APPROVAL SHEET

Title of Thesis: EFFICIENT RECOVERY FROM REPEATED DOMAIN SHIFTS IN STREAMING DATA

Name of Candidate: Richa Gandhewar
MS Computer Science, 2016

Thesis and Abstract Approved: ________________________
Dr. Tim Oates
Professor
Department of Computer Science and
Electrical Engineering

Date Approved: ______________________________

ABSTRACT

Title of Thesis: EFFICIENT RECOVERY FROM REPEATED DOMAIN SHIFTS IN STREAMING DATA

Richa Gandhewar, MS Computer Science, 2016

Thesis directed by: Dr. Tim Oates, Professor
Department of Computer Science and Electrical Engineering

Humans have a remarkable ability to learn how to learn, what to learn, and when to learn. We are able to assess the utility of learned knowledge to achieve an objective and adapt our learning strategies accordingly. Likewise, we want machine learning systems trained in one domain to adapt well to different domains. If a classifier system encounters a distribution which it has seen previously, it should remember the previously learned knowledge and classify accordingly. This thesis addresses the problem of recovering efficiently from repeated domain shifts in streaming data for a classifier system.

This problem can be divided into two sub-problems. The first sub-problem is detecting a domain shift in a data stream representing learned knowledge. Like (Dredze, Oates, & Piatko 2010), we also use the $\mathcal{A}$-distance (Kifer, Ben-David, & Gehrke 2004) over the absolute value of classification margin of support vector machines for this task. The second sub-problem is deciding what action to take after a domain shift is detected. We propose and evaluate approaches to training new models and deciding when to reuse old models to minimize cost and maximize accuracy in the face of repeated domain shifts. We use the Amazon product reviews dataset for evaluating our algorithm.

EFFICIENT RECOVERY FROM REPEATED DOMAIN SHIFTS IN STREAMING DATA

by

Richa Gandhewar

Thesis submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Master of Science
2016

Dedicated

To my father, Rajendra Gandhewar, who has put in tremendous efforts to make me what I am today

&

To my mother, Anushree Gandhewar, for all the love and being a constant support to me and my family

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to the University of Maryland, Baltimore County for letting me fulfill my dream of pursuing Masters. I would also like to thank the Computer Science Department for giving me an opportunity to write an honors thesis. I am extremely grateful to my adviser and mentor, Dr. Tim Oates, for his guidance, patience, understanding and encouragement. I would like to thank Dr. Tim Finin and Dr. Charles Nicholas for agreeing to be on my committee.

I would like to thank my friends Girish Kotecha, Vaishali Narkhede, Neha Tilak, Sunil Gandhi, Aishwarya Bhide, Mandar Haldekar and Ajinkya Borale for helping me survive all the stress and making my Masters experience worthwhile. I would also like to thank all my friends back home for encouraging me. Lastly, I am extremely thankful to my family for always supporting me and believing in me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Motivation

Consider a named entity recognition (NER) system trained on news articles. Semantic analyzers build ontologies using such systems by extracting entities from text documents. The objective is to learn a model that would enable the detection of entities like “Tesla” and “Apple” given a sentence like “Tesla is not the next Apple”. This model tends to work well as long as the data is coming from a single data distribution, like new articles. However, we cannot be confident that the system will perform as expected when sentences like “JOBS MYSTERY, KRUGMAN BLAMES APPLE ...” are given as some of the features used by the NER system are no longer available. For instance, capitalization patterns are not an indicator of the presence of a named entity, and parts of speech are not easily distinguishable. This problem of domain shift is a pervasive problem in NLP in which any kind of model - a parser, POS tagger, sentiment classifier - is tested on data that do not match training data.

Another interesting instance of domain shift can be seen in sentiment classifiers of e-commerce applications. A classifier trained on book reviews learns a model which may heavily weigh features like “page-turner”, “enchanting” and “esoteric” to make confident predictions about the sentiment in a review. In real-time however, some home appliance reviews may get interspersed with the data given to the analyzer. The features relevant for correct classification on this data like “power-efficient”, “water-resistant” and “unreliable” might be missed as they will not be in the classification model at all. To say the least, it is completely possible for the learned knowledge (e.g. weights in SVMs) to become less useful over time resulting in more errors.

The first step in addressing this problem is to recognize there is indeed a problem. Most learning systems are trained in the lab by human experts and then deployed in the wild. The system itself does not have the ability to detect when the use of the knowledge it has learned through training is growing less effective over time. This problem can be treated as detecting changes in data streams of real numbers, where the streams contain information about learned knowledge. Given a classification model and a stream of instances, we want to automatically detect changes in feature distribution that negatively affect classification accuracy. We would also want a new model, which exploits the features of the new distribution and improves accuracy on future instances. If we encounter previously seen distributions during classification, we want the classifier system to use the correct pre-trained model and thus avoid the cost of retraining.

1.2 Thesis Statement

This thesis investigates the following claim:

“Accuracy of a classifier system while classifying instances in streaming data can be improved by detecting a domain shift if it exists and adapting the classifier to the new domain while minimizing retraining cost by reusing previously trained models when applicable.”

This thesis proposes a method for the classifier system to adapt to a new domain of instances. The classifier system adapts to the new domain if it detects a domain change in a stream of instances for which it computes output. Domain changes can be detected by measuring changes in the absolute value of the classification margin of support vector machines as the classification margin incorporates information about the domain of the instances. Our thesis problem can be divided into two sub problems: detecting a domain change from a stream of real numbers containing domain information, and deciding whether to retrain a model, use a previously trained model or use the current model for classification of future instances when a domain shift is detected. We use the $\mathcal{A}$-distance metric (Kifer, Ben-David, & Gehrke 2004) to solve the first sub-problem as it has been previously used in domain adaptation work (Blitzer *et al.* 2007). Our main contribution is towards the second sub-problem, deciding what to do after a domain shift is detected. We analyze the three approaches stated above that can be followed after detecting a domain shift. Our experiments include evaluations on the Amazon product reviews dataset (McAuley 2014).

To summarize, the following are our contributions:

1. A method to decide if data samples should be labeled to create a new model, or reuse a previously trained model, if a domain shift is detected on a stream of data instances being classified by support vector machines.

2. Analysis of improvement in classification accuracy on the Amazon product reviews dataset when our model is used.

3. Analysis of the cost incurred for retraining vs the cost incurred due to incorrectly classified samples.

1.3 Thesis Overview

Chapter 2 provides a basic background of concepts and algorithms used in this thesis. Chapter 3 consists of a survey of previous work related to the tasks described in this thesis. We briefly summarize the contributions made in the area of domain adaptation. Chapter 4 begins with an overview of the our general approach to the problem of domain adaptation. It follows with a detailed explanation of the algorithms and tools used. Chapter 5 starts with a description of the dataset used for our experiments. It follows with the experimental evaluation of our domain adaptation algorithm. It also shows that our algorithm has good efficiency in the long run. Finally, chapter 6 lists the conclusions drawn from this work and discusses future scope in this area.

Chapter 2

BACKGROUND

This chapter provides an overview of the concepts in machine learning that we used in this thesis.

2.1 Supervised Learning

Supervised learning is the machine learning task of inferring a function from labeled training data. In supervised learning, each training example is a pair consisting of an input object (feature vector) and a desired output value (class label). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new unlabelled examples to their correct class label. An optimal scenario will allow for the algorithm to correctly determine the class labels for instances not seen in the training data.

2.2 Support Vector Machines (SVM)

Support Vector Machines are supervised learning models for classification of data samples. Given a set of training samples labeled as either class A or B, an SVM training algorithm builds a model that assigns new examples into one of the two categories. The core SVM algorithm has also been generalized to multi-class problems.

An SVM model represents the data samples as points in a multi-dimensional space, defined by the kernel. The mapping is done such that the samples of the separate classes are divided by a clear gap that is as wide as possible. The learning algorithm finds a hyperplane that maximizes this gap. New samples are then mapped into that same space and predicted to belong to a class based on which side of the gap they fall. In addition to performing linear classification, SVMs can efficiently perform non-linear classification using the kernel trick, i.e., mapping their inputs into high-dimensional feature spaces.

