatives in performance and/or efficiency in cross-domain semantic concept detection as well as other tasks such as the adaptation of text classifiers.

- Based on this basic framework, this thesis proposes **an extended framework for multi-classifier adaptation**, which adapts multiple source classifiers into one target classifier. This is, to our knowledge, the first method for many-to-one adaptation. Since the source classifiers are not equally useful, weights are introduced into the framework to control their contribution. These weights are learned automatically to reflect the utility of each source classifier w.r.t the target domain. From this extended framework, we derive multi-adaptive support vector machine (ma-SVM) as a counterpart of a-SVM for multi-classifier adaptation. Experiments show the benefit of multi-classifier adaptation over single-classifier adaptation in cross-domain concept detection.

- We have proposed several **selective adaptation** methods to improve the overall cost-efficiency of adaptation. Based on the knowledge about the generalizability and/or learnability of each classifier, these methods select and prioritize adaptation tasks involving multiple classifiers, such that their overall performance is maximized after adaptation using a fixed number of training examples. To support selective adaptation, we have explored **an empirical approach to predicting the generalizability of semantic concept classifiers**. Our study has has revealed strong correlations between the generalizability of a classifier and its various meta-features, including the ratio of positive training data, model structure, and the distribution of its output. We have built a regression model to predict the generalizability of a classifier based on these meta-features. Experiments have shown that this model is capable of accurately predicting the performance of a concept classifier on a new, unlabeled domain other than its training domain.

The remainder of the proposal is organized as follows. Chapter 2 surveys related works on model adaptation in the area of multimedia, machine learning, data mining, and others. Chapter 3 studies the issue of generalizability in semantic concept detection, a representative learning task in multimedia and the benchmark task for our proposed approaches. Chapter 4 describes the general framework for function-level classifier adaptation, and a-SVM and a-KLR as two concrete algorithms derived from this framework. Chapter 5 further extends this framework to enable adapting multiple

classifiers into one classifier in a way that their utility is properly modeled. Chapter 6 investigates the generalizability of semantic concept classifiers as well as how to predict generalizability and use it for selective adaptation. Chapter 7 explores the application of the proposed approaches in adaptation tasks outside the multimedia area, such as cross-domain text categorization. Finally, Chapter 8 summarizes the thesis and discusses further directions.

Chapter 2

Literature Review

In this section, we review previous work related to cross-domain model adaptation in different research areas. We first discuss the approaches to adaptation problems in multimedia, for which the proposed techniques are intended. This is followed by a review of related works on transfer and multi-task learning, sample bias correction, and incremental learning in the machine learning area, concept drift detection in data mining, and adaptation techniques for specific tasks such as speech recognition and natural language processing. In our review, we will also discuss the connections and differences between these existing techniques and our approach.

2.1 Adaptation in Multimedia

The analysis, classification, and retrieval of multimedia data heavily involve the training of various machine learning models, particularly supervised classifiers. In image classification and semantic concept detection [4, 25, 49, 77, 89, 91, 95, 101, 106, 112], supervised classifiers are used to map features representing images or video shots into labels indicating the categories or concepts they belong to. These classifiers are trained from manually labeled data using classification algorithms such as support vector machines (SVM) and neural network. Multimedia retrieval models [4, 60, 104, 111, 110] are used to combine the relevance scores computed from multiple knowledge sources, such as keyword similarity and image similarity. The combination weights are often estimated using classifiers such as logistic regression, which are trained from previous

queries and their (labeled) results [60, 111, 110]. As suggested by Natsev et al. [71], both semantic concept detection (classification) and retrieval can be viewed as the same problem, i.e., classifying multimedia data as relevant or irrelevant w.r.t categories or queries. While supervised classifiers are perhaps the most widely used learning models in multimedia, there are many other types of models, such as models for annotating images with keywords [11, 15, 47, 54], for labeling video shots with person names [85, 114] and with locations [113].

In terms of models' performance and labeling cost, there are significant benefits in adapting existing models across different domains. However, the problem of model adaptation is in general overlooked or at best understudied, partly due to the evaluation method used in multimedia. Models for multimedia data are often evaluated with the training and test data coming from the *same* domain, which eliminates the need for model adaptation. For example, image retrieval and image annotation models [11, 25, 43] are often trained and tested on selected subsets of the Corel image collection [1], which contains high-quality photos. Most semantic concept detection and video search models [4, 49, 71, 77, 91] are trained and tested on the TRECVID collection [89] of a particular year, which contains video footage of either documentary or broadcast news video. While this evaluation setting is legitimate as long as the training and test data do not overlap, it does not represent the real-world scenarios where we have to deal with data from multiple domains simultaneously. The need for adaptation becomes apparent if we want to reuse existing models while shifting from one domain to another.

Previous work along several directions can be related to model adaptation in multimedia. One of them is semantic concept detection based on the correlations between multiple concepts. Methods for this problem learn classifiers of multiple concepts simultaneously and allow these models to influence each other, so that they perform better than classifiers learned independently for individual concepts. For example, Naphade et al. [69] explicitly modeled the linkages between concepts via a Bayesian network that implicitly offered ontology semantics in a video collection. Amir et al. [4] 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. Yan et al. [112] proposed a series of probabilistic graphical models to mine the relationships

between concepts. In addition, Qi et al. [77] proposed correlative multi-label (CML) framework, and Chen and Hauptmann [28] proposed multi-concept discriminative random field (MDRF) to automatically identify concept correlations and learn concept classifiers simultaneously. These last two methods are similar in spirit because they both capture the correlations between concepts by modifying the regularization term in the objective function of the respective learning algorithm (support vector machines in [77] or logistic regression in [28]). Additionally, this problem has also been studied by Snoek et al. [91] and by Wu et al. [109]. However, their methods focus on the correlation between models for different semantic concepts, while the focus of this thesis is on the adaptation of models across data domains.

Other related research is on adapting multimedia retrieval models towards new queries, either with or without users' feedback on retrieval results. Relevance feedback methods for content-based image retrieval [29, 84] update the initial query representations and/or distance metrics in the feature space based on user feedbacks in order to improve retrieval results. Since query point and distance metric are integral part of a retrieval model, relevance feedback can be understand as the adaptation of the retrieval model. For video retrieval, Yan [110] proposed to construct the retrieval model for a new query as a mixture of existing models of several predefined "query classes", and update the model based on users' implicit and explicit feedbacks to better reflect the characteristics of the new query. This work is extended by Kennedy et al. [60] to allow query classes to be automatically discovered. These methods are designed for specific feature representation and retrieval algorithms and not applicable to general adaptation problems.

The work on these two problems is mainly about the adaptation or knowledge transfer between semantic concepts and/or user queries. The problem of adapting between two data domains has been overlooked, although due to the diversity of multimedia data the problem is by no means less important. To our knowledge, the only work besides ours on this problem has been that from the Digital Video and Multimedia (DVMM) Lab at Columbia University. Jiang et al. [55] introduced cross-domain SVM (CDSVM) as an algorithm for adapting an existing SVM model to a new domain based on a (small) number of labeled examples in it, and later used this algorithm in the concept detection task in TRECVID 2007 [23]. CDSVM shares the same goal

as our proposed algorithm, adaptive SVM (a-SVM), but the two algorithms are very different. CDSVM essentially adds the support vectors of the original model, weighted by their similarity to the distribution of the new domain, as additional training data besides those from the new domain. This means it re-trains the model over a merged set of labeled data from both old and new domain. In comparison, a-SVM directly modifies the decision function of the existing model and avoids using any data from the old domain, which implies better efficiency and applicability. Their difference will be discussed in detailed in Section 4.3.2.

To summarize, the issue of model adaptation has been studied for some *specific* problems in multimedia, yet there lacks a *generic* and *systematic* approach. In this thesis, we take a unified view of various adaptation problems in multimedia, and attempt to find a generic and systematic approach to these problems. Specifically, we will focus on adapting *supervised classifiers* since they are arguably the most widely used model in multimedia. A generic approach avoids the need of developing algorithms for different problems, and allows people to treat emerging adaptation problems with ease.

2.2 Transfer and Multi-Task Learning, Incremental Learning, and Sample Bias Correction

In machine learning, several directions of research are related to the problem of cross-domain model adaptation. Transfer learning and multi-task learning are both about applying knowledge learned from some tasks to help other related tasks. Specifically, the former focuses on the transfer of knowledge from one task to another, while the latter focuses on learning models for multiple related tasks together. Cross-domain model adaptation can be viewed as transfer learning between multiple datasets on the same task, and our approach in this thesis can be treated as a novel type of transfer learning approach. Also related to model adaptation are incremental learning, which continuously updates a model based on subsets of the training data, and sample bias correction, which deals with learning problems where the distribution of test and training data are different. We review previous work in these directions and discuss their relation to our problem.

2.2.1 Transfer learning

Transfer learning (TL) aims to apply knowledge learned from *auxiliary tasks*, where labeled data are usually plenty, to help develop an effective model for a related *target task* which has only limited labeled data. There is no formal definition of "related tasks", and in practice it refers to either related learning problems on the same dataset (e.g., detecting *Sky* and *Outdoor* in news video), or the same learning problem on different datasets (e.g., detecting *Outdoor* in news video as well as in movies). After the notion was first introduced by Thrun [97] about a decade ago, techniques have been proposed to transfer the knowledge at different levels of abstraction, including the data level, representation level, and parametric level.

Data-level transfer techniques augment the training data of the target task with labeled data from auxiliary tasks in order to build a better model for the target task. This has been the approach taken by the TL algorithm based on k-nearest neighbor [97], based on support vector machines by Wu and Dietterich [105], based on logistic regression by Liao et al. [65], and based on AdaBoost by Dai et al. [32]. While some of these methods (e.g., [105, 32]) do not explicitly add auxiliary data into the training set, a close examination reveals that such data implicitly play the role of additional training data. A key issue in these methods is the weight of the data from the auxiliary tasks, which can be specified manually [105] or according to the degree of "mismatch" between the auxiliary and target data [65, 32]. Inefficiency is the major disadvantage of these methods, because training can be expensive when there is a large amount of auxiliary data. This poses a serious problem for multimedia, where data features of hundreds or even thousands of dimension are typical, making training over large data set particularly inefficient.

Representation-level approaches learn an effective feature representation and/or distance metric from the auxiliary tasks and use it for the target task. The first work on transfer learning by Thrun [97] proposed to learn both a new data representation and a distance function from the labeled data in auxiliary tasks. In order to help models generalize, Ben-David et al. [33] derived a common representation such that the distribution of the auxiliary and target data appear to be similar. Raina [79] used unlabeled images collected from various sources to learn high-level feature representations that can make image classification tasks easier in general. These approaches are

obviously more general and efficient than data-level transfer learning methods, since the representation needs to be learned only once and is applicable to many other tasks.

There has been some work in multimedia that shares the same spirit as representation-level adaptation. Probabilistic graphic models such as Latent Dirichlet Allocation (LDA) [16] and its variations like Correspondence-LDA [15], exponential-family Harmonium [103] and its variations like Dual-wing Harmonium [108], have been used to derive lower-dimensional representations of image/video data from their raw features, in either unsupervised or supervised manner. In the supervised case, these models learn new data representations using the data labels w.r.t some semantic categories, making them essentially equivalent to representation-level adaptation. Unlike in the text domain, however, no significant performance improvement has been reported on large-scale multimedia data analysis. For example, the Oxford team in TRECVID 2007 concept detection task shows that submission based on LDA-derived features offers little or no improvement over submissions using raw features [75], and harmonium model was also ineffective in similar tasks [3]. This shows the difficulties of learning better feature representations from multimedia data.

Parameter-level approaches use the parameters of the models learned previously from related tasks to form a "prior" for the model parameters to be learned for the target task. The new model can be thought as a "posterior" obtained by updating the prior based on the labeled examples in the target task. Many approaches chose Bayesian logistic regression with a Gaussian prior on the parameters as the learning algorithm for the target task. In Marx [67], both the mean and variance of the prior are inherited from the parameters of the existing models for related tasks. Similarly, Raina et al. [80] constructed a Gaussian prior with its covariance matrix encoding the word correlations derived from text classification tasks and applied it to similar tasks. Zhang [120] combined Rocchio algorithm with logistic regression via a Gaussian prior to yield a low-variance model for adaptive filtering. Fei-fei [42] incorporated the Bayesian prior into an object recognition model for images such that it can recognize new object categories with very limited training examples.

There are more specialized TL approaches. For instance, Taylor and Stone [96] proposed a transfer algorithm for reinforcement learning, and Heitz et al. [51] described a landmark-based model for transfer of the "essence" of object classes learned from

cartoon images to natural images. While most TL techniques assume only one auxiliary task, Marx et al. [67] explored transfer learning with *multiple* auxiliary tasks and provided a simple solution. Rosenstein et al. [83] studied the impact of the relatedness between the target and auxiliary tasks on the performance of transfer learning.

The framework to be proposed in this thesis can be viewed as transfer learning at a more abstract level: the *function level*. Our framework aims to directly adapt the decision function of one or more auxiliary classifiers into the decision function of a classifier for the target data. Function-level transfer enjoys many benefits, including high efficiency due to the freedom from training over auxiliary data and broad applicability to adapt any type of classifiers. To our knowledge, the most similar work in the literature has been that by Schapire et al. [86], which modifies the AdaBoost algorithm such that the Kullback-Leibler divergence between the classifier learned and a prior model representing human knowledge is minimized. However, this method is specialized for AdaBoost, while our framework is able to adapt classifiers of any type.

2.2.2 Multi-task learning

Multi-task learning (MTL) explores the dependency between related tasks and learns models for them simultaneously, in order to achieve better performance than learning each task independently. Unlike in TL where the knowledge transfer is unidirectional (from auxiliary tasks to target task), in MTL the knowledge transfer is be mutual and between any related tasks. Despite the difference on problem setting, MTL approaches provide important references to our model adaptation approach because both of them deal with knowledge transfer between tasks.

Similar to the approaches for TL, most MTL approaches support the transfer of knowledge either at the representation level or at the parametric level. *Representation-level* approaches [5, 6, 7, 20, 119] map original data features into a latent feature space (i.e., representation) shared by all the tasks, and then learn models for each task independently based on this common feature space. This shared feature representation can be derived using different principles and techniques, such as maximizing independence through latent independent component analysis in Zhang et al. [119], maximizing sparsity (i.e., lower dimension) in Argyriou and Evgeniou [7], and so on. A related method is that by Niculescu-Mizil and Caruana [73], which learns the structure of a Bayes

net from related tasks. *Parametric-level* approaches assume the model parameters of related tasks are related in a certain way and are to be learned together. In Bakker and Heskes [10], the model parameters of related tasks must share one Bayesian prior, while in Evgeniou et al. [39, 38] they are constrained by a regularization term in the objective function that penalizes their difference. The work in [116] extended the idea to a hierarchical Bayesian method, where the models of related tasks are represented by Gaussian processes (GPs) constrained in the function space, i.e., their decision functions must be close to each other in the function space. In addition, Lawrence and Platt [63] studied sample selection strategies in MTL, and Ben-David and Schuller [12] provided a mathematical notion of task relatedness.

2.2.3 Incremental learning

Incremental learning, or online learning, is to continuously update a model based on (different) subsets of the training data. It is preferred over training a model in one batch using all the data, when the latter is computationally expensive due to large data size, or when new training data become available after the model is trained. There are many incremental learning methods for updating classifiers especially SVMs. For example, Syed et al. [94] proposed to update a SVM model by re-training it from the support vectors of the existing model and the new data combined. Cauwenberghs and Poggio [21] further extended it by allowing also decremental learning of SVM.

On the surface, our adaptation approach is similar to incremental learning methods because both update existing classifiers based on additional labeled examples. Nevertheless, there are fundamental difference between them in terms of their assumption, goal, and approach. Incremental learning assumes that all the training data come from the same distribution, and their goal is that the incrementally trained models are identical or close to models trained in one batch. In comparison, our adaptation approach accommodates distribution changes across domains, and it does not require the adapted models to be close to batch-learned models (and they should not be). Moreover, most incremental learning methods involve at least part of old training data (e.g., support vectors) in the training, so they are not as efficient as our approach which directly adapts the existing classifier and uses absolutely *no* old data. Given these differences, our adaptation approach can be used as an efficient incremental learning algorithm,

but existing incremental learning algorithms are inappropriate for model adaptation.

2.2.4 Sample bias correction

Another line of research focuses on learning tasks where the training and test data distribution is known to be different, a problem known as *sample selection bias* or *covariance shift*. In this problem setting, the training sample is governed by an unknown distribution $p(\mathbf{x}|\lambda)$ while the unlabeled test data is governed by a different unknown distribution $p(\mathbf{x}|\theta)$. The training and test distribution may differ arbitrarily, but the basic assumption is that there is only one true unknown conditional distribution $p(y|\mathbf{x})$. The goal is to find a classifier $f : \mathbf{x} \longmapsto y$ that can accurately classify the test data.

Many sample bias correction methods are based on the theorem that the expected loss on test distribution $p(\mathbf{x}, y|\theta)$ equals the *weighted* expected loss on the training distribution as $p(\mathbf{x}, y|\lambda)\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}$. This means one can train a classifier that minimizes the loss on the test distribution by minimizing the weighted loss on the training data, where the loss on each instance $\mathbf{x}$ needs to be weighted with instance-specific scaling factor $\frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}$. Since $p(\mathbf{x}|\lambda)$ and $p(\mathbf{x}|\theta)$ are typically unknown, Shimodaira [88] and Sugiyama et al. [93] proposed to estimate $\hat{p}(\mathbf{x}|\lambda)$ and $\hat{p}(\mathbf{x}|\theta)$ from the training and test data using kernel density estimation, and then use $\frac{\hat{p}(\mathbf{x}|\theta)}{\hat{p}(\mathbf{x}|\lambda)}$ to resample or weight the training examples in the training of classifiers. Instead of estimating the data distribution $p(\mathbf{x}|\lambda)$ and $p(\mathbf{x}|\theta)$, the methods of Zadronzy [118] and of Bickel and Scheffer [14] directly estimate the ratio $p(s = 1|\mathbf{x}, \lambda, \theta) \propto \frac{p(\mathbf{x}|\theta)}{p(\mathbf{x}|\lambda)}$, where s is a selector variable that decides whether an example $\mathbf{x}$ drawn under the test distribution $p(\mathbf{x}|\theta)$ is moved into the training set ($s = 1$) or not ($s = 0$). In practice, $p(s = 1|\mathbf{x}, \lambda, \theta)$ is estimated from the training and test data, usually by a discriminative approach [14]. Besides, Huang et al. [52] proposed a kernel mean matching method that sets the weights of training instances such that the first moment of training and test data matches. Bickel et al. [13] improved upon these methods by integrating the weight estimation and model training into a unified discriminative framework. Fan [41] suggested to use simple model averaging technique to alleviate the effect of sample bias.

Despite the advance on sample bias correction methods, several difficulties limit their applicability on our problem. One of them is the difficulty of estimating the

distribution of multimedia data $p(\mathbf{x})$, not to mention its change across different domains. While several generative models for image and video data exist [15, 108], they are specialized for certain features (e.g., color histogram) and have not shown to be effective. Moreover, in multimedia the class-conditional $p(y|\mathbf{x})$ often changes across domains. This violates the basic assumption of sample bias correction and renders all the aforementioned methods inapplicable. Thus, in this thesis we try to adapt models *without* articulating how the data distribution changes across domains.

2.3 Concept Drift in Data Mining

Domain adaptation is also related to the problem of concept drift in data mining research on streaming data. Concept drift means the statistical properties of a target variable or *concept*, which a model is trying to predict, changes over time due to the change of some hidden context. An example of concept drift is spam detection from a user's daily emails, where the definition of spam may change over time (e.g., to include email advertisements) and the types of spam can change drastically with time. Concept drift may involve changes of the target concept definition (i.e., $p(y|\mathbf{x})$) and/or the change of the data distribution (i.e., $p(\mathbf{x})$).

In data mining, there are two major approaches to detecting a drifting concept in time-evolving data. The first approach selects training instances using a temporal window with fixed or adaptive size that moves over the data stream, possibly weights the selected instances by their utility to the target concept, and uses them to build a single classifier. This has been the approach employed by Klinkenberg and Joachims [61], and by Cunningham et al. [31]. The second approach maintains a set of base classifiers trained from each segment of the data stream, and combines their outputs on the test data in the form of a weighted ensemble. Wang et al. [102], Street and Kim [92], and Kolter and Maloof [62] adopted this approach.

Despite being proposed for streaming data, the aforementioned methods can be used to solve domain adaptation problems if we treat each domain as a data chunk from a data stream. However, methods for concept drift usually make implicit assumptions about the underlying distribution of the data stream. One assumption in many approaches [102, 61] is that the distribution of current data chunk is similar to that of

the most recent data chunks. Based on this assumption, models for detecting concepts in the current data should be built mainly from the recent data, and the ensemble approach should also select and weight the base classifiers from recent data chunks. While such assumption may be reasonable in temporally correlated streaming data, no similar assumptions can be made on the distributions of data domains, which may differ arbitrarily. In fact, as discussed in Chapter 1, measuring the relatedness between different domains is one of the key challenges in this thesis. A related work from data mining is from Fan [40], which analyzes the impact of combining "old data" for training and proposes an efficient and systematic way to selecting useful data.

2.4 Adaptation in Other Areas

Model adaptation has been also studied in the context of specific problem domains, such as natural language processing and speech recognition. In natural language processing (NLP), there is often a need to adapt language models, parsers, and models for other tasks (e.g., named-entity detection) from one corpus to other corpora. Through experiments on Wall Street Journal corpus and Brown corpus, Gildea [46] found that statistical parsing models, especially the bi-gram statistics, are corpus-specific, and suggested a technique to pruning model parameters to achieve better generalization ability. Hwa [53] proposed a two-stage adaptation process to first train a grammar from a fully-labeled old domain, and then re-estimate the probabilities of the grammars from a sparsely-labeled new domain. Roark and Bacchiani [82] proposed to adapt a lexicalized probabilistic context-free grammar (PCFG) from an old domain to a new domain. Their approach is to compute the maximum a posterior (MAP) estimation of the model parameters under a prior distribution given by the old ("out-of-domain") models. In [9], the same approach is applied to adapt language models across corpora, where a language model is a generative model governing the distribution of terms and phrases in documents. In addition, Shen et al. [87] explored adapting a general HMM-based named-entity recognizer trained from newswire documents to biomedical documents. While most of the aforementioned models are generative ones, Chelba and Acero [27] proposed to adapt maximum entropy classifiers for recovering correct capitalizations in uniformly-cased text based on MAP estimation of model parameters.

Model adaptation has also been extensively studied in speech recognition, where the model mismatch problem can be caused by different speakers, dialects, speaking styles, and environments (noise levels). Since the acoustic model of a speech recognition system is usually based on hidden Markov model (HMM), most adaptation methods adjust the parameters of the HMM model to better fit the target data. Such methods include speaker adaptation methods based on maximum-a-posterior (MAP) [45], Maximum Likelihood Linear Regression (MLLR) [64], Vocal Tract Length Normalization (VTLN) [99], and noise adaptation methods such as parallel model combination (PMC) [44]. All the above methods are specialized to the (generative) acoustic models used in speech recognition systems.

The aforementioned techniques exploited the domain knowledge of the highly specialized models in NLP or speech recognition. They can offer little help to solving the general adaptation problem in multimedia.

Chapter 3

Generalizability in Semantic Concept Detection

Semantic concept detection is a critical learning task in multimedia and also the foundation of many other tasks. The goal of semantic concept detection is to automatically determine whether certain semantic concepts (e.g., *Studio*, *Outdoor*, and *Sports*) are present in images and/or video shots, where a video shot is a sequence of video frames taken by a single camera operation. The general approach is to build a supervised classifier for each concept from labeled images or video shots. The classifier can be represented as a function $f : \mathbf{x} \rightarrow y$ that maps the low-level feature $\mathbf{x}$ of an image or video shot into a score y which indicates its degree of relevance to concept (larger scores mean higher relevance). The score y can be converted into a binary label that indicates the presence or absence of the concept in the image or video shot. Such *concept classifiers* can be trained using various classification algorithms such as support vector machines (SVMs), logistic regression, etc.

Concept classifiers are typically evaluated by their *within-domain performance*, or performance on data from the same domain as their training data, often through cross validation. The performance can be high due to overfitting, i.e., the classifier fits the training data very well without the ability to generalize beyond the training data. There has been very little study on the *cross-domain performance* of concept classifiers [100], defined as their performance on data from domains other than their training domain. We believe cross-domain performance is a more rigorous and realistic metric,

as it measures not only how well a classifier fits its training data but more importantly how well it generalizes beyond such data. In this section, we conduct a comprehensive survey on the generalizability of semantic concept classifiers by comparing their within-domain performance to their cross-domain performance. We also investigate the reason behind the poor generalizability of concept classifiers. This study provides insights on the importance and challenges of the model adaptation problem in multimedia.

3.1 Experiment Set-up

We design a series of experiments to explore the generalizability issue in semantic concept detection. This involves building concept classifiers from one domain and comparing their performance on the data from the same domain (within-domain performance) to their performance on data from other domains (cross-domain performance). To make the findings general and convincing, we choose the data, semantic concepts, and classification algorithms that are frequently used in the literature and representative of the state-of-the-art. Meanwhile, we propose new performance metrics which overcome the limitations of existing metrics and measure generalizability better.

3.1.1 Test data

An ideal corpus for studying generalizability must consist of data from multiple domains. A good choice is the video collections used in TREC Video Retrieval Evaluation (TRECVID) [89]. TRECVID is an annual workshop sponsored by the National Institute of Standards and Technologies (NIST) to promote research in content-based video retrieval in large collections via an open, metrics-based evaluation. It has defined a set of retrieval-related tasks for evaluation, including shot boundary detection, high-level feature extraction (a.k.a semantic concept detection), and automatic and interactive search. From 2002 to 2007, the TRECVID collection varies on a yearly basis from movies to broadcast news video and documentary video. Among all the TRECVID collections, we select the development set of the collection used in 2005 and in 2007, which are referred to as TV05DEV and TV07DEV respectively in this thesis.

The TV05DEV collection contains 86 hours of broadcast news video from 6 TV channels, including CNN, NBC, MSNBC, CCTV, NTDTV, and LBC. Among them,

CCTV and NTDTV are in Chinese (Mandarin), LBC is in Arabic, while the others are in English. The data in each channel come from 2-3 different news programs in that channel, e.g., the footage from CNN is from "Live From CNN" and "Anderson Cooper 360". Due to the difference on editing styles, target audience, and many other factors, the data from different channels exhibit very different characteristics. The 86-hour footage has been automatically partitioned into 61,901 video shots [74] and the shot boundaries are provided. The footage is relatively evenly distributed across different channels, with largest channel containing 11,025 shots and the smallest one having 6,481 shots.

The TV07DEV collection contains 50 hours of news magazine, science news, news reports, documentaries, educational programming, and archival video provided by the Netherlands Institute for Sound and Vision, which can be collectively described as documentary video. This set has 21,532 shots, and there is no further partitioning into subsets.

In both TV05DEV and TV07DEV collection, each video shot is represented by the frame in the middle of its temporal duration, denoted as its "keyframe". A keyframe is described by a 225-d color moment feature computed from 5×5 grids and a 48-d Gabor texture feature. We concatenate them into a 273-d feature vector representing the video shot. In several TRECVID evaluations through 2006 (e.g., [24]), this frequently used feature has shown to provide performance on par with other state-of-the-art features.

3.1.2 Semantic concepts

Labels for a set of semantic concepts are available on the shots of the TV05DEV and TV07DEV collection. The shots in TV05DEV were manually annotated with respect to 39 concepts as part of the Light Scale Concept Ontology for Multimedia (LSCOM-Lite) project [68]. The shots in TV07DEV were annotated through a collaborative effort [8] with respect to the same set of concepts expect 3 concepts (thus totally 36 concepts). The 36 common concepts are *Airplane*, *Animal*, *Boat_Ship*, *Building*, *Bus*, *Car*, *Charts*, *Computer_TV-Screen*, *Court*, *Crowd*, *Desert*, *Explosion_Fire*, *Face*, *Flag-US*, *Maps*, *Meeting*, *Military*, *Mountain*, *Natural-Disaster*, *Office*, *Outdoor*, *People-Marching*, *Person*, *Police_Security*, *Prisoner*, *Road*, *Sky*, *Snow*, *Sports*, *Studio*, *Truck*, *Urban*, *Vegetation*, *Walking_Running*, *Waterscape_Waterfront*, and *Weather*.

The 3 concepts left out from TV07DEV are *Entertainment*, *Government-Leader*, and *Corporate-Leader* because they have no or very few relevant shots in TV07DEV.

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. There is also a large difference between concepts in terms of their frequency, which is defined as the ratio of shots relevant to a given concept against all the shots. Some general concepts have frequency around 50%, such as *Face* and *Outdoor*, while many rare concepts have frequency below 1%, such as *Prisoner* and *Airplane*. The frequency of a concept also varies between the two collections and between different channels of the TV05DEV collection.

3.1.3 Performance metric: AP and ΔAP

Average precision (AP) has been frequently used as the performance metric in semantic concept detection. Given a concept classifier's output in the form of relevance scores on a set of test video shots, we rank the shots in descending order of their score, and compute AP as the average of the precisions of this ranked list truncated at each of the relevant shots:

$$AP = \frac{\sum_{r=1}^{N}(P(r) \times rel(r))}{\#\text{ of relevant shots}} \tag{3.1}$$

where r is the rank, N is the total number of shots, $rel(r)$ is a binary function with output 1 or 0 indicating whether the r^{th} shot is relevant to the concept or not, and $P(r)$ is the precision of the list truncated at rank r. To measure the performance of multiple classifiers, we use *mean average precision* (MAP) which is equal to the average of multiple APs.

While AP is a good metric of rank quality, it is incomplete and misleading as the metric of classifier performance. The baseline of AP should be the AP of a "random classifier" that sorts the test shots *completely randomly*. Given the definition in Eq.(3.1), it is easy to see that the baseline AP of a concept is not zero, because no matter how low the relevant shots are ranked, $P(r)$ becomes non-zero after r reaches the first relevant shot. Although the baseline AP is not constant due to random ranking, it is easy to show mathematically that the *expectation of baseline AP* of a given concept is equal to the frequency of that concept, i.e., the ratio of positive shots against all

shots. This means different concepts have different AP baselines, and these baselines have nothing to do with how well concept classifiers perform.

The difference on baselines causes several problems in the use of AP. First, it makes concept classifiers less comparable. For example, an *Outdoor* classifier with 0.9 AP is not necessarily better than a *Studio* classifier with 0.8 AP, since the baseline (frequency) of the latter is much lower. Even for the same concept, the classifiers built on different collections are still not comparable because the concept's frequency varies with collections. The second problem comes with using MAP as the metric of average performance on multiple concepts. Because concept frequency varies greatly, sometimes by orders of magnitude, MAP can be easily dominated by the AP of a concept with a much higher frequency than the rest. Meanwhile, a rare concept whose positive instances occur in nearly identical form (e.g., commercials) typically has a very high AP if one such instance is in the training data. This results in a large but somewhat deceptive improvement on MAP, without any generalizability to other domains. Last but not the least, AP alone is unable to capture the generalizability of a concept classifier. For example, if we know the within-domain and cross-domain performance of a classifier is 0.9 AP and 0.8 AP respectively, we are still unable to judge how generalizable the classifier is. The answer also depends on the baseline. If the baseline is 0.7 AP, the classifier in fact loses half of its performance; if the baseline is 0.4 AP, then it loses only 20% of its performance. Since our focus is on the generalizability of classifiers, we must introduce new metrics that capture generalizability more precisely.

A concept classifier can be more precisely evaluated in terms of how much *better* it performs than a random classifier. Thus, we define a *normalized* performance metric *delta AP* (or ΔAP), which is set to the *difference* between the AP of a concept classifier and the expected baseline AP of this concept in a particular dataset. In other words, ΔAP measures the *improvement* of rank quality as the result of using a classifier. Since the baseline of ΔAP is zero, it overcomes all the problems mentioned above. Most importantly, we can use the difference (either absolute or relative) between within-domain and cross-domain performance measured by ΔAP to represent a classifier's generalizability. Therefore, we will mainly use ΔAP to measure generalizability in the rest of the chapter. Note that ΔAP can occasionally be negative if the classifier performs worse than random. We also define ΔMAP as the mean of multiple ΔAPs.

3.1.4 Classification algorithm

Most existing methods on semantic concept detection chose support vector machines (SVMs) with radius-basis kernel function (RBF) for training concept classifiers due to its good performance in practice. The sophisticated methods for concept detection are often built on top of SVM-based concept classifiers as their basic building blocks. For examples, some methods combine SVM classifiers trained on different features as an ensemble, while some combine SVM classifiers trained for correlated concepts [77, 112, 19, 72, 90, 117]. Therefore, SVM-based concept classifiers represent the mainstream of semantic concept detection, and their generalizability represent the generalizability of the more sophisticated methods. To make the results of our study representative, we choose to train *all* concept classifiers by SVMs with RBF kernel, using the widely-used LIBSVM package [22].

