# A Study of Boosting based Transfer Learning for Activity and Gesture Recognition

by

Ashok Venkatesan

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved May 2011 by the
Graduate Supervisory Committee:

Sethuraman Panchanathan, Chair
Baoxin Li
Jieping Ye

ARIZONA STATE UNIVERSITY

August 2011

# ABSTRACT

Real-world environments are characterized by non-stationary and continuously evolving data. Learning a classification model on this data would require a framework that is able to adapt itself to newer circumstances. Under such circumstances, transfer learning has come to be a dependable methodology for improving classification performance with reduced training costs and without the need for explicit relearning from scratch. In this thesis, a novel instance transfer technique that adapts a “Cost-sensitive” variation of AdaBoost is presented. The method capitalizes on the theoretical and functional properties of AdaBoost to selectively reuse outdated training instances obtained from a “source” domain to effectively classify unseen instances occurring in a different, but related “target” domain. The algorithm is evaluated on real-world classification problems namely accelerometer based 3D gesture recognition, smart home activity recognition and text categorization. The performance on these datasets is analyzed and evaluated against popular boosting-based instance transfer techniques. In addition, supporting empirical studies, that investigate some of the less explored bottlenecks of boosting based instance transfer methods, are presented, to understand the suitability and effectiveness of this form of knowledge transfer.

ACKNOWLEDGEMENTS

Working on this thesis has allowed me to learn and understand some of the most complex and interesting problems in Computer Science. Over the time that I spent on this study, I have had the honor and opportunity to interact with some of the most knowledgeable and highly accomplished people in this field. I take this opportunity to thank each and every one from the bottom of my heart.

First and foremost, I