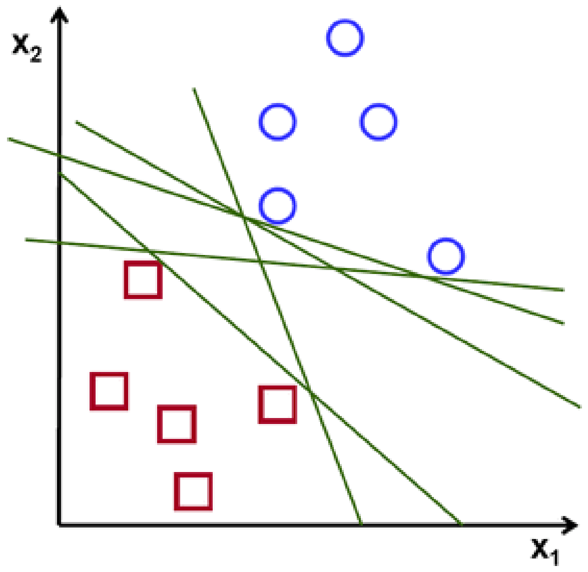

FIG. 2.1: Multiple separating lines for classification (OpenCV)

Fig. 2.1 represents samples from two classes (circle and square) as points in a 2D plane. Multiple lines are drawn on the plane that are able to separate the two classes. A separating line which passes too close to the data points is bad because it will be sensitive to noise and will not generalize well. We want the separating line to be as far from the training samples as possible. This is called the optimal hyperplane shown in Fig. 2.2. The margin of a classifier is defined as the maximum width that a hyperplane can grow to before hitting a data point. The data points the margin pushes up against are called support vectors. In Fig. 2.2 the support vectors are shown by filled squares or circles.

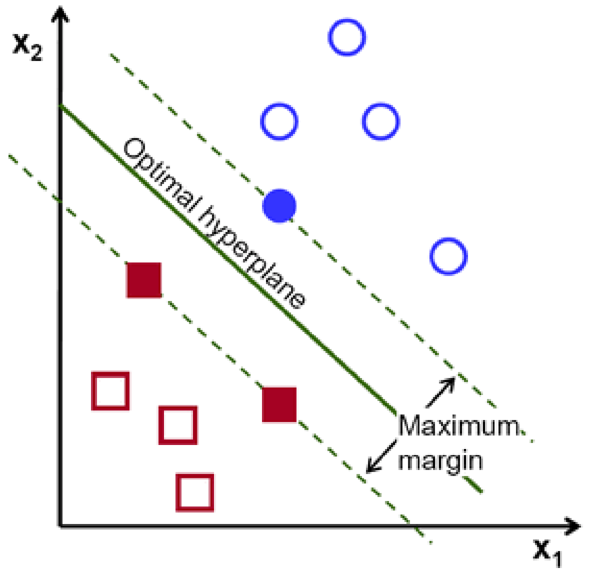

FIG. 2.2: Optimal separating hyperplane in SVM (OpenCV)

Some advantages of support vector machines are as follows. SVMs use a subset of the training points called support vectors in the decision function, which makes it memory efficient. It is versatile due to kernel functions which define the feature space and thus the shape of the hyperplane. It is effective in high dimensional spaces due to the kernel trick.

2.3 Sentiment Classification as a task

Text classification generally involves categorizing based on topic or subject matter of a text sample. Sentiment classification (Pang, Lee, & Vaithyanathan 2002) can be considered a special type of text classification where the objective is to classify samples according to the sentimental polarities they contain, i.e, overall opinion towards the subject. It determines the attitude of a person with respect to a certain topic. Sentiment analysis is widely applied to reviews and social media. Consumers can check reviews for the products or services they wish to purchase. Companies can monitor their social media accounts to check public opinion on their products or services. They can also analyze customer satisfaction depending on their comments. Organizations can take business decisions based on user reviews for their products.

A certain product, e.g., a laptop, might have a review like “Great Laptop! Excellent display and light weight” which is a positive review, or “Poor quality and very low battery life” which is a negative review. A sentiment classifier must correctly label the review as positive or negative.

Machine learning algorithms can be applied to data samples to determine the positive or negative sentiment they imply. First, the text is modeled as a bag-of-words, i.e., a set of content words. We could assume that the sentimental orientation of the whole text depends on the sum of polarities of content words. But this may not be always true since many words can shift the polarities of the text. For example, in the sentence “The laptop was not worth the money”, the polarity of the word “worth” is positive but the polarity of the whole sentence is reversed because of the negation word “not”. Thus, for sentiment classification, it is better to consider the bag-of-words as bigrams which considers “not worth” as a single feature, instead of the unigram model which considers every word as a separate feature.

2.4 Domain Adaptation

As we saw in the previous section, sentiment classification is a widely used task. Let us consider a classifier that is trained on electronics review data. It will be able to determine for a new electronics review sample whether it is positive or negative with good accuracy. What happens if the classifier tries to determine the polarity of a book review? It might not be able to classify the review correctly. In the electronics domain, words like “compact” and “power-efficient” convey positive sentiment while words like “blurry” convey negative sentiment. But in the books domain, words like “informative” and “engaging” express positive opinion and words like “boring” expresses negative opinion. If the domains on which the classifier is trained and tested are different, the classifier’s accuracy might reduce significantly.

We want classifiers to perform well and with high accuracy even if the domain shifts. The domain on which the classifier is trained is called the source domain and the domain on which it is tested is called the target domain. Domain adaptation is the ability of a classifier system trained in a source domain to perform well in a target domain.

Different types of domain adaptation are:

1. Unsupervised domain adaptation: The training sample contains a set of labeled source examples, a set of unlabeled source examples, and an unlabeled set of target examples.

2. Semi-supervised domain adaptation: Here, we also consider a small set of labeled target examples along with a set of labeled source examples, a set of unlabeled source examples and an unlabeled set of target examples.

3. Supervised domain adaptation: In this situation, all the examples considered are supposed to be labeled.

Our thesis concentrates on supervised domain adaptation. In this setting, we also have a small budget for acquiring some labels in the target domain. We use human generated class labels for a small number of target examples for our experiments.

2.5 The $\mathcal{A}$-Distance

For a classifier system to adapt to a new domain, it should first detect that there is indeed a domain shift. We can detect a domain shift for a support vector machine classifier by detecting a change in the distribution of the absolute value of the classification margin. For detecting this change in distribution, we use the $\mathcal{A}$-distance (Kifer, Ben-David, & Gehrke 2004) metric.

The $\mathcal{A}$-distance detects differences between two arbitrary probability distributions by dividing the range of a random variable into a set of (possibly overlapping) intervals, and then measures changes in the probability that a value drawn for that variable falls into any one of the intervals. If such a change is large, a change in the underlying distribution is declared. Let $\mathcal{A}$ be a set of intervals and let A $\in \mathcal{A}$ be one such interval. For that interval, P(A) is the probability that a value drawn from some unknown distribution falls in A. The $\mathcal{A}$-distance between P and P', i.e., the difference between two distributions over the intervals, is defined as follows:

$$d_{\mathcal{A}} \text{ (P, P')} = 2\ sup_{A \in \mathcal{A}} \mid P(A) - P'(A) \mid$$

Two distributions are said to be different when for a user-specified threshold ϵ, $d_{\mathcal{A}}$(P,P')$> \epsilon$. It means that the $\mathcal{A}$-distance is the largest change in probability of a set that the user cares about. The $\mathcal{A}$-distance is distribution independent. That is, it makes no assumptions about the form of the underlying distribution nor about the form of the change that might occur, either algorithmically or in the underlying theory. The $\mathcal{A}$-distance can be shown to require finitely many samples to detect distribution differences, a property that is crucial for streaming, sample-based approaches.

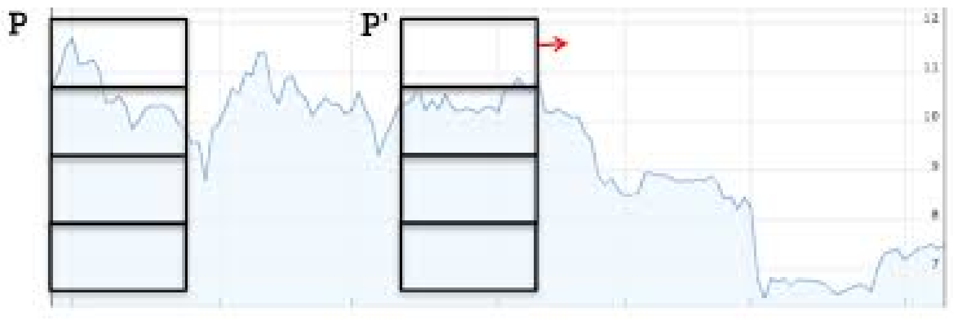

FIG. 2.3: $\mathcal{A}$-distance between two windows

Since the $\mathcal{A}$-distance processes a stream of real numbers, we need to represent the activity of a learner using a real number. We can use the classification margin in support vector machines for a test example or activation values of nodes in a neural network. The first n of these numbers in the stream are a sample from P, and the most recent n are a sample from P'. We signal a domain shift when the $\mathcal{A}$-distance between P and P' is greater than ϵ. Larger values of n result in more accurate estimates of P(A) and slower detection of changes.