The model parameters of SVM, especially the cost factor and gamma parameter in the RBF kernel [17], have shown to have a large impact on the performance and were heavily tuned in practice. In our study, we choose the model parameters to values that lead to respectable performance in TRECVID 2005 evaluation [49], which used the same data and semantic concepts as in this experiment. They are not necessarily the parameters that achieve the highest AP or MAP. This is not a problem since our focus is on the generalizability of performance across different domains. In other words, we care about the *change* of performance across different domains instead of the absolute performance in one domain.

3.1.5 Experiment settings

Our study consists of two settings: the *cross-channel setting* and *cross-genre setting*. In the cross-channel setting, we build concept classifiers from one channel (of news video) in the TV05DEV collection, and evaluate their performance on the other channels in TV05DEV. In the cross-genre setting, we build concept classifiers trained from the entire TV05DEV (or TV07DEV) collection, and apply them to the TV07DEV (or TV05DEV) collection. In the former setting each news video channel is a domain, while in the latter all the 6 news channels together are treated as one domain. This is reasonable because each news channel has its own characteristics to justify being

		Training channel					
		CCTV	CNN	LBC	MSNBC	NBC	NTDTV
Test channel	CCTV	**.332**	.167	.172	.166	.172	.138
	CNN	.145	**.319**	.151	.193	.168	.132
	LBC	.161	.152	**.306**	.159	.180	.159
	MSNBC	.123	.176	.129	**.312**	.179	.122
	NBC	.140	.147	.152	.180	**.329**	.138
	NTDTV	.127	.129	.165	.141	.150	**.313**

(a) **MAP**

		Training channel					
		CCTV	CNN	LBC	MSNBC	NBC	NTDTV
Test channel	CCTV	**.256**	.088	.093	.089	.095	.061
	CNN	.075	**.250**	.080	.124	.100	.062
	LBC	.080	.070	**.226**	.079	.101	.077
	MSNBC	.058	.111	.064	**.249**	.116	.057
	NBC	.074	.081	.086	.116	**.265**	.072
	NTDTV	.058	.059	.094	.072	.081	**.242**

(b) **ΔMAP**

Table 3.1: Average performance as (a) MAP and (b) ΔMAP of detecting 39 concepts with training and test data from 6 different news channels in the TV05DEV collection.

a domain, and at the meantime all news channels share the common properties of broadcast news video. The two settings contrast well in terms of the domain definition, data size, and the difference between domains. The hope is that by experimenting with two contrasting settings the results of this study are more general and convincing.

3.2 Generalizability of Concept Detectors

3.2.1 Cross-channel performance

We first examine how well concept classifiers trained from one channel of news video in TV05DEV perform on another channel. For each of the 6 channels, we build concept classifiers for the 39 LSCOM-Lite concepts using *all* the video data in that channel. This results in a total of 6 × 39 classifiers. Then we apply these classifiers to the data in each of the 6 channels. The channel used for training a classifier is the *training channel*, while the channel to which the classifier is applied is the *test channel*.

Table 3.1 shows the average performance of concept classifiers under different training and test channels in terms of MAP and ΔMAP over 39 concepts. The numbers

off the diagonal denote the average *cross-channel performance*, which is the performance of concept classifiers trained from one channel applied on a different channel. The numbers on the diagonal denote the average *within-channel performance*, namely the performance when the training and test data are from the same channel. The within-channel performance are obtained using 5-fold cross validation in order to avoid overfitting caused by using the same data for both training and testing. In cross validation, we evenly divide the data in a channel into 5 subsets, where 4 subsets are used for training and the remaining one for testing. The training-testing process is repeated 5 times so that each subset has been used for testing once, and we compute the MAP as the average performance of the 5 iterations.

It is clear from Table 3.1 that the cross-channel performance of concept classifiers is *consistently* and *substantially* lower than the within-channel performance. While all within-channel MAPs are above 0.3, none of the cross-channel MAPs are above 0.2. For most training-test configurations, the relative decline of MAP is around 50%. Note that because of its non-zero baseline, the decline measured in terms of MAP is smaller than the *actual* decline of a classifier's performance. The decline in ΔMAP gives a more realistic measure of how poorly concept classifiers generalize. As shown in Table 3.1(b), the average decline in ΔMAP is more significant at about 60-70%. While performance decline is expected, the amount of average decline is surprisingly large given that all the video data are broadcast news video. Overall, the result shows that the concept classifiers generalize poorly to channels other than its training channel, and because this happens to all training-test configurations, it is a general problem than an individual case.

The amount of performance decline varies between concepts. Figure 3.1 shows the average within- and cross-channel ΔAP of each concept, where the concepts are ordered from left to right in descending frequency. The gap between the two performance numbers on each concept, which indicates its generalizability, is much larger for some concepts than the others. For example, this gap for concepts like *Person* and *Crowd* is much smaller than the gap for *Animal* and *Charts*. A closer examination reveals that more frequent concepts tend to have smaller decline, and thus are more generalizable than rare concepts. In terms of the relative decline from within-channel to cross-channel performance, the 5 concepts with the highest frequency are also the ones with

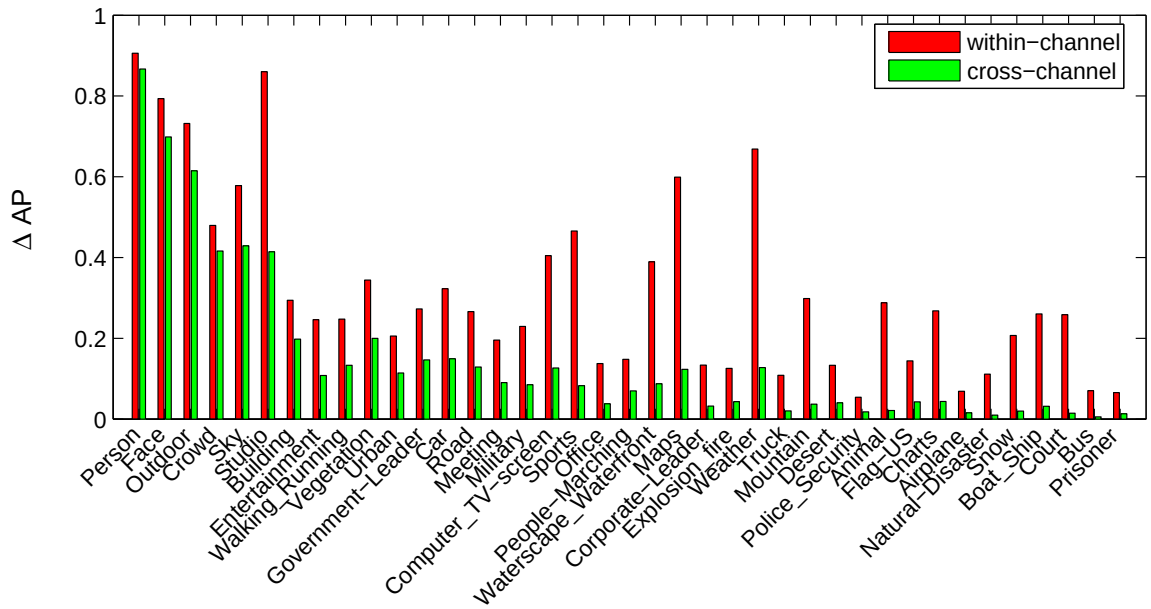


Figure 3.1: Within-channel ΔAP and cross-channel ΔAP on 39 concepts, from left to right in descending order of frequency.

the smallest decline. Among the 24 concepts with over 70% relative decline, 19 of them have frequency below 0.02. One reason for the better generalizability of frequent concepts is that they are generic concepts and generic concepts are relatively insensitive to domain changes (e.g., *Outdoor*). Another possible reason is that frequent concepts have a large number of positive data, which are critical for building reliable classifiers.

The cross-channel performance is affected by, and indicates, the similarity between the training and test channel in terms of data distribution. As supervised classifiers assume identical distribution between the training and test data, the decline of performance is determined by how much this assumption is violated. In Table 3.1, for example, when MSNBC is the training channel, the average performance of concept classifiers is 0.124 ΔMAP on CNN, but only 0.072 ΔMAP on NTDTV. This shows that CNN is more similar to MSNBC than to NTDTV.

By treating cross-channel performance as an indicator of the pairwise similarity between channels, we visualize such similarities in Figure 3.2 using a visualization technique named multidimensional scaling (MDS) [30]. As expected, the three English news channels (all produced in the United States), namely CNN, MSNBC, and NBC,

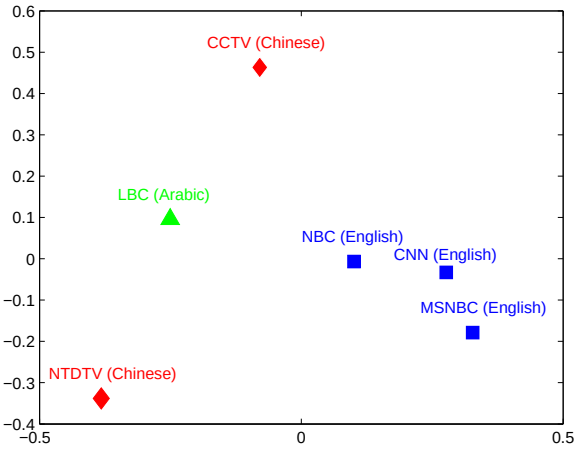


Figure 3.2: The relationships of 6 news channels illustrated using multidimensional scaling (MDS), based on the cross-channel performance between any two of them as an indicator of their similarity.

are close to each other. Meanwhile, the other 3 channels, which are the Chinese channel CCTV and NTDTV and the Arabic channel LBC, are far away from each other and the English channels. This is intuitive because language and country influences the editing styles and consequently the data characteristics of each news channel. The two Chinese news channels are far apart perhaps because one is produced in China while the other is produced in the US.

3.2.2 Cross-genre performance

The cross-genre experiment evaluates the performance of concept classifiers when they are applied to data of a different genre from their training data. To obtain cross-genre performance, we build concept classifiers for the 36 common concepts from the entire TV05DEV collection (news video) and evaluate them on the entire TV07DEV (documentary video) collection. Then we repeat the experiment with training and test collection reversed. We compare that to within-genre performance, which is obtained by 5-fold cross validation in either collection.

As shown in Table 3.2, the cross-genre performance is significantly lower than the within-genre performance in terms of either MAP or ΔMAP. The relative decline in terms of ΔMAP is 61% when applying TV05DEV classifiers on TV07DEV data, and

MAP / Δ MAP		Training collection	
		TV05DEV	TV07DEV
Test collection	TV05DEV	0.294 / 0.223	0.143 / 0.073
	TV07DEV	0.166 / 0.086	0.201 / 0.122

Table 3.2: Average performance as MAP and ΔMAP of detecting 36 concepts with training and test data from either TV05DEV (news video) or TV07DEV (documentary)

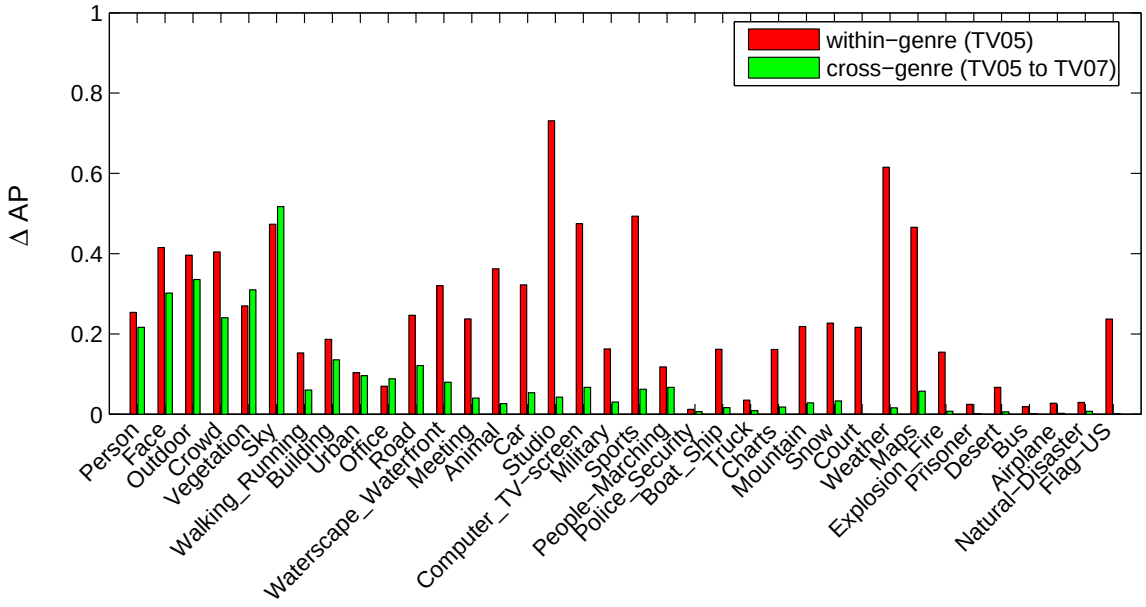


Figure 3.3: Within-genre ΔAP and cross-genre ΔAP on 36 concepts, from left to right in descending order of frequency.

about 40% the other way around. Surprisingly, while the training and test data are perhaps more dissimilar here than in the cross-channel setting, the performance decline is not larger. A possible explanation is that the concept classifiers are built on a larger amount of training data (the entire TV05DEV data are used for training, instead of a single channel in it) and therefore provide more reliable performance.

Figure 3.3 shows the within- and cross-genre ΔAP on a per-concept basis. Similar to the cross-channel experiment, we see a large variation between concepts in terms of generalizability, indicated by the gap between the two performance numbers of each concept. The trend that frequent concepts are more generalizable is even more pronounced here.

Given the results on both cross-channel and cross-genre experiments, we conclude that the change of data domains has a large impact on the performance of concept

detection, and concept classifiers in general do not generalize well beyond their own training data. Meanwhile, frequent concepts tend to generalize better than the rare concepts.

3.3 Explaining Classifier Generalizability

We have learned that most semantic concept classifiers over-fit their training data and do not generalize well. In order to provide insights into their poor generalizability, we examine the model structure of these classifiers and compare them to memory-based learning methods. We will see that, despite training using state-of-the-art algorithms and features, concept classifiers learn little beyond memorizing the training data and perform nearest-neighbor type of prediction, which helps explain their poor generalizability.

3.3.1 Model (SVM) structure

Similar to many existing methods to concept detection, in our experiment the concept classifiers are trained with SVM. The decision function of a SVM classifier is expressed as:

$$f(x) = \sum_{x_i \in D_{SV}} \alpha_i y_i \mathcal{K}(x, x_i) \tag{3.2}$$

where D_{SV} is a subset of training data called support vectors (SVs), x_i and $y_i \in \{-1, 1\}$ denotes the feature vector of a SV and its label which indicates its relevance to a given concept, and $\mathcal{K}(x, x_i)$ is the kernel function determining the similarity between the query instance x and x_i. Thus, the decision boundary of a classifier is completely determined by the set of SVs, which are chosen by the SVM algorithm as representative instances to define the separation between positive and negative data. For example, training instances close to the decision boundary are typically chosen as SVs.

The number of SVs in a SVM classifier indicates the complexity of the decision boundary, especially when a non-linear kernel function such as RBF is used. Think SVM training as a data compression process. If the SVM algorithm finds a simple and smooth boundary to separate the data in two classes, it would need only a small number of SVs to represent the boundary. In other words, it compresses the data into

Dataset	# of shots	Percentage of SVs in positive data	negative data	all data
CCTV	10896	92.1%	10.5%	13.2%
CNN	11025	93.8%	9.3%	12.0%
LBC	15272	94.0%	10.4%	12.8%
MSNBC	8905	96.2%	9.7%	12.2%
NBC	9322	95.6%	11.4%	14.5%
NTDTV	6481	94.1%	10.9%	13.6%
TV05DEV	61901	94.7%	9.8%	12.4%
TV07DEV	21532	94.3%	12.6%	15.5%

Table 3.3: For the SVM classifiers of 39 concepts, the average ratio of support vectors (SVs) in the positive data, in the negative data, and in all the data.

a small set of SVs while retaining the information about classification. If a smooth boundary cannot be found, the SVM algorithm would produce a convoluted boundary that zigzags through the feature space in order to separate the positive and negative instances as much as possible. Such a complex boundary needs to be "supported" by a large number of SVs. In this case, it fails to compress the training data, and it has to "memorize" most of them in the form of SVs. As we will see, this also means this classifier does nearest-neighbor type of classification, i.e., judging the label of an instance based on whether most of its neighboring instances are positive or negative.

In Table 3.3, we show the average ratio of SVs among the entire training data and among the positive and negative data in concept classifiers trained from different domains. All the ratios are averaged across the 39 concepts (36 for TV07DEV). We see that while the SVM classifiers use only 12% to 15% of the training data as SVs, they retain most of the positive instances as SVs. The ratio of SVs in positive data is over 90% for all the domains, implying that these SVM classifiers are unable to summarize the positive data into general and concise representations. As a result, they retain most of the positive training data as SVs. In fact, for about 25-30 concepts out of the 39 concepts (depending on domains), *all* the positive data are used as SVs by their classifiers. For only 2-3 concepts this ratio is below 50%. On the other hand, the classifiers are able to aggressively compress the negative data, which are abundant. Since positive data are far more valuable to classification than negative data due to their relative scarcity, it is legitimate to say that SVM concept classifiers fail to learn the essential patterns in the data.

Why do SVM classifiers retain most positive data as SVs? One explanation is that positive data are insufficient. However, as shown in Table 3.3, the ratio of SVs in positive data stays within the small range of 92% to 96% when the data size increases almost 10 times from the smallest dataset (NTDTV) to the largest one (TV05DEV). We do not see a clear connection between the *absolute* size of positive data and the ratio of positive SVs, although one may argue that even the largest collection is not large enough and the ratio of positive SVs will finally drop if the data size grows further. A better explanation is that the relative percentage of positive data in all the data is very small. Nevertheless, we find that more frequent concepts do not necessarily have smaller ratio of SVs in positive data. For example, the frequency of *Sky* (11%) is slightly higher than that of *Studio* (10%), but the ratio of SVs in positive data for *Sky* classifiers (97.3%) is much higher than that for *Studio* classifiers (40.6%). This is because *Studio* shots are visually similar and can be well represented by a small set of SVs, while *Sky* shots are more diversified.

Therefore, how many positive SVs are needed is more related to the intrinsic property of each concept, such as the irregularity of data distribution. When positive and negative data are intermingled in the feature space and a clear separation is impossible, using most positive data to form a convoluted boundary is unavoidable. So this is mainly due to the limitation of our visual features rather than the incompetence of classification algorithms.

3.3.2 Comparison with memory-based models

In SVM's decision function Eq.(3.2), each term $y_i\mathcal{K}(x, x_i)$ in the summation can be seen as an *atomic classifier* which predicts the label of x to be the same as the label of SV x_i with confidence weighted by their similarity measured by kernel function K. Thus, we can treat a SVM classifier as an *ensemble* of such atomic classifiers, and its prediction as a weighted sum of the labels of all its SVs. This reveals an analogy between SVM and memory-based learning models such as k-nearest neighbor (kNN). A kNN model predicts the label of query instance x as the normalized weighted sum of the labels $y_i \in \{0, 1\}$ of its K nearest neighbor $x_1, ..., x_K$:

$$f(x) = \frac{\sum_{i=1}^{K} w_i y_i}{\sum_{i=1}^{K} w_i} \tag{3.3}$$

where the weight is set to the inverse of the Euclidean distance between x and each of its neighbors, i.e., $w_i = 1/D(x_i, x)$. Except for the denominator, Eq.(3.3) has a similar form to SVMs' decision function in Eq.(3.2), because the kernel function $\mathcal{K}(x, x_i)$ is essentially a similarity (distance) measure between x and x_i and plays a similar role as w_i. The key difference is that kNN makes predictions based on nearest neighbors while SVMs make predictions based on SVs. However, since most positive data are used as SVs in the concept classifiers as shown earlier, a SVM classifier is structurally similar a kNN classifier with a large K.

Given the structural similarity between SVM and kNN, it is interesting to compare their performance in concept detection. Note that a memory-based model like kNN needs no training at all; it simply memorizes all the training data. If SVM performs no better than the learning-less kNN, it means SVM classifiers indeed "degenerates" to a memory-based model and fails to learn from the data despite its complicated training.

Table 3.4 compares the performance of SVM and kNN, where K is set to 100 based on our preliminary experiments, in concept detection on 39 concepts evaluated in both within-domain and cross-domain settings. In the two within-domain settings, we show the 5-fold cross-validation performance of classifiers trained on NBC and on TV05DEV. In two cross-domain settings, we show the performance of NBC classifiers applied to CNN, and TV05DEV classifiers applied to TV07DEV. We see that SVM and kNN performs comparably in the two within-domain settings, with very close AP on most concepts and almost identical average performance. One may argue that with careful parameter tuning SVMs may outperform kNN. While this is possible, we can also fine-tune kNN by varying K and ways of computing weight w_i. This shows that when the training and test data are from the same domain, SVM behaves close to kNN which does no training.

In the two cross-domain settings, SVM on average performs slightly better than kNN, although the difference is marginal. A close examination reveals that the difference mainly comes from a few concepts, such as *Crowd*, *Face*, and *Vegetation*, on which SVM outperforms kNN by a large margin. Note that these concepts are mainly frequent concepts. This echoes with our earlier finding that SVM classifiers for frequent concepts tend to be more generalizable, as shown in Figure 3.1 and 3.3. So SVM is no more generalizable than kNN except on a few frequent concepts.

	within-domain				**cross-domain**			
	NBC		TV05		NBC/CNN		TV05/TV07	
	SVM	kNN	SVM	kNN	SVM	kNN	SVM	kNN
airplane	0.077	0.145	0.028	0.109	0.016	0.012	0.005	0.018
animal	0.036	0.061	0.337	0.427	0.033	0.042	0.070	0.065
boat_ship	0.147	0.195	0.155	0.190	0.006	0.006	0.033	0.046
building	0.288	0.325	0.279	0.278	0.187	0.173	0.245	0.268
bus	0.144	0.200	0.018	0.056	0.004	0.004	0.005	0.006
car	0.542	0.552	0.356	0.378	0.292	0.218	0.093	0.076
charts	0.528	0.509	0.156	0.144	0.038	0.030	0.026	0.021
computer_tv-screen	0.429	0.427	0.400	0.491	0.143	0.113	0.102	0.061
corporate-leader	0.133	0.052	n/a	n/a	0.016	0.024	n/a	n/a
court	0.341	0.231	0.181	0.147	0.006	0.003	0.004	0.005
crowd	0.424	0.338	0.516	0.421	0.393	0.267	0.407	0.302
desert	0.231	0.304	0.067	0.116	0.050	0.030	0.010	0.012
entertainment	0.015	0.018	n/a	n/a	0.010	0.012	n/a	n/a
explosion_fire	0.091	0.133	0.144	0.140	0.020	0.053	0.012	0.008
face	0.692	0.658	0.762	0.768	0.781	0.663	0.788	0.678
flag-us	0.068	0.053	0.238	0.145	0.050	0.081	0.002	0.001
government-leader	0.211	0.147	n/a	n/a	0.180	0.150	n/a	n/a
maps	0.794	0.826	0.417	0.408	0.084	0.077	0.064	0.041
meeting	0.136	0.116	0.280	0.248	0.055	0.058	0.085	0.075
military	0.259	0.297	0.196	0.227	0.122	0.137	0.056	0.039
mountain	0.298	0.286	0.219	0.192	0.078	0.089	0.036	0.017
natural-disaster	0.084	0.265	0.034	0.038	0.012	0.007	0.010	0.009
office	0.128	0.123	0.088	0.098	0.036	0.064	0.172	0.116
outdoor	0.722	0.720	0.682	0.704	0.605	0.573	0.700	0.642
people-marching	0.134	0.132	0.131	0.159	0.098	0.097	0.084	0.076
person	0.835	0.808	0.896	0.903	0.898	0.839	0.873	0.822
police_security	0.043	0.029	0.022	0.046	0.026	0.012	0.023	0.023
prisoner	0.171	0.195	0.019	0.094	0.019	0.006	0.006	0.005
road	0.421	0.392	0.287	0.275	0.166	0.133	0.198	0.239
sky	0.580	0.484	0.564	0.498	0.496	0.409	0.654	0.571
snow	0.582	0.681	0.221	0.311	0.021	0.011	0.041	0.041
sports	0.393	0.277	0.499	0.413	0.014	0.014	0.086	0.033
studio	0.826	0.794	0.767	0.815	0.713	0.567	0.081	0.067
truck	0.089	0.127	0.043	0.069	0.009	0.015	0.021	0.028
urban	0.269	0.273	0.162	0.210	0.083	0.131	0.203	0.264
vegetation	0.326	0.259	0.351	0.278	0.299	0.193	0.462	0.395
walking_running	0.302	0.283	0.226	0.222	0.164	0.143	0.173	0.175
waterscape_waterfront	0.165	0.193	0.319	0.405	0.165	0.050	0.132	0.142
weather	0.878	0.850	0.540	0.626	n/a	n/a	0.023	0.011
average	**0.329**	**0.327**	**0.294**	**0.307**	**0.168**	**0.145**	**0.166**	**0.150**

Table 3.4: Per-concept performance of SVM and kNN ($k = 100$) in two within-domain settings (NBC and TV05DEV) and two cross-domain settings (NBC/CNN, TV05/TV07).

3.3.3 Discussion

The structural and performance similarity between SVM concept classifiers and kNN ones may explain their poor generalizability. Despite the learning process, SVM classifiers perform nearest-neighbor type of prediction, i.e., classifying data close to the positive SVs as positive and so on. The nearest-neighbor approach is fragile when the test data are distributed differently from the training data. For example, there are some gray-scale shots in TV07DEV which distribute far away from the color shots in

TV05DEV in the color feature space, and as a result, concept classifiers trained on TV05DEV perform poorly on TV07DEV. So the real reason is the limitation of the features used for classification rather than the (SVM) classification algorithm. However, until better features become available that make the data more separable, we need to work on algorithms to make classifiers generalize better.

We attempt to explain why classifiers for frequent concepts generalize better than classifiers for rare ones. One reason is that the frequent concepts are mostly generic concepts, such as *Outdoor* and *Crowd*, whose definition and visual appearance are relatively less sensitive to the change of domains. Moreover, frequent concepts have more positive training data, which are valuable to the classification performance due to their relative scarcity. The classifiers of rare concepts may suffer from high model variance due to the deficiency of positive data. This is especially true for models like kNN, including the SVM concept classifiers which are structurally similar to kNN. While a classifier with low variance may not generalize well, a classifier with high variance almost certainly generalize poorly.

3.4 Summary

Our study has shown that concept classifiers do not generalize well beyond their training domains. Meanwhile, building new classifiers for every new domain can be prohibitively expensive in terms of the effort of labeling training data. As an example, for TRECVID evaluation, every year a news video collection of moderate size, varying from 67 to 176 hours, was manually labeled w.r.t a small set of no more than 39 semantic concepts as the training data for concept detection. Based on the "rule of thumb" that a person on average needs one second to label one video shot for one concept, the manual effort is measured at 190.7 man-hours in 2003, 260 man-hours in 2006, 807 man-hours in 2005, and 1652 man-hours in 2006. Despite such a high labor cost, the classifiers trained in each year work well only on data very similar to the training data (i.e., news video form the same sources produced in approximately the same time) and do not generalize to other data. In comparison, the footage in many video archives (e.g., YouTube.com) can easily exceeds tens of thousands of hours, and a reasonably comprehensive semantic ontology contains hundreds of semantic concepts [2]. The practice of building new

models for each and every domain in TRECVID certainly does not scale to any real-world video archives.

Since existing classifiers generalize poorly to new domains, and building new classifiers require expensive labeling effort, adapting existing classifiers to new domains provides a good trade-off between performance and human cost. We call the task of detecting semantic concepts in one domain using classifiers adapted from other domains *cross-domain semantic concept detection*. We use this task as a benchmark task to evaluate the adaptation approach to be proposed in this thesis, because this task represents a set of common challenges faced by many adaptation problems in multimedia and other areas: Classifiers are vulnerable to the change of underlying data distributions across domains; The distribution change is arbitrary and hard to model; Labeling training data is tedious and time-consuming. If our approach is able to address the challenges in cross-domain semantic concept detection, it is reasonable to believe that it will work well on many other adaptation problems. Chapter 4 and 5 will focus on the technical details and evaluation of our adaptation approach.

The study in this chapter also encourages further investigation of the generalizability of concept classifiers. We have shown in Figure 3.1 and 3.3 that certain concept classifiers, especially those for frequent concepts, tend to generalize better than the others. It is an interesting and challenging research topic to find whether generalizability can be predicted from the properties of a concept and/or its classifier. Chapter 6 is dedicated to this question, where we will show predicting a concept classifier's generalizability is feasible, and the predicted generalizability can improve the cost-efficiency of adaptation through adapting classifiers selectively.

Chapter 4

Function-level Classifier Adaptation

In Chapter 3, we have shown the importance of adapting classifiers due to their inferior ability to generalize across domains. In this chapter, we describe a generic and principled framework for *function-level classifier adaptation*, which adapts a classifier by directly modifying its decision function based on regularized loss minimization. This framework enjoys great efficiency and broad applicability due to its freedom from using the "old data". We also introduce two concrete adaptation algorithms derived from the framework, namely adaptive support vector machines (a-SVM) and adaptive kernel logistic regression (a-KLR), and evaluate their performance on cross-domain semantic concept detection.

4.1 Problem Settings and Notations

We begin by defining a general problem setting and introducing the terminologies and notations used in this thesis. We consider a binary classification task in a *target domain*, where only a limited set of data are labeled while most data are unlabeled. 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 $y_i \in \{+1, -1\}$ [1] is a binary label indicating whether $\mathbf{x}_i$ is relevant (or irrelevant) to a concept or a category. 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.

[1] The binary label takes value of {+1,-1} instead of {0,1} only for the sake of notational convenience, i.e., it makes the representation of the loss functions discussed later much simpler.

In addition to the target domain, there is a *source domain* which contains a large set of data $\mathcal{D}^s$ labeled w.r.t the same concept or category. Similarly, we have $\mathcal{D}^s = \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{N^s}$, where N^s is the number of instances, $\mathbf{x}_i^s \in \mathbb{R}^{d+1}$ and $y_i^s \in \{-1, +1\}$. The distribution of source data $\mathcal{D}^s$ may be relevant but different from the distribution of target data $\mathcal{D}^t$ in an unknown way. A binary classifier has been trained from the source data $\mathcal{D}^s$ and is denoted as *source classifier* $f^s(\mathbf{x})$. This classifier can be trained using *any* classification algorithms (e.g., SVM, decision tree), but it is subject to a uniform representation as a decision function that predicts the data label as its sign, i.e., $\hat{y} = sgn(f^s(\mathbf{x}))$. For simplicity, we do not distinguish classifier and the decision function of classifier.

The goal of our research is to propose a general framework for adapting a source classifier $f^s(\mathbf{x})$ into a *target classifier* $f(\mathbf{x})$ that works well on the target domain, based on the limited number of labeled examples $\mathcal{D}^t$ in the target domain. As a fundamental difference from many existing methods, our adaptation approach directly updates the decision function of the classifier. It does not use any data in the source domain or even require them to be available.

4.2 A Framework for Classifier Adaptation

The framework for function-level classifier adaptation is proposed based on regularized loss minimization principle. In this framework, the target classifier $f(\mathbf{x})$ has an *additive form*: it is the sum of the source classifier $f^s(\mathbf{x})$ and a delta function $\Delta f(\mathbf{x})$:

$$f(\mathbf{x}) = f^s(\mathbf{x}) + \Delta f(\mathbf{x}) \tag{4.1}$$

This means that the transition from $f^s(\mathbf{x})$ to $f(\mathbf{x})$ is realized by adding a small function $\Delta f(\mathbf{x})$ on top of $f^s(\mathbf{x})$. While it is hard to prove that the additive form is theoretically better than other forms (e.g., multiplicative form $f = f^s \cdot \Delta f$), we will show that this form leads to clear interpretation, great flexibility, and efficient learning algorithms. In Section **??**, we will further show the analogy between this additive adaptation and a Bayesian approach which treats the source classifier $f^s(\mathbf{x})$ as a "prior model".

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 under the influence of source classifier $f^s(\mathbf{x})$. We propose to learn $\Delta f(\mathbf{x})$ in a framework that aims to minimize the regularized empirical risk [48]:

$$\min_{\Delta f} \sum_{i=1}^{N} L(y_i, f(\mathbf{x}_i)) + \lambda \Omega(\|\Delta f\|_{\mathcal{H}}) \tag{4.2}$$

where L is the empirical loss function, $\Omega(\cdot)$ is some monotonically increasing regularization function on the domain $[0, +\infty]$, $\|\cdot\|_{\mathcal{H}}$ is the norm of a function in a reproducing kernel Hilbert space (called RKHS) $\mathcal{H}$ which is a space of functions with certain properties [2], and λ is a scalar.

In Eq.(4.2), the first term measures the classification error (loss) of the target classifier $f(\mathbf{x})$ on the training examples; 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 measures the distance between the source and target classifier in the function space. Hence, the target classifier $f(\mathbf{x})$ learned under this framework must satisfy two goals:

1. *minimal classification error on the training examples;*

2. *minimal distance from the source classifier* $f^s(\cdot)$.

While the second goal does not seem to be as intuitive as the first one, it is as important. If minimal classification error is the only goal, one may find a large number of classifiers achieving the same classification error (even zero classification error when the training size is small), although many of them do not generalize well beyond the training examples. The regularizer in Eq.(4.2) uses the distance to the source classifier as a second criterion for ranking candidate classifiers. This can be justified by our assumption that the source classifier has better-than-random performance on the target domain. The two goals is balanced by constant λ, and in practice its value needs to be determined based on the utility of the source classifier.

We can understand this adaptation approach as making *minimum* changes to the source classifier that are *necessary* to correctly classify the labeled examples. This *"minimum necessary changes"* principle is what underlies our adaptation framework.

[2] A RKHS is a Hilbert space of functions in which pointwise evaluation is a continuous linear functional. If X is an arbitrary set and H a Hilbert space of functions on X, then H is a RKHS if if every linear map of the form $f \mapsto f(x)$ is continuous on any x in X.

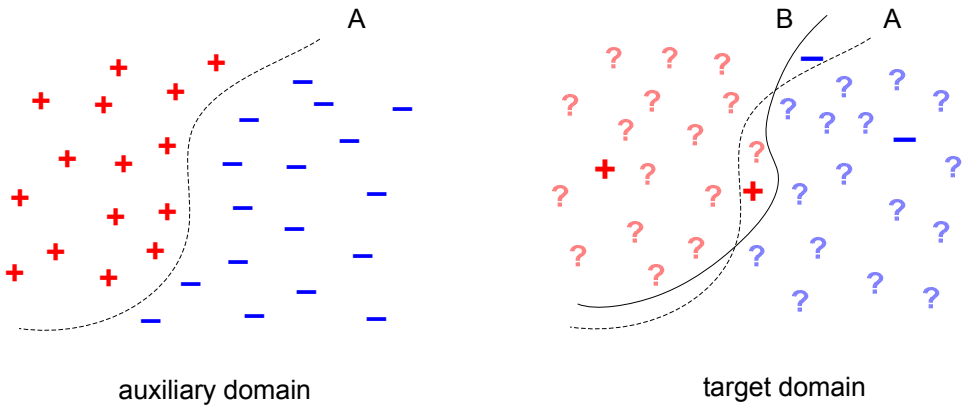


Figure 4.1: An illustration of classifier adaptation, where red '+' denotes positive instances, blue '-' denotes negatives, and red and blue '?' denotes unlabeled positive and negative instances. The decision boundary A is trained from the labeled data in the source domain, B is trained from the labeled data in the target domain, and C is adapted from A.

This is illustrated in Figure 4.1. The classification boundary A is trained from the source domain, and its performance on the target domain is suboptimal due to the distribution change. Our adaptation approach tries to find a new decision boundary B which is slightly modified from A but can classify the target data well.

In terms of bias-variance tradeoff, this framework attempts to reduce the high variance caused by limited training examples using 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 exclusively from limited examples, and using the low-variance but probably-biased source classifier. We expect the adapted classifier achieves better bias-variance tradeoff.

Taking a function-level adaptation approach results in great flexibility and efficiency. This is because our framework directly exploits the source classifier as a *summary* of the knowledge of the source domain, instead of using the raw data in the source domain. By using this compact representation of knowledge, our approach is more efficient than existing adaptation methods that train models over typically a large number of source data [32, 65, 105]. On the other hand, this framework is applicable even when the source data are not available or accessible, which is usually the case in applications involving copyright-protected or privacy-related data such as surveillance video.

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 will focus on two specific algorithms coming out of this framework, which adopt the loss function of support vector machines (SVM) and of kernel logistic regression (KLR) respectively. The first takes the advantage of the discriminative power of SVM, while the second gives a probabilistic interpretation of our framework.

4.3 Adaptive Support Vector Machines (a-SVM)

The empirical success of support vector machines (SVM) in various classification problems has demonstrated the effectiveness of its hinge loss function. By plugging SVM's loss function into the adaptation framework, we reach a specific adaptation algorithm named *Adaptive Support Vector Machines* or *a-SVM*.

4.3.1 Model formulation

In a-SVM, the delta function takes a linear form either in the original feature space as $\Delta f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$, where $\mathbf{w} \in \mathbb{R}^{d+1}$ are the parameters, or in a transformed feature space as $\Delta f(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x})$, where $\phi(\cdot)$ is the feature map projecting the original feature $\mathbf{x}$ into the transformed space. In the latter case, $f^s(\mathbf{x})$ is in fact a non-linear function in the original feature space.

We adopt the hinge loss function of SVM which is expressed as $L(y, f(\mathbf{x})) = (1 - yf(\mathbf{x}))_+ = \max(1-yf(\mathbf{x}), 0)$. Moreover, we use a trivial regularization function $\Omega(x) = x$, and as we will show later, the regularizer $\Omega(\|\Delta f\|_{\mathcal{H}})$ can be rewritten as $\|\mathbf{w}\|^2$. The objective function of a-SVM is given by plugging this loss function and regularizer into the adaptation framework in Eq.(4.2):

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}(1 - y_i f(\mathbf{x}_i))_+ \tag{4.3}$$

where the first term is the regularizer and the second term is the classification error. C is a constant that plays the same role as λ in Eq.(4.2). We use C instead of λ in order to be consistent with the notation of cost factor in standard SVM. By introducing the

slack variable ξ_i to represent the loss on each instance $\mathbf{x}_i$, we can rewrite Eq.(4.3) as:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \tag{4.4}$$
$$\text{s.t. } \xi_i \geq 0, \quad y_i f^s(\mathbf{x}_i) + y_i \mathbf{w}^T \phi(\mathbf{x}_i) \geq 1 - \xi_i, \ \ \forall (\mathbf{x}_i, y_i) \in \mathcal{D}^t$$

While this objective function is very similar to the objective function of standard SVM (Eq.(12.8) in [48]), there is a fundamental difference: here $\mathbf{w}$ denotes the parameters of $\Delta f(\mathbf{x})$ instead of $f(\mathbf{x})$. In fact, we will show that $\|\mathbf{w}\|^2 = \|\Delta f\|^2_{\mathcal{H}} = \|f - f^s\|^2_{\mathcal{H}}$, which shows the regularizer is the distance between the source and target classifier in the function space, instead of "margin" in the case of SVM. Since $\sum_i \xi_i$ measures the classification error of the target classifier $f(\mathbf{x})$, the objective function in Eq.(4.5) seeks a classification boundary (hyperplane) that is close to the boundary of the source classifier, and is meanwhile able to correctly classify the labeled examples in $\mathcal{D}^t$. The cost factor C in a-SVM balances the contribution between the source classifier (through the regularizer) and the training examples. Larger C indicates smaller influence of the source classifier, and vice versa. In practice, C should be decided based on the utility of source classifiers.

By integrating the constraints in Eq.(4.5) using Lagrange multipliers, we can rewrite the objective function as the following (primal) Lagrangian function:

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\mu_i\xi_i - \sum_{i=1}^{N}\alpha_i(y_i f^s(\mathbf{x}_i) + y_i\mathbf{w}^T\phi(\mathbf{x}_i) - (1-\xi_i)) \tag{4.5}$$

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:

$$\mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i \phi(\mathbf{x}_i), \quad \alpha_i = C - \mu_i, \ \ \forall i \tag{4.6}$$

From the above, it is easy to show that $\Delta f(\cdot) = \sum_{i=1}^{N}\alpha_i y_i K(\cdot, \mathbf{x}_i)$, which is a function in the RKHS. Given the definition of inner product in RKHS, we can prove the regularizer $\|\mathbf{w}\|^2$ indeed is equal to the distance between the target classifier $f(\mathbf{x})$ and the source classifier $f^s(\mathbf{x})$ in RKHS.

$$\|f - f^s\|^2 = \|\Delta f\|^2 = \langle \Delta f, \Delta f\rangle = \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{w}\|^2 \tag{4.7}$$

In addition to Eq.(4.6), the Karush-Kuhn-Tucker (KKT) conditions, which the optimal solution of Eq.(4.5) must satisfy, also include:

$$\begin{aligned} \alpha_i\{y_i f^s(\mathbf{x}_i) + y_i \mathbf{w}^T \mathbf{x}_i - (1-\xi_i)\} &= 0 \\ \alpha_i &\geq 0 \\ y_i f^s(\mathbf{x}_i) + y_i \mathbf{w}^T \mathbf{x}_i - (1-\xi_i) &\geq 0 \\ \mu_i \xi_i &= 0 \\ \mu_i &\geq 0 \\ \xi_i &\geq 0 \end{aligned} \tag{4.8}$$

Substituting Eq.(4.6) into Eq.(4.5), we get the Lagrange dual objective function:

$$L_D = \sum_{i=1}^{N}(1-\lambda_i)\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{4.9}$$

where $\lambda_i = y_i f^s(\mathbf{x}_i)$. 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 4.3.3. Given the solutions $\hat{\alpha}$, the target classifier is written as:

$$f(\mathbf{x}) \;\; = \;\; f^s(\mathbf{x}) + \sum_{i=1}^{N}\hat{\alpha}_i y_i K(\mathbf{x}, \mathbf{x}_i) \tag{4.10}$$

where $(\mathbf{x}_i, y_i) \in \mathcal{D}^t$. 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 data.

4.3.2 Discussion

In this section, we discuss several key issues of a-SVM in order to gain deeper insights of its properties and its connections/differences with other methods.

On support vectors. Support vectors of SVM are training examples that are on the classification boundary or on the wrong side of the boundary. Support vectors of a-SVM have a different interpretation. We start by comparing the dual objective function of SVM (Eq.(12.13) in [48]) and a-SVM in Eq.(4.9). The only difference is that the latter has $\{\lambda_i\}_{i=1}^N$ in the first term, where $\lambda_i = y_i f^s(\mathbf{x}_i)$. It is interesting to

see how λ affect the estimation of α. In Eq.(4.9), if $\lambda_i = y_i f^s(\mathbf{x}_i) < 0$, which means the source classifier f^s misclassifies $\mathbf{x}_i$, a larger α_i is desired in order to maximize L_D, and vice versa. This is intuitive because the target classifier f is adapted from f^s with the support vectors $\mathbf{x}_i \in \mathcal{D}^t$, and α_i can be seen as the weight of each support vector. If the source classifier f^s misclassifies $\mathbf{x}_i$, which means the boundary of f^s around $\mathbf{x}_i$ is wrong, then the boundary of the target classifier f around $\mathbf{x}_i$ needs to be changed from f^s in order to correctly classify $\mathbf{x}_i$. This is realized by adding $\mathbf{x}_i$ as a support vector with a large weight α_i. On the contrary, if the source classifier correctly classifies $\mathbf{x}_i$, $f(\mathbf{x}_i)$ does not need to be different from $f^s(\mathbf{x}_i)$, so the weight α_i can be small or even zero. This shows that the support vectors in a-SVM are used to *correct the misclassifications* of the source classifier.

On training cost. A key benefit of function-level adaptation is high efficiency as the result of avoiding training over source data. We show why it is the case in a-SVM. It is clear form from Eq.(4.9) that the number of parameters $\{\alpha_i\}_{i=1}^N$ in a-SVM is equal to the number of target examples N, and not related to the number of source data N^s. It has the same number of parameters as a standard SVM model trained from $\mathcal{D}^t$. Thus, adapting $f^s(\mathbf{x})$ to $f(\mathbf{x})$ using a-SVM is no more expensive than training a SVM model entirely from $\mathcal{D}^t$, except the cost associated with computing $\{\lambda_i\}_{i=1}^N$. Since $\lambda_i = y_i f^s(\mathbf{x}_i)$ remains as a constant throughout the optimization process (see Section 4.3.3), this is an one-time cost of evaluating $f^s(\mathbf{x})$ for N data instances in $\mathcal{D}^t$.

While the cost of computing $\{f^s(\mathbf{x}_i)\}_{i=1}^N$ depends on the complexity of the source classifier f^s, it is linear with N. It is also linear with feature dimension d, because the cost for evaluating a kernel function is linear with d no matter it is linear, polynomial, and RBF kernel. So the extra cost of a-SVM exceeding the cost of SVM is $O(dN)$. Meanwhile, even the most efficient training methods for SVM, such as SVM-Light [56], SMO [76], LIBSVM [22], all have superlinear scaling factor with N. In [56], it has been shown that the time complexity of SVM is linear with N^k where $k \approx 1.7$ for real-valued feature vectors. So the time complexity of training an adapted classifier using a-SVM from $\mathcal{D}^t$ is $O(dN^k + dN) = O(dN^k)$, which in terms of the big-O notation is the same as the complexity of training a SVM model from $\mathcal{D}^t$. It is considerably smaller than the complexity of training a SVM model over all training examples $\mathcal{D}^s \bigcup \mathcal{D}^t$, which is $O((N + N^s)^k)$, because the size of source data is typically much larger than that of

labeled target data, i.e., $N^s >> N$. The experiments to be presented support this analysis.

On the cost factor C. In a-SVM, C balances the classification error and the deviation from the source classifier f^s, with large C emphasizing small classification error and small C emphasizing closeness to f^s. Intuitively, one should use smaller C for a "better" source classifier that works well on the target data, and vice versa. That being said, in practice the absolute value of C is also influenced by the range of $f^s(\cdot)$. Since $0 \leq \alpha_i \leq C$, the range of delta function $\Delta f(\mathbf{x}) = \sum_{i=1}^{N} \hat{\alpha}_i y_i K(\mathbf{x}, \mathbf{x}_i)$ is constrained by C. If the range of $f^s(\cdot)$ is large, say, $[-100, +100]$, C needs to be large enough so that $\Delta f(\mathbf{x})$ is not overwhelmed by $f^s(\mathbf{x})$ when they add up to form the target classifier $f(\mathbf{x})$ (see Eq.(4.10)). Another possibility is to normalize $f^s(\cdot)$ to a fixed range such as $[0, 1]$. We will explore such engineering issues in the later experiments.

Comparison with existing methods. We now discuss the connections and differences between a-SVM and several related methods. First, a-SVM is fundamentally different from methods that re-train a classifier from the examples from source and target domain combined. a-SVM directly modifies the decision function $f^s(\cdot)$, and does not use raw data from the source domain $\mathcal{D}^s$. For this reason, it is also different from cross-domain SVM (CDSVM) proposed in by Jiang et al. [55], which includes part of the source data (support vectors of the source classifier) as additional training data for the new classifier. Second, the decision function of a-SVM as defined in Eq.(4.10) has an additive form, which is similar to an ensemble classifier that combines the source classifier $f^s(x)$ and the delta function $\Delta f(x)$. However, it is different from a genuine ensemble classifier where the component classifiers are trained *independently* from different datasets ($\mathcal{D}^s$ and $\mathcal{D}^t$ in our case). It is clear from Eq.(4.9) that α as parameters of $\Delta f(\cdot)$ is estimated under the influence of $f^s(\cdot)$, so its value would be different if it is estimated *exclusively* from $\mathcal{D}^t$. Finally, a-SVM should not be mixed up with some incremental learning algorithms for SVM, such as those proposed by Syed et al. [94] and by Cauwenberghs and Poggio [21]. Although these methods add support vectors from new data into the existing SVM model, they also modify the original support vectors in the existing model. In comparison, a-SVM does not change the original support vectors in the source classifier.

4.3.3 Learning algorithm for a-SVM

The parameters α of a-SVM are estimated by maximizing the dual objective function defined in Eq.(4.9). This is a quadratic programming (QP) problem, where the number of variables $\{\alpha_i\}_{i=1}^N$ is equal to the number of labeled examples in D^t. The sequential minimal optimization (SMO) algorithm proposed by Platt [76] can efficiently solve a large QP problem by decomposing it into a series of QP subproblems and optimizing them iteratively. We modify the original SMO algorithm to solve a-SVM efficiently.

Before the technical details, it is worthwhile to note an important difference between the SMO algorithm for a-SVM and that for standard SVM. For SVM, the minimum sub-problem tackled in each iteration of SMO optimizes *two* variables. One cannot optimize a single variable at a time because of the linear constraint $\sum_i \alpha_i y_i = 0$ derived from $\frac{\partial L_P}{\partial b} = 0$, where b is the intercept in a decision function $f(x) = \mathbf{w}^T\phi(x) + b$. In contrast, such constraint does not exist in a-SVM and therefore its SMO algorithm optimizes *only one* variable in each iteration. On the surface, as the constraint is derived from $\frac{\partial L_P}{\partial b} = 0$, its absence is because our decision function $f(x) = \mathbf{w}^T\phi(x)$ does not explicitly represent the intercept b but implicitly includes it as a component of $\mathbf{w}$. This notational difference is not critical. The real reason is that in a-SVM, the intercept is implicitly involved in the regularizer $\|\mathbf{w}\|^2$, while in SVM the intercept is *not* part of the regularizer (since it does not affect margin). Even if we explicitly include b in $f(x)$, one cannot derive this linear constraint as long as b is included as part of regularizer, say, in the form of $\|b\|^2$.

The parameter α is the optimal solution to the QP problem in Eq.(4.9) *if and only if* the KKT conditions in Eq.(4.8) are fulfilled. We decompose the optimality condition in Eq.(4.8) according to the value of α_i:

$$\begin{aligned} \alpha_i = 0 \quad &\Rightarrow \quad \mu_i = C, \xi_i = 0 \Rightarrow y_i f(\mathbf{x}_i) \geq 1 \\ 0 < \alpha_i < C \quad &\Rightarrow \quad \mu_i > 0, \xi_i = 0 \Rightarrow y_i f(\mathbf{x}_i) = 1 \\ \alpha_i = C \quad &\Rightarrow \quad \mu_i = 0, \xi_i \geq 0 \Rightarrow y_i f(\mathbf{x}_i) \leq 1 \end{aligned} \tag{4.11}$$

If the above optimal conditions are satisfied for every i, the QP problem is solved, otherwise it is not. Eq.(4.11) provides a method to check the optimality condition of the problem, and also to find variables α_i that violate such condition and need to be optimized. Our SMO algorithm chooses one variable to optimize in each iteration.

While there are many ways to select working variables, we use an intuitive heuristic of selecting the variable α_{i^*} that violates the optimality condition the most:

$$\begin{aligned} i^* &= \underset{i\in\{i_{up},i_{low}\}}{\operatorname{argmax}} |y_i f(\mathbf{x}_i) - 1| \\ \text{where} \quad i_{up} &= \underset{i\in\{t|\alpha_t<C\}}{\operatorname{argmin}} y_i f(\mathbf{x}_i), i_{low} = \underset{i\in\{t|\alpha_t>0\}}{\operatorname{argmax}} y_i f(\mathbf{x}_i) \end{aligned} \tag{4.12}$$

Without loss of generality, 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:

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

where $f^{old}(x)$ is the target classifier Eq.(4.10) evaluated using the existing value of α. This leads to an analytical solution of α_1:

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

Due to $0 \le \alpha_i \le C$, the constrained optimal of α_1 is given by clipping the unconstrained optimal using the following bounds:

$$\alpha_1^{new,clipped} = \begin{cases} C, & \text{if } \alpha_1^{new} \ge C; \\ \alpha_1^{new}, & \text{if } 0 < \alpha_1^{new} < C; \\ 0, & \text{if } \alpha_1^{new} \le 0 \end{cases} \tag{4.13}$$

To summarize the SMO learning algorithm, we start with certain initializations of α, and iteratively choose working variables using Eq.(4.12) and optimize them one at a time using Eq.(4.13). This process continues until the optimality condition in Eq.(4.11) is satisfied up to a certain accuracy.

4.4 Adaptive Kernel Logistic Regression (a-KLR)

In this section, we derive *Adaptive Kernel Logistic Regression* or *a-KLR* from our adaptation framework by adopting the logistic loss function. We will present an alternative, probabilistic interpretation of a-KLR, where it is derived from the maximum-a-posterior (MAP) estimation of a kernel logistic regression (KLR) model with a Gaussian Process prior.

4.4.1 Model formulation

To derive a-KLR from the adaptation framework in Eq.(4.2), we adopt the logistic loss function $L(y, f(\mathbf{x}) = \log(1 + \exp(-yf(\mathbf{x})))$ used by (kernel) logistic regression, and set the delta function as $\Delta f(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x})$ and the regularizer as $\|\mathbf{w}\|^2$. This leads to the following objective function:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \log(1 + \exp(-y_i f(\mathbf{x}_i))) \tag{4.14}$$

By defining $\xi_i = -y_i f(\mathbf{x}_i)$, we can rewrite it as the (primal) Lagrangian form:

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \log(1 + e^{\xi_i}) - \sum_{i=1}^{N} \alpha_i(\xi_i + y_i f^s(\mathbf{x}_i) + y_i \mathbf{w}^T\mathbf{x}_i) \tag{4.15}$$

where $\{\alpha_i\}_{i=1}^N$ are Lagrange multipliers. To minimize L_P, we set its derivative against $\mathbf{w}$ and ξ_i to zero, which gives $\mathbf{w} = \sum_{i=1}^N \alpha_i y_i \phi(\mathbf{x}_i)$ and $\xi_i = \log \frac{\alpha_i}{C-\alpha_i}$. Plugging them into the primal form in Eq.(4.15), we have the dual Lagrangian function:

$$\begin{aligned} L_D &= -\sum_{i=1}^{N} \alpha_i y_i f^s(\mathbf{x}_i) - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ &\quad -\sum_{i=1}^{N} (C - \alpha_i)\log(C - \alpha_i) - \sum_{i=1}^{N} \alpha_i \log \alpha_i \end{aligned} \tag{4.16}$$

The parameters $\{\alpha_i\}_{i=1}^N$ are estimated by maximizing the above dual form using the learning algorithm to be described in Section 4.4.3. The target classifier has exactly the same form as that of a-SVM, i.e., $f(\mathbf{x}) = f^s(\mathbf{x}) + \sum_{i=1}^N \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i)$.

4.4.2 Probabilistic interpretation

We show that a-KLR can be also derived from a probabilistic perspective where the source classifier is treated as "prior model" of the target classifier. This alternative derivation is based on *maximum-a-posterior* (MAP) estimation. Following the representation of (kernel) logistic regression [48], we define the conditional probability as $p(y|\mathbf{x}) = 1/(1 + \exp(-yf(\mathbf{x})))$. Thus, the log-likelihood of the training examples is represented as:

$$l(\mathcal{D}^t; f) = \sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i) = -\sum_{i=1}^{N} \log(1 + e^{\xi_i}) \tag{4.17}$$

where $\xi_i = -y_i f(\mathbf{x}_i)$. This log-likelihood corresponds to the second term of a-KLR's objective function in Eq.(4.14), which represents empirical loss.

From a Bayesian point of view, the regularizer $\|\mathbf{w}\|^2$ is the outcome of a Gaussian prior distribution on parameters, i.e., $\mathbf{w} \sim \mathcal{N}(0, C\mathbf{I})$, where $\mathbf{I}$ is an identity matrix. The distribution $p(\mathbf{w})$ specifies the properties of the considered function space of $f(\mathbf{x})$ in terms of the *mean function*:

$$E(f(\mathbf{x})) = f^s(\mathbf{x}) + E(\mathbf{w}^T)\phi(\mathbf{x}) = f^s(\mathbf{x})$$

and *covariance function*:

$$Cov(f(\mathbf{x}), f(\mathbf{x}')) = E(\mathbf{w}^T\phi(\mathbf{x}) \cdot \mathbf{w}^T\phi(\mathbf{x}')) = Var(\mathbf{w})K(\mathbf{x}, \mathbf{x}') = CK(\mathbf{x}, \mathbf{x}')$$

This means the target classifier $f(\mathbf{x})$ follows a prior *Gaussian process* (GP) [81] with mean function $f^s(x)$ and its covariance function $CK(\mathbf{x}, \mathbf{x}')$:

$$f(\mathbf{x}) \sim \mathcal{GP}(f^s(\mathbf{x}), CK(\mathbf{x}, \mathbf{x}'))$$

GP is a distribution in the function space. This GP prior in particular favors target classifiers $f(\mathbf{x})$ that are close to the source classifier $f^s(\mathbf{x})$ in the function space. According to the definition of GP [81], a finite sample of $f(\mathbf{x})$ as $\mathbf{f} = \{f(\mathbf{x}_1), .., f(\mathbf{x}_N)\}$, where $\mathbf{x}_i \in \mathcal{D}^t$, follow a joint Gaussian distribution with mean as $\mathbf{f}^s = \{f^s(\mathbf{x}_1), .., \mathbf{f}^s(\mathbf{x}_N)\}$ and covariance matrix as $C\mathbf{K}$ where $\mathbf{K} = [K(\mathbf{x}_i, \mathbf{x}_j)]_{N\times N}$, i.e., $\mathbf{f} \sim \mathcal{N}(\mathbf{f}^s, C\mathbf{K})$.

To learn the target classifier $f(\mathbf{x})$, we resort to the *maximum-a-posterior* (MAP) estimation instead of a fully Bayesian approach. If we treat $\mathbf{f}$ as the model parameters, the logarithm of the posterior distribution is specified by the log-likelihood of data and the prior $p(\mathbf{f})$:

$$\begin{aligned}
\log p(\mathbf{f}|D) &\propto l(D; \mathbf{f}) + \log p(\mathbf{f}) \\
&= -\sum_{i=1}^{N} \log(1 + e^{\xi_i}) - \frac{1}{2C}(\mathbf{f} - \mathbf{f}^s)^T\mathbf{K}^{-1}(\mathbf{f} - \mathbf{f}^s) + const \\
&= -\sum_{i=1}^{N} \log(1 + e^{\xi_i}) - \frac{1}{2C}\|\mathbf{w}\|^2 + const
\end{aligned} \tag{4.18}$$

which is to be maximized subject to $\xi_i = -y_i f(\mathbf{x}_i)$. This maximization problem is identical to the minimization problem of a-KLR in Eq.(4.14). This shows that the a-KLR model can be equivalently derived from a fully probabilistic perspective.

This probabilistic interpretation does not extend to other methods under our adaptation framework, such as a-SVM, because their loss functions cannot be interpreted as conditional probabilities. Nevertheless, it sheds light on the role of source classifier in this framework, which is a prior in the hypothesis (function) space for the target classifier $f(\mathbf{x})$. The regularizer $\|\Delta f\|_{\mathcal{H}}$ penalizes any hypothesis $f(\mathbf{x})$ that deviates from this prior.

4.4.3 Learning algorithm for a-KLR

Similar to a-SVM, the parameters of a-KLR are also estimated by minimizing its dual Lagrangian form L_D in Eq.(4.16). This is a quadratic programming (QP) problem solved using another variation of SMO algorithm we proposed. It is inspired by the SMO algorithm for kernel logistic regression proposed by Keerthi et al. [59].

For simplicity, we define $F_i = -y_i f(\mathbf{x}_i) + \log(C - \alpha_i) - \log \alpha_i$. To maximize L_D defined in Eq.(4.16), we can set its derivative against every α_i to zero:

$$\frac{\partial L_D}{\partial \alpha_i} = -y_i f(\mathbf{x}_i) + \log(C - \alpha_i) - \log \alpha_i = F_i \triangleq 0 \tag{4.19}$$

This provides a method for checking the optimality condition (i.e., $F_i = 0, \forall i$) and for finding variables that violate the condition. Following the same reason discussed in Section 4.3.3, only one working variable is optimized in each iteration of this SMO algorithm. We select the one that violates the optimality condition the most as our working variable, i.e., $i* = \text{argmax}_i |F_i|$.

Without loss of generality, let α_1 be the current working variable. Unlike in a-SVM, here we cannot derive an analytical solution of α_1 from Eq.(4.19), so we resort to the Newton-Raphson (NR) method to optimize α_1 iteratively. Suppose $\alpha_1^{new} = \alpha_1^{old} + t$. In the NR method, t is iteratively updated using the following equation:

$$t^{l+1} = t^l - \frac{\partial L_D}{\partial t} \left(\frac{\partial^2 L_D}{\partial t^2} \right)^{-1} \tag{4.20}$$

where

$$\begin{aligned} \frac{\partial L_D}{\partial t} &= \frac{\partial L_D}{\partial \alpha_1} \frac{\partial \alpha_1}{\partial t} = -y_1 f^{old}(\mathbf{x}_1) - tK(\mathbf{x}_1, \mathbf{x}_1) + \log(C - \alpha_1^{old} - t) - \log(\alpha_1^{old} + t) \\ \frac{\partial^2 L_D}{\partial t^2} &= -K(\mathbf{x}_1, \mathbf{x}_1) - \frac{1}{C - \alpha_1^{old} - t} - \frac{1}{\alpha_1^{old} + t} \end{aligned}$$

Starting from $t^0 = 0$, we repeatedly invoke Eq.(4.20) to update t until a certain accuracy is reached or the constraint $0 < \alpha_1 < C$ becomes tight. The resulting t is used to update α_1. We iteratively optimize selected working variables until the optimality condition is satisfied, i.e., $F_i = 0, \forall i$.

4.5 Experimental Results

We experiment with adaptive SVMs (a-SVM) as the representative of the proposed adaptation methods on both synthetic data and the application of cross-domain video concept detection.

4.5.1 Alternative approaches

Given a fully-labeled source domain with a classifier trained from it and a sparsely-labeled target domain, there are many approaches to classifying the unlabeled data in the target domain. Adapting the source classifier is one approach, but there are other alternative approaches, which are discussed below. The first three approaches exploit the knowledge in one of the two domains, while the last three approaches exploit both the source and target domain.

- **Keeping source classifiers:** This approach directly applies the classifier $f^s(\mathbf{x})$ trained from the source domain directly to the target domain, without any adaptation. It does not use any labeled data from the target domain either.

- **Building new classifiers:** Contrary to the first approach, this approach ignores the source domain completely and builds a new classifier classifier $f^t(\mathbf{x})$ entirely from the labeled examples $\mathcal{D}^t$ in the target domain.

- **Semi-supervised learning (SSL):** The limited number of labeled data and the large number of unlabeled data in the target domain makes semi-supervised learning [26] an appealing approach, which aims to improve the classification performance by learning from both the labeled and unlabeled data. Transductive SVMs (TSVM) [58] is a widely used SSL method adapted from SVMs. We build classifiers using TSVM from both the labeled and unlabeled data in the target

domain. The labels for the unlabeled data are predicted in the training process of TSVM.

- **Aggregation (data-fusion):** This computationally expensive approach learns a single classifier using *all* the labeled instances combined from the source and target domain, i.e., $\mathcal{D}^s \bigcup \mathcal{D}^t$. The instances from the two domains are weighted according to their relative importance. In the case of SVM, the weights are implemented through the different cost factors of the instances. The decision function is expressed as $f^{aggr}(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x})$ with $\mathbf{w}$ estimated from the following objective function:

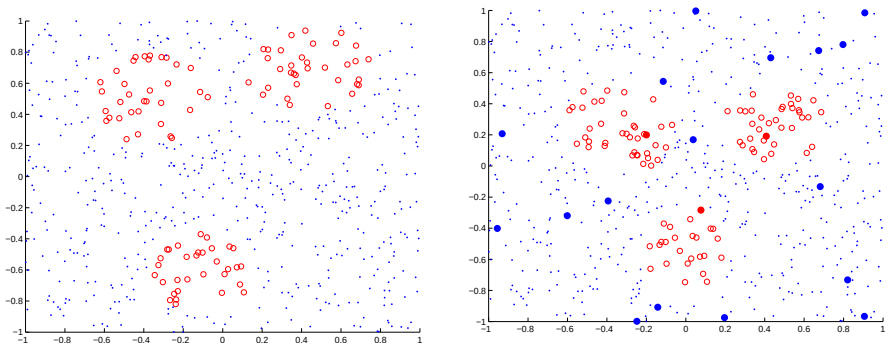
$$\begin{aligned} & \min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i + C^s\sum_{i=1}^{N^s}\xi_i^s \\ \text{s.t.} \quad & \xi_i \geq 0, \quad y_i\mathbf{w}^T\phi(\mathbf{x}_i) \geq 1-\xi_i, \ \ \forall(\mathbf{x}_i, y_i) \in \mathcal{D}^t \\ & \xi_i^s \geq 0, \quad y_i^s\mathbf{w}^T\phi(\mathbf{x}_i^s) \geq 1-\xi_i^s, \ \ \forall(\mathbf{x}_i^s, y_i^s) \in \mathcal{D}^s \end{aligned} \tag{4.21}$$

  where C and C^s are the cost factors for instances in the target domain and in the source domain, respectively. This approach represents the data-level transfer learning techniques used in many existing works [34, 61, 65, 105]. It is fundamentally different from our adaptation approach in that it involves the source data in the training process while our approach directly manipulates the source classifiers.