The two windows of samples of size n are shown graphically in Fig. 2.3. Each increment on the horizontal axis represents the arrival of a new test example. The vertical axis is some value computed from each example, such as its classification margin. To compute P and P', one needs to specify $\mathcal{A}$ and n, which are shown as two stacks of boxes that are identical except for their position. The width of each box is n, the number of examples used to estimate P(A) and P'(A) for A $\in \mathcal{A}$, where the real interval A corresponds to the vertical span of the box. The value P(A) is simply the number of examples whose real value falls inside interval A divided by n. Note that the first n examples in the stream are used to compute P, and as each new example arrives the location of the stack of boxes used to compute P' is shifted to the right by one.

In Fig. 2.3, the number of examples whose real value falls in the top two intervals for P is approximately the same, with no example's value falling in the lower two intervals. For P', almost every one of the n example values falls in the second interval from the top, virtually assuring that $d_{\mathcal{A}}$ (P, P') will be large. Though the intervals in the figure do not overlap, they typically do.

Given n and intervals A, the value of ϵ is chosen by randomization testing. Because the $\mathcal{A}$-distance is distribution independent, a sample of size m $\gg$ n is drawn from any distribution that spans A. This sample is treated as a stream as described above, and the largest value of $d_{\mathcal{A}}$ (P, P') is stored. The sample is permuted and this process is repeated *l* times. Note that any change detection would be a false positive because all values were sampled from the same distribution. The values $d_{\mathcal{A}}$ (P, P') are sorted from largest to smallest, and ϵ is chosen to be the $[\alpha l]^{th}$ value where parameter α is a user specified false positive probability.

Both the time and space complexity of our approach based on the $\mathcal{A}$-distance are small. Given n and $\mathcal{A}$, n instances must be stored in the sliding window and $2|\ \mathcal{A}\ |$ counters are required to represent P and P'. Note that both values are constants based on user specified parameters, not on the size of the stream. Processing a new instance involves computing its margin and updating P and P', all of which can occur in constant time.

Chapter 3

RELATED WORK

In this chapter, we briefly discuss about previous works that are relevant to the topic of this thesis.

3.1 Domain shift

Discriminative learning methods for classification perform well when training and test data are drawn from the same distribution. In many situations, though, we have labeled training data for a source domain, and we wish to learn a classifier that performs well on a target domain with a different distribution. For instance, while the vast majority of object recognition methods today are trained and evaluated on the same image distribution, real world applications often present changing visual domains. In general, visual domains could differ in some combination of factors, including scene, intra-category variation, object location and pose, view angle, resolution, motion blur, scene illumination, background clutter, camera characteristics, etc. Recent studies have demonstrated a significant degradation in the performance of state-of-the-art image classifiers due to domain shift from pose changes (Farhadi & Tabrizi 2008) or a shift from commercial to consumer video (Duan *et al.* 2009). This makes it very important to be able to detect a domain shift so that certain steps can be taken to improve the performance of the classifiers.

Recent work by (Dredze, Oates, & Piatko 2010) has shown that changes in the distribution of margin values produced for test instances by an SVM are correlated with changes in classification accuracy. They approach this problem of domain shift detection by representing a stream of examples as a stream of real numbers informative for distribution change detection. The stream of real numbers here is the absolute value of classification margin values produced for test instances by an SVM. They detect a change in the distribution of input samples by using the $\mathcal{A}$-distance metric which detects differences between two arbitrary probability distributions.

3.2 Domain Adaptation for Sentiment Classification

Previous work has been done on the problem of recovering from changes in either the input distribution or the labeling function in the case of supervised learning, known as domain adaptation. There are multiple versions of this problem depending on the availability of labeled target data. (Blitzer, McDonald, & Pereira 2006) use only unlabeled target domain data in their work (unsupervised domain adaptation). They proposed structural correspondence learning (SCL) algorithms to learn the common feature representation across domains based on some heuristic selection of pivot features. Another version is adapting using a limited amount of target domain labeled data (supervised domain adaptation) as done by (Daumé III 2009) who designed a heuristic kernel to augment features for solving some specific domain adaptation problems in NLP. (Dredze & Crammer 2008) developed a framework for learning across multiple domains simultaneously in an online setting.

Sentiment classification aims to predict the sentiment polarity of text data, e.g., text sentences and review articles. It has drawn much research attention recently. Many machine learning techniques have been proposed for sentiment classification, such as unsupervised learning techniques (Turney 2002), supervised learning techniques (Pang, Lee, & Vaithyanathan 2002), and semi-supervised learning techniques (Sindhwani & Melville 2008). However, most sentiment classifiers are domain dependent. It is challenging to adapt a classifier trained on one domain to another domain.

To address this problem, (Blitzer *et al.* 2007) proposed the SCL algorithm to exploit domain adaptation techniques for sentiment classification. SCL is motivated by a multi-task learning algorithm, called alternating structural optimization (ASO). SCL tries to construct a set of related tasks to model the relationship between "pivot features" and "non-pivot features". Then "non-pivot features" with similar weights among tasks tend to be close with each other in a low-dimensional latent space. Although it is experimentally shown that SCL can reduce the difference between domains based on the $\mathcal{A}$-distance measure (Ben-David *et al.* 2007), the heuristic criterion of pivot feature selection may be sensitive to different applications.

(Pan *et al.* 2010) proposed a general framework for cross-domain sentiment classification. In this framework, they built a bipartite graph between domain-independent and domain-specific features and proposed a spectral feature alignment (SFA) algorithm to align the domain-specific words from the source and target domains into meaningful clusters, with the help of domain independent words as a bridge.

(Xia *et al.* 2013) consider both labeling adaptation (model changes of labeling function) and instance adaptation (model the change of instance probability) in their feature ensemble plus sample selection approach. (Tan *et al.* 2009) pick out only the generalizable features and use these features as a bridge linking an old domain to a new domain. The main idea was to employ a weighted Expectation Maximization algorithm to combine the old-domain data with the new-domain data, and gradually enlarge the weight for the new-domain data while decreasing the weight for the old-domain data with each iteration, with the hope to fit the new-domain data as well as possible.

Delta idf weighting proposed by (Martineau & Finin 2009) is a supervised variant of idf weighting in which the idf calculation is done for each document class and then one value is subtracted from the other. They present evidence that this weighting helps with sentiment classification. (Chinavle *et al.* 2009) present a method based on an ensemble of classifiers that automatically, without human intervention, deals with determining when classifier performance has degraded and how the classifiers are retrained. They use mutual agreement between classifiers in the ensemble to detect possible changes in classifier accuracy. The mutual agreement between a pair of classifiers is the fraction of time they assign an instance the same class label. They use the output of the ensemble as a proxy for the true label for new instances and retrain individual classifiers identified as possibly weak via mutual agreement.

To the best of our knowledge, there has not been much work done with respect to a classifier recovering from repeated domain shifts. None of the above domain adaptation methods save the previous learned information which could be reused if there are instances from a previous domain again.

Chapter 4

METHODOLOGY

4.1 Approach

Our problem of domain adaptation can be divided into two sub-problems.

1. Detecting domain shifts from a stream of data

2. Deciding whether to use the current model, retrain or use a previous model.

Domain shift detection requires samples to be represented as a stream of real numbers which contain information about distributional changes. We use support vector machines for building models. We use the absolute value of the classification margin as our data stream - a stream of real values. We show that classification margins become significantly lower when domain changes occurs. After training a model and deploying it, we label the unlabeled instances in batches of 1000 samples. As the model is labeling the instances, we use the $\mathcal{A}$-distance over classification margins to detect domain changes. The $\mathcal{A}$-distance detector detects a change if a certain threshold is reached. If a domain shift is detected, then one of the following is done.

1. If there is no other trained model - Get the correct class labels for the last 500 instances before the point of domain shift and the next 500 instances after the shift and retrain. Save this new model and new average classification margin.

2. If there exists a previously trained model - If the average classification margin for these samples on a previous model is within 20% of the average, then swap in that model for classification of future instances. Otherwise retrain the samples with correct class labels and save the new model and average classification margin.

Fig. 4.1 shows a high-level view of the system that we propose in this work.