- **Ensemble (result-fusion):** While the aggregation approach combines the training data, the ensemble approach combines the output of two independently trained classifiers, namely the source classifier $f^s(\mathbf{x})$ and the new classifier $f^t(\mathbf{x})$ trained from the labeled examples $\mathcal{D}^t$ in the target domain. Its result is equal to a weighted sum of output of these two classifiers:

$$f^{ens}(\mathbf{x}) = Cf^t(\mathbf{x}) + C^sf^s(\mathbf{x}) \tag{4.22}$$

  where C and C^s are used as the weights of the two classifiers. This has been the approach used in [62, 102]. Note that although in our approach an adapted classifier also has an additive form $f(\mathbf{x}) = f^s(\mathbf{x}) + \Delta f(\mathbf{x})$, it is different from



(a) Source domain D_{src} (b) Target domain D_{tgt}

Figure 4.2: The data distribution of (a) source domain D_{src} and (b) target domain D_{tgt}. Small red circles denote positive instances and small blue dots denote negative instances. The larger blue dots in D_{tgt} indicate 20 labeled instances.

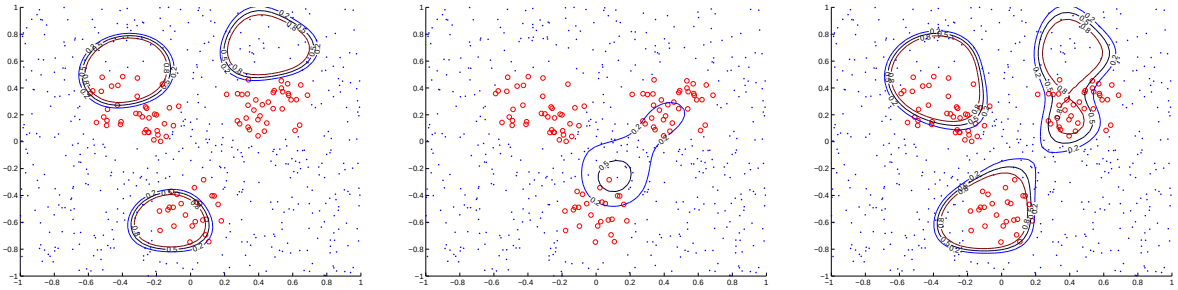
this ensemble approach in that $\Delta f(\mathbf{x})$ is learned under the influence of $f^s(\mathbf{x})$. In contrast, in this ensemble approach the two base classifiers $f^s(\mathbf{x})$ and $f^t(\mathbf{x})$ are learned independently.

- **Adaptation approach:** We use the proposed a-SVM algorithm to adapt the classifier $f^s(\mathbf{x})$ trained from the source domain to a new classifier $f(\mathbf{x})$ based on the labeled target examples $\mathcal{D}^t$. The cost factor in a-SVM is set to C.

In the experiments, care is taken to ensure that these approaches are comparable. All the approaches use SVM as the classification algorithm and the empirically successful RBF kernel function, i.e., $K(x_i, x_j) = e^{-\gamma\|x_i - x_j\|^2}$. Moreover, we ensure that the cost factor C^s for the source domain as well as C^s for the target domain are the same across different approaches. We set $C^s = 1$ in all the experiments, but vary the value of C to study its impact on the classification performance.

4.5.2 Synthetic data

To illustrate how a-SVM works, we generate two synthetic data sets D_{src} and D_{tgt} from two different distributions in a 2-d feature space, which represent the data in the source and target domain, respectively. Each data set has 100 positive and 500 negative data. The positive data in each set are generated from a Gaussian mixture model with 3 Gaussian components, and the negative data are sampled uniformly outside the area



(a) SVM$_{src}$ (20.5% error) (b) SVM$_{tgt}$ (16.7% error) (c) a-SVM (15.0% error)

Figure 4.3: Plotted on the target domain D_{tgt}, the decision boundary of (a) SVM$_{src}$, the source classifier trained from all the 600 instances in D_{src}, (b) SVM$_{tgt}$, a new classifier trained from the 20 labeled instances in D_{tgt}, and (c) a-SVM, the classifier adapted from SVM$_{src}$ using 20 instances in D_{tgt}.

of the positive data. In source domain D_{src}, the 3 Gaussian components are centered at $(-0.4, 0.5)$, $(0.5, 0.7)$, and $(-0.1, -0.6)$, while in target domain D_{tgt} their means shift to $(-0.4, 0.3)$, $(0.5, 0.3)$, and $(0, -0.65)$. Figure 4.2 (a) and (b) shows the distribution of D_{src} and D_{tgt}.

We assume *all* the instances in D_{src} are labeled, while only 20 instances are labeled in D_{tgt}, including 3 positive instances and 17 negative instances. Suppose SVM$_{src}$ is the (source) classifier trained from all the 600 instances in D_{src} using SVM, and SVM$_{tgt}$ is a new classifier trained from the 20 labeled instances in D_{tgt}. As shown in Figure 4.3(a), the decision boundary of SVM$_{src}$ centers around the positive data in D_{src} and does not align well with the positive data in D_{tgt}, whose centers have sifted from D_{src}. Therefore, SVM$_{src}$ is biased with respect to data in D_{tgt} and has a relatively high error rate. On the other hand, the unbiased new classifier SVM$_{tgt}$ suffers a large variance due to the limited training data, and its boundary in Figure 4.3(b) does not correspond well to the distribution of D_{tgt} either. This shows that relying on knowledge of a single domain is insufficient.

The a-SVM classifier is adapted from SVM$_{src}$ based on the 20 labeled instances in D_{tgt}. As shown by Figure 4.3(c), its decision boundary captures the locations of positive data more precisely than SVM$_{src}$ and SVM$_{tgt}$, and its error rate is also the smallest among the three. We can conceive the adaptation process visually as "dragging" the boundary of SVM$_{src}$ towards the shifted centers of positive data in D_{tgt}. We attribute

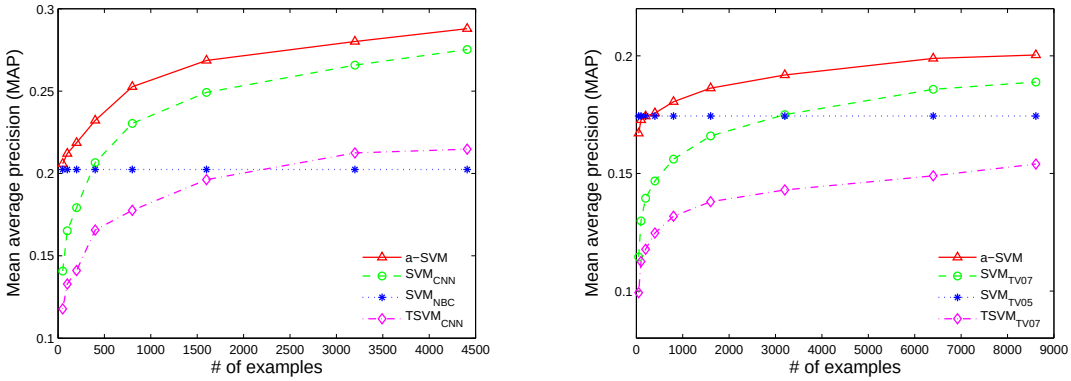
the effectiveness of a-SVM to a good bias-variance tradeoff: the adapted classifier has a low bias by training from only the instances in D_{tgt}, and meanwhile achieves a low variance through a regularizer penalizing its difference from SVM_{src} as a prior model.

4.5.3 Cross-domain semantic concept detection

In cross-domain concept detection, we adapt classifiers of semantic concepts trained from one video domain to the other domains. The experiment is conducted on the TRECVID 2005 development set (TV05DEV), which contains news video from 6 channels, and the TRECVID 2007 development (TV07DEV) containing documentary videos. The details of the two collections can be found in Section 3. The labels of 39 semantic concepts are available on both collections. Two types of experiments are performed: cross-channel experiment and cross-genre experiment.

In the cross-channel experiment, we treat NBC data as the source domain, and CNN data as the target domain, which are two new channels in TV05DEV. We assume *all* the video shots in NBC are labeled w.r.t the 39 concepts, and a classifier is trained for each concept from the data. The video shots in CNN are partitioned into a *development set* and an *evaluation set*. Along the temporal axis, the first 40% of the shots in CNN belong to the development set, and the remaining 60% belong to the evaluation set. We label a certain number of shots *randomly* selected from the development set, and treat them as the training examples. Our goal is to adapt a concept classifier trained from NBC to a classifier for CNN based on its limited training examples. The adapted classifier is evaluated on the evaluation set in terms of average precision (AP). In the experiment, we iterative over all the 39 concepts, and for each concept repeat the experiment using 4 sets of random samples in order to reduce the variance caused by random sampling. The performance is measured by mean average precision (MAP) averaged over all 39 concepts and 4 random iterations for each concept.

In the cross-genre experiments, we treat the entire TV05DEV collection as the source domain and entire TV07DEV as the target domain. Similarly, TV07DEV is partitioned to a development set and a evaluation set in 40/60 ratio. Classifiers trained from the entire TV05DEV set are adapted to TV07DEV based on the randomly selected examples in its development set, and the adapted classifiers are evaluated on its evaluation set.



(a) Performance on CNN (b) Performance on TV07DEV

Figure 4.4: Comparison of a-SVM and non-adaptation methods in (a) cross-channel (NBC/CNN) setting and (b) cross-genre (TV05/TV07) setting.

Figure 4.4 (a) compares the average performance of 4 methods in cross-channel experiment, including SVM_{NBC} as source classifiers trained from NBC data, SVM_{CNN} as new classifiers trained exclusively from the labeled data in CNN, a-SVM as classifiers adapted from SVM_{NBC} using a-SVM, and TSVM_{CNN} as transductive SVM classifiers trained using both the labeled and unlabeled data in CNN. All the classifiers are trained using cost factor $C = 1$ and $C^s = 1$, and RBF kernel function $K(x_i, x_j) = e^{-\rho\|x_i - x_j\|^2}$ with $\rho = 0.1$. We increase the number of training examples in CNN from 50 to 4410, which is the size of the entire development set, each time doubling the number of examples. Because the average percentage of positive shots for the 39 concepts is only 7%, the number of positive data available for training is actually very small despite the seemingly large number of total training examples.

From Figure 4.4 (a) we see that on average, the adapted classifiers trained by a-SVM outperform both SVM_{NBC} and SVM_{CNN} by substantial margins. The gap between a-SVM and SVM_{CNN} is significant when the labeled examples are scarce, because SVM_{CNN} relies exclusively on the labeled examples while $a - SVM$ also relies on the source classifier. The performance of both methods improves as more labeled data become available, and the gap between them diminishes. Nevertheless, a-SVM is still better than SVM_{CNN} even when all the data in the development set are used for training. This shows for difficult tasks such as semantic concept detection, adaptation is beneficial even when there is a large amount of training data. From another

perspective, compared with building new classifiers, adaptation requires much fewer training data to reach the same performance. For example, $a - SVM$ needs only 50 instances to reach the same MAP that SVM_{CNN} achieves using 400 examples. This shows the benefit of leveraging the knowledge of source domain (classifiers) in terms of improving performance or reducing labeling effort. Compared with source classifier SVM_{NBC}, a-SVM performs close to it when the training data are scarce, and then outperforms it with an increasing margin as more training data arrives. The performance of SVM_{NBC}, on the other hand, remains the same due to no adaptation. TSVM_{CNN} has the worst performance, suggesting that using unlabeled data hurts the performance in this particular case.

This result shows that a-SVM is able to take advantages of "the best of two worlds". When there is little or no labeled data, a-SVM exploits the knowledge in source classifiers SVM_{NBC} and avoid a "cold start" which happens to the new classifiers. When more labeled data become available, it takes advantages of the labeled data and quickly improves its performance. While SVM_{NBC} suffers bias due to distribution shift, and SVM_{CNN} suffers large variance due to limited training data, a-SVM hits a good bias-variance tradeoff and outperforms the two.

We compare the performance of the same set of methods in cross-genre experiment in Figure 4.4 (b). The relationship between these methods stay the same, with a-SVM outperforming both SVM_{TV05}, the source classifiers trained from TV05DEV, and SVM_{TV07}, the new classifiers trained from TV07DEV. The only difference is that a-SVM and SVM_{TV07} improves more slowly as the training examples increase, suggesting a greater difficulty of concept detection in TV07DEV.

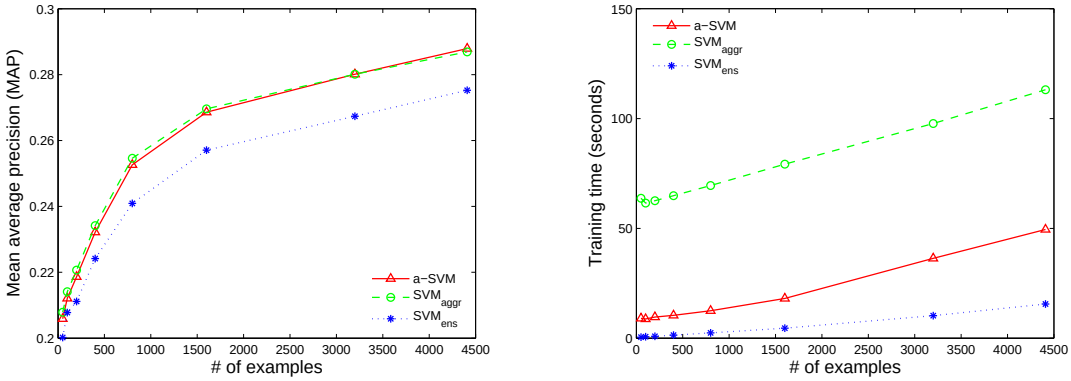
While a-SVM has the highest average performance, it is also important to know whether it consistently excels on each single concept. Table 4.1 compares the performance of different classifiers on the 39 concepts in the NBC/CNN setting and TV05/TV07 setting. In either setting, a-SVM is the top performer on majority of the concepts, beating the source classifiers (SVM_{NBC} or SVM_{TV05}) and new classifiers (SVM_{CNN} or SVM_{TV07}) in all but 9 concepts. Sometimes, the source classifiers generalize so well to the new domains that their performance cannot be further improved through adaptation, especially in the TV05/TV07 setting. In other cases, the source classifier is unless and it actually hurts a-SVM such that it performs worse than the

	Performance on CNN			Performance on TV05DEV		
	SVM_{NBC}	SVM_{CNN}	a-SVM	SVM_{TV05}	SVM_{TV07}	a-SVM
Airplane	0.021	0.030	**0.032**	0.007	0.010	**0.011**
Animal	0.012	**0.639**	0.635	0.041	**0.044**	0.043
Boat_Ship	n/a	n/a	n/a	0.021	0.030	**0.032**
Building	0.152	0.187	**0.199**	0.252	0.302	**0.336**
Bus	n/a	n/a	n/a	0.006	0.006	**0.007**
Car	0.306	0.220	**0.340**	**0.104**	0.095	0.100
Charts	0.021	0.049	**0.050**	**0.030**	0.022	0.021
Computer_TV-screen	0.102	0.129	**0.145**	0.093	0.128	**0.148**
Corporate-Leader	0.010	0.017	**0.021**	n/a	n/a	n/a
Crowd	**0.372**	0.238	0.295	0.412	0.367	**0.413**
Desert	0.050	0.063	**0.071**	0.012	0.013	**0.013**
Entertainment	0.004	0.013	**0.014**	n/a	n/a	n/a
Explosion_fire	0.031	**0.106**	0.100	**0.006**	0.004	0.005
Face	0.808	0.799	**0.829**	0.812	0.775	**0.812**
Flag-US	0.077	0.065	**0.083**	n/a	n/a	n/a
Government-Leader	**0.151**	0.105	0.110	n/a	n/a	n/a
Maps	0.106	**0.470**	0.429	0.020	0.026	**0.032**
Meeting	0.024	0.033	**0.035**	0.088	0.114	**0.120**
Military	0.159	0.285	**0.299**	**0.073**	0.030	0.033
Mountain	0.051	0.177	**0.181**	**0.030**	0.017	0.023
Natural-Disaster	n/a	n/a	n/a	**0.013**	0.004	0.005
Office	0.039	0.046	**0.052**	0.171	0.280	**0.282**
Outdoor	0.671	0.739	**0.766**	0.670	0.715	**0.742**
People-Marching	0.140	0.087	**0.151**	0.065	0.094	**0.105**
Person	0.879	0.882	**0.900**	0.890	0.866	**0.891**
Police_Security	n/a	n/a	n/a	**0.034**	0.023	0.023
Prisoner	n/a	n/a	n/a	**0.008**	0.007	0.007
Road	**0.210**	0.127	0.203	0.207	0.260	**0.285**
Sky	**0.559**	0.347	0.481	0.657	0.637	**0.686**
Snow	0.030	**0.149**	0.074	0.031	0.039	**0.049**
Sports	0.026	0.137	**0.161**	0.108	0.055	**0.108**
Studio	0.685	0.758	**0.778**	0.030	0.034	**0.043**
Truck	0.013	0.021	**0.021**	0.025	0.024	**0.027**
Urban	0.116	0.165	**0.183**	0.213	0.293	**0.304**
Vegetation	0.264	0.304	**0.332**	0.509	0.564	**0.588**
Walking_Running	**0.199**	0.183	0.187	0.174	0.208	**0.217**
Waterscape_Waterfront	0.187	0.403	**0.439**	0.094	0.199	**0.219**
Weather	n/a	n/a	n/a	0.024	0.030	**0.032**
average	0.202	0.249	0.269	0.174	0.186	0.199

Table 4.1: Per-concept performance of several methods in NBC/CNN setting (with 1,600 training examples in CNN) and in TV05/TV07 setting (with 6,400 training examples in TV07DEV). "n/a" denotes skipped concepts due to insufficient positive data (below 10).

new classifier trained from scratch.

While a-SVM performs well on the majority of the concepts, its robustness can be further improved. One possibility is to make more educated decisions on whether to adapt each classifier, rather than blindly adapting every classifier. Ideally, one should be able to predict how well a classifier generalize to the new data in order to decide whether to keep the original classifier, improve it by adaptation, or replace it with a new classifier built from scratch. We will investigate this problem in Section 6.



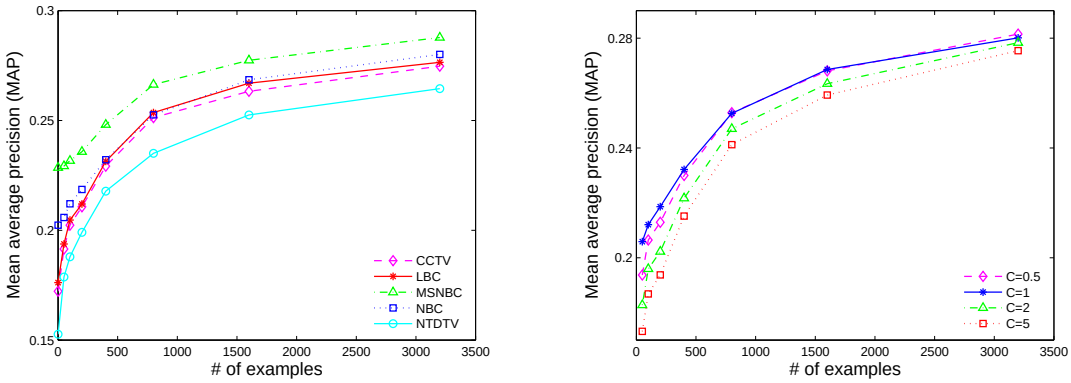
(a) Performance on CNN

(b) Average training time

Figure 4.5: Comparison of alternative adaptation methods (a) performance on CNN, and (b) average training time per concept.

We have shown the advantages of a-SVM over methods that relies on knowledge from only a single domain. It is important to compare it with methods that exploit the knowledge in both the source and target domain, namely the aggregation and ensemble method. The former trains a classifier SVM$_{aggr}$ from labeled data combined from both domains, while the latter employs an ensemble SVM$_{ens}$ that combines the outputs of two classifiers trained separately from the labeled data in the two domains. These two methods can be seen as alternative adaptation methods as well.

Figure 4.5(a) shows the performance of a-SVM, SVM$_{aggr}$, and SVM$_{ens}$ in the cross-channel (NBC/CNN) setting described above. We find a-SVM performs comparably with SVM$_{aggr}$, and better than SVM$_{ens}$ by a moderate margin. All the three methods perform better than SVM$_{CNN}$ and SVM$_{NBC}$, showing the benefit of exploiting knowledge from both domains rather than from a single domain. Moreover, to find out the training cost of these methods, we compare their average (per-concept) training time in Figure 4.5(b). Here, we assume source classifiers already exist, and their training cost are not taken into account. We see that the training time of a-SVM is considerably lower than that of SVM$_{aggr}$, especially when the size of training examples is small. This is due to the fact that while a-SVM is trained over only the limited examples in the target domain, the training of SVM$_{aggr}$ involves a large amount of data in the source domain. Because these two methods have comparable performance, we prefer a-SVM over SVM$_{aggr}$ based on its higher efficiency.



(a) a-SVM with different source domains

(b) a-SVM with different cost factors C

Figure 4.6: Sensitivity of a-SVM's performance on CNN data w.r.t (a) the choice of source domain (while C is fixed to 1), and (b) cost factors C (while the source domain is NBC).

We proceed to investigate the sensitivity of a-SVM. We first examine the choice of source domain (classifier) and its impact on the performance of adaptation. While previously we use NBC as the only source channel, we now use each of the 6 news channels in TV05DEV besides CNN as the source domain, which include CCTV, LBC, MSNBC, NBC, and NTDTV. For each source channel, we build a source classifier from all of its data, and apply a-SVM to adapt it to CNN based on the setting described above. Figure 4.6 (a) plots the average performance of 5 a-SVM runs, each using a different source channel (classifier), against the number of labeled examples. We also include the performance with zero examples, which is the performance of the *original* source classifiers on CNN before adaptation. It is clear that the choice of source channels has a substantial impact on the performance of the adapted classifiers, since the gap between different runs is non-trivial. So if more than one source domain exist, choosing the best one is another important research problem.

It worths noting that the performance of an adapted classifier is closely tied to the performance of the original source classifier on CNN. As shown in Figure 4.6 (a), MSNBC's classifiers have the highest average performance before adaptation (i.e., when the number of examples is zero), and the classifiers adapted from them also have the highest performance. This is intuitive because a better source classifier provides a higher starting point for adaptation, and this advantage is "inherited" by the classifier

adapted from it. However, it is infeasible to choose source classifiers based on their cross-domain performance, because such performance is unknown unless the target CNN data are completely labeled (which, however, would make adaptation unnecessary). In Section 6, we will discuss an empirical approach to predict such performance without labeling the target data.

Another performance-sensitive factor is the trade-off between the dependence on the source classifier and on the labeled examples. This is essentially the trade-off between the two objectives of our adaptation framework. The trade-off is controlled by the cost factor C in the objective function of a-SVM given by Eq.(4.5). As discussed in Section 4.3, a larger C emphasizes more on minimizing the classification error on the labeled examples, while smaller C emphasizes on the similarity between the source and adapted classifier. Figure 4.6 (b) compares several a-SVM runs with difference values of C, all trained with NBC as the source domain and CNN as the target domain. The choice of C indeed has an impact on the performance of a-SVM, but the impact is not as great as that of the choice of source classifier. In this particular setting, $C = 1$ has the best average performance, but it is unlikely the optimal choice for every concept. The optimal C depends on the source and target domain as well as the concept in question.

The choice of cost factor should be based on the same factor as in the choice of source domain: the utility of a source classifier with respect to the target domain. One should use a smaller C for a source classifier that is useful to the target domain, and a larger C for a useless source classifier. We will explore this problem in Section 6. So far C is chosen through cross-validation which is computationally expensive.

4.6 Summary

We have proposed a general framework for function-level classifier adaptation, which enjoys high efficiency and broad applicability due to its direct manipulation of classifiers' decision function and the freedom from training on "old data". We have also described adaptive support vector machine (a-SVM) and adaptive kernel logistic regression (a-KLR) as two concrete algorithms derived from this framework. The experiments on cross-domain concept detection have demonstrated the effectiveness of our approach in

comparison with alternative methods.

The proposed method is for adapting one classifier into another. In practice, there is often a need and benefit in adapting multiple classifiers into one classifier. For example, instead of adapting a NBC's classifier to CNN, we can leverage the complementary knowledge in various news channels (NBC, CCTV, LBC, etc) by incorporating the classifiers trained from all these channels in the adaptation process. As implied by Figure 4.6(a), different source classifiers are not equally useful in adaptation, so weighting these classifiers according to their utility is an important question. This kind of many-to-one adaptation paradigm will be addressed in Chapter 5.

Chapter 5

Multi-Classifier Adaptation

In Chapter 4, we have described a framework and two algorithms for adapting one classifier into another, namely single-classifier adaptation. In practice, there are usually multiple existing classifiers (and domains) that are helpful to the classification task in the target domain. To take the advantage of them, we extend our previous framework to perform multi-classifier adaptation, namely adapting multiple source classifiers into one target classifier. From this extended framework, we derive *multi-adaptive SVM* as a specific algorithm for multi-classifier adaptation.

The following notations are used in this chapter. Suppose there are M the source domains denoted as $\mathcal{D}_1^s, ..., \mathcal{D}_M^s$. In each domain $\mathcal{D}_k^s$, there are N_k^s labeled instances as $\mathcal{D}_k^s = \{(\mathbf{x}_i^k, y_i^k)\}_{i=1}^{N_k^s}$, where $\mathbf{x}_i^k \in \mathbb{R}^{d+1}$ denotes features and $y_i^k \in \{-1, +1\}$ denotes the label. The data distribution in these source domains may differ from that in the target data in different ways. We have trained a source classifier $f_k^s(\mathbf{x})$ from the data in each source domain $\mathcal{D}_k^s$.

5.1 An Extended Framework for Multi-Classifier Adaptation

As the first step of adaptation, we combine multiple source classifiers as a weighted sum, or an *ensemble*, in the form of $\sum_k t_k f_k^s(\mathbf{x})$, where $\mathbf{t} = \{t_k\}_{k=1}^M$ are their weights. We treat this ensemble as a single classifier to be adapted to the target classifier $f(\mathbf{x})$. Similar to the framework for single-classifier adaptation, this target function is the sum

of the ensemble of source classifiers and a delta function, expressed as:

$$f(\mathbf{x}) = \sum_{k=1}^{M} t_k f_k^s(\mathbf{x}) + \Delta f(\mathbf{x}) \tag{5.1}$$

The key to multi-classifier adaptation is determining the weights $\{t_k\}_{k=1}^M$ of source classifiers. The weights can be set manually based on their utility to the target domain. In this case, the ensemble $\sum_k t_k f_k^s(\mathbf{x})$ is *fixed* before the adaptation process, and this becomes a single-classifier adaptation problem which can be solved using our approaches in Chapter 4. However, manually defining the weights is rarely practical or desirable given the difficulty of knowing a classifier's utility to the (unlabeled) target domain. Therefore, we extend our adaptation framework to allow the weights $\{t_k\}_{k=1}^M$ learned *automatically* and *simultaneously* with the target classifier $f(\mathbf{x})$ in one process. This is realized by adding another regularizer $\Psi(\|\mathbf{t}\|)$ to the regularized loss minimization framework:

$$\min_{\Delta f, \mathbf{t}} \sum_{i=1}^{N} L(y_i, f(\mathbf{x}_i)) + \lambda \Omega(\|\Delta f\|_{\mathcal{H}}) + \beta \Psi(\|\mathbf{t}\|^2) \tag{5.2}$$

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, which means this framework seeks to minimize the overall contribution of source classifiers as measured by $\|\mathbf{t}\|^2$. While this appears to be counter-intuitive given that this framework is for adaptation, 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.

5.2 Multi-Adaptive Support Vector Machine (ma-SVM)

From the above framework, we derive *multi-adaptive SVM* as the counterpart of adaptive SVM for multi-classifier adaptation. We first describe the formulation of this model

and its alternative interpretation, and then presents its learning algorithm.

5.2.1 Model formulation

Multi-adaptive SVM (ma-SVM) is derived by plugging in SVM's hinge loss function $L(y, f(\mathbf{x}) = \max(0, 1-yf(\mathbf{x}))$ and trivial regularization functions $\Omega(x) = x$ and $\Psi(x) = x$. 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:

$$\begin{aligned} L_P \;=\; & \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{2}B\|\mathbf{t}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N} u_i\xi_i \\ & - \sum_{i=1}^{N}\alpha_i(y_i \sum_{k} t_k f_k^s(\mathbf{x}_i) + y_i \mathbf{w}^T\phi(\mathbf{x}_i) - (1-\xi_i)) \end{aligned} \tag{5.3}$$

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

$$\mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i \phi(\mathbf{x}_i), \;\; t_k = \frac{1}{B}\sum_{i=1}^{N}\alpha_i y_i f_k^s(\mathbf{x}_i), \;\; \alpha_i = C - \mu_i \tag{5.4}$$

The equation 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 regularized loss minimization framework. It also justifies the introduction of regularizer $\|\mathbf{t}\|^2$ in the objective function in Eq.(5.2) and in Eq.(5.3).

By plugging Eq.(5.4) into the primal Lagrangian Eq.(5.3), we obtain the dual Lagrange function as:

$$L_D = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \left(K(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{B}\sum_{k} f_k^s(\mathbf{x}_i) f_k^s(\mathbf{x}_j) \right) \tag{5.5}$$

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

$$f(\mathbf{x}) = \sum_{i=1}^{N} \hat{\alpha}_i y_i \left(K(\mathbf{x}_i, \mathbf{x}) + \frac{1}{B} \sum_{k=1}^{M} f_k^s(\mathbf{x}_i) f_k^s(\mathbf{x}) \right) \tag{5.6}$$

5.2.2 Discussion

From the form of the resulting target classifier in Eq.(5.6), we can interpret ma-SVM as a standard SVM that treats *the outputs of source classifiers as additional features*. For each instance $\mathbf{x}$, we treat the output of auxiliary classifiers $\mathbf{f} = [f_1^a(\mathbf{x}), ..., f_M^a(\mathbf{x})]$ on it as an extra feature vector besides its original feature vector $\mathbf{x}$. Similarly, $\mathbf{f}_i = [f_1^a(\mathbf{x}_i), ..., f_M^a(\mathbf{x}_i)]$ is an additional feature vector of $\mathbf{x}_i$. Thus, we can rewrite the target classifier in Eq.(5.6) 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 $\mathbf{f}_i \cdot \mathbf{f}$ can be treated as their similarity in terms of the output of source classifiers on them. If the classifiers' outputs on $\mathbf{x}_i$ and $\mathbf{x}$ are close, $\mathbf{f}_i \cdot \mathbf{f}$ is large and thus their similarity is high, and vice versa. Compared with a standard 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 $\mathbf{x}_i$ by including the similarity in the classifier-output space. In other words, it treats the output of source classifiers as additional features.

In a trivial linear kernel function $K(\mathbf{x}_i, \mathbf{x}) = \mathbf{x}_i \cdot \mathbf{x}$ is used, we can concatenate the original feature $\mathbf{x}$ with the output of source classifiers $\mathbf{f}$ to form a "hybrid" feature vector as $\mathbf{z} = \begin{bmatrix} \mathbf{x} \\ \mathbf{f} \end{bmatrix}$, and similarly, $\mathbf{z}_i = \begin{bmatrix} \mathbf{x}_i \\ \mathbf{f}_i \end{bmatrix}$. Assuming $B = 1$ without loss of generality, the target classifier of ma-SVM can be written as $f(\mathbf{x}) = \sum_i \alpha_i y_i \mathbf{z}_i \cdot \mathbf{z}$, which is identical to a linear SVM model based on the concatenated feature vector $\mathbf{z}$. This "feature concatenation" view does not apply when the kernel function is not the linear one. This is because while feature similarity is computed in any kernel space, the classifier-output similarity is computed in linear space. Therefore, we cannot implement ma-SVM as SVM over concatenated feature vectors. Neither can we use the training algorithm for SVM to train ma-SVM by replacing $K(\mathbf{x}_i, \mathbf{x}_j)$ with $K(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{B} \mathbf{f}_i \cdot \mathbf{f}_j$. We will explore this issue in Section 5.2.3.

The idea of treating model (classifier) outputs as additional features is not new. For example, Natsev et al. [70] adopted the so called "model-vector" approach which constructed semantic feature of a video shot from the scores of video concept classifiers and used it in retrieval. Wu et al. [106] used the outputs of concept classifiers built on different modalities as meta-features for concept detection. Here, we offer a more principled interpretation of this seemingly ad-hoc technique: using classifier outputs as features is equivalent to adapting these classifiers under the regularized loss minimization framework.

The above discussion also provides insight on the scalar B. We can see the role of B as to balance the contribution between feature similarity and the similarity based on source classifier outputs. The scale of these two similarity terms is affected by the feature dimension d and the number of source classifiers M. Typically, we have $d \gg M$ since features for multimedia data is of very high dimensionality. We need to set $\frac{1}{B} > 1$ to avoid the classifier-output similarity being overwhelmed by the feature similarity. A good starting point is to set $B = \frac{M}{d}$ such that that the two similarity terms have equal contribution.

The training cost of ma-SVM can be analyzed similarly to the analysis of a-SVM in Section 4.3.2. From the dual form in Eq.(5.5) it is clear that ma-SVM has exactly the same set of parameters $\{\alpha_i\}_{i=1}^N$ as a standard SVM model trained from the same data $\mathcal{D}^t$. Due to the similarity of their learning algorithm, ma-SVM has the about same training cost as SVM, except the cost of computing $f_k^s(\mathbf{x}_i)$ for each of the M source classifiers $\{f_1^s, .., f_M^s\}$ and on every training example in $\mathcal{D}^t$. So this extra cost of ma-SVM is $O(dMN)$, where d is the feature dimension, N is the data size of $\mathcal{D}^t$. Meanwhile, the training cost of SVM is $O(dN^k)$, where k has been empirically determined to be 1.7 in [56]. Therefore, the total time complexity of ma-SVM is $O(dN^k + dMN)$. Given that the number of source classifiers cannot be very large compared with the number of training examples (i.e., $M << N$), the complexity of ma-SVM is only slightly higher than that of a-SVM in terms of big-O notation.

5.2.3 Learning algorithm for ma-SVM

We propose another modified version of the standard sequential minimal optimization (SMO) algorithm for solving the optimization problem in dual form of ma-SVM.

The dual form of ma-SVM as shown in Eq.(5.5) can be derived from SVM's dual form by replacing $K(\mathbf{x}_i, \mathbf{x}_j)$ with $K(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{B}\sum_k f_k^a(\mathbf{x}_i) f_k^a(\mathbf{x})$. This observation gives the impression that ma-SVM can be implemented using SVM's learning algorithm by simply replacing the kernel term. This impression is not true. Despite the correspondence on their dual form, the intercept term of a classifier is (implicitly) involved in the objective function of ma-SVM through the regularizer $\|\mathbf{w}\|^2$, but it is not involved in the objective function of SVM. Based on our discussion in Section 4.3.3, this allows the SMO algorithm for ma-SVM to optimize only one working variable in each iteration, instead of two variables as in the case of SVM. This changes the structure of the learning algorithm.

The SMO algorithm for ma-SVM is modified from the SMO algorithm for a-SVM described in Section 4.3.3. In fact, the optimality condition and selection criterion of working variable remain exactly the same as in the case of a-SVM, which are given by Eq.(4.11) and Eq.(4.12). The only difference is the analytical solution of each working variable, because ma-SVM has a different optimization function. Suppose α_1 is the working variable. We set the derivative of the dual form in Eq.(5.5) against it to zero:

$$\frac{\partial L_D}{\partial \alpha_1} = 1 - y_1 f^{old}(\mathbf{x}_1) - (\alpha_1^{new} - \alpha_1^{old})\left(K(\mathbf{x}_1, \mathbf{x}_1) + \frac{1}{B}\sum_{k=1}^{M} f_k^a(\mathbf{x}_1)^2\right) \triangleq 0 \qquad (5.7)$$

which leads to the analytical solution of α_1:

$$\alpha_1^{new} = \alpha_1^{old} + \frac{1 - y_1 f^{old}(\mathbf{x}_1)}{K(\mathbf{x}_1, \mathbf{x}_1) + \frac{1}{B}\sum_{k=1}^{M} f_k^a(\mathbf{x}_1)^2} \qquad (5.8)$$

This optimal solution may need to be clipped to satisfy the constraint $0 < \alpha_1 < C$. Thus, the learning algorithm of ma-SVM is the same as that for a-SVM except that the variable update equation in Eq.(4.13) needs to be replaced by Eq.(5.8).

5.3 Experimental Results

We evaluate our multi-classifier adaptation method ma-SVM in cross-domain semantic concept detection. As in our previous experiment in Section 4.5, we treat CNN data in TV05DEV as the target domain. But instead of using another channel as the only source domain, we use all the other 5 channels in TV05DEV (besides CNN) as source

	a-SVM					ma-SVM
source domain	CCTV	LBC	MSNBC	NBC	NTDTV	all 5 channels
airplane	0.036	**0.040**	0.035	0.037	0.031	0.027
animal	0.618	0.598	0.523	0.571	**0.626**	0.551
building	0.187	0.191	0.204	0.190	0.180	**0.208**
car	0.161	0.179	0.249	0.332	0.185	**0.332**
charts	0.046	0.046	0.055	0.031	0.031	**0.056**
computer_tv-screen	0.176	0.128	0.117	0.134	0.106	**0.193**
corporate-leader	0.014	0.026	**0.030**	0.019	0.010	0.022
crowd	0.335	0.344	0.318	0.317	0.323	**0.351**
desert	0.056	0.090	**0.145**	0.065	0.057	0.125
entertainment	0.012	0.012	**0.013**	0.013	0.008	0.011
explosion_fire	0.105	0.115	**0.193**	0.082	0.086	0.123
face	0.815	0.807	0.824	0.824	0.794	**0.827**
flag-us	0.041	0.063	**0.085**	0.060	0.038	0.069
government-leader	0.121	0.109	0.108	0.106	0.107	**0.121**
maps	0.476	0.418	**0.543**	0.317	0.147	0.413
meeting	0.031	0.050	0.040	0.035	0.035	**0.050**
military	0.354	0.290	0.349	0.305	0.292	**0.354**
mountain	0.143	0.156	0.151	0.152	0.144	**0.156**
office	0.057	0.047	**0.095**	0.059	0.048	0.067
outdoor	0.750	0.722	0.765	0.752	0.735	**0.772**
people-marching	0.083	0.110	0.113	0.168	0.125	**0.126**
person	0.891	0.891	0.894	0.896	0.881	**0.898**
road	0.108	0.129	0.144	0.196	0.124	**0.208**
sky	0.518	0.527	0.546	0.521	0.483	**0.549**
snow	0.171	**0.194**	0.132	0.066	0.187	0.106
sports	0.132	0.069	0.050	0.068	0.034	**0.144**
studio	0.764	0.755	0.742	**0.773**	0.752	0.770
truck	0.021	0.020	**0.029**	0.016	0.023	0.025
urban	0.142	0.158	0.172	0.166	0.146	**0.185**
vegetation	0.287	0.280	0.304	0.276	0.259	**0.321**
walking_running	0.178	0.178	**0.194**	0.178	0.178	0.179
waterscape_waterfront	0.212	0.375	0.363	0.359	0.346	**0.400**
average (MAP)	0.251	0.254	0.266	0.253	0.235	0.273

Table 5.1: Per-concept performance on CNN (with 800 training examples) of 5 a-SVM runs and a ma-SVM run. The 5 a-SVM runs use each of the 5 news channels in TV05DEV besides CNN as the source domain, while the ma-SVM run uses all the 5 news channels as source domains.

domains. We assume for each semantic concept, a source classifier has been trained independently from each of the 5 source domains, whose data are are fully labeled. As before, 40% of the CNN data are used for development and 60% are used for evaluation. For each concept, we use ma-SVM to adapt the 5 source classifiers trained from the 5 channels into a new classifier for CNN based on labeled instances randomly selected from its development set. The adapted classifier is evaluated against the evaluation set of CNN. To compare single-classifier adaptation to multi-classifier adaptation, we also use a-SVM to adapt each source classifier into a classifier for CNN.

Table 5.1 compares the performance of a ma-SVM run, which uses all the 5 source channels in adaptation, to 5 a-SVM runs, each using a different channel as the source

domain. The average performance of ma-SVM is 0.273 MAP, higher than any of a-SVM runs, which confirms the benefit of using multiple source domains (classifiers). On a per-concept basis, ma-SVM is the best performer on majority of the concepts, although it is occasionally outperformed by the best a-SVM run. This is because for detecting a concept on CNN, not every source domain (classifier) is equally useful and some can be useless. Combining "bad" source classifiers in adaptation brings in noises, and even if ma-SVM tries to weight them by utility, the result is not as good as that of using the single best source classifier. However, note that which the source classifier is the best for a given concept is unknown and difficult to find out automatically.

We notice from Table 5.1 that the gap between different a-SVM runs is quite large. This suggests a large variation on the utility of different source channels. So it is especially important that the weights of source classifiers learned automatically by ma-SVM indeed reflect their utility. In order to see whether the learned weights are effective, we compare the standard ma-SVM to two modified ma-SVM runs with alternative weighting strategies. The *uniform weighting* strategy assigns equal weights to all source classifiers and ensures the weights sum up to 1. The *oracle weighting* approach assigns weights proportional to the *actual* performance (as AP) of each source classifier on CNN, and also ensures the weights sum up to 1. This performance-based weighting does not guarantee optimal performance in adaptation, but it provides a quasi-optimal reference performance of multi-classifier adaptation. It is called "oracle" because the actual performance on CNN is *unknown* unless all the data in CNN are labeled. In this experiment, we intentionally label all the CNN data for evaluation purpose. In practice, the target domain is unlabeled and therefore it is infeasible to obtain oracle weights. In both strategies, since the weights are fixed before adaptation rather than learned on the fly, we can combine the source classifiers using these weights into a "ensemble" classifier and adapt it as a single classifier.

Figure 5.1 compares the performance of three ma-SVM runs under different weighting strategies, all using CNN as the target domain and the other 5 channels as source domains. As a reference, it also includes an "average a-SVM" run whose performance is set to the average performance of the 5 a-SVM runs shown in Table 5.1, which use different source domains. We see that when training examples are abundant, the proposed automatic weighting strategy in ma-SVM outperforms uniform weighting by a

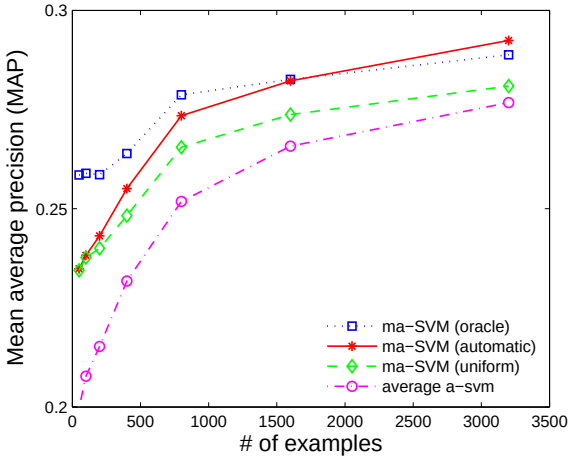


Figure 5.1: Comparison of different weighting strategies in ma-SVM.

non-trivial gap, and even slightly outperforms oracle weighting at 3,200 examples (as we discussed, the oracle weighting does not guarantee the highest performance). When training data are scarce, its performance is close to that of uniform weighting and below oracle weighting. This is because, as discussed in Section 5.2, the weights learned by ma-SVM are based on the performance of source classifiers on the labeled examples. The size of the training data there has a direct impact on the quality of the weights. The weights learned from little training data are unreliable, and the performance is no better than that of uniform weights. When training data are abundant, the learned weights lead to performance as good as that of oracle weights, which are set according to the (unknown) performance on the entire target data. It worths nothing that, despite the weighting strategies, the performance of ma-SVM is significantly higher than that of average a-SVM, confirming the benefit of using multiple source domains.

5.4 Summary

We have addressed the problem of adapting multiple source classifiers into a classifier for the target domain by proposing an extended framework for multi-classifier adaptation. Based on the loss minimization principle, the extended framework allows the weights of source classifiers learned automatically to reflect their utility in adaptation. We have derived multi-adaptive SVM (ma-SVM) from this framework and proposed another modified SMO algorithm for the learning of ma-SVM. We have shown that

in cross-domain semantic concept detection, adaptation based on multiple source domains achieves better performance than adaptation using a single source domain. The weights of source classifiers learned automatically by ma-SVM have also shown to be effective.

The weights of source classifiers are expected to reflect their utility in the target domain, while in ma-SVM they are determined based on the performance on the labeled data. When there are no or very little labeled data, it is impossible to learn the weights reliably, as reflected in the performance in Figure 5.1. It is natural to ask whether we can predict a source classifier's performance on the target domain *without* any labels. If that is possible, we can set the weights of the source classifiers according to their predicted performance, which would further improve the performance of ma-SVM. This would also help identify the single best source classifier for each concept, and therefore improve the performance of single-classifier adaptation based on a-SVM. Given all the benefits of predicting a classifier's generalization performance, we will explore this topic through an in-depth study of classifier's generalizability in Chapter 6.

Chapter 6

Generalizability Analysis and Selective Adaptation

In Chapter 4 and 5, we have presented several approaches to adapting classifiers in different scenarios and demonstrated their *effectiveness* in semantic concept detection. These approaches treat the classifiers to be adapted *equally*, i.e., each and every classifier is adapted with the same number of labeled examples. There is little discussion on the *cost-efficiency* of adaptation, or return-to-cost ratio, where return here is the improvement from pre-adaptation performance to post-adaptation performance, and the cost is the human effort on labeling training examples used in adaptation. As we have shown in Table 4.1, given the same number of training examples, the improvement from adaptation is much larger for some concepts (e.g., *Waterscape*) than for other concepts (e.g., *Face*). This shows a large variation in the cost-efficiency of adapting different concept classifiers. Therefore, when there are multiple classifiers, adapting every classifier equally is *suboptimal* in terms of overall cost-efficiency, namely the total performance improvement on all classifiers for a fixed number of total training examples. To maximize the overall cost-efficiency, we should select and prioritize these classifiers such that concepts like *Waterscape* are adapted with higher priority (e.g., with more training examples) than concepts like *Face*.

A critical factor related to the cost-efficiency of adapting a classifier is its generalization ability. Given the same number of training examples, adapting a less-generalizable classifier is likely to produce larger improvement than adapting a more-generalizable

one, because the former has a lower pre-adaptation performance and thus potentially a larger room for improvement. Meanwhile, we have learned in Chapter 3 that generalizability indeed varies significantly between concept classifiers. Therefore, prioritizing classifiers to be adapted based on their generalizability may enhance the overall cost-efficiency of adaptation. However, measuring a classifier's generalizability to a new domain is not as straightforward as in the study of Chapter 3, because in practice the data in new domains are usually unlabeled (if they are, one can build new classifiers from them instead of having to adapt existing ones). The challenge is to predict a classifier's generalizability *a priori*, namely to predict its performance on a new domain *without* labeling its data. This is a new research problem not yet studied in the literature.

In order to predict classifiers' generalizability and finally improve the cost-efficiency of adaptation, we first present an empirical analysis of the generalizability of semantic concept classifiers. We identify various meta-features of a classifier that are related to its generalizability, including the model structure and parameters, the properties of the concept intended, and the distribution of its output. Based on these meta-features, we build *generalizability models* to predict how well an existing classifier generalizes to new data. Based on this generalizability model, we propose several *selective adaptation* strategies for the selection and prioritization of adaptation tasks, which achieve better cost-efficiency than adapting every classifier equally. The generalizability model can be also used for choosing, for each concept, the parameter setting that leads to the best generalized performance.

It is worth nothing that, while our adaptation approaches are generic and principled, the analysis in this chapter is *empirical* and *specific* to semantic concept detection. Some of the meta-features investigated are specific to SVM classifiers, and generalizability is measured using the performance metric unique to concept detection. This allows us to leverage the domain knowledge of concept detection in order to accurately predict classifier generalizability. Nevertheless, such an empirical study is justified by the critical importance of semantic concept detection. We believe it is also possible to conduct similar analysis of generalizability in other areas.

6.1 Meta-features for Generalizability

The generalizability of a concept classifier is influenced by several factors. The first impacting factor is the *semantic concept* the classifier is intended. We have observed in Section 3.2 that classifiers of frequent concepts tend to be more generalizable, so we will further study their relationship in this section. The second factor is the *feature* used as the input to classifiers. While only one type of feature was used in our previous experiments, we will study whether other features make classifiers generalize better or worse. The third factor is *model configuration*, which includes the classification algorithm, kernel function, and various model parameters used for training classifiers. The impacting factors are by themselves meta-features that can be used for predicting generalizability.

Generalizability can be also determined by examining the classifier after it is trained. In the case of SVM classifiers, we can examine how the number and ratio of support vectors (SVs) in a classifier is related to its generalizability. We can also apply a classifier on (unlabeled) target data and examine the distribution of its output to find clues of its generalizability. These meta-features are unlikely the result of any single factor mentioned above, but the join effect of multiple factors. For example, the ratio of SVs in a classifier is influenced by the concept, the feature, and the model parameters.

6.1.1 Experiment set-up

To study the connections between various meta-features of a classifier and its generalizability, we expand our experiments on the generalizability of semantic concept detection presented in Chapter 3.1. The new experiments include two settings. In the NBC/CNN (cross-channel) setting, we train concept classifiers from NBC data and apply them to CNN data, where NBC and CNN are two news channels in TV05DEV. In the TV05/TV07 (cross-genre) setting, we build classifiers from the entire TV05DEV collection and apply them to TV07DEV. We still use the 39 semantic concepts provided by the LSCOM-lite project [2], and SVM with RBF kernel as the classification method. However, instead of using a single feature and fixed model parameters, here we evaluate a variety of features and model parameters in order to study their impact on generalizability. We use 6 different features that describe the color, texture or edge

characteristics of an image:

- **gcm**: 225-dimensional color moment feature which contains the mean, standard deviation, skewness of LUV color feature in each image region of 5×5 grids.
- **gbr**: 48-d Gabor texture feature which includes the mean and standard deviation of the output of Gabor filters in 4 scales and 6 orientations.
- **gcm-gbr**: the concatenated feature of **gcm** and **gbr**. This is the feature used in all previous experiments.
- **edh**: 73-d edge direction histogram, where 72 bins denote edge direction quantized at 5 degrees each and the remaining one bin is for non-edge points.
- **3hvc**: 375-d color histogram in HVC color space, computed from 3×3 image grids.
- **3hvc.stats**: 150-d feature as the compact representation of **3hvc** feature.

Moreover, we experiment with different choices of two critical parameters of SVM, which have been shown to have a significant impact on performance. The cost factor C in the objective function of SVM [17] controls the tradeoff between minimizing classification error and maximizing margin. In the RBF kernel function $K(x_i, x_j) = e^{-\gamma\|x_i - x_j\|^2}$ used in SVM, gamma γ controls the size of the region influenced by a support vector and consequently, the smoothness of the decision boundary (hyperplane). Smaller γ leads to a smoother boundary, and vice versa. We vary C between 3 values as 1, 3, and 10, and γ between 6 values as 0.01, 0.05, 0.1, 0.2, 0.4, and 1.

In the experiments, the performance of a classifier on its training domain, or within-domain performance, is typically higher than that on a new domain, or cross-domain performance. Naturally, the relative difference between the two performance is a good indicator of how well the classifier generalizes from its training domain to the new domain. In this chapter, we use the relative decline (in percentage) from a classifier's within-domain performance ΔAP to its cross-domain performance ΔAP as a quantitative measure of its generalizability:

$$d = \frac{\Delta AP_{src} - \Delta AP_{tgt}}{\Delta AP_{src}} \tag{6.1}$$

We call this *cross-domain relative performance decline* or simply *decline*. Obviously, smaller decline indicates better generalizability, and vice versa. Because ΔAP is not influenced by the ratio of positive data (a.k.a. concept frequency) and has a zero baseline, the decline computed based on ΔAP is also unrelated to the ratio of positive data. The specific meaning of ΔAP_{within} and ΔAP_{cross} depends on the experiment setting. For example, in the NBC/CNN setting, ΔAP_{within} represents the performance of 5-fold cross-validation within NBC data, while ΔAP_{cross} represents the performance of NBC's classifier on CNN data.

Now we can study the connection between a meta-feature and generalizability by examining whether and how it is related to the decline in all the classifiers we built. We will use correlation coefficient, a metric of the linear relationship between two random variables, for this purpose.

6.1.2 Overall analysis

The generalizability of a concept classifier is affected by the concept, feature, and model parameters. We want to first gain an impression on which factor has the largest overall impact on generalizability. To compare the impact of concept and model parameters, we build classifiers for each of the 39 concepts using the **gcm-gbr** feature and using 6 values of gamma γ and 3 values of cost factor C. This results in 18 classifiers for each concept. Figure 6.1 shows for each concept, the mean and standard deviation of the relative performance declines of the 18 classifiers in the NBC/CNN setting and TV05/TV07 setting. The mean decline is mainly determined by concept, while the standard deviation is caused by the variation of model parameters. We see that the mean decline differs significantly from one concept to another, but the variance of decline within each concept is relatively small. This suggests that a classifier's generalizability is influenced more by the concept it is intended than by the choice of model parameters in training.

Features may also have a large impact on the generalizability of classifiers built on them. If the data distribution under a certain feature space is consistent and insensitive to domain changes, the classifiers built on top of this feature is likely to generalize well, and vice versa. Considering applying a classifier for *Building* trained from color images

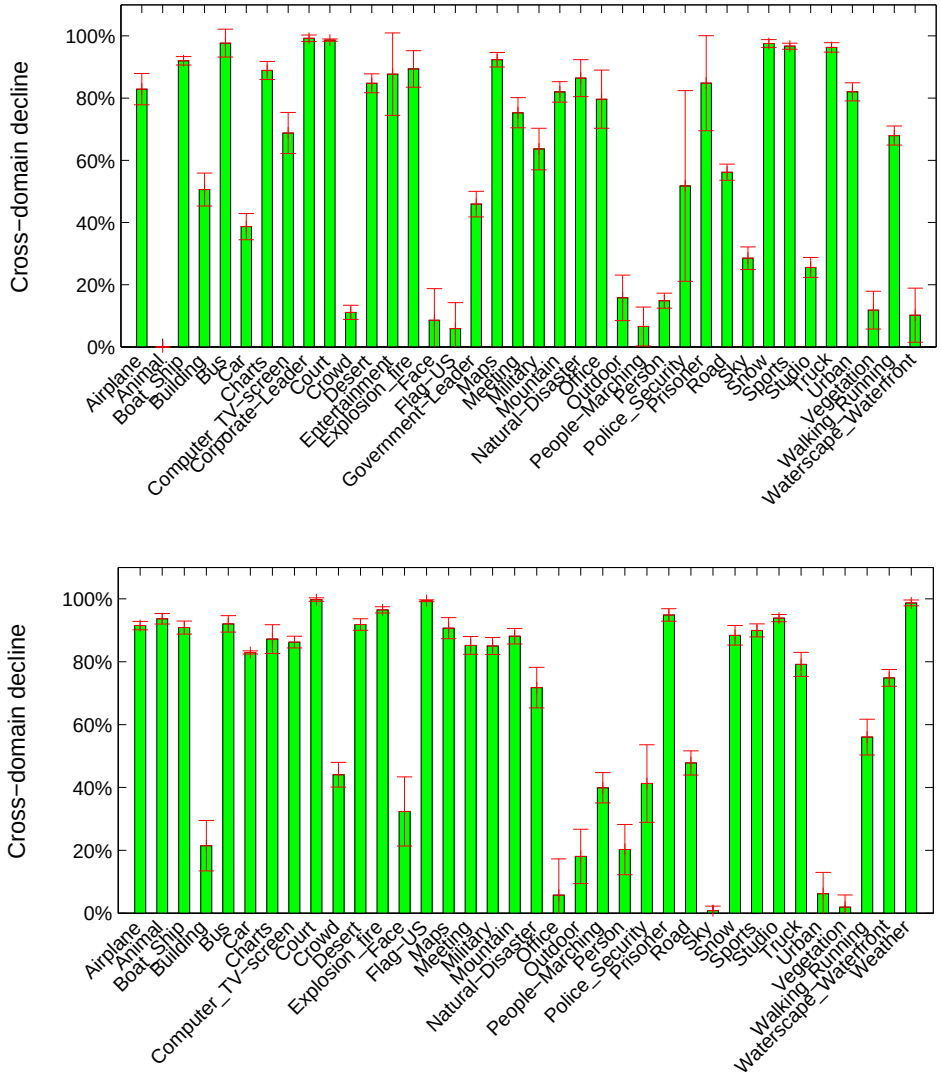


Figure 6.1: For each concept, the mean and standard deviation of relative performance decline of classifiers trained with different parameter settings.

to gray-scale images. In this case, a classifier built on Gabor texture feature is expected to generalize better than one built on color histogram feature, because texture features are more insensitive to the transition from color to gray-scale images.

Table 6.1 compares six features in terms of their within-domain and cross-domain performance and the decline between the two. We find that features indeed have a strong influence on generalizability (decline), but it is difficult to conclude on the most generalizable feature. In the NBC/CNN setting, most features have about the same level of decline, while **edh** and **hvc.comp** has much higher decline. In the TV05/TV07

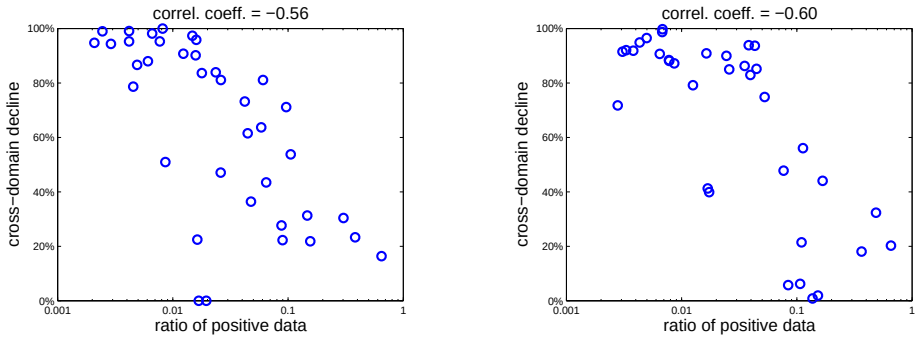
	ΔMAP of SVM_{NBC}			ΔMAP of SVM_{TV05}		
Feature	NBC	CNN	decline	TV05Dev	TV07Dev	decline
gcm	0.257	0.093	63.8%	0.211	0.067	68.3%
gbr	0.160	0.053	67.2%	0.120	0.053	55.9%
gcm+gbr	0.280	0.105	62.5%	0.287	0.089	69.1%
edh	0.034	0.003	92.2%	0.028	0.027	2.7%
hvc	0.205	0.066	67.8%	0.163	0.044	73.2%
hvc.comp	0.202	0.011	94.3%	0.162	0.044	73.1%

Table 6.1: Comparison of the generalizability of 6 features. SVM_{NBC} denotes classifiers trained from NBC, and SVM_{TV05} denotes classifiers trained from TV05Dev. The performance is the average over 39 concepts.

setting, however, **edh** has extremely low decline, and **gbr** has the second smallest decline, while all color features have higher decline. This is perhaps because TV05DEV has only color footage while TV07DEV has both color and gray-scale footage, making color-independent features like **edh** and **gbr** more robust than color features. This is not the case in NBC/CNN setting because both NBC and CNN only has color footage. So the generalizability of a feature depends on the relationship between the source and target domain, and no feature appears to have the smallest decline across two settings.

Meanwhile, what really matters in practice is to find features that lead to good performance on the target domain. Therefore, the most "generalizable" feature defined by the smallest decline may not be the most desirable feature. For example, the edge feature **edh** has the smallest performance decline (only 2%) in TV05/TV07 setting, yet its performance on either the source or target domain is the lowest among all the features. So it is in fact the least desirable feature in terms of its cross-domain performance. When cross-domain performance is the criterion, we find that in both settings, the **gcm-gbr** feature has the highest cross-domain performance among all the features, beating other features including **gcm** and **gbr** by large margins. This shows that combining complementary features on texture and color improves the performance and robustness of the resulting feature. Since **gcm-gbr** is the clear winner, we focus on this feature in our subsequent experiments.

We conclude that concept and feature are the major impacting factors of a classifier's generalizability. Our explanation is that these two factors together completely



(a) NBC/CNN setting (b) TV05/TV07 setting

Figure 6.2: Relative performance decline vs. the ratio of positive data in 39 concept classifiers.

defines how the data points are distributed in the feature space, and how that distribution changes across domains. This in turns determines how well the positive and negative data can be separated by a decision boundary and how this boundary generalizes to a different domain. In comparison, model parameters have only a small influence the precise placement of the decision boundary, and therefore the generalizability.

6.1.3 Positive ratio (concept frequency)

We have found that classifiers of frequent concepts, namely concepts with a higher ratio of positive data, tend to generalize better. In Figure 3.1 and 3.3, the gap between within-domain and cross-domain ΔAP is smaller for frequent concepts than for rare concepts. In Figure 6.2, we plot the relative performance decline of the classifiers of the 39 concepts against the ratio of positive data for each concept, in both NBC/CNN and TV05/TV07 setting. To avoid the influence of other factors, all the classifiers are trained with **gcm-gbr** feature and fixed model parameter ($\gamma = 1$ and $C = 1$).

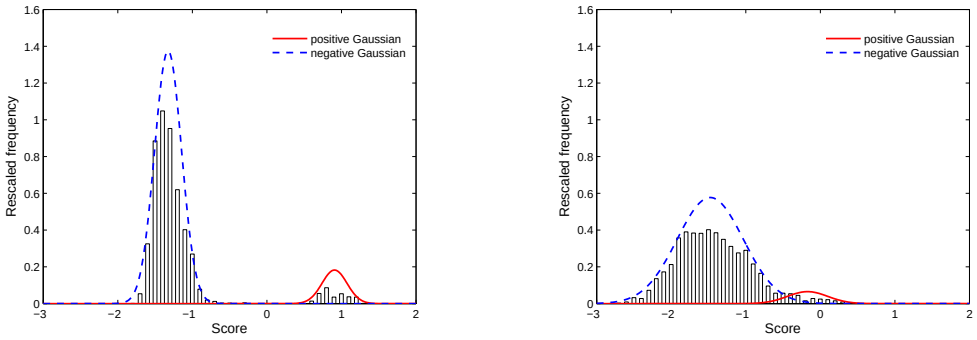
We find that in both settings, larger ratio of positive data corresponds to smaller performance decline, and vice versa. The correlation coefficient between this ratio and decline is -0.56 in the NBC/CNN setting and -0.60 in the TV05/TV07 setting, indicating a pronounced correlation between them (it is negative because the decline is higher than the ratio is lower). This suggests that in general, classifiers of frequent concepts generalize better than those of rare concepts. Note that this is not because frequent concepts have higher baseline performance, because the decline is computed using ΔAP which always has zero baseline despite the ratio of positive data. Our explanation is

that frequent concepts are usually generic concepts and the visual appearance of generic concepts is relatively insensitive to domain changes. For example, we expect shots of frequent concepts like *Outdoor* and*Person* are visually similar across different domains, while shots of rare concepts such as *Studio* may appear differently across domains. As a result, for frequent concepts the data distribution change between domains is relatively small and thus their classifiers are more robust. Another reason is that rare concepts do not have sufficient positive training data necessary for building reliable classifiers, leading to inferior performance.

6.1.4 Score distribution

When a concept classifier is applied to predict the label of a data instance, it produces a numeric score indicating the degree of relevance between this instance and the given concept. In the case of SVM, the score is in the range of $(-\infty, +\infty)$, with positive scores indicating positive (predicted) labels and negative scores indicating negative labels. The absolute value of the score indicates the confidence of the prediction. If we apply a classifier on a data set, the distribution of all the scores provides clues as to how well this classifier performs on the data, even without knowing the true labels of the data. As exemplified in Figure 6.3, one can easily find connections between the score distribution and the performance of a classifier. For a classifier with good performance, the scores of positive data are much higher than the scores of negative data, with a large separation between them. In contrast, for a classifier with poor performance, the separation between the scores of positive data and scores of negative data is not obvious. Thus, to find out how well a classifier generalizes to a target domain, we can apply it to the (unlabeled) target data and examine the distribution of the scores for clues related to its performance.

We start with meta-features of score distribution that are easy to compute. One of such features is the *maximum score*, namely the largest score the classifier produces on the target data. If the maximum score is low, which means the classifier is not sure about any positive instance, then its performance is likely to be poor (with a small risk that no positive data exist in the target domain). Another feature is *score range*, which is the distance between the largest and the smallest score on the target data. This range indicates how well a classifier separates the positive data and the negative



(a) Output of a "good" classifier (b) Output of a "bad" classifier

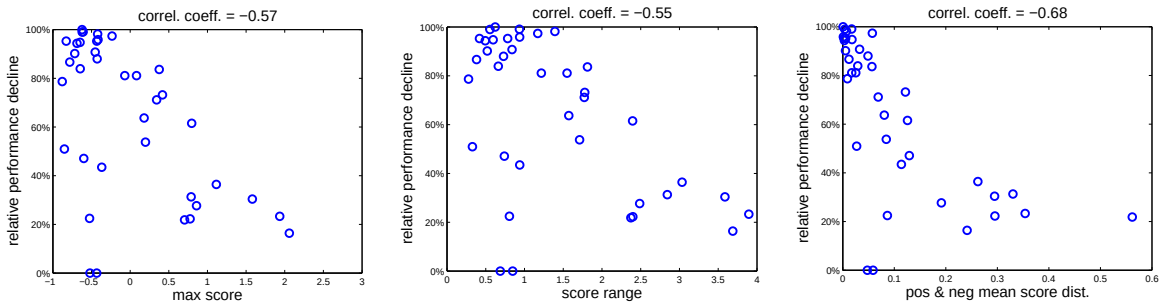
Figure 6.3: The score distribution of (a) a SVM classifier with high performance, and (b) a SVM classifier with low performance. The histograms denote the distribution of the actual scores, while the red/blue curves show the estimated Gaussian distributions.

data.