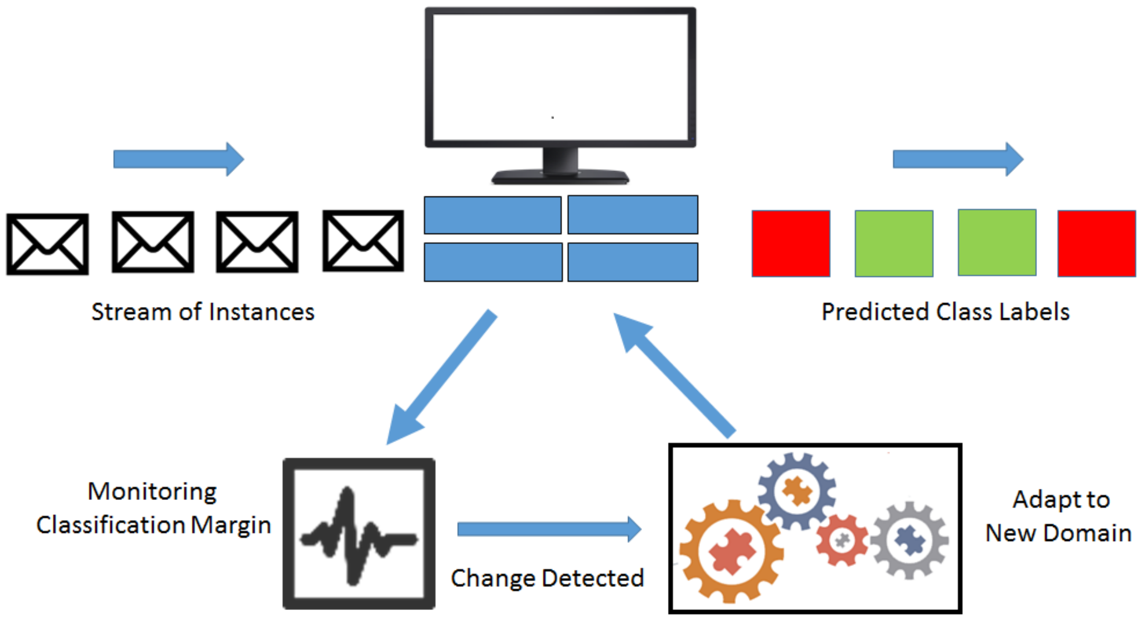

FIG. 4.1: Schematic diagram of the system

4.2 Training the classifier

The method for detecting a change in domain depends on the type of classifier used in the system. Different classifiers store knowledge gained from samples in different ways, for example classifier margin in SVMs or activation levels in neural networks. Changes in the distribution of margin values of SVM classifiers leads to changes in accuracy. This change can be detected using the $\mathcal{A}$-distance algorithm.

The domain of instances on which the support vector machine is trained is called the source domain or known domain. The domain of instances which is not seen by the support vector machine classifier during training is called the target domain or unknown domain. We train the support vector machine classifier in a supervised setting. In a supervised setting, both the text (input) and class label (output) are provided to the system.

4.3 Variation in Classification Margin

Classification margins in support vector machines contain information for detecting changes in the domain of instances that are being classified by the support vector machine classifier (Dredze, Oates, & Piatko 2010). The classifier margin is a function of the weights of features of test samples. As the important features disappear, the magnitude of the classification margin decreases. Also, the classifier margin remains uninfluenced by the features that do not affect performance of the classifier.

We show that SVM margin values are related to accuracy by analyzing changes in margin values and accuracy when the support vector machine classifier labels instances in known domains and unknown domains. We begin by examining visually the information content of the margin with regards to predicting a domain shift. We use a dataset containing 2000 samples of electronics, book and kitchen product reviews. We train the support vector machine classifier on 1500 instances from the source domain. We test the classifier on the remaining 500 instances from the source domain followed by 2000 instances from a target domain. We plot the accuracy and absolute value of classification margin for each experiment.

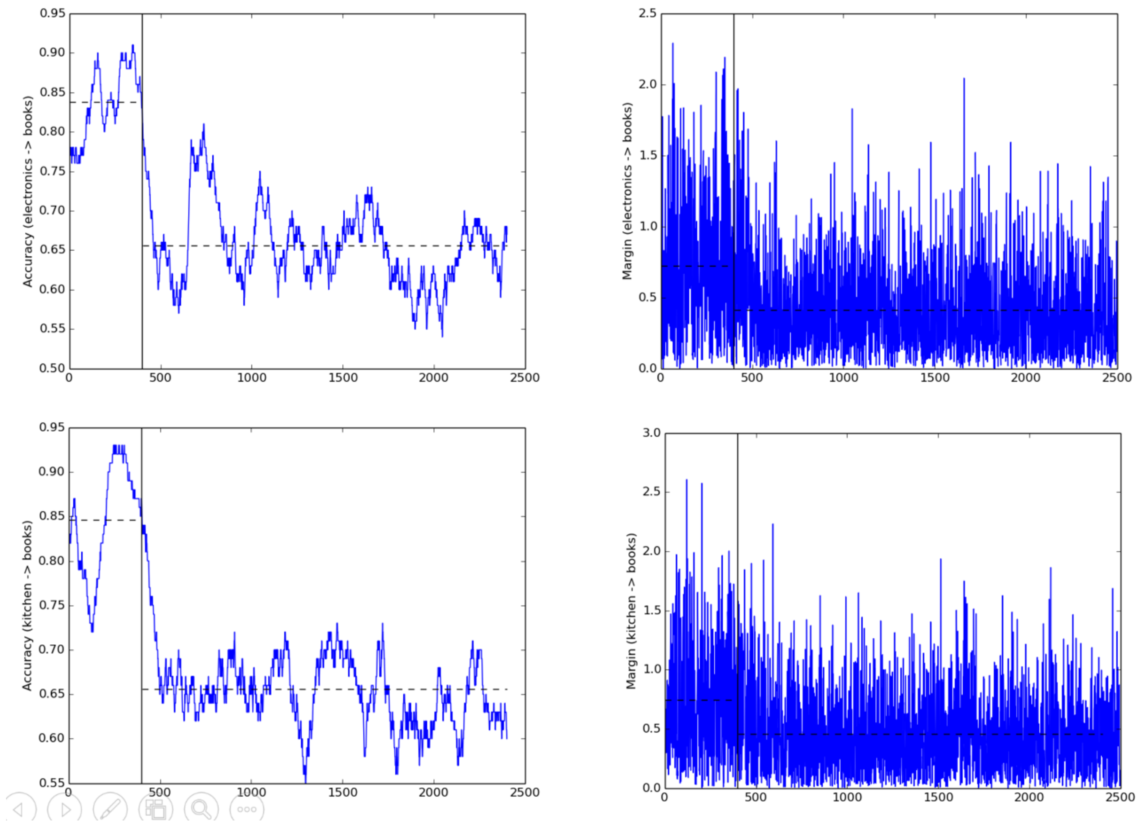

FIG. 4.2: Change in accuracy and the absolute value of classification margin after a domain shift

In Fig. 4.2, the first row demonstrates the effect on accuracy and absolute value of classification margin for domain shift from electronics to books reviews and the second row demonstrates for domain shift from kitchen products to book reviews. The horizontal axis is the number of instances from the stream processed by the classifier. The left half in the figure shows change in accuracy of the classifier on a moving window of the last 50 instances and the right half shows change in absolute value of classification margin when a domain shift occurs. The vertical line at 450 instances marks the point of domain shift. Horizontal dotted lines indicate the mean of the accuracy or margin before and after the domain shift. We observe that in both cases, the mean accuracy drops, as do the mean margin values, demonstrating that both can indicate domain shifts.

4.4 $\mathcal{A}$-distance for detecting domain shift

We use the $\mathcal{A}$-distance algorithm to detect distributional changes in streams of classification margin values. $\mathcal{A}$-distance is ideal for such a stream-based setting where we have to detect change in the distribution of a stream of values. We run the $\mathcal{A}$-distance algorithm on sample streams of instances to calibrate the parameters. Once the calibration is done, we apply the $\mathcal{A}$-distance detector on the classification margin values. As the SVM classifier begins to label examples from a stream of instances (simulating the stream of instances in the real-world), the $\mathcal{A}$-distance detector processes the margin values. The initial n numbers in the stream are extracted from known domain instances, and the most recent n are extracted from unknown domain instances. The detector registers a domain change when the $\mathcal{A}$-distance between the two windows of samples is greater than the threshold value.

4.5 After detecting domain shift - Domain Adaptation

If a domain shift occurs, classifier accuracy often decreases. Once the $\mathcal{A}$-distance detector detects a domain shift, we need to take action so that the accuracy improves. The classifier must adapt itself to the new domain to provide good results. Fig. 4.3 represents our domain adaptation algorithm.