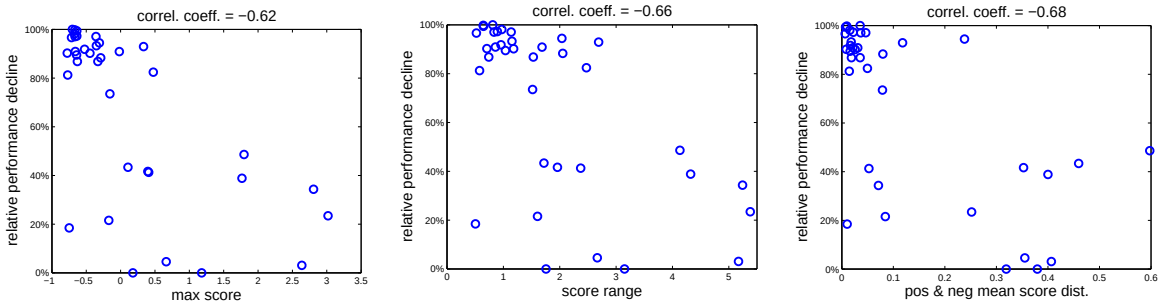
Note that because target data are unlabeled, it is impossible to know precisely how the scores of the positive and negative data are distributed. The features mentioned above provides hits about the separation between the scores of the two classes, but they do not directly capture such separation. Here we take a model-based approach to recover the score distribution of the positive data and of the negative data in the target domain without the true data labels. Assuming that the scores of positive data and scores of negative data follow distributions of a certain family, we estimate their respective distribution from the scores of all the data using the Expectation Maximization (EM) algorithm [48]. Modeling score distribution have been discussed in the context of information retrieval [66] and rank aggregation [35]. Similar to [35], we use Gaussian distributions to model the scores of positive data and scores of negative data.

Suppose $z = f(x)$ is the score of instance x produced by a concept classifier f. 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, $p(z|y = -1) = \mathcal{N}(u_n, \sigma_n)$ is the distribution of 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:

$$p(z) = \pi\mathcal{N}(u_p, \sigma_p) + (1 - \pi)\mathcal{N}(u_n, \sigma_n) \tag{6.2}$$



(a) NBC/CNN setting



(b) TV05/TV07 setting

Figure 6.4: Relative performance decline vs. meta-features of classifiers' score distribution on the target data, including the maximum score, the score range, and the distance between the estimated mean score of the positive and negative data.

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 known. Because the true labels $\{y_i\}$ of the target data are unknown, we can still use the EM algorithm to estimate these parameters from only the scores $\{z_i\}$. The EM algorithm iteratively optimizes the model parameters starting from their initial values until it finds two Gaussian components that best fit the scores. In Figure 6.3, the red (solid) and blue (dashed) curve respectively represents the estimated Gaussian distribution for the positive data and for the negative data, which actually fit the actual score distributions denoted as histograms. Suppose $(\hat{\pi}, \hat{u}_p, \hat{\sigma}_p, \hat{u}_n, \hat{\sigma}_n)$ are the parameters estimated by the EM algorithm. We can measure the score separation between the two classes in terms of the distance between the estimated mean of the two Gaussian components, which is $\hat{u}_p - \hat{u}_n$.

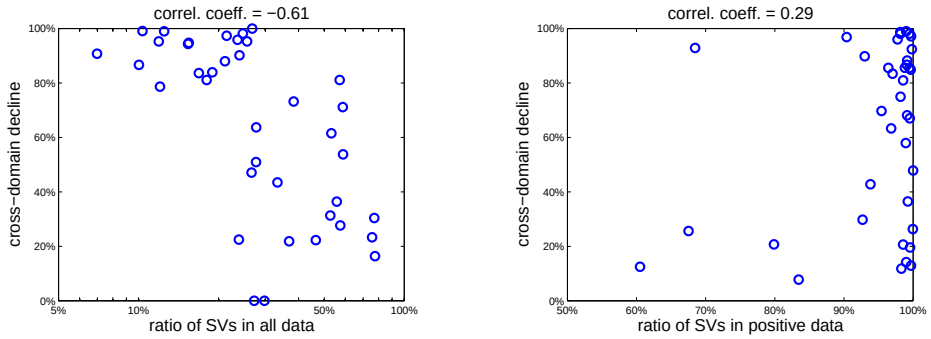
Figure 6.4 shows in two settings, the relationship between relative performance decline of 39 concept classifiers and meta-features extracted from their score distribution on the target domain, including the maximum score, the distance between the largest

and smallest score, and the distance between the estimated mean score of the positive and negative data. For all the three meta-features, which indicate the separation between the scores of two classes, larger feature values correspond to smaller performance decline. So it is true that the between-class separation is a strong clue of generalizability. Among the three meta-features, the difficult-to-compute feature of the distance between the estimated mean score of two classes is slightly better than the other two feature, as indicated by their correlation coefficients with performance decline.

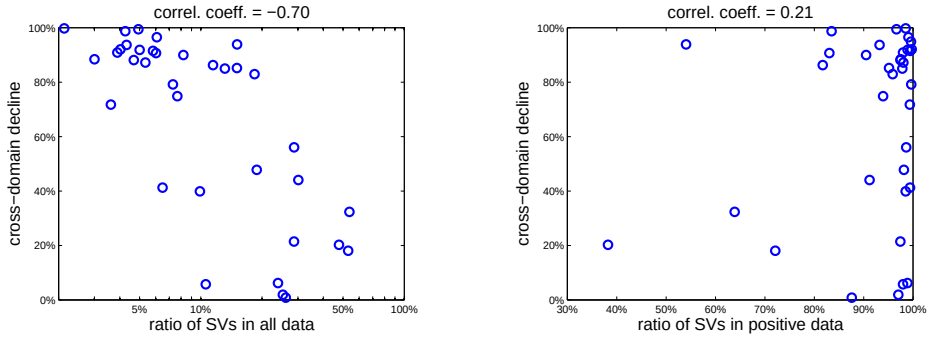
6.1.5 Model structure

The structure of a SVM classifier also reveals its generalizability. As we discussed in Section 3.3, the decision boundary of a SVM classifier is completely determined by support vectors (SVs), so the number of SVs is related to the model complexity. Figure 6.5 shows for the classifiers of 39 concepts, the relative performance decline against the ratio of SVs in the entire training data, in the NBC/CNN and TV05/TV07 setting. To eliminate the distractions from other factors, all the classifiers are trained with **gcm-gbr** feature and fixed parameters ($C = 1, \gamma = 1$). Because the number of training data is the same for all the concepts, the ratio of SVs is proportional to the number of SVs. The trend is that classifiers with more SVs have the smaller decline, or better generalizability, than classifiers with fewer SVs. This means more complex and "bloated" classifiers generalize better than simpler ones. Our observation in Section 3.3 has shown that SVM concept classifiers are analogous to memory-based models such as kNN, if we treat SVs as the training data in kNN. For memory-based models, more training data lead to smaller model variance and more reliable performance. The same is true between the performance of SVM and the number of SVs.

More interesting is the ratio of SVs in positive data, which are scarcer and more valuable to the performance of classifiers compared with the usually abundant negative data. Figure 6.5 also shows the relationship between performance decline and the ratio of SVs in *positive* training data, i.e., the percentage of the positive data chosen as SVs. In the NBC/CNN setting, although many concept classifiers use nearly *all* the positive data as SVs, five classifiers use less than 90% of positive data as SVs. For all but one of the five classifiers, the performance decline is around or below 30%, indicating very good generalizability. Similar observations can be made in the TV05/TV07 setting.



(a) NBC/CNN setting



(b) TV05/TV07 setting

Figure 6.5: Relative performance decline vs. the ratio of SVs in training data or the ratio of SVs in positive training data.

So the observation is that classifiers that are able to "compress" the positive data tend to generalize better. Our explanation is that, if the distribution of the positive data of a specific concept displays certain general patterns, it is possible for a classifier to represent such pattern using only part of positive data, leading to a small ratio of SVs in positive data. Otherwise, if the distribution of positive data is close to random, then the classifier has to memorize most of them and resort to a nearest-neighbor type of prediction. Intuitively, a classifier capturing the data pattern is likely to generalize better than a nearest-neighbor type of classifier, because while the general pattern may still hold true in new domains, randomly distributed positive data shift their locations from domain to domain.

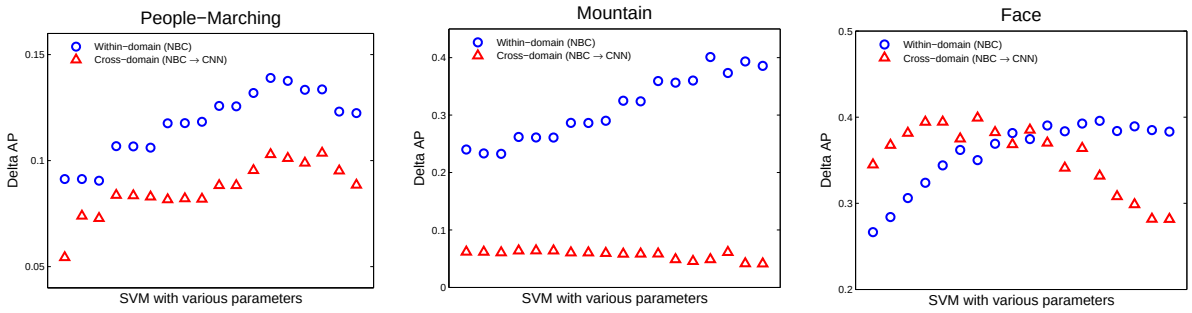
Classifiers using *more* data as SVs generalize better; Meanwhile, classifiers using *fewer* positive data as SVs generalize better. There are no contradictions between the two seemingly conflicting observations, though. If a classifier is able to represent the

general pattern in the positive data by a small subset of them, it is likely to generalize well thanks to a general and smooth decision boundary. However, this is the case only with a handful of concepts, and the other concepts do not display general pattern to be captured by their classifiers. For these concepts, the classifiers have to memorize most positive data and resort to a memory-based approach for classification. In this case, more SVs help reduce model variance and yield better performance.

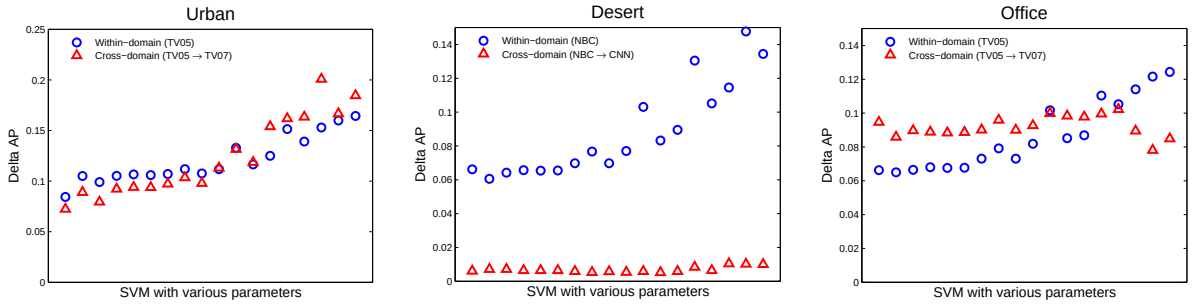
6.1.6 Model parameters

We explore the influence of two model parameters on a classifier's generalizability, namely the cost factor C and the gamma γ parameter in the RBF kernel function. We have shown in Figure 6.1 that the influence of model parameters is relatively small compared with that of the concept. Here, we are interested in two questions: 1) how does C and γ affect the cross-domain performance ΔAP_{cross}? and 2) does ΔAP_{cross} and the within-domain performance ΔAP_{within} change in the same way as parameters change? The second question is important because a common practice in training concept classifiers is to choose model parameter that leads to the highest cross-validation performance, or the highest ΔAP_{within}. It is interesting to know whether this practice will also find the classifier with the best performance on the new domains, or the highest ΔAP_{cross}. This will be addressed by the second question.

To answer these questions, we examine several concepts in each of the two settings. For each concept, we train classifiers using 3 values of C and 6 values of γ, which results in a total of 18 classifiers. In Figure 6.6, each figure shows the ΔAP_{within} and ΔAP_{cross} of the 18 classifiers of a specific concept. We find that the answer to both questions above depends on concept. Let us first look at the NBC/CNN setting. For concept *People-Marching* and *Face*, ΔAP_{cross} does change with parameters, but not always in accordance with ΔAP_{within}: for *People-Marching* the two performance changes basically in synchronization, while for *Face* they sometimes change in opposite directions. For concept *Mountain*, however, ΔAP_{cross} basically remains the same (and very low) despite the change of model parameters. Moreover, there is no single optimal parameter setting that leads to the highest ΔAP_{cross} in all the concepts. Similar observations can be made in the TV05/TV07 setting, where ΔAP_{cross} changes with parameters for *Urban* and *Office* but not for *Desert*. Therefore, the influence of model



(a) NBC/CNN setting

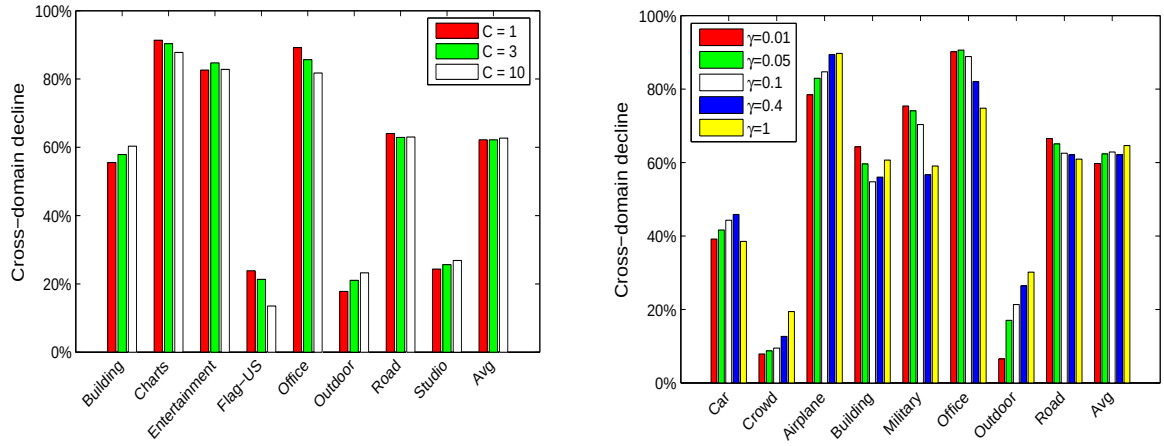


(b) TV05/TV07 setting

Figure 6.6: For several concepts, the within-domain ΔAP and cross-domain ΔAP of classifiers trained with different C and γ.

parameters on ΔAP_{cross} is not consistent across concepts, and the relation between ΔAP_{within} and ΔAP_{cross} also varies from concept to concept. This also means that classifiers (model parameters) with the highest cross-validation performance do not always generalize the best to new domains.

What is the relationship between a model parameter and the generalizability of a classifier? For example, does a larger cost factor C lead to better (or worse) generalizability? Figure 6.7 shows the relationship between the relative performance decline as an indicator of generalizability and parameter γ and C on several concepts in the NBC/CNN setting. We find that the relationship changes from concept to concept. For example, on some concepts such as *Flag-US* and *Office* larger C leads to smaller decline, while on other concepts such as *Building* and *Studio* larger C leads to larger decline. The average decline of all concepts, shown by the bars on the right of Figure 6.7(a), is about the same for different C. The same observations can be made on gamma γ. So there is no consistent relationship between generalizability and parameter C or γ.



(a) decline vs. cost factor C (b) decline vs. gamma γ

Figure 6.7: Relative performance decline vs. cost factor C and gamma γ in NBC/CNN setting.

6.2 Generalizability Model

We have examined the correlations between a concept classifier's generalizability and various model meta-features. The findings show that the concept is the major impacting factor of generalizability, while the influence of model parameters is relatively small and inconsistent across different concepts. Therefore, building a *universal* model that predicts the generalizability of classifiers for any concept and trained with any model parameters is not a good idea. Instead, we divide the problem into estimating two types of generalizability: (1) *between-concept generalizability*, or the generalizability of classifiers for different concepts trained with fixed model parameters, and (2) *within-concept generalizability*, or the the generalizability of classifiers of a specific concept trained with different model parameters. In this section, we focus on predicting the generalizability between different concepts. Later in the chapter, we will discuss finding the most generalizable classifier for each concept.

Table 6.2 lists the meta-features and their correlation coefficient with the relative performance decline as an indicator of generalizability. These correlation coefficients are computed based on classifiers of the 39 concepts trained with fixed feature and parameter (**gcm-gbr** feature, $C = 1$, and $\gamma = 1$), so the concept is the only important factor of generalizability. We see that most correlation coefficients have a high absolute value (the maximum is 1), which suggests that the listed meta-features are useful in

Feature f	**Correl**(f, *decline*)	
	NBC/CNN	TV05/TV07
ratio of positive data	-0.56	-0.60
SV ratio of in training data	-0.61	-0.70
SV ratio of in positive training data	0.29	0.21
maximum output of classifier on test data	-0.60	-0.66
range of classifier's output on test data	-0.54	-0.62
distance b/w estimated mean score of positive and negative data	-0.45	-0.57

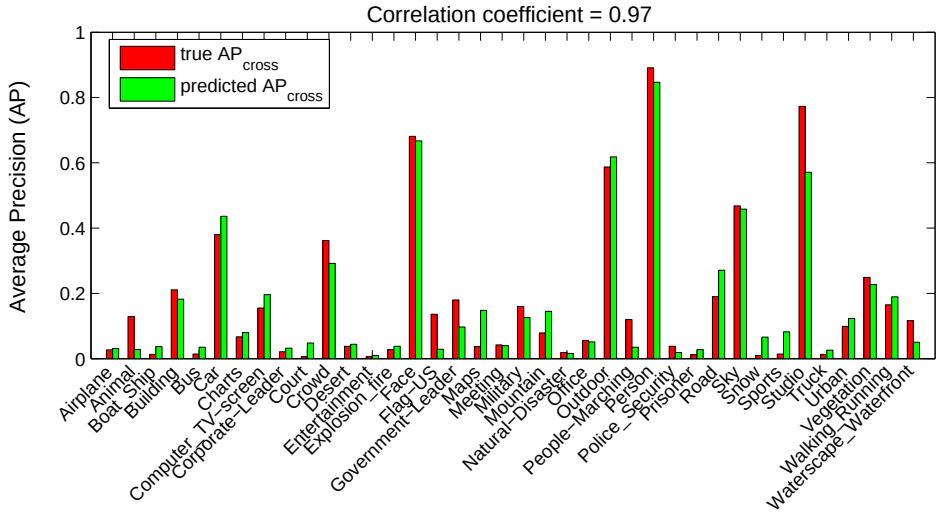
Table 6.2: Meta-features and their correlation coefficients with relative performance decline, computed across 39 concepts.

distinguishing the generalizability between different concepts.

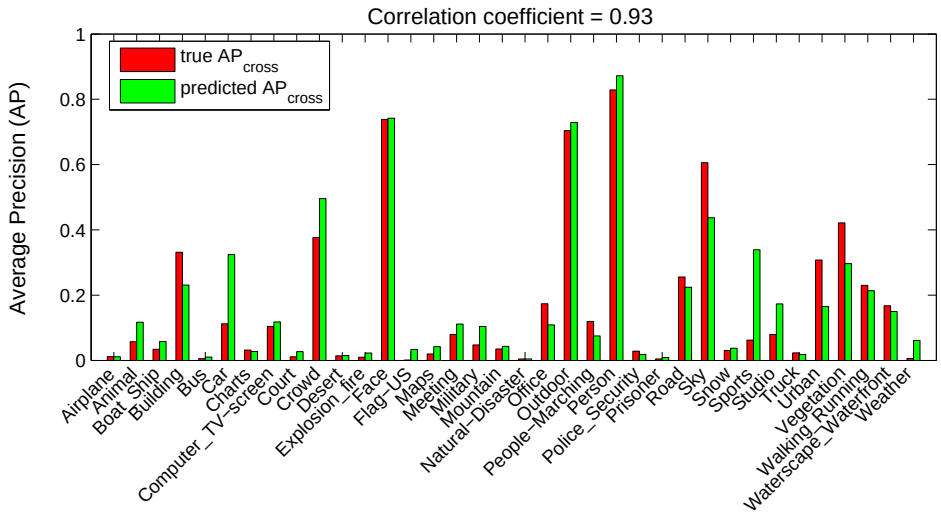
We build a generalizability model based on linear regression to predict the generalizability of concept classifiers. The input to the model includes all the meta-features of a classifier listed in Table 6.2, and the output is the prediction of the relative decline $\hat{d}$) from its performance in the source domain to the performance in the target domain. The linear regression model is trained using support vector regression (SVR) algorithm [36], a variant of standard SVM algorithm for regression, with linear kernel function. Since it is easy to compute a classifier's performance in its training domain ΔAP_{within} by cross-validation, we can compute its (normalized) performance on the target domain as $\widehat{\Delta AP}_{cross} = \Delta AP_{within}(1-\hat{d})$. The un-normalized performance $\widehat{AP}_{cross}$ can be obtained by adding the concept frequency to $\widehat{\Delta AP}_{cross}$. Therefore, the generalizability model allows us to predict a classifier's performance on an unlabeled new domain different from its training domain.

6.2.1 Prediction of cross-domain performance

We evaluate the generalizability model by examining the accuracy of the cross-domain performance $\widehat{AP}_{cross}$ predicted based on this model. The experiment is conducted in a leave-one-out fashion. In the NBC/CNN setting, from the classifiers of 39 concepts trained from NBC, we use the meta-features extracted from 38 classifiers to train the generalizability model, and apply the model to predict the decline on the remaining classifier, based on which we can compute $\widehat{AP}_{cross}$ as its predicted performance on target domain CNN. We repeat this experiment 39 times such that every concept has been held-out once for evaluation. We then compare the predicted performance $\widehat{AP}_{cross}$ of each classifier on CNN with its true performance AP_{cross}, which is calculated with



(a) NBC/CNN setting



(b) TV05DEV/TV07DEV setting

Figure 6.8: Comparison between the true and predicted cross-domain performance (AP_{cross} and $\widehat{AP}_{cross}$) based on the generalizability model.

the entire CNN data labeled. The accuracy of the prediction can be measured by correlation coefficient or by mean square error (MSE) between $\widehat{AP}_{cross}$ and AP_{cross} across all the concepts. This experiment is valid because the model used to predict the performance on each concept is not trained using the meta-features related to that concept. The same leave-one-out experiment is conducted on the 36 concepts in the TV05DEV/TV07DEV setting.

In Figure 6.8, we compare the predicted performance $\widehat{AP}_{cross}$ and true performance AP_{cross} in both settings. It is clear that the predicted performance is very close to

Correlation Coeff. ($\widehat{AP}_{cross}$, AP_{cross})		**Target domain**							
		CCTV	CNN	LBC	MSNBC	NBC	NTDTV	TV05	TV07
Source domain	CCTV	N/A	.931	.949	.946	.923	.963		
	CNN	.877	N/A	.959	.930	.965	.956		
	LBC	.888	.899	N/A	.909	.890	.895		
	MSNBC	.940	.951	.960	N/A	.842	.954		
	NBC	.934	.968	.914	.925	N/A	.960		
	NTDTV	.970	.965	.966	.964	.898	N/A		
	TV05							N/A	0.941
	TV07							0.918	N/A

Table 6.3: Correlation coefficient between $\widehat{AP}_{cross}$ and AP_{cross} over 39 concepts under different source and target domains. The generalizability model is trained in NBC/CNN setting.

MSE($\widehat{AP}_{cross}$, AP_{cross})		**Target domain**							
		CCTV	CNN	LBC	MSNBC	NBC	NTDTV	TV05	TV07
Source domain	CCTV	N/A	.0064	.0046	.0042	.0056	.0026		
	CNN	.0136	N/A	.0046	.0076	.0031	.0034		
	LBC	.0102	.0083	N/A	.0083	.0081	.0088		
	MSNBC	.0055	.0058	.0035	N/A	.0154	.0035		
	NBC	.0063	.0029	.0080	.0076	N/A	.0030		
	NTDTV	.0022	.0027	.0028	.0027	.0074	N/A		
	TV05							N/A	.0059
	TV07							.0072	N/A

Table 6.4: Mean square error (MSE) between $\widehat{AP}_{cross}$ and AP_{cross} over 39 concepts under different source and target domains.

the true one for most of the concepts. The correlation coefficient between them is as high as 0.97 in NBC/CNN setting and 0.93 in TV05DEV/TV07DEV setting. This demonstrates our generalizability model is capable of accurately predicting a classifier's performance on unlabeled out-of-domain data.

While the generalizability model trained from NBC/CNN setting is able to accurately predict the performance of NBC classifiers on CNN, it is interesting to see whether this generalizability model itself is "generalizable" to settings with other source and target domains. Is it able to accurately predict, for example, the performance of concept classifiers trained from CCTV on LBC data? To answer this question, we iterate over all 6 channels in TREC05DEV and each time pick two different channels as the source and target domain. This results in 30 unique pairs of source and target

domain. For each unique pairs of source and target domain, we apply the generalizability model trained from NBC/CNN setting to predict the performance of concept classifiers trained from that source domain and applied on that target domain. We also apply this model to predict the performance of concept classifiers trained from TV05DEV on TV07DEV, and performance of classifiers trained from TV05DEV on TV07DEV. Since the documentary video in TV07DEV is very different from the news video in NBC or CNN, this test poses a greater challenge than settings with two news channels as the source and target domain. For each source-target pair, the accuracy of prediction is evaluated in terms of the correlation coefficient and MSE between the predicted and true cross-domain performance.

Table 6.3 shows the correlation coefficients of our prediction for each unique pair of source and target domain. We find most correlation coefficients are higher or close to 0.9, and some are even higher than the coefficient for the NBC/CNN setting from which the generalizability model is trained. Even the correlation coefficients in the TV05DEV/TV07DEV and TV07DEV/TV05DEV setting are very high, showing that the generalizability model can be generalized to other types of video data. In addition, Table 6.4 shows that the mean square error of our prediction is very small for most source-target pairs. This confirms that our generalizability model is general and domain-independent, which implies generalizable (and un-generalizable) concept classifiers share common properties that are not related to the data domains from which they are trained.

6.3 Selective Adaptation

The generalizability model can be used to enhance the cost-efficiency of adaptation through selection and prioritization of multiple adaptation tasks. In our previous experiments, all concept classifiers are adapted with the same number of labeled examples, which is suboptimal because the improvement from adaptation is different across these concepts. In Table 4.1, we have compared the pre-adaptation and post-adaptation performance of 39 concept classifiers trained from NBC and evaluated on CNN. For concepts like *Crowd* and *Person*, the original classifier generalizes so well to CNN that adaptation generates little or no improvement. This contrasts to concepts like

Military and *Sports*, where the original classifier performs poorly on CNN and the improvement from adaptation is significant. The same observation can be made on the pre-adaptation and post-adaptation performance in TV05/TV07 setting. There is a large difference between different concepts in terms of "return to cost ratio", where the return is the performance improvement and the cost is the manual effort on labeling training data.

A reasonable goal for adaptation is to maximize the overall performance on multiple concept classifiers for a fixed budget of training examples. Adapting every classifier with an equal number of examples is certainly not the best approach to achieve this goal. A better idea is to first identify the potential performance improvement of adapting each concept classifier, and prioritize these classifiers such that those with larger improvement-to-example ratio are the only ones being adapted or are adapted with more examples. In this section, we propose several *selective adaptation* approaches, which prioritize adaptation tasks to maximize the overall post-adaptation performance for a fixed number of examples. These methods differ in the predictions they use for potential improvement of adapting a classifier.

6.3.1 Generalizability-based methods

The generalizability of a classifier is an important clue for selective adaptation because classifiers with low generalizability have a greater potential of improvement as a result of adaptation. If the generalizability model predicts the relative performance decline of a classifier to be $\hat{d}$, then the absolute decline of its performance is $\Delta AP_{within} \times \hat{d}$. The larger this decline is, the lower the classifier's performance on the target domain, implying a larger room for potential improvement. It is reasonable to use the predicted absolute performance decline of a classifier as an estimate of improvement resulted from adaptation. The underlying assumption is that after adaptation with sufficient training data, the classifier will reclaim all the lost performance. There are exceptions, for example, when detecting the same concept is much harder in the target domain than in the source domain.

The most straightforward method is to adapt only the classifiers with poor generalizability. We first calculate the predicted absolute performance decline $\Delta AP_{within} \times \hat{d}$ of each concept classifier, and then adapt only half of the classifiers whose decline is

above the average decline, and leave the other half of the classifiers unchanged. Meanwhile, we assume the average number of training examples for each concept is fixed at n, although different concepts do not have to have the same number of examples. This is important because we cannot compare different selective adaptation methods if they consume different number of training examples. In this simple method, because only half of the classifiers are adapted, each of them receives $2n$ examples so that the per-concept number of examples is still n.

This simple method can be generalized as follows. Instead of making a hard decision on whether to adapt a classifier, we can make a soft decision as to the number of training examples each classifier receives for adaptation. Suppose we divide all the concept classifiers into k buckets according to their decline $\Delta AP_{within} \times \hat{d}$, with each bucket having (approximately) the same number of concepts in it. These buckets are numbered $1, 2, .., k$, starting from the one containing concepts with the smallest declines. The declines of the concepts falling into bucket i are above the $(i-1)/k \times 100$ percentile of all declines and below the $i/k \times 100$ percentile. We distribute the training examples among the concepts such that the concepts in bucket $i$receive more examples than those in bucket $i-1$. Each concept in bucket k receives $2n$ examples, while each concept in bucket 1 receives no examples (therefore no adaptation is performed on it). As a result, concepts with larger decline, or worse generalizability, receive more examples for adaptation than those with smaller decline. Specifically, the number of examples $S(i)$ a concept in the i_{th} bucket receives is given by the following equation:

$$S(i) = \begin{cases} 2n \times i/k, & \text{if } i > k/2 \\ 2n \times (i-1)/k, & \text{if } i <= k/2 \end{cases} \tag{6.3}$$

It is easy to prove that the average number of examples of a concept is still n. We call this method k-way selective adaptation. The simple method discussed at the beginning is a special case of this approach at $k = 2$.

We further extend this k-way selection strategy into a weighted sampling strategy. When k exceeds the number of concepts, each bucket has only one concept in it, and each concept receives a different number of examples according to the predicted performance decline. This is similar (but not equivalent) to a weighted sampling strategy, where each example is assigned to a concept with the probability proportional to the

Performance (MAP)		Average Sample Size (per concept)							
		200	400	600	800	1000	1200	1400	1600
Non-Selective adaptation		0.219	0.232	0.240	0.253	0.259	0.263	0.265	0.267
Generalizability-based selection	2-way	**0.231**	**0.246**	**0.257**	**0.259**	**0.261**	0.263	0.264	0.268
	4-way	0.227	0.241	0.256	0.257	**0.261**	0.265	0.266	**0.269**
	8-way	0.227	0.238	0.253	0.257	0.258	**0.267**	**0.267**	0.268
	sampling	0.225	0.236	0.247	0.255	**0.261**	0.264	0.266	0.266

Table 6.5: Comparison of non-selective adaptation and several methods of generalizability-based selective adaptation in NBC/CNN setting in terms of MAP over 39 concepts.

predicted decline. We implement such a weighted sampling method which assigns training examples to concepts in this way until all the available examples are assigned. As a result, the number of examples each concept receives is approximately proportional to its predicted decline.

We apply the generalizability-based selective adaptation methods to adapt the classifiers of 39 concepts from NBC to CNN. Table 6.5 compares the performance of these methods to the performance of non-selective adaptation as the average number of examples per concept increases. For a certain average number of examples, say, 200, the non-selective method adapts each classifier using a-SVM based on exactly 200 examples. In contrast, the selective methods adapt different classifiers using different number of training examples (some with zero examples), while keeping the per-concept number of examples at 200. The performance is averaged over the classifiers (adapted or unadapted) of all the concepts. We see that the selective adaptation methods have a clear advantage over the non-selective method, especially when the training examples are scarce. The advantage shrinks as the number of examples increases. This is because for any concept, when the number of training examples increases, the performance improvement *per example* typically decreases and the performance levels off after reaching a certain number of examples. Therefore, when training examples are plentiful, the improvement-per-example is small for any concept, and different methods for allocating training examples do not cause as great a difference as when the training examples are scarce. Moreover, among different selective adaptation methods, the simple 2-way selection method, which adapts only half of the concept classifiers, has the best overall performance than more sophisticated methods.

6.3.2 Learnability-based methods

The generalizability of a concept classifier is related to the *potential* room of improvement of adaptation. It may not reflect the *actual* improvement, and it certainly does not reflect the *rate* of improvement. For example, a concept classifiers with 0.6 AP in its source domain is predicted to achieve 0.2 AP in the target domain. Even if the prediction is accurate, it does not mean the improvement from adaptation will be 0.4 AP, because detecting the same concept in the target domain may not as easy (or difficult) and the highest performance is not necessarily 0.6 AP. More importantly, it does not tell how fast this improvement can be achieved, namely how many training examples are needed for such improvement. This leads us to examine the *learnability* of a concept, which is measured by the amount of improvement achieved through adaptation based on a certain amount of examples. In order to maximize the overall performance for a fixed training data size, more training examples should be assigned to concepts that are more learnable as they bring larger improvement per example than the other concepts.

It is extremely difficult to predict the improvement-per-example of adapting a classifier, unless we actually perform the adaptation and evaluate the adapted classifier on the entire target domain. In practice, however, we want to measure learnability *before* adaptation, and the target domain is unlabeled besides a small set of training data. To overcome these problems, instead of the batch-mode adaptation where each classifier is adapted once using all its examples, we adapt a classifier *iteratively* based on a set of examples that increase by a certain number in each step. This allows us to compute the average performance improvement of a concept in the previous one or more steps based on the labeled examples. The average historical improvement can be used as a measure of the learnability of a concept. We propose a selective adaptation approach which iteratively chooses the most learnable concept classifier to adapt based on gradually increasing training examples. In each iteration, we compare the average historical improvement across all concepts and choose the one with the largest. Then we increase the training examples of the chosen concept by a certain amount, and re-adapt the classifier based on the expanded training data. Then we start a new iteration where the most learnable concept is identified and its classifier is re-adapted based on more labeled data. As a result, the classifiers with a history of fast improvement receive more

examples for adaptation, while classifiers with slow improvement receive less examples.

There are two design issues regarding this learnability-based selection strategy. First, how many steps shall we look back into the adaptation history of a concept to calculate its average historical improvement? This is defined by window size w, which can be any value from 1 to infinity (infinity means we look back all the way to the first step). Small window sizes may not accurately capture the learnability of a concept due to the variance caused by randomly selected examples used in each step. Large window sizes may not be appropriate too, because future improvement can be better approximated by improvement in more recent steps than in earlier steps. This leads to the second issue, namely whether and how to weight the improvement in different steps according to recency. A solution is to use a decay factor $\alpha \leq 1$ to discount the historical improvement at a certain step as many time as the number of steps between it and the current step. The smaller α is, the larger the discount. When $\alpha = 1$, the improvement from each step in the window is treated with equal importance. Therefore, the average historical improvement as a measure of the learnability of a classifier after it has been adapted t times is defined as

$$L(t) = \frac{\sum_{i=\max(1,t-w+1)}^{t} \alpha^{t-i}(AP_i - AP_{i-1})}{\sum_{i=\max(1,t-w+1)}^{t} \alpha^{t-i}} \tag{6.4}$$

where AP_i is the average precision of the adapted classifier after it is adapted i times, and AP_0 is the performance of the classifier before adaptation, all evaluated on the current set of labeled examples. For the classifier with the highest $L(t)$, we increases its training examples by labeling a certain number of new examples (100 in our case) and re-adapt the its classifier. A remaining problem is that before any adaptation is performed ($t = 0$), there is no history of improvement and $L(0) = 0$ for every concept classifier, so it is unable to distinguish the most learnable classifier. To overcome this problem, we adapt each classifier once using an initial set of 100 examples for each concept, after which we can use Eq.(6.3.2) to compute the learnability. As a side effect, this method has a "cold start" problem, since the performance after the first 100 examples per concept is equal to that of non-selective adaptation despite the window size and decay factor.

In Table 6.6, we compare several learnability-based selective adaptation methods to non-selective adaptation in terms of their performance in NBC/CNN setting. These

Performance (MAP)		Average Sample Size (per concept)							
		200	400	600	800	1000	1200	1400	1600
Non-Selective adaptation		0.219	0.232	0.240	0.253	0.259	0.263	0.265	0.267
Learnability-based selection	$w = 1, \alpha = 1$	0.227	0.231	0.232	0.238	0.240	0.240	0.242	0.253
	$w = 2, \alpha = 1$	0.229	0.240	0.243	0.244	0.259	0.263	0.265	0.268
	$w = 2, \alpha = 0.8$	0.229	0.240	0.241	0.247	0.257	0.263	0.262	0.268
	$w = 5, \alpha = 1$	0.229	**0.248**	0.253	0.254	0.264	0.266	0.266	0.268
	$w = 5, \alpha = 0.8$	**0.230**	0.247	0.252	0.254	0.263	0.264	0.264	0.268
	$w = \infty, \alpha = 1$	0.227	**0.248**	**0.256**	**0.258**	0.261	0.266	**0.267**	0.267
	$w = \infty, \alpha = 0.8$	0.229	**0.248**	**0.256**	**0.258**	**0.265**	**0.267**	**0.267**	**0.269**

Table 6.6: Comparison of non-selective adaptation and several methods of learnability-based selective adaptation in NBC/CNN setting in terms of MAP over 39 concepts. w denotes the window size, and α denotes the decay factor.

methods differ in terms of window size, which is set to 1, 2, 5, and ∞ (infinity), and decay factor, which is set to 1 or 0.8. The results show that this learnability-based method consistently outperforms non-selective adaptation when the window size is equal or larger than 5. The run with window size ∞ and decay factor 0.8 has the best performance, suggesting that all the historical improvements are useful in measuring learnability. When the window size is small (1 or 2), the selective adaptation method sometimes performs worse than the non-selective method.

6.3.3 Hybrid methods

We have shown that both generalizability and learnability are useful clues in the selection and prioritization of adaptation tasks. These two clues are complementary: generalizability defines the starting performance of a classifier before adaptation, which is related to the potential room of improvement from adaptation, while learnability defines how fast the performance improves during adaptation. It is natural to assume that a selective adaptation method combining both clues will perform better. By comparing Table 6.5 and Table 6.6, we see that the best learnability-based method outperforms than the best generalizability-based method except when the training examples are very rare, which can be explained by the former's "cold start" problem. Therefore, we design a hybrid method which adopts the same iterative structure as the learnability-based method, but circumvents its cold start problem by using generalizability as the initial estimate of learnability. Specifically, we add an imaginary

step ($t = 0$) before the first step and set the improvement of this initial step to the predicted absolute performance decline, namely $AP_{within} \times \hat{d}$. The intuition is that before we have a chance to observe the improvement from actual adaptation, we can prioritize the classifiers based on their generalizability such that classifiers with larger decline are adapted first. Therefore, this hybrid method does not have to assign an initial set of training examples to every concept like in the learnability-based method. Instead, it makes educated decisions on allocating examples from the very beginning, so we expect it to perform better especially when the training examples are limited. Until reaching a certain number of adaptation steps, this generalizability factor will still affect the calculation of historical improvement, but its influence is discounted. After we have seen a long history of improvement from actual adaptation, it is safe to ignore the generalizability in deciding the priority of adaptation tasks.

In this hybrid method, we modify the calculation of average historical improvement from Eq.(6.3.2) to the following:

$$L(t) = \begin{cases} \frac{\sum_{i=1}^{t} \alpha^{t-i}(AP_i - AP_{i-1}) + \alpha^t \hat{d} AP_{within}}{\sum_{i=0}^{t} \alpha^{t-i}}, & \text{if } t < w \\ \frac{\sum_{i=t-w+1}^{t} \alpha^{t-i}(AP_i - AP_{i-1})}{\sum_{i=t-w+1}^{t} \alpha^{t-i}}, & \text{if } t >= w \end{cases} \tag{6.5}$$

Therefore, when the initial step is still in the window ($t < w$), the predicted decline has an impact on $L(t)$ which is discounted as t increases. When the initial step is out of the window ($t >= w$), the decline no longer has any impact on $L(t)$ and Eq.(6.3.3) is exactly the same as Eq.(6.3.2).

Figure 6.9 compares the performance of the hybrid method with parameter $w = \infty$ and $\alpha = 0.8$ to that of the learnability-based method with the same parameter, the generalizability-based method (2-way), and non-selective method. The first two are iterative methods where the number of examples is increased by 100 in each step. All the three selective adaptation methods are consistently better than the non-selective method, showing the benefit of selective adaptation in improving cost-efficiency. The hybrid method outperforms both the learnability-based and the generalizability-based method by allowing them to complement each other. As we have expected, the hybrid method overcomes the cold start problem in the learnability method with a high starting performance. Note that although the hybrid method initially relies on the

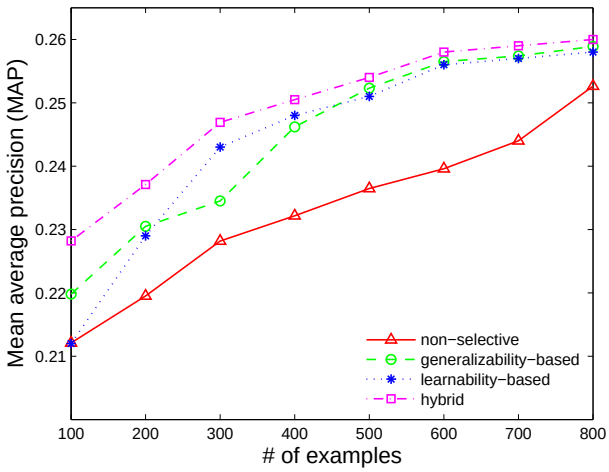


Figure 6.9: Average performance on 39 concepts using selective and non-selective adaptation methods in NBC/CNN setting.

predicted generalizability for allocating examples, its starting performance (at 100 average examples) is different from the starting performance of the generalizability-based method. This is because the former performs adaptation iteratively while the latter performs in batch mode, and as a result the number of examples for each concept is not the same between these two methods.

6.4 Generalizability for Parameter Selection

The common practice in detecting a semantic concept is to build *many* classifiers for this concept using different features and model parameters, and to choose the one with the highest cross-validation performance on the training data. This is because features and certain model parameters, such as cost factor and gamma parameter in SVM, have a large impact on performance. The goal is to achieve the best performance on test data, which cannot be directly measured due to the lack of truth labels. So the hope is that the classifier with the highest performance on the training data also has the highest performance on the test data.

In a cross-domain setting, the goal is to identify, among classifiers of a given concept trained with different model parameters, the one generalizing the best to target domains, namely the one with the highest cross-domain performance AP_{cross}. (We do not consider different features because as Table 6.1 shows the **gcm-gbr** feature is

Feature f	**Correl** (f, AP_{cross})	
	NBC/CNN	TV05/TV07
cost factor C	0.013	-0.288
gamma γ *in RBF kernel*	-0.045	-0.031
SV ratio in training data	0.732	0.750
SV ratio in positive training data	-0.177	0.731
maximum output of classifier	0.250	0.393
range of classifier's output	0.056	0.342

Table 6.7: Meta-features and their correlation coefficients with cross-domain performance.

significantly better than other features.) However, due to the distribution difference between domains, the best classifier in the source domain is not necessarily the best performer of the target domain. This has been shown in Figure 6.6, where for many concepts the highest AP_{within} and the highest AP_{cross} are achieved using different parameter settings. This means we cannot choose the most generalizable classifier based on AP_{within}. Our solution is to build a model that directly predicts for each concept, AP_{within} of a classifier trained with certain parameters. This model is similar to the generalizability model discussed in Section 6.2, except that it focuses on the generalizability of classifiers of a specific concept while the latter focuses on the generalizability between different concepts.

Table 6.7 lists several model meta-features and their correlation coefficients with the cross-domain performance AP_{cross}. To calculate each coefficient in the table, we first compute the correlation coefficient of each concept from all its classifiers trained with different model parameters, and then average the coefficients across the 39 concepts. We build a regression model to predict AP_{cross} from these meta-features of a classifier, using the SVR algorithm with linear kernel function [36]. A key question here is whether to build such a model for each concept, or to build a universal model for all concepts. A concept-specific model is obviously very restrictive since it cannot be used for other concepts. To train a universal model for predicting AP_{cross}, however, a major difficulty is that the AP_{cross} of some concepts are at a much higher range than that of other concepts, and the difference overwhelms the relatively small variation on AP_{cross} caused by model parameters. To overcome this problem, we normalize the AP_{cross} of a concept classifier by dividing it using a *reference performance*, which is

Parameter selection	**NBC classifiers on CNN**			**TV05 classifiers on TV07**		
	CrossVal	ModelPred	Oracle	CrossVal	ModelPred	Oracle
Airplane	0.018	0.020	0.027	0.011	0.011	0.013
Animal	0.129	0.136	0.136	0.058	0.069	0.072
Boat_Ship	0.013	0.013	0.013	0.029	0.038	0.039
Building	0.198	0.203	0.211	0.330	0.330	0.331
Bus	0.014	0.012	0.017	0.008	0.006	0.008
Car	0.367	0.367	0.380	0.112	0.108	0.113
Charts	0.038	0.056	0.077	0.039	0.032	0.045
Computer_TV-screen	0.160	0.215	0.215	0.104	0.096	0.117
Corporate-Leader	0.018	0.020	0.021	n/a	n/a	n/a
Court	0.006	0.006	0.008	0.010	0.009	0.011
Crowd	0.400	0.400	0.413	0.407	0.408	0.410
Desert	0.024	0.037	0.053	0.014	0.014	0.014
Entertainment	0.006	0.010	0.010	n/a	n/a	n/a
Explosion_fire	0.016	0.016	0.028	0.009	0.008	0.012
Face	0.714	0.777	0.781	0.785	0.784	0.791
Flag-US	0.116	0.128	0.136	0.002	0.003	0.003
Government-Leader	0.172	0.182	0.191	n/a	n/a	n/a
Maps	0.067	0.088	0.092	0.055	0.054	0.068
Meeting	0.049	0.060	0.062	0.079	0.090	0.090
Military	0.187	0.181	0.205	0.051	0.055	0.059
Mountain	0.067	0.082	0.082	0.035	0.038	0.044
Natural-Disaster	0.013	0.019	0.020	0.011	0.011	0.012
Office	0.055	0.063	0.063	0.169	0.162	0.186
Outdoor	0.599	0.585	0.613	0.694	0.715	0.724
People-Marching	0.119	0.112	0.120	0.123	0.123	0.123
Person	0.896	0.900	0.904	0.861	0.843	0.880
Police_Security	0.033	0.033	0.073	0.026	0.026	0.029
Prisoner	0.011	0.027	0.069	0.005	0.006	0.006
Road	0.190	0.179	0.192	0.241	0.237	0.259
Sky	0.479	0.437	0.496	0.648	0.654	0.654
Snow	0.017	0.021	0.041	0.023	0.038	0.041
Sports	0.014	0.014	0.017	0.076	0.077	0.088
Studio	0.771	0.769	0.777	0.081	0.087	0.100
Truck	0.012	0.011	0.013	0.021	0.021	0.023
Urban	0.088	0.093	0.099	0.291	0.291	0.308
Vegetation	0.302	0.292	0.302	0.435	0.437	0.454
Walking_Running	0.165	0.167	0.177	0.230	0.217	0.230
Waterscape_Waterfront	0.168	0.197	0.197	0.177	0.167	0.177
Weather	n/a	n/a	n/a	0.009	0.023	0.023
average	0.177	0.183	0.193	0.174	0.175	0.182

Table 6.8: Cross-domain performance of classifiers selected using three methods. "CrossVal" selects the classifier with highest cross-validation performance in source domain; "ModelPred" selects the one with the highest *predicted* performance on the target domain using our model; "Oracle" selects the one with the highest *actual* performance on the target domain.

set to the AP_{cross} of a classifier of the same concept trained with particular parameter setting ($\gamma = 1, C = 1$). The normalized performance, which is equal to the multiple of AP_{cross} to the reference performance, is not affected by concept and only influenced by model parameters. Therefore, the model can be trained using meta-features from the classifiers of all the concepts, using the normalized performance as the output (response), and it can be applied to any classifier.

We use the above model to predict the normalized performance of each classifier (i.e., multiple of reference performance), which can be used to distinguish the classifier with the highest AP_{cross} for each concept. This provides an alternative to selecting classifiers based on cross-validation performance in the source domain. To compare the two strategies, we use them to select the best-performing classifier on target domain for each concept, among the candidate classifiers trained with different parameters. In the NBC/CNN setting, for example, we build 18 classifiers for each concept from NBC data using the **gcm-gbr** feature with 6 γ values (0.01, 0.05, 0.1, 0.2, 0.4, 1) and 3 C values (1, 3, 10), and the goal is to find the one with the best performance on CNN. The **CrossVal** method selects the one with the highest cross-validation performance on NBC, and the **ModelPred** selects the one with the highest *predicted* performance on CNN computed based on our model. For reference purpose, we include the **Oracle** method which selects the classifier with the highest *true* performance on CNN. While this method is guaranteed to be optimal, it is also impractical because we cannot compute the true performance since CNN data are assumed to be unlabeled.

Table 6.8 compares the performance of the three selection methods in two settings. We see that on average, classifiers selected based on our model perform better than classifiers selected based on cross-validation performance, especially in the NBC/CNN setting. On 27 out of 38 concepts in the NBC/CNN setting, or 23 out of 36 concepts in the TV05DEV/TV07DEV setting, the classifier selected based on $\widehat{AP}_{cross}$ performs better or at least equally well on the target domain as the one selected based on cross-validation. In both settings, the performance of **ModelPred** is only moderately better than **CrossVal**, and well below the performance of the **Oracle** method. This shows that our model predicts the cross-domain performance of classifiers trained with different parameters relatively well, but there is still a large room for improvement.

6.5 Summary

In this chapter we have focused on improving the cost-efficiency of adaptation through selecting and prioritizing classifiers to be adapted based on their generalizability. We

have conducted an empirical analysis of the generalizability of semantic concept classifiers, which reveals strong correlation between generalizability and various meta-features of a concept classifier, including the ratio of positive training data, its model structure such as the ratio of support vectors, the distribution of its output. We have built a regression model based on these meta-features to predict classifiers' generalizability, which has been shown capable of accurately predicting a classifier's performance on a new (unlabeled) domain other than its training domain. Based on this generalizability model, we have then proposed several selective adaptation methods, which selective and prioritize adaptation tasks involving multiple classifiers such that the less-generalizable and more-learnable classifiers are adapted with higher priority. Experiments on cross-domain concept detection have shown that compared with treating every classifier equally in adaptation, these selective adaptation methods achieve higher overall performance for the same number of training examples. Another use of such generalizability model is to identify, among the classifiers of a given concept trained with different model parameters, the one that generalizes the best to the target domain.

Chapter 7

Classifier Adaptation in Other Areas

In previous chapters, we have shown the effectiveness of the proposed approach in cross-domain semantic concept detection. While motivated by the need for classifier adaptation in multimedia, our approach is not restricted to this single application or even the entire multimedia area. In fact, our approach is very general and flexible because it imposes almost no restrictions on the data distribution, features, or classifiers. For example, data distribution is allowed to change in arbitrary and unknown ways across domains; classifiers of any type can be adapted; features can be continuous or discrete and of any dimension. To demonstrate its broad applicability, we apply our approach to adaptation problems in other areas, including the adaptation of text document classifiers and of a pioneering "mind-reading" model in human-computer interaction.

7.1 Adaptation in Text Categorization

Text categorization [37, 57, 115] is an extensively studied problem in both research and industry. In text categorization, classifiers trained from labeled textual documents are used to determine whether or not a document belongs to certain semantic categories. A large diversity of text corpora have been used for training these classifiers, such as Wall Street Journal (WSJ), Web pages, and so on. From the perspective of scalability and

efficiency, it is desirable to re-use text classifiers trained from one corpus on documents from other corpora. However, similar to our findings in the multimedia area, the performance of text classifiers may decline when they are applied on documents from other-than-training corpora due to the difference on data distribution. Our adaptation approach provides a way to improve such performance by adapting the original text classifiers towards the target corpus with reasonably small effort on labeling training documents.

The adaptation in text categorization differs from cross-domain concept detection in several aspects. Text documents are represented by discrete, high-dimensional term vectors, instead of continuous, lower-dimensional visual features describing images and video shots. Moreover, while SVM is also also extensively used for text categorization, linear kernel function is more popular than non-linear kernel functions such as RBF. Also, the accuracy of text categorization in general is at a much higher level than that of semantic concept detection. Overall, text categorization contrasts well to semantic concept detection, making it a good alternative task for evaluating our adaptation approach. In the following, we show the experiment results of cross-domain text categorization in 20-newsgroup data and Reuters-21578 data, and a sentiment classification task on user reviews.

7.1.1 Experiment: Reuters-21578 corpus

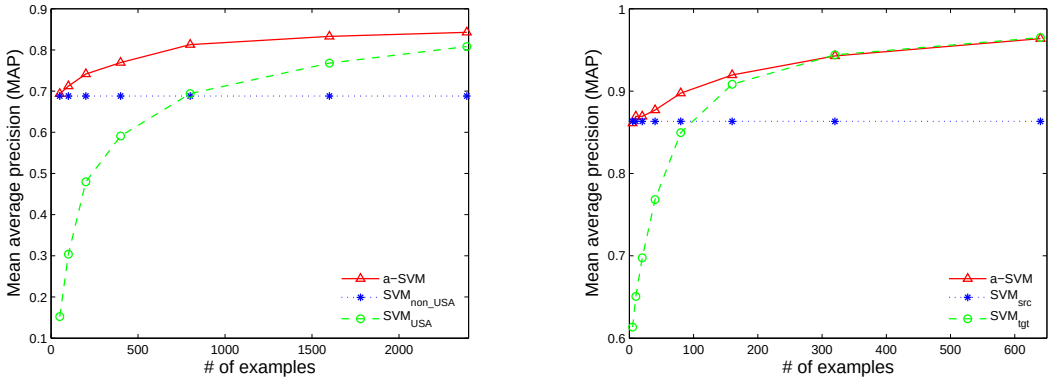
Reuters-21578 is one of the most widely used collections for text categorization research. This collection has 21,578 documents, which appeared on the Reuters newswire in 1978. Each document was labeled with tags in several categories, including TOPIC, PLACE, PEOPLE, ORG. Our experiment of cross-domain text categorization is designed by taking advantage of such multi-dimensional labels. By examining the PLACE tag, we find that about half of the documents are labeled as "USA", while the others are labeled with the names of other countries. We use the set of "non-USA" documents, denoted as $D_{\overline{USA}}$, to be the source domain, and the set of "USA" documents, denoted as D_{USA}, to be the target domain. This split has been used in related works (e.g.,[107]), and it produces related and yet different source and target domain of comparable size. The documents in D_{USA} are further split into a 40% development set and a 60% evaluation set.

Task	cross-validation		Performance on D_{USA}				
	$D_{\overline{USA}}$	D_{USA}	$\text{SVM}_{\overline{USA}}$	SVM_{USA}		a-SVM	
				$S = 100$	$S = 800$	$S = 100$	$S = 800$
acq vs. other	0.888	0.983	0.949	0.820	0.970	0.950	0.972
money-fx vs. other	0.823	0.749	0.560	0.423	0.644	0.576	0.743
grain vs. other	0.873	0.944	0.889	0.429	0.840	0.893	0.931
trade vs. other	0.711	0.841	0.730	0.147	0.666	0.727	0.769
wheat vs. other	0.919	0.774	0.729	0.077	0.498	0.736	0.809
ship vs. other	0.809	0.660	0.679	0.071	0.436	0.696	0.788
corn vs. other	0.863	0.868	0.638	0.165	0.801	0.649	0.801
crude vs. other	0.906	0.896	0.801	0.414	0.844	0.811	0.855
oilseed vs. other	0.730	0.817	0.447	0.177	0.463	0.491	0.624
money-supply vs. other	0.914	0.877	0.459	0.316	0.778	0.592	0.828
average	0.843	0.841	0.688	0.304	0.694	0.712	0.812

Table 7.1: Performance as average precision (AP) on the evaluation set of D_{USA} for 10 topics. The first two columns show the performance of 10-fold cross validation on D_{USA} and $D_{\overline{USA}}$. The classifiers being compared include $\text{SVM}_{\overline{USA}}$ as the (source) classifier trained from entire $D_{\overline{USA}}$, SVM_{USA} as a new classifier trained from the labeled examples in D_{USA}, and a-SVM is a classifier adapted from $\text{SVM}_{\overline{USA}}$ based on the same set of examples.

We identify the top 10 most frequent topics from the TOPIC tags. For each of the top 10 topics, all the documents labeled with this topic are treated as positive, and the other documents are treated as negative. For this topic, we build a classifier from its positive and negative documents in source domain $D_{\overline{USA}}$, which is the source classifier. We then adapt this classifier using a-SVM to the target domain D_{USA} based on a small number of labeled examples randomly selected from its development set. The performance of the adapted classifier is evaluated on the evaluation set of D_{USA} using average precision (AP) as the metric. This experiment is repeated such that each of the 10 topics is used once to define positive and negative data.

Table 7.1 compares the performance of several methods on classifying documents in D_{USA} w.r.t the top 10 topics. Before adaptation, source classifiers trained from $D_{\overline{USA}}$ on average perform worse on D_{USA} (0.688 MAP) than on their training data $D_{\overline{USA}}$ (MAP = 0.843). This means the change of corpus compromises the performance of text classifiers, although the impact is smaller than that in semantic concept detection and marginal for some topics (e.g., "acq", "ship"). So there is still a clear need for adaptation. Meanwhile, the performance of source classifiers on D_{USA} is also much lower than the cross-validation performance on D_{USA} (MAP = 0.841), implying a



(a) Reuters-21578 (b) 20-newsgroups

Figure 7.1: Comparison of different classifiers on (a) Reuters-1578 and (b) 20-Newsgroups data.

large room of improvement for adaptation. The average performance of the adapted classifiers on D_{USA} is 0.712 MAP with 100 examples and 0.812 MAP with 800 examples, both significantly higher than that achieved by training new classifiers exclusively from the examples in D_{USA}. With 800 samples, the adapted classifiers perform very close to the cross-validation performance on D_{USA}, which is achieved using all the training examples. This shows the effectiveness of the adaptation approach based on a-SVM.

In Figure 7.1 (a), we show how the average performance on D_{USA} under different methods changes with the number of training examples. Similar to the observation in cross-domain semantic concept detection in Section 4.5, a-SVM performs close to the source classifier $\text{SVM}_{\overline{USA}}$ when labeled data are scarce, but improves when more labeled data become available. It performs better than the new classifiers SVM_{USA} trained exclusively from the labeled data in D_{USA}, although the advantage shrinks as training examples increase. The main difference is that the performance numbers here are at a much higher level than those in concept detection, implying that text categorization is an "easier" task. This shows a-SVM is able to perform well despite the difficulty of the task.

Task	cross-validation		Performance on D_{tgt}				
	D_{src}	D_{tgt}	SVM_{src}	SVM_{tgt}		a-SVM	
				$S = 20$	$S = 160$	$S = 20$	$S = 160$
comp vs. sci	0.965	0.992	0.834	0.782	0.905	0.831	0.891
rec vs. talk	0.984	0.992	0.801	0.672	0.926	0.835	0.924
rec vs. sci	0.981	0.992	0.824	0.631	0.878	0.820	0.886
sci vs. talk	0.968	0.979	0.834	0.667	0.868	0.826	0.882
comp vs. rec	0.996	0.990	0.911	0.630	0.926	0.935	0.958
comp vs. talk	0.994	0.992	0.975	0.801	0.943	0.976	0.980
average	0.981	0.990	0.863	0.697	0.908	0.871	0.920

Table 7.2: Performance as average precision (AP) on the evaluation set of D_{tgt}. The two columns on the left show the performance of 10-fold cross validation on D_{src} and D_{tgt}. The classifiers being compared include SVM_{src} as the (source) classifier trained from entire D_{src}, SVM_{tgt} as a new classifier trained from the labeled examples in D_{tgt}; and a-SVM is a classifier adapted from SVM_{src} based on the same set of examples.

7.1.2 Experiment: 20-Newsgroups corpus

20-Newsgroup collection is another widely used corpus for text categorization. It contains 18,774 articles labeled against a two-level category system. That is, each article belongs to a top-level category (e.g., "comp") as well as a sub-category (e.g., "comp.graphics") under the top-level category. We design the adaptation task based on such two-level category structure. For each top-level category, we divide its sub-categories into two non-overlapping groups, and treat one group as part of the source domain D_{src} and the other group as part of the target domain D_{tgt}. We build classifiers to distinguish two top-level categories from the labeled documents in D_{src}, and then adapt them to the documents in D_{tgt}. For example, for a classifier that distinguishes "comp" (computer) documents and "rec" (recreation) documents, the source domain D_{src} consists of documents in sub-category 'comp.graphics", "comp.os.ms-windows", "rec.autos", and "rec.motocycles", and the target domain D_{tgt} consists of documents in sub-category "comp.sys.ibm", "comp.sys.mac", "rec.sport.baseball", and "rec.sport.hockey".

Table 7.2 shows the performance on 6 classification tasks. Similar to the results on Reuters-21578, the source classifiers SVM_{src} trained from D_{src} have lower performance on D_{tgt} than on D_{src}, but their performance improves after adaptation using a-SVM based on limited labeled examples. The adapted classifiers outperform classifiers SVM_{tgt} trained from scratch, although the gap is smaller than that found on

Reuters-21578. This is confirmed in Figure 7.1(b), which shows the average performance of the 6 tasks against the number of training examples per task. While a-SVM performs much better than SVM_{tgt} with limited training examples, the two quickly converge and perform close to each other after the number of labeled data reaches 300 per task. This is different from the findings in Reuters-21578, where a-SVM performs better even when over 2,000 examples are used. This shows the classification tasks in 20-Newsgroups are more learnable than in Reuters-21578. Nevertheless, using a-SVM avoids the problem of low initial performance that occurs on the new classifiers built from scratch.

7.1.3 Experiment: sentiment classification

Sentiment classification aims to determine user opinions from text documents such as online product reviews. This research becomes very important as the number of online reviews of products, movies, and other information has increased significantly. Sentiment classification is more challenging than conventional text categorization, because determining user opinion requires deeper understanding of the meanings of documents than classifying topics. Nevertheless, existing text categorization methods are still applicable.

Two types of user reviews are used in this experiment. There are 2,000 online movie reviews, half of which are labeled as positive (like) based on human judgments and the other half are labeled as negative (dislike). Moreover, there are also 11,727 reviews of a software for personal finance management called Quicken, of which 5,680 are positive and the rest are negative. Among them, a small subset of only 30 product reviews is used as the development data, and all the other reviews besides the 30 are used for evaluation. Term vectors are used to represent these review documents after removing stop-words and stemming. In the experiment, we build a SVM classifier with linear kernel function from the 2,000 labeled movie reviews, and adapt it towards the product reviews based on the 30 labeled product reviews in the development set. The adapted classifier is evaluated on the product reviews excluding the development set.

Table 7.3 compares the performance of several methods on classifying the user opinions in the product reviews. SVM_{mov} is a SVM classifier trained from the 2,000 movie reviews with *no* adaptation; SVM_{pod} is a SVM classifier trained from the development

	Average precision (AP)		
	binary	tf	tf-idf
SVM_{mov}	0.603	0.636	0.658
SVM_{pod}	0.590	0.516	0.557
a-SVM	**0.715**	**0.687**	**0.712**
$SVM_{mov+pod}$	0.653	0.657	0.685

Table 7.3: Performance (AP) of sentiment classification in product reviews using different methods and term weighting schemes.

set of 30 labeled product reviews; a-SVM is the classifier adapted from SVM_{mov} based on the 30 labeled product reviews using the proposed a-SVM algorithm; $SVM_{mov+pod}$ is a SVM classifier trained using the 2,000 movies reviews and 30 product reviews combined.

Despite the weighting scheme of term vectors (binary, tf, or tf-idf), a-SVM outperforms other methods by significant margins. Unlike in concept detection task, a-SVM is even better than $SVM_{mov+pod}$, the expensive data-fusion method which builds a classifier from the combined training data. Overall, adaptation has shown to be effective in this difficult type of text categorization task.