Consider a scenario where we train the support vector machine classifier on 1000 instances from electronics product reviews. We have a model list to store all the past models and their corresponding average classification margin values. We store this model and its average margin in the model list. We use this model to initialize our *currentModel* which classifies future test samples. Then we use another 1000 instances from the source do-

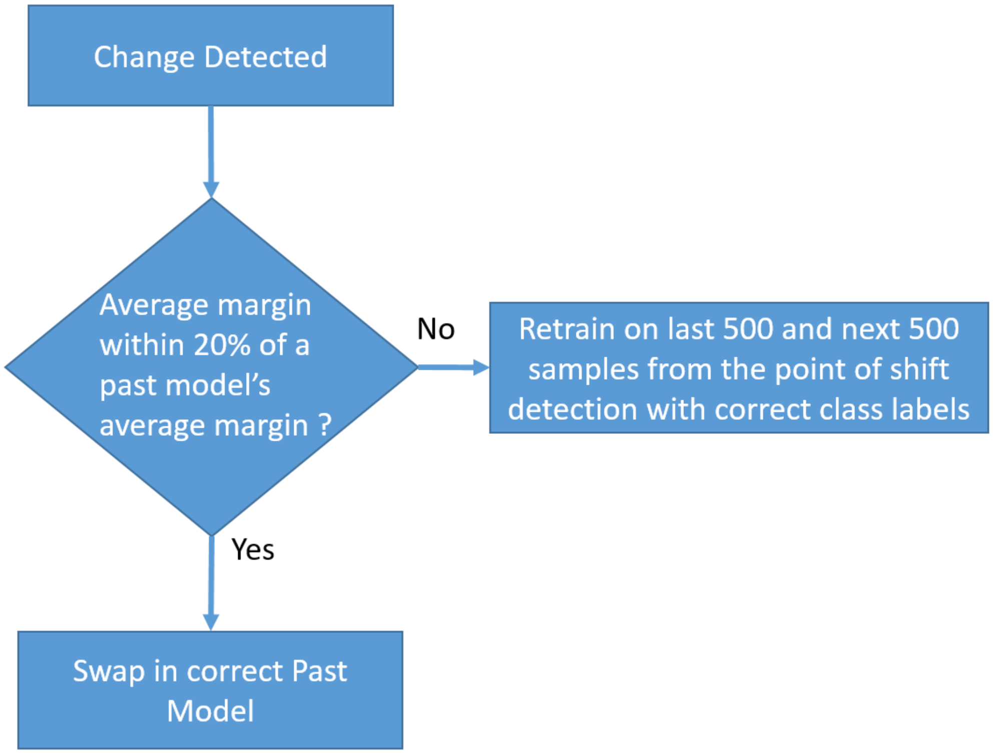
Change Detected
Average margin within 20% of a past model's average margin ?
No
Retrain on last 500 and next 500 samples from the point of shift detection with correct class labels
Yes
Swap in correct Past Model

FIG. 4.3: Steps in Domain Adaptation Algorithm

main for configuring the parameters of the $\mathcal{A}$-distance tracker. If the test samples are also from the electronics domain, no change is detected and the classification continues using the currentModel. Now, if the input samples to the classifier are from the book domain, a change is detected. If the average classification margin on the 1000 samples before the point of change detection is within 20% of any of the past model's corresponding average margin value, we swap in that past model and set currentModel to that past model. If it is within 20% of currentModel's corresponding average margin, we consider this as a false detection and continue classification using currentModel. If the average value over the last 1000 instances is not within 20% of any of the past model's average margin values, we get the correct class labels for the last 500 instances and the next 500 instances after the point of shift detection and retrain to generate a new model. We retrain on 500 instances before and after the point of domain shift detection because we assume that the input stream will have at least 1000 consecutive samples from the same domain and we assume that the A-distance tracker detects changes with considerable accuracy (within 700 samples). We save this model and its corresponding average margin value in our model list and set currentModel to this model for classification of future instances. Now, if the input samples are from the electronics domain again, the past model trained on the electronics domain will be swapped in. If the input samples are from the clothing domain, our algorithm will correctly detect this domain shift. Since no model exists which is trained on clothing data, ideally, the average margin value on these samples will not be within 20% of any of the past models. So, a new model will be generated by retraining the last 500 and the next 500 samples after the domain shift with their correct class labels. Thus our algorithm can be used to detect domain shifts for multiple domains and adapt to new domains while efficiently reusing past models.

We used the average value of classification margins in our algorithm because we assume that the margin values of test samples belonging to the same domain do not vary much. So the average value is representative of all the margin values. Our algorithm uses a 20% threshold to select a past model. If the threshold is very low like 5% or 10%, we might not be able to reuse the past models because the average margin value on one set of samples will not be exactly the same as on another set of samples. We might end up retraining every time we detect a domain shift. If we keep the threshold high, like 30%, there could be errors in selecting the correct past model. We might end up in selecting a past model more frequently even if that domain was never seen before and retrain less frequently. We want a balance between these two cases, hence we chose 20% as our threshold.

Chapter 5

EXPERIMENTS AND RESULTS

5.1 Dataset

We used Amazon product reviews (McAuley 2014) for creating a new dataset for our experiments. Among the various product reviews, we selected books, clothing, electronics, health care, music and pet supplies reviews. The reviews are in JSON format. Each review consists of a reviewer Id, name, a rating of 0-5 stars and review text. An example of a review for a product in the electronics category is shown below.

{“reviewerID”: “A284PZTO3H9FEN”, “asin”: “0972683275”, “reviewerName”: “Jason Pecina”, “helpful”: [0, 0], “reviewText”: “fit perfect for samsung 50’ led tv sturdy material cant beat the price plan on buying another for my other tvs”, “overall”: 5.0, “summary”: “great buy”, “unixReviewTime”: 1371686400, “reviewTime”: “06 20, 2013”}

We created a new dataset which consisted of two classes - positive reviews and negative reviews. For this we extracted the fields named “overall” and “reviewText” from each review. The reviews with rating (overall) $>$ 3 were labeled positive and the reviews with rating $<$ 3 were labeled negative. We did not use reviews with rating 3 because they were mostly neutral. For our purpose, we extracted 20,000 items for each category. After the extraction and labeling, the items in our new dataset consisted of a class label (positive or negative) followed by a tab, and then followed by the review text. An example of an instance from the new dataset is shown below.

Positive	fit perfect for samsung 50' led tv sturdy material cant beat the price plan on buying another for my other tvs

If we considered reviews with rating 3, then our system could be fooled by a shift in sentiment. Reviews which are strongly positive or negative have high values of classification margin but reviews with rating 3 are generally neutral and hence the margin values on these reviews will be low. Consider a classifier system trained on reviews in a source domain. If it gets reviews from the same domain for classification with rating 3, the $\mathcal{A}$-distance detector might detect a domain shift because of the difference between margin values on these samples (low margin values) and margin values on the training data (high margin values). This is a false detection because even tough the margin values are low, there is no change in the domain.

5.2 SVM Classifier using Scikit-learn

Scikit-learn (Pedregosa *et al.* 2011) is an open source machine learning library for the Python programming language. It features various classification, regression and clustering algorithms like support vector machines, nave bayes, decision trees, k-means etc. Support vector machines are implemented by a Cython wrapper around LIBSVM.

5.2.1 Feature Extraction

We extracted features from the review text before performing machine learning. The text needs to be converted into numerical feature vectors. We used the bag of words model in which a fixed integer ID is assigned to each word occurring in any document of the training set (building a dictionary from words to integer indices). Then, for each document #i, the number of occurrences of each word w are stored in X[i,j] as the value of feature #j where j is the index of word w in the dictionary.

Scikit-learn provides CountVectorizer to convert a collection of text documents to a matrix of token counts. It produces a sparse representation of the counts. We used bigrams for our classification by specifying ngram_range= $(1, 2)$.

The number of occurrences of a word in a document cannot be used for classification because longer documents will have more occurrences of many words than smaller documents. So we used term frequency, which is the number of occurrences of each word in a document divided by the total number of words in the document. Another refinement can be done to term frequency called "term frequency times inverse document frequency" which downscales weights for words that occur in many documents in the corpus. These words are less informative than the words occurring only in a smaller portion of the corpus.

5.2.2 Training

After extracting features, we trained a classifier to predict the type of review given the review text. We used support vector machines provided by scikit-learn to train our model. We used the SVC linear kernel for our model. From the 20,000 samples extracted for each category, we took 1000 samples of a category randomly for training. The remaining samples were used for testing. The category of reviews on which we train the classifier is called as the source domain and the category of reviews the classifier has not seen during training is called as the target domain.

5.3 Parameters for $\mathcal{A}$-distance

Given a trained support vector machine classifier and a stream of absolute margin values, to use the $\mathcal{A}$-distance to find significant changes in the distribution of margin values we do the following:

- Choose a window size, *n*
- Choose a set of intervals over the real line, $\mathcal{A}$
- Choose a minimum detected change, ϵ

We can choose intervals with a fixed length. The intervals could be like A1 : 0.0-0.2, A2 : 0.2-0.4 and so on. Choosing intervals with a fixed length might not always be the best approach. The probability of a new value falling in each interval will not be uniform. It is good to have a distribution dependent way of choosing intervals. In this method, we divide part of the original distribution into an equal number of points. This makes the probability of a value falling in each interval uniform. For this, we sort part of the original distribution and have a fixed number of data points in each interval. We need to decide the percent of data in each interval and we can have some percent of an interval overlapping with the next interval. After several runs, we observed that this method performed better than the fixed intervals method on the our dataset.