7.2 Adaptation of EEG-based Relevance Models

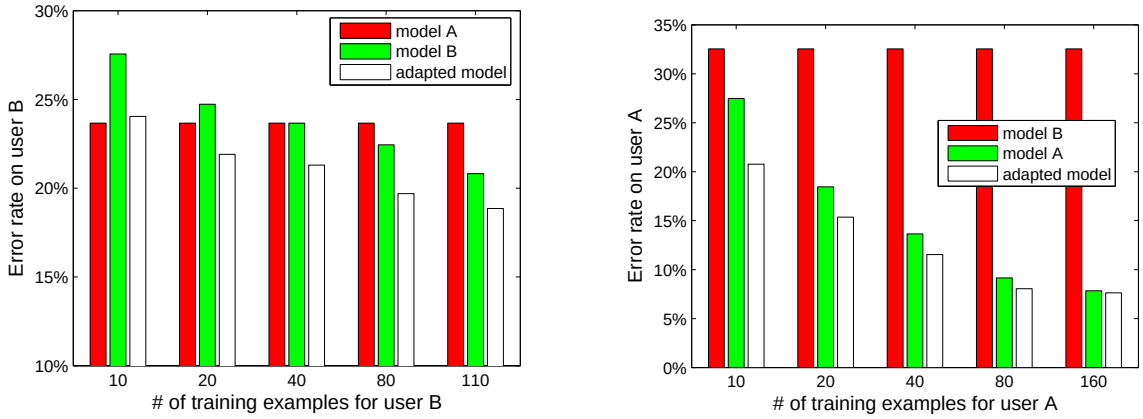
Another interesting application of our approach is the adaptation of a pioneering "mind-reading" model for human-computer interaction, which is used to determine the relevance judgement of humans based on their electroencephalogram (EEG) brain signals. In various multimedia systems, human subjects are often asked to judge whether images (or video shots) are relevant to a given semantic concept or query, and convey such judgments to the system through interactions such as pressing certain keys on the keyboard or clicking a button using mouse. This scenario occurs in interactive image or video retrieval as well as in collecting training data for semantic concept detection. Although this is a simple and intuitive action for human, the time required for judging whether an image is relevant is *much shorter* than that for marking that image through physical interactions. Time can be saved if the system is able to immediately learn the subject's judgment without physical interactions, and the saving can be significant when the number of images to be judged is large. This would also release the

human subject from the stress of repeating the same actions (e.g., pressing a key) for hundreds or even thousands of times. One potential solution is for the system to read the EEG signals from the subject's brain to determine whether he/she recognizes an image as relevant or not.

The EEG signals are electrical impulses within human brain caused by post-synaptic potentials from firing neurons. We use a portable device called Pocket Neurobics Pendant EEG, and put its electrodes on two locations on the head close to the parietal lobe. The parietal lobe is where P300 brain wave, an indicator of cognitive function in decision making, is mostly readily extracted. To minimize the noise in the signals, we record signals in windows of 200 milliseconds and average the signals recorded during each window. The signal of each channel (electrode) is segmented into different frequency bands (up to 50Hz), and fast Fourier transformation (FFT) is applied on the signal in each band, which produces a feature vector of 170 components. During the time a human subject is judging an image, we collect EEG signals from the two channels in two 200-millisecond window, which results in a 680-d feature vector.

The goal is to build a model that determines the relevance judgment of a human subject from the EEG signal represented by this 680-d feature vector collected while he/she is making judgments. The training data are collected using our extreme video retrieval (XVR) system [50], which allows users to quickly scan through an ordered list of images returned by keyword-based search and mark them as relevant or irrelevant by pressing the "U" key on the keyboard. Users wearing the EEG device is asked to use the system to mark images' relevance to given queries while their EEG signals are being collected. The marked relevance label on each image and the corresponding EEG signal feature constitute one training example. Based on hundreds or thousands of such training examples, the relevance model can be trained using a (binary) supervised classification algorithm, such as logistic regression or SVM. The experiments show that the model trained with sufficient data of a user achieves over 90% accuracy in "guessing" the user's relevance judgement. This allows us to use the EEG-based relevance model to automatically determine users' judgments in the XVR system, instead of having them do that through physical interactions.

A potential weakness of this approach is that the relevance model is sensitive to many factors, varying from the user identity to the placement of the electrodes. Due to



(a) Performance on user B (b) Performance on user A

Figure 7.2: Error rate of EEG-based relevance models on user A and user B. In the left figure, **model A** is trained from all the examples for user A, **model B** is trained from part of the examples for user B, and the **adapted model** is adapted from model A using the examples for user B. In the right figure, the role of user A and B are reversed.

the difference on the brain signals between users, the model trained on the data of one user is unable to determine the relevance judgment of other users as accurately as on that particular user. Meanwhile, training a brand-new model for each user is tedious because it requires every user to do a hours-long training session in order to collect sufficient training data. A better idea is to adapt the model trained from another user to the current user who provides only a small number of training examples. The approaches proposed in this thesis can be used for the adaptation of such relevance models.

We collected the training data from two users while they were using the XVR system and having their EEG signals recorded. There are 1538 examples for user A and 275 examples for user B. In the experiment, we first build a relevance model from *all* the data of user A using SVM with linear kernel, and adapt it towards user B using a-SVM. To evaluate the adapted model, we partition the training examples of user B into a development set of 40% data and an evaluation set of 60% data. The adaptation uses training examples randomly selected from the development set, and the adapted model is evaluated on the evaluation set, using the error of predicted user judgments as the performance metric. The performance is compared to that of the original model of user A (without adaptation) and a new model for user B trained exclusively from the examples of B. Then we repeat the experiment by adapting B's model towards user A.

As Figure 7.2 shows, in both experiments the adapted models outperform the original (unadapted) models and new models built from scratch. This is true no matter how well the model from a different user performs on the current user. In the adaptation from A to B, the model of user A has a decent performance on user B, and the model adapted from it keeps a non-trivial advantage over the new model even when the number of training examples increases. In the adaptation from B to A, the model for B does not perform well on A, and it is in fact worse than a new model trained with only 10 examples from A. This is perhaps due to the insufficient training data for user B. Nevertheless, the adapted model still performs better than the new model, although its advantage fades as more training examples become available.

7.3 Summary

We have shown the application of our approach in adapting models of several research areas besides multimedia. The adaptation tasks evaluated in this chapter, together with cross-domain concept detection evaluated in previous chapters, contrast well in terms of the type of feature, the level of difficulty, classification algorithm, etc. Being able to consistently achieve good results in various tasks has shown that our approach, while motivated from problems in multimedia, is flexible and applicable to adaptation in many other areas.

Chapter 8

Conclusion and Future Directions

In this chapter, we summarize the major contributions of this thesis, and discuss the future directions of model adaptation techniques in multimedia and in other areas.

8.1 Summary of Contributions

Machine learning techniques have been extensively used to build models for the analysis and retrieval of multimedia data. Because of the diversity of multimedia data and the limitations of their representations, models trained from one data domain do not usually generalize well to other domains. To overcome the poor generalizability of existing models and avoid the cost of building new models, this thesis is dedicated to the problem of re-using and generalizing learning models across different domains with relatively little human intervention. Specifically, its focus is mainly on developing a systematic approach to adapting supervised classifiers trained from some domains to other domains, a generic problem in multimedia and many other areas. This includes not only principled methods for adapting classifiers under different circumstances, but also empirical analysis of classifier generalizability for better selection and prioritization of adaptation tasks. The proposed approaches have been shown effective in solving adaptation problems in multimedia as well as in other areas. The major contributions of this thesis are summarized as follows:

- This thesis has presented **a study on the generalizability issue in semantic concept detection**, an important task in multimedia and the foundation of

many other tasks. Based on experiments on TRECVID video corpora, this study has shown that in general, concept classifiers fail to generalize well to new domains other than their training domain, with an average decline of 60% to 70% of its performance. The poor generalizability is due to the fact that majority of concept classifiers are unable to "summarize" the data and resort to a nearest-neighbor approach in classification. This is to our knowledge the first comprehensive study on the generalizability of semantic concept classifiers, and the results fully justify the need for classifier adaptation techniques.

- The foundation of our adaptation approaches is **a general framework for function-level classifier adaptation** based on regularized loss minimization. It adapts an existing classifier trained from a source domain to a classifier for a target domain which has only limited labeled examples. As a fundamental difference from existing techniques, in this framework the adaptation of a classifier is done by directly modifying its decision function, rather than re-using its training data to build a new classifier. This approach makes the framework highly efficient, capable of adapting any type of classifiers, and applicable to tasks where previous training data are not available. The framework employs a generic objective function which ensures that only *minimal but necessary* changes are made to an existing classifier such that it can correctly classifies the examples in the target domain. By plugging in different loss and regularization functions into this objective function, one can derive many concrete adaptation algorithms from this framework. We elaborate on two of such algorithms, **adaptive support vector machine** (a-SVM) and **adaptive kernel logistic regression** (a-KLR). The derivation of a-KLR provides a probabilistic interpretation of our framework, which shows that the source classifier plays the role of a "prior model" in the training of the adapted classifier. Experiments on cross-domain semantic concept detection based on TRECVID corpora have shown that our approaches outperform existing (adaptation and non-adaptation) methods in terms of accuracy and/or efficiency.

- We have further extended this basic framework into **a framework for multi-classifier adaptation**, which adapts multiple source classifiers into one classifier

for the target domain. This is to our knowledge the first approach for many-to-one adaptation. The key idea is to combine all the source classifiers into a weighted ensemble and adapt this ensemble to the target domain. We have introduced into the framework the weights of source classifiers to reflect their utility w.r.t the target domain, and modified the objective function to allow the weights to be learned together with the adapted classifier. From this extended framework, we have derived **multi-adaptive support vector machine** (ma-SVM) as a counterpart of a-SVM for multi-classifier adaptation. Experiments has confirmed the benefit of multi-classifier adaptation over single-classifier adaptation in cross-domain semantic concept detection.

- We have proposed several **selective adaptation** methods to improve the overall cost-efficiency of adaptation. Based on the knowledge about the generalizability and/or learnability of each classifier, these methods select and prioritize adaptation tasks involving multiple classifiers, such that their overall performance is maximized after adaptation using a fixed number of training examples. Experiments in semantic concept detection have shown that selective adaptation achieves higher cost-efficiency than blindly adapting every classifier with the same number of training data.

- This thesis has investigated **an empirical approach to predicting the generalizability of semantic concept classifiers**, in order to better determine the need for adaptation and prioritize adaptation tasks. Our study has shown a large variation in concept classifiers' generalizability, influenced mainly by the concept they intended and also by the features and model parameters used in their training. It has revealed strong correlations between generalizability and various meta-features of a classifier, including the ratio of positive training data, model structure, and the distribution of its output. A regression model has been proposed to predict the generalizability of a classifier from these meta-features. We have shown that using this model we can accurately predict the performance of a concept classifier on a new, unlabeled domain other than its training domain.

- We have applied the proposed techniques on **a diversity of adaptation problems in different areas**. These include adapting semantic concept detectors

across video collections, adapting of text classifiers across text corpora, and adapting an EEG-based relevance model across human subjects. The promising results on these tasks have shown that our approach, designed for a generic setting and with few restrictions, is capable of solving adaptation problems not only in multimedia but also in other areas.

8.2 Future Directions

While this thesis has addressed different aspects of classifier adaptation, it still leaves many related problems to be tackled in future work. In the following, we discuss future directions on sample selection for adaptation, the combination of adaptation and semi-supervised learning, etc.

In many applications, there is a typically a large amount of unlabeled data in the target domain in addition to a small set of labeled data. In our approaches, only the labeled data are used in learning the adapted classifiers. Meanwhile, if we do not consider the source domain, semi-supervised learning (SSL) techniques [26] can be an appealing choice for classifying the target data, which make use of both labeled and unlabeled data for the training of classifiers. In a sense, the source classifier in our adaptation approaches and the unlabeled data in SSL techniques play a similar role, which is to reduce the model variance caused by the limited training data. It is an interesting future direction to combine our adaptation approach with semi-supervised learning into an unified learning process. Such a combination would allow the adapted classifier to benefit not only from the source classifier but also from the unlabeled target data, which may further improve its performance. An easy-to-implement approach is to combine a-SVM with transductive SVM (TSVM) [58], an extensively used SSL algorithm, because both of them are based on SVM's loss function. One can extend the objective function of a-SVM defined in Eq.(4.5) to include a new term that measures the loss on the unlabeled data of the target domain.

However, the use of unlabeled data needs to be handled with great care, as there is no guarantee that it always helps the the performance. In fact, as shown in Section 4.5, TSVM which uses unlabeled data performs worse than SVM in semantic concept

detection. The possible reasons can be that the positive and negative data are intermingled in the feature space, or that the negative data overwhelms the positive data in number. Therefore, while combining our adaptation approach with an SSL algorithm is technically feasible, investigation into the data distribution of the source and target domain is needed to tell whether the use of unlabeled data is helpful at all.

Another area that deserves investigation is the selection of training examples used for adaptation. In all the experiments, the training examples are *randomly* selected from the target domain. However, not every example is equally useful from the perspective of adaptation. If labeling each example requires a constant amount of human effort, it is more cost-efficient to select examples that help improve the performance of the adapted classifier the greatest. This is similar to active learning [18, 98], which selects the most informative examples as the training data for building a new classifier. In the context of adaptation, the goal should be finding examples that provides *complementary knowledge* to the classifier being adapted. As we have shown in Section 4.3.2, in a-SVM the training examples are used to modify the decision boundary of an existing classifier, so they are most useful when they fall into the wrong side of the existing decision boundary (if they are already on the right side, there is little need to adapt the boundary). Therefore, training examples which the source classifier misclassifies are more valuable than those it classifies correctly. The key question is that, without first labeling a data instance, how to estimate its probability of being classified incorrectly. One choice is to follow the risk minimization framework in several active learning techniques [18], which aims to find examples that achieves the maximum reduction of the expected risk of the classifier. Moreover, since many classification problems in multimedia are *imbalanced*, which is the case in semantic concept detection, examples from the rare class are more valuable than those from the frequent class. It would be interesting to compare the performance of adaptation based on randomly selected examples and based on examples selected using these clues.

In Section 6.3 we have discussed methods for allocating training examples between multiple classifiers to maximize the overall performance after adaptation. It is *orthogonal* to the future work on finding the most effective training examples for each classifier, and these two techniques can be combined to complement each other. That is, we can jointly optimize the priority between classifiers to be adapted and the choice of training

examples for adapting each classifier. The key question to answer is, *which* training example for *which* classifier shall we label in order to maximize the overall performance after adaptation. We expect a combination of the two techniques will achieve better performance than using either technique by itself.

In Section 6, the model for predicting concept classifiers' generalizability from meta-features is shown to be effective and beneficial to improving the cost-efficiency of adaptation. This result should be received with an understanding of the empirical, application-specific nature of this generalizability model. Some of the meta-features, such as the ratio of support vectors, are unique to the SVM algorithm and have no counterpart in other classification algorithms. The relationship between generalizability and certain meta-features, e.g., higher ratio of positive data means better generalizability, is true perhaps only in semantic concept detection. Thus, the findings of our analysis and the use of our generalizability model are both restricted to the domain of concept detection. As a future direction, it is of great research interest to conduct a *general*, *domain-independent* analysis of generalizability, and to find out whether generalizability can be predicted without domain knowledge. The key question is whether we can find domain-independent clues that reveals a classifier's generalizability.

One of these clues must come from comparing the data distribution between the source and target domain. The similarity (or difference) between two distributions is more directly related to how well a classifier would generalize than the meta-features discussed in our analysis, such as the model complexity. However, measuring the distribution similarity is not easy, especially when feature spaces are of hundreds or even thousands of dimension. Due to the curse of dimensionality, every data point in a high-dimensional feature space is far away from probably any other data point, making the comparison of two sets of data points difficult. One solution is to assume the data in each domain are generated by a certain model such as Gaussian mixture model (GMM), whose parameters can be estimated from the data. Then the similarity between two domains can be measured by the distance between two GMM models, using metrics like KL divergence. Moreover, if the data in the two classes are very imbalanced, there is an even greater difficulty. In this case, any change to the distribution of the rare class has a significant impact on classification performance, but such change is not reflected in the similarity of the overall distribution because the rare class is overwhelmed by

the dominant class. Meanwhile, the distribution of the rare class cannot be estimated because the target domain is unlabeled. Overall, this direction deserves more research and may be very useful for predicting generalizability.

Bibliography

[1] Corel stock photo images. Corel Corporation. http://www.corel.com.

[2] LSCOM lexicon definitions and annotations version 1.0. In *DTO Challenge Workshop on Large Scale Concept Ontology for Multimedia, Columbia University ADVENT Technical Report 217-2006-3*, 2006.

[3] M. C. R. C. J. G. Q. J. W. H. L. J. Y. P. S. M. S. R. Y. J. Y. A. G. Hauptmann, R. Baron and Y. Zhang. CMU informedia's TRECVID 2005 skirmishes. In *TREC Video Retrieval Evaluation Proceedings*, 2005.

[4] A. Amir, M. Berg, S. Chang, W. Hsu, G. Iyengar, C. Lin, M. Naphade, A. Natsev, C. Neti, H. Nock, and et al. IBM research TRECVID-2003 video retrieval system. In *Worshop of TRECVID 2003*, 2003.

[5] Y. Amit, M. Fink, N. Srebro, and S. Ullman. Uncovering shared structures in multiclass classification. In *Proc. of the 24th Int'l Conf. on Machine learning*, pages 17–24, 2007.

[6] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *J. Mach. Learn. Res.*, 6:1817–1853, 2005.

[7] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. *Advances in Neural Information Processing Systems*, 2007.

[8] S. Ayache and G. Quenot. TRECVID 2007 collaborative annotation using active learning. In *Workshop of TRECVID 2007*.

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

[10] B. Bakker and T. Heskes. Task clustering and gating for bayesian multitask learning. *J. Mach. Learn. Res.*, 4:83–99, 2003.

[11] K. Barnard, P. Duygulu, D. Forsyth, N. de Freitas, D. Blei, and M. Jordan. Matching words and pictures. *Journal of Machine Learning Research*, 3, 2002.

[12] S. Ben-David and R. Schuller. Exploiting task relatedness for multiple task learning. In *Proc. of Computational Learning Theory*, 2003.

[13] S. Bickel, M. Brückner, and T. Scheffer. Discriminative learning for differing training and test distributions. In *Proc. of the 24th Int'l Conf. on Machine Learning*, pages 81–88, 2007.

[14] S. Bickel and T. Scheffer. Dirichlet-enhanced spam filtering based on biased samples. *Advances in neural information processing systems*, 19, 2007.

[15] D. M. Blei and M. I. Jordan. Modeling annotated data. In *Proc. of 26th Annual Int'l ACM SIGIR Conf. on Research and development in informaion retrieval*, pages 127–134, 2003.

[16] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. volume 3, pages 993–1022, 2003.

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

[18] C. Campbell, N. Cristianini, and A. Smola. Query learning with large margin classifiers. In *Proc. of Int'l Conf. on Machine Learning*, pages 111–118, 2000.

[19] M. Campbell, A. Haubold, S. Ebadollahi, M. Naphade, A. Natsev, J. Smith, J. Tesic, and L. Xie. IBM Research TRECVID-2006 Video Retrieval System. In *TREC Video Retrieval Evaluation Proceedings*, 2006.

[20] R. Caruana. Multitask learning. *Mach. Learn.*, 28(1):41–75, 1997.

[21] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Proc. of Neural Information Processing Systems*, pages 409–415, 2000.

[22] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001.

[23] S. Chang, W. Jiang, A. Yanagawa, and E. Zavesky. Columbia University TRECVID 2007 High-Level Feature Extraction. In *TREC Video Retrieval Evaluation Proceedings*, 2007.

[24] S. F. Chang, W. Hsu, L. Kennedy, L. Xie, A. Yanagawa, E. Zavesky, and D. Zhang. Columbia University TRECVID-2005 Video Search and High-Level Feature Extraction. In *TREC Video Retrieval Evaluation Proceedings*, 2005.

[25] O. Chapelle, P. Haffner, and V. Vapnik. Support vector machines for histogram-based image classification. *IEEE Trans. on Neural Networks*, 10(5):1055–1064, 1999.

[26] O. Chapelle, B. Scholkopf, and A. Zien. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[27] C. Chelba and A. Acero. Adaptation of maximum entropy capitalizer: Little data can help a lot. *Computer Speech & Language*, 20(4):382–399, 2006.

[28] M.-Y. Chen and A. Hauptmann. Discriminative fields for modeling semantic concepts in video. In *Eighth Conference on Large-Scale Semantic Access to Content*, 2007.

[29] I. Cox, M. Miller, S. Omohundro, and P. Yianilos. Pichunter: Bayesian relevance feedback for image retrieval. *Proc. of the 13th Int'l Conf. on Pattern Recognition*, 3, 1996.

[30] T. Cox and M. Cox. *Multidimensional Scaling*. Chapman and Hall, 2001.

[31] 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.

[32] 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.

[33] S. B. David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. *Advances in Neural Information Processing Systems*, 2007.

[34] 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.

[35] 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.

[36] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. *Advances in Neural Information Processing Systems*, pages 155–161, 1996.

[37] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. pages 148–155, 1998.

[38] T. Evgeniou, C. A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *J. Mach. Learn. Res.*, 6:615–637, 2005.

[39] T. Evgeniou and M. Pontil. Regularized multi–task learning. In *Proc. of the 10th ACM SIGKDD Int'l Conf. on Knowledge discovery and data mining*, pages 109–117, 2004.

[40] W. Fan. Systematic data selection to mine concept-drifting data streams. In *Proc. of 10th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, pages 128–137, 2004.

[41] W. Fan and I. Davidson. On sample selection bias and its efficient correction via model averaging and unlabeled examples. In *Proc. of SIAM Data Mining Conference 2007*, 2007.

[42] L. Fei-Fei, R. Fergus, and P. Perona. A bayesian approach to unsupervised one-shot learning of object categories. In *Proc. of the Ninth IEEE Int'l Conf. on Computer Vision*, page 1134, 2003.

[43] S. L. Feng, R. Manmatha, and V. Lavrenko. Multiple bernoulli relevance models for image and video annotation. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 1002–1009, 2004.

[44] M. Gales and S. Young. Robust continuous speech recognition using parallel modelcombination. *IEEE Trans. on Speech and Audio Processing*, 4(5):352–359, 1996.

[45] 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.

[46] D. Gildea. Corpus variation and parser performance. In *Conf. on Empirical Methods in Natural Language Processing*, pages 167–202, 2001.

[47] Z. Guo, Z. Zhang, E. P. Xing, and C. Faloutsos. Enhanced max margin learning on multimodal data mining in a multimedia database. In *Proc. of 13th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, 2007.

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

[49] A. Hauptmann and et al. Informedia at TRECVID 2003: Analyzing and searching broadcast news video. In *Workshop of TRECVID*, 2003.

[50] A. Hauptmann, W. Lin, R. Yan, J. Yang, R. Baron, M. Chen, S. Gilroy, and M. Gordon. Exploring the Synergy of Humans and Machines in Extreme Video Retrieval. In *Int'l Conf. on Image and Video Retrieval (CIVR)*, pages 514–522, 2006.

[51] G. Heitz, G. Elidan, and D. Koller. Transfer learning of object classes: From cartoons to photographs. In *NIPS 2005 Workshop on Inductive Transfer: 10 Years Later*, 2005.

[52] J. Huang, A. Smola, A. Gretton, K. Borgwardt, and B. Scholkopf. Correcting sample selection bias by unlabeled data. *Advances in neural information processing systems*, 19, 2007.

[53] R. Hwa. Supervised grammar induction using training data with limited constituent information. In *Proc. of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 73–79, 1999.

[54] J. Jeon, V. Lavrenko, and R. Manmatha. Automatic image annotation and retrieval using cross-media relevance models. In *Proc. of the 26nd Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 119–126, 2003.

[55] W. Jiang, E. Zavesky, S. Chang, and A. Loui. Cross-domain Learning Methods for High-level Visual Concept Classification. In *IEEE International Conference on Image Processing*, 2008.

[56] T. Joachims. Making large-scale support vector machine learning practical. In *Advances in Kernel Methods: Support Vector Machines*. 1998.

[57] T. Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *Proc. of the 10th European Conf. on Machine Learning*, pages 137–142. Springer-Verlag, 1998.

[58] T. Joachims. Transductive Inference for Text Classification using Support Vector Machines. In *Proc. of 16th Int'l Conf. on Machine Learning*, pages 200–209, 1999.

[59] S. S. Keerthi, K. B. Duan, S. K. Shevade, and A. N. Poo. A fast dual algorithm for kernel logistic regression. *Machine Learning*, 61(1-3):151–165, 2005.

[60] L. S. Kennedy, A. P. Natsev, and S.-F. Chang. Automatic discovery of query-class-dependent models for multimodal search. In *Proc. of the 13th annual ACM Int'l Conf. on Multimedia*, pages 882–891, 2005.

[61] 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.

[62] J. Z. Kolter and M. A. Maloof. Dynamic weighted majority: A new ensemble method for tracking concept drift. In *Proc. of Int'l Conf. on Data Mining*, page 123, 2003.

[63] N. D. Lawrence and J. C. Platt. Learning to learn with the informative vector machine. In *Proc. of Int'l Conf. on Machine Learning*, 2004.

[64] C. Leggetter and P. Woodland. Flexible Speaker Adaptation Using Maximum Likelihood Linear Regression. In *Proc. ARPA Spoken Language Technology Workshop*, pages 104–109, 1995.

[65] 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.

[66] 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, 2001.

[67] 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.

[68] 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.

[69] M. R. Naphade, T. Kristjansson, B. Frey, and T. Huang. Probabilistic multimedia objects Multijects: A novel approach to video indexing and retrieval in multimedia systems. In *Proc. of ICIP*, 1998.

[70] 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.

[71] 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.

[72] C. Ngo, Y. Jiang, X. Wei, F. Wang, W. Zhao, H. Tan, and X. Wu. Experimenting VIREO-374: Bag-of-Visual-Words and Visual-Based Ontology for Semantic

Video Indexing and Search. In *TREC Video Retrieval Evaluation Proceedings*, 2007.

[73] A. Niculescu-Mizil and R. Caruana. Learning the Structure of Related Tasks. In *Proc. of NIPS-2005 Workshop on Inductive Transfer*, volume 10, 2005.

[74] C. Petersohn. Fraunhofer HHI at TRECVID 2004: Shot Boundary Detection System. In *TREC Video Retrieval Evaluation Online Proceedings*, 2004.

[75] J. Philbin, O. Chum, J. Sivic, V. Ferrari, M. Marin, A. Bosch, N. Apostolof, and A. Zisserman. Oxford TRECVid 2007–Notebook paper. In *TREC Video Retrieval Evaluation Proceedings*, 2007.

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

[77] G.-J. Qi, X.-S. Hua, Y. Rui, J. Tang, T. Mei, and H.-J. Zhang. Correlative multi-label video annotation. In *Proc. of the 15th ACM Int'l Conf. on Multimedia*, pages 17–26, 2007.

[78] G. Quenot and S. Ayache. TRECVID 2007 collaborative annotation. In *http://mrim.imag.fr/tvca/.*

[79] 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.

[80] 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.

[81] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning.* The MIT Press, 2005.

[82] B. Roark and M. Bacchiani. Supervised and unsupervised pcfg adaptation to novel domains. In *Proc. of the 2003 Conf. of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 126–133, 2003.

[83] M. Rosenstein, Z. Marx, L. Kaelbling, and T. Dietterich. To transfer or not to transfer. In *NIPS 2005 Workshop on Inductive Transfer: 10 Years Later*, 2005.

[84] Y. Rui, T. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: a power tool for interactive content-basedimage retrieval. *IEEE Trans. on Circuits and Systems for Video Technology*, 8(5):644–655, 1998.

[85] S. Satoh and T. Kanade. Name-it: Association of face and name in video. In *Proc. of the Conf. on Computer Vision and Pattern Recognition*, pages 368–373. IEEE Computer Society, 1997.

[86] R. E. Schapire, M. Rochery, M. G. Rahim, and N. Gupta. Incorporating prior knowledge into boosting. In *Proc. of the 9th Int'l Conf. on Machine Learning*, pages 538–545, 2002.

[87] D. Shen, J. Zhang, G. Zhou, J. Su, and C.-L. Tan. Effective adaptation of a hidden markov model-based named entity recognizer for biomedical domain. In *Proc. of the ACL 2003 workshop on Natural language processing in biomedicine*, pages 49–56, 2003.

[88] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.

[89] A. Smeaton and P. Over. Trecvid: Benchmarking the effectiveness of infomration retrieval tasks on digital video. In *Proc. of Conf. on Image and Video Retrieval*, 2003.

[90] C. Snoek, I. Everts, J. van Gemert, J. Geusebroek, B. Huurnink, D. Koelma, M. van Liempt, O. de Rooij, K. van de Sande, and A. Smeulders. The MediaMill TRECVID 2007 Semantic Video Search Engine. In *TREC Video Retrieval Evaluation Proceedings*, 2007.

[91] C. Snoek, M.Worring, J. Geusebroek, D. Koelma, and F. Seinstra. The mediamill trecvid 2004 semantic viedo search engine. In *Proc. of TRECVID*, 2004.

[92] W. N. Street and Y. Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In *Proc. of ACM SIGKDD Int'l Conf. on Knowledge discovery and data mining*, pages 377–382, 2001.

[93] M. Sugiyama and K. Muller. Model selection under covariate shift. Springer, 2005.

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

[95] M. Szummer and R. Picard. Indoor-outdoor image classification. In *IEEE Int'l Workshop on Content-based Access of Image and Video Databases, in conjunction with ICCV*, volume 98, pages 42–51, 1998.

[96] M. Taylor and P. Stone. Cross-domain transfer for reinforcement learning. *Proc. of the 24th Int'l conference on Machine learning*, pages 879–886, 2007.

[97] 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.

[98] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In *Proc. of Int'l Conf. on Machine Learning*, pages 999–1006, 2000.

[99] L. Uebel and P. Woodland. An Investigation into Vocal Tract Length Normalisation. In *Proc. of Eurospeech-99*, 1999.

[100] A. Ulges, M. Koch, C. Schulze, and T. Breuel. Learning TRECVID 08 High-Level Features from YouTube TM.

[101] A. Vailaya, M. Figueiredo, and A. Jain. Image classification for content-based indexing. *IEEE Trans. on Image Processing*, 10(1):117–130, 2001.

[102] H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proc. of ACM SIGKDD Int'l Conf. on Knowledge discovery and data mining*, pages 226–235, 2003.

[103] M. Welling, M. Rosen-Zvi, and b. . A. p. . . y. . . Geoffrey Hinton, title = Exponential Family Harmoniums with an Application to Information Retrieval.

[104] T. Westerveld, T. Ianeva, L. Boldareva, A. de Vries, and D. Hiemstra. Combining information sources for video retrieval. In *TRECVID 2003 Workshop*, 2004.

[105] 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.

[106] 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.

[107] L. Xiao, D. Wenyuan, X. Gui-Rong, Y. Qiang, and Y. Yong. Spectral domain-transfer learning. In *Proc. of the 14th ACM SIGKDD Int'l Conf. on Knowledge discovery and data mining (KDD)*, pages 488–496, 2008.

[108] E. Xing, R. Yan, and A. Hauptmann. Mining associated text and images with dual-wing harmoniums. In *Proc. of the 21th Annual Conf. on Uncertainty in Artificial Intelligence (UAI-05)*, 2005.

[109] B. L. T. Y. Wu and J. R. Smith. Ontology-based multiclassification learning for video concept detection. In *IEEE Int'l Conference on Multimedia and Expo*, 2004.

[110] R. Yan. Probabilistic models for combining diverse knowledge sources in multimedia retrieval. In *Ph.D Thesis*, 2006.

[111] R. Yan, J. Yang, and A. G. Hauptmann. Learning query-class dependent weights in automatic video retrieval. In *Proc. of the 12th ACM Int'l Conf. on Multimedia*, pages 548–555, 2004.

[112] R. Yan, M. yu Chen, and A. G. Hauptmann. Mining relationship between video concepts using probabilistic graphical model. In *IEEE Int'l Conf. on Multimedia and Expo*, 2006.

[113] J. Yang and A. Hauptmann. Annotating news video with locations. In *Proc. of 5rd Int'l Conf. on Image and Video Retrieval*, pages 153–162, 2006.

[114] J. Yang and A. G. Hauptmann. Naming every individual in news video monologues. In *Proc. of the 12th annual ACM Int'l Conf. on Multimedia*, pages 580–587, 2004.

[115] Y. Yang and J. Pedersen. A comparative study on feature selection in text categorization. In *14th Int'l Conf. on Machine Learning*, pages 412–420, 1997.

[116] K. Yu, V. Tresp, and A. Schwaighofer. Learning gaussian processes from multiple tasks. In *Proc. of the 22nd Int'l conference on Machine learning*, pages 1012–1019, 2005.

[117] J. Yuan, Z. Guo, L. Lv, W. Wan, T. Zhang, D. Wang, X. Liu, C. Liu, S. Zhu, and D. Wang. THU and ICRC at TRECVID 2007. In *TREC Video Retrieval Evaluation Proceedings*, 2007.

[118] B. Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Proc. of the 21st Int'l Conf. on Machine learning*, page 114, 2004.

[119] J. Zhang, Z. Ghahramani, and Y. Yang. Learning multiple related tasks using latent independent component analysis. In *Advances in Neural Information Processing Systems 18*, pages 1585–1592, 2006.

[120] Y. Zhang. Using bayesian priors to combine classifiers for adaptive filtering. In *Proc. of the 27th Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 345–352, 2004.