Given n and intervals $\mathcal{A}$, the value of ϵ is chosen by randomization testing. Because the

$\mathcal{A}$-distance is distribution independent, a sample of size m $\gg$ n is drawn from any distribution that spans $\mathcal{A}$. This sample is treated as a stream as described above, and the largest value of $\mathcal{A}$-distance is stored. The sample is permuted and this process is repeated *l* times. Note that any change detection would be a false positive because all values were sampled from the same distribution. The values of $\mathcal{A}$-distance are sorted from largest to smallest, and ϵ is chosen to be the $[\alpha l]^{th}$ value where parameter α is a user specified false positive probability. We used m = 5000, α = 0.05 and *l* = 50.

We select the window size (n) experimentally by comparing the average error in shift detection (the number of examples observed after a shift occurred before the $\mathcal{A}$-distance detector registered a change). From our experiments, the window size of 100 with the other parameters as mentioned above gives the most accurate detection of distributional change in classification margin values. Window size is related to how quickly changes in the data can be measured. If n is small, sudden changes can be detected and the sensitivity (true positive rate) is high, but the specificity of changes is low. We seek to strike a balance between sensitivity (detection of even smaller changes) and specificity (low shift errors) by choosing the $\mathcal{A}$-distance parameters like window size (n). This choice is most of the times application dependent and represents trade-offs between false positives and false negatives. For example, for medical applications we might allow some false positives to avoid false negatives, while for other applications, like monitoring social networks, one might allow more false negatives.

5.4 An experiment to analyze cost incurred by our model

We use the Amazon product reviews dataset from (Blitzer *et al.* 2007) to analyze the cost of retraining vs the cost incurred due to errors. Prior work (Blitzer *et al.* 2007), (Pan *et* *al.* 2010), (Xia *et al.* 2013) and (Dredze & Crammer 2008) demonstrated domain adaptation for a classifier system using different techniques on this dataset. This dataset contains reviews for four product types: books, dvds, electronics, and kitchen products. For each category there are 2000 samples, of which 1000 samples are positive reviews and the remaining 1000 samples are negative reviews. Since this dataset does not have enough samples required to demonstrate our algorithm effectively, we re-sample the 2000 instances in each category and create a new dataset of 10,000 instances in each category.

Consider the following experiment setup. The SVM classifier is trained on 1000 samples from one product type reviews and tested on another product type reviews. The classifier takes input in batches of 1000 samples and predicts a class label for each sample. We show empirically that our algorithm improves overall accuracy of the classifier system significantly.

We train the classifier on 1000 samples from kitchen product reviews. We use 1000 more samples from kitchen product reviews for configuration of the $\mathcal{A}$-distance parameters. The classifier then gets 2000 kitchen reviews followed by 8000 books reviews for classification. We calculate accuracy over a sliding window of 100 instances for all the test samples. The left side of Fig. 5.1 shows the accuracy of the classifier without domain adaptation and the right side shows the accuracy of the classifier using our method for domain adaptation.
The left side of Fig. 5.1 shows a drop in accuracy after 3000 instances, i.e., when the domain shift to book reviews occurs. The mean accuracy of the classifier in this case is 68.74% (horizontal line in the figure). The total number of correct predictions is 7562 out of 11000 (including configuration samples). The right side of Fig. 5.1 shows a drop in accuracy after 3000 instances after a domain shift. The $\mathcal{A}$-distance detector detects a domain shift and then the classifier is retrained on the last 500 and next 500 samples from the point

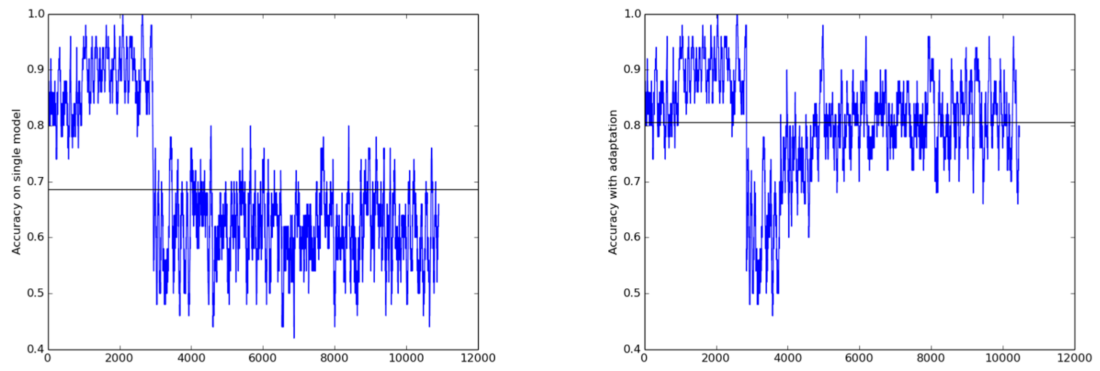

FIG. 5.1: Difference in accuracy for the two models

of shift detection after determining the correct class labels. We can see that the accuracy improves after retraining. The mean accuracy over all samples in this case is 80.70%. The total number of correct predictions is 8877 out of 11000. The number of errors reduced due to our model, *e* = (11000 - 7652) - (11000 - 8877) = 1225.

Let's analyze if this training was worth the improvement in accuracy. Let *n* be the number of instances on which we retrain, *l* be the cost of obtaining the correct class label, e_1 be the number of errors with a single model, e_2 be the number of errors on our model (with domain adaptation), *e* be the reduction in number of errors with our model ($e_1 - e_2$) and *c* be the cost incurred due to an error.

Cost incurred using single model = cost of errors = e_1c

Cost incurred using our model = retraining cost + cost of errors = $nl + e_2c$

We can say that our method is more effective than the single model if the cost incurred for our model is less than that for the single system.

$$e_1c > nl + e_2c$$

$$(e_1 - e_2)c > nl$$

$$ec > nl$$

This means that if $e > (nl)/c$ our method is effective. Here, l and c are constant. We can also fix the value of n like we did in the above example where n=1000. So, if the classifier system has a long lifetime, the reduction in number of errors (e) also increases with time (with more test samples). Thus in the long run, our method will always be worth the expense it incurs for retraining.

5.5 Experiments

Now we perform our experiments on the Amazon product reviews dataset from (McAuley 2014) on these categories of reviews - books, clothing, electronics, health care, music and pet supplies. All the experiments are performed several times (5-10 times depending on the variation in results) and the mean values are reported here. For all the experiments we use 16000 samples of data. We show the utility of our approach for a single domain shift, repeated shifts between two domains, and repeated shifts to multiple domains. We show 3 examples each for a classifier system initially trained on books and music reviews. In our experiments, we refer to the following definitions of worst case (single) model and best case model.

- Worst Case (Single) Model - A scenario where the classifier system does not adapt to the domain of input samples. It continues classification using the same model which it was trained on, irrespective of the domain of input samples
- Best Case Model- A scenario where the classifier system adapts to the domain of input samples exactly at the point of domain shift. It means that the input samples are always classified using a model trained on the sample's domain.

1. Train - books reviews, Test - books, pet supplies and health care product reviews.

Fig. 5.2 represents the accuracy for worst case, our model and best case for Case (A), Case (B) and Case (C), each of which is defined below.

Case (A) Test data - 16000 pet supplies reviews:

In this case the domain shift occurs only once. This is the worst case scenario for the single model trained on book reviews. The best case scenario for this test data would be when the classifier is trained on pet supplies reviews.

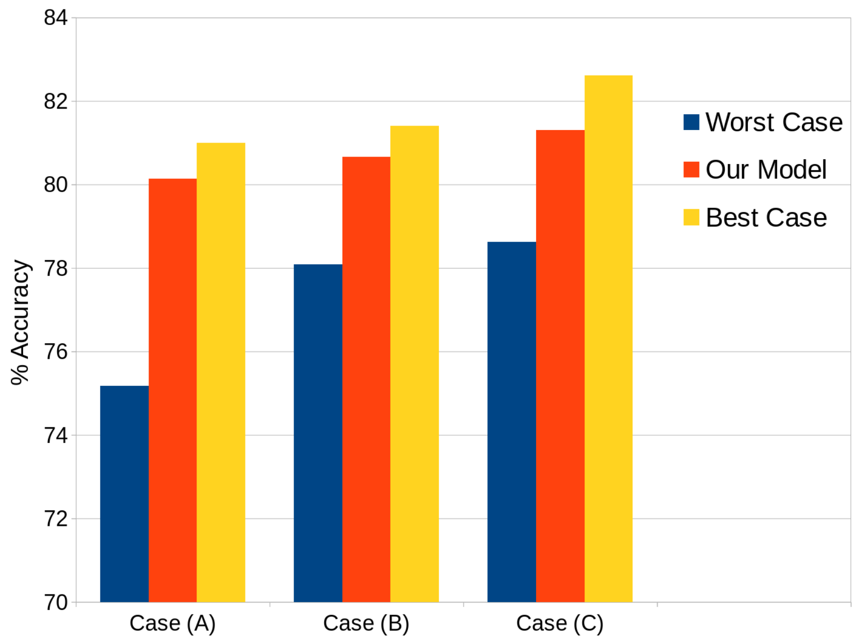

FIG. 5.2: Scenarios for books->pet supplies shift

Case (B) Test data - 1000 books, 5000 pet supplies, 4000 books, 6000 pet supplies reviews:

In this case repeated domain shifts occur between books and pet supplies domains. The worst case scenario is using the single model trained on books reviews. The best case scenario would be when the right model is swapped in exactly at the point of each shift.

Case (C) Test data - 1000 books, 5000 pet supplies, 7000 health care, 3000 pet supplies reviews:

In this case, there are more than two domains. Multiple shifts occur between new domains as well as previously seen domains. The worst case scenario is using the single model trained on books. The best case scenario is shifting to the right model exactly at the point of domain shift.

2. Train - music reviews, Test - music, books and pet supplies reviews.

Fig. 5.3 represents the accuracy for worst case, our model and best case for Case (A), Case (B) and Case (C), each of which is defined below.

Case (A) Test data - 16000 pet supplies reviews:

In this case a single domain shift occurs from music reviews to pet supplies reviews.

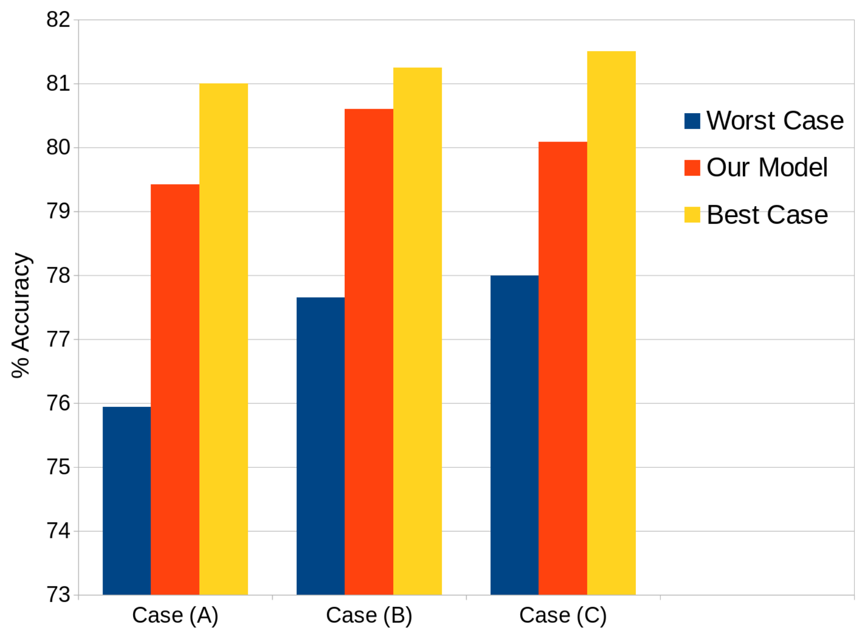

FIG. 5.3: Scenarios for music->pet supplies shift

Case (B) Test data - 6000 pet supplies, 4000 music, 6000 pet supplies reviews:

Here, repeated domain shifts occur between the music and pet supplies domains.

Case (C) Test data - 5000 pet supplies, 3000 music, 4000 pet supplies and 4000 books reviews:

In this case, there are more than 2 domains. Multiple shifts occur between new domains as well as previously seen domains by the classifier.

3. Train - books reviews, Test - books and electronics reviews.

Fig. 5.3 represents the accuracy for worst case, our model and best case for Case (A), Case (B) and Case (C), each of which is defined below.

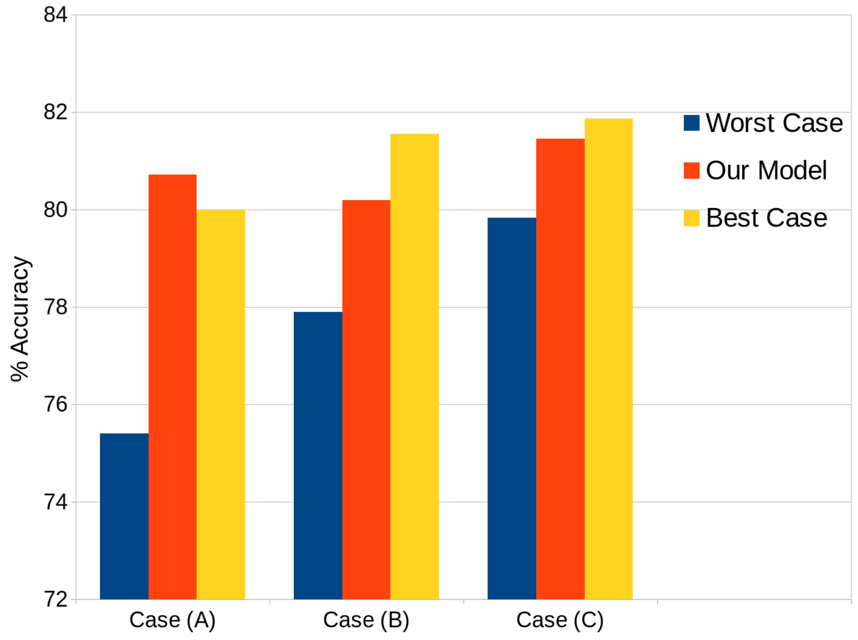

FIG. 5.4: Scenarios for books->electronics shift

Case (A) Test data - 16000 electronics reviews:

In this case there exists only one domain shift from books domain to electronics domain.

Case (B) Test data - 5000 electronics, 4000 books, 5000 electronics and 2000 books reviews:

In this case multiple domain shifts occur between books and electronics domains.

Case (C) Test data - 2000 books, 4000 electronics, 3000 books and 7000 clothing reviews:

In this case, there are more than 2 domains. Multiple shifts occur between new domains (clothing and electronics) as well as previously seen domains (books) by the classifier.

5.6 Tests of Statistical Significance

We use both unpaired and paired t-tests to show that the worst case scenario (single model) accuracies are significantly different from our model's accuracies in all the experiments. We also show that our model's accuracies are not significantly different from best case scenario accuracies. The accuracies for the single model, our model and the best case model from the experiments are shown in Table 5.1.

No.	Single Model Accuracy%	Our Model Accuracy%	Best Case Accuracy%
1	75.18	80.14	81
2	78.08	80.67	81.4
3	78.63	81.3	82.61
4	75.94	79.42	81
5	77.65	80.6	81.25
6	78	80.09	81.51
7	75.4	80.72	80
8	77.9	80.19	81.55
9	79.83	81.45	81.87
Mean	77.4011	80.5089	81.3544

Table 5.1: Accuracies for single model, our model and best case in all the experiments

1. Unpaired t-tests

To calculate the value of *t* and degrees of freedom (*dof*) for an unpaired t-test, we use the following formulae.

$$t = \frac{\overline{X}_1 - \overline{X}_2}{\sqrt{\dfrac{var_1}{n} + \dfrac{var_2}{n}}}$$

$$dof = \frac{\left(\dfrac{var_1}{n_1} + \dfrac{var_2}{n_2}\right)^2}{\dfrac{\left(\dfrac{var_1}{n_1}\right)^2}{n_1 - 1} + \dfrac{\left(\dfrac{var_2}{n_2}\right)^2}{n_2 - 1}}$$

In these formulae, $\overline{X}_1$ represents the mean of the first distribution, var_1 represents the variance of the first distribution and *n* represents the number of samples.

Worst Case vs Our Model - Substituting the values in the formula to calculate t:

$$t = \frac{77.4011 - 80.5089}{\sqrt{\dfrac{2.4516}{9} + \dfrac{0.3987}{9}}} = \frac{-3.1078}{\sqrt{0.2724 + 0.0443}} = \frac{-3.1078}{0.0.5628} = -5.522$$

The calculated *t* value is -5.522. The degrees of freedom are 11. Entering the *t* table with 11 degrees of freedom, we see that for α = 0.01, the tabled value is 3.106. The absolute value of the calculated *t* exceeds the tabled value at alpha = 0.01, so we reject the null hypothesis (the two means are same) and accept the alternate hypothesis that the mean accuracy of our model is significantly higher than the mean accuracy of the single model and is not the result of random chance. Thus, we prove that our model is significantly better than the single model.

Best Case vs Our Model - Substituting the values in the formula to calculate t:

$$t = \frac{81.3544 - 80.5089}{\sqrt{\dfrac{0.5004}{9} + \dfrac{0.3987}{9}}} = \frac{0.8455}{\sqrt{0.0556 + 0.0443}} = \frac{0.8455}{0.3161} = 2.6747$$

The calculated *t* value is 2.6747. The degrees of freedom are 16. Entering the *t* table with 16 degrees of freedom, we see that for α = 0.01, the tabled value is 2.921. The absolute value of the calculated *t* is less than the tabled value at alpha = 0.01, so we cannot reject the null hypothesis that the two distributions are *not* significantly different. We accept the null hypothesis that the two distributions are similar. Thus, we prove that our model is similar to the best case model.

2. Paired t-tests

We also use a paired test to show the significance of our model as it takes into account the difference in the accuracies of the two models for each experiment. To calculate the value of *t* and degrees of freedom (*dof*) for a paired t-test, we use the following formulae.

$$t = \frac{\overline{X}_D}{\sqrt{\dfrac{var_D}{n}}}$$

$$dof = n - 1$$

In these formulae, $\overline{X}_D$ represents the mean of the differences between the pairs, var_D represents the variance of the differences between the pairs and *n* represents the number of samples.

Worst Case vs Our Model - Substituting the values in the formula to calculate t:

$$t = \frac{-3.1078}{\sqrt{\dfrac{1.6083}{9}}} = \frac{-3.1078}{\sqrt{0.1787}} = \frac{-3.1078}{0.4227} = -7.3522$$

The calculated *t* value is -7.3522. The degrees of freedom are 8. Entering the *t* table with 8 degrees of freedom, we see that for α = 0.01, the tabled value is 3.355. The absolute value of the calculated *t* exceeds the tabled value at alpha = 0.01, so we reject the null hypothesis (the two means are same) and accept the alternate hypothesis that the mean accuracy of our model is significantly higher than the mean accuracy of the single model and is not the result of random chance. Thus, we prove that our model is significantly better than the single model.

Best Case vs Our Model - Substituting the values in the formula to calculate t:

$$t = \frac{-0.8454}{\sqrt{\dfrac{0.7101}{9}}} = \frac{-0.8454}{\sqrt{0.056}} = \frac{-0.8454}{0.2366} = -3.5731$$

The calculated *t* value is -3.5731. The degrees of freedom are 8. Entering the *t* table with 8 degrees of freedom, we see that for α = 0.01, the tabled value is 3.355. The absolute value of the calculated *t* is greater than the tabled value at alpha = 0.01, so we reject the null hypothesis that the two distributions are similar. Even tough the t-test shows that the two distributions are different, we can see that the difference between the calculated t-value (3.5731) and the tabled t-value at alpha = 0.01 (3.355) is very small (0.2181). So, we can still say that our model performs nearly as good as the best case model.

Chapter 6

CONCLUSION AND FUTURE WORK

It is very important for a classifier system to learn about the changes in distribution in a stream of data and adapt with the changes in order to perform well and maintain good accuracy. In this thesis, we have presented a method for the support vector machine classifier to adapt to a new domain and recover from multiple domain shifts effectively. Since, the classification margin of support vector machines incorporates information about the domain of test samples, we used the absolute value of the classification margin for measuring distribution changes in a stream of instances. The streaming classification margin values were monitored using the $\mathcal{A}$-distance metric of (Kifer, Ben-David, & Gehrke 2004) and changes in distribution that adversely affect the performance of the classifier were detected. After detecting these changes (domain shift), the classifier could be retrained on a small subset of test samples with the correct class labels to create a new model or a previously trained model could be used or the current model could be used for classification of future instances. Our proposed method decides what action to take after detecting a domain shift. Our method is effective as it avoids unnecessary computations for retraining by storing the learned knowledge and reusing the model whenever the test samples shift again to an old domain. The results with support vector machine classifier for sentiment classification on the Amazon product reviews dataset showed the utility of the overall approach.

As a part of future enhancements to this approach, we could derive expressions for true positives and true negatives as a function of retraining cost, number of test samples and error cost. We could also derive expressions for false positives and false negatives depending on the type of application. For example, for an application in the medical domain we would want to minimize false negative as much as possible. Besides sentiment classification, we would also like to apply our method on other domain adaptation scenarios like part of speech tagging, named-entity recognition and shallow parsing. Another enhancement that could be done is automatically deciding the parameters - window size, distribution of intervals and threshold for change detection, for the A-distance metric depending on the type of application to improve the results. Our algorithm reuses a past model only if the average margin value on the input samples with that model is within 20% of the model's standard average margin value. More work can be done to analyze the best threshold to improve the accuracy of the classifier system. Nonetheless, our current method is a promising tool for recovering effectively from multiple domain shifts.

REFERENCES

[1] Ben-David, S.; Blitzer, J.; Crammer, K.; Pereira, F.; et al. 2007. Analysis of representations for domain adaptation. *Advances in neural information processing systems* 19:137.

[2] Blitzer, J.; Dredze, M.; Pereira, F.; et al. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, volume 7, 440–447.

[3] Blitzer, J.; McDonald, R.; and Pereira, F. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, 120–128. Association for Computational Linguistics.

[4] Buitinck, L.; Louppe, G.; Blondel, M.; Pedregosa, F.; Mueller, A.; Grisel, O.; Niculae, V.; Prettenhofer, P.; Gramfort, A.; Grobler, J.; Layton, R.; VanderPlas, J.; Joly, A.; Holt, B.; and Varoquaux, G. 2013. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 108–122.

[5] Chinavle, D.; Kolari, P.; Oates, T.; and Finin, T. 2009. Ensembles in adversarial classification for spam. In *Proceedings of the 18th ACM conference on Information and knowledge management*, 2015–2018. ACM.

[6] Cui, H.; Mittal, V.; and Datar, M. 2006. Comparative experiments on sentiment classification for online product reviews. In *AAAI*, volume 6, 1265–1270.

[7] Daumé III, H. 2009. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*.

[8] Dredze, M., and Crammer, K. 2008. Online methods for multi-domain learning and adaptation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 689–697. Association for Computational Linguistics.

[9] Dredze, M.; Oates, T.; and Piatko, C. 2010. We're not in kansas anymore: detecting domain changes in streams. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 585–595. Association for Computational Linguistics.

[10] Duan, L.; Tsang, I. W.; Xu, D.; and Chua, T.-S. 2009. Domain adaptation from multiple sources via auxiliary classifiers. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 289–296. ACM.

[11] Farhadi, A., and Tabrizi, M. K. 2008. Learning to recognize activities from the wrong view point. In *Computer Vision–ECCV 2008*. Springer. 154–166.

[12] Kifer, D.; Ben-David, S.; and Gehrke, J. 2004. Detecting change in data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, 180–191. VLDB Endowment.

[13] Martineau, J., and Finin, T. 2009. Delta tfidf: An improved feature space for sentiment analysis. *ICWSM* 9:106.

[14] McAuley, J. 2014. Amazon product reviews. `http://jmcauley.ucsd.edu/data/amazon/links.html`.

[15] OpenCV. Introduction to support vector machines. `http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html`.

[16] Pan, S. J.; Ni, X.; Sun, J.-T.; Yang, Q.; and Chen, Z. 2010. Cross-domain sentiment

classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*, 751–760. ACM.

[17] Pang, B.; Lee, L.; and Vaithyanathan, S. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, 79–86. Association for Computational Linguistics.

[18] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

[19] Sindhwani, V., and Melville, P. 2008. Document-word co-regularization for semi-supervised sentiment analysis. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, 1025–1030. IEEE.

[20] Tan, S.; Cheng, X.; Wang, Y.; and Xu, H. 2009. Adapting naive bayes to domain adaptation for sentiment analysis. In *Advances in Information Retrieval*. Springer. 337–349.

[21] Turney, P. D. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, 417–424. Association for Computational Linguistics.

[22] Uribe, D. 2010. Domain adaptation in sentiment classification. In *Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on*, 857–860. IEEE.

[23] Wikipedia. 2016. Support vector machine – wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Support_vector_machine&oldid=713798909.

[24] Xia, R.; Zong, C.; Hu, X.; and Cambria, E. 2013. Feature ensemble plus sample selection: domain adaptation for sentiment classification. *Intelligent Systems, IEEE* 28(3):10–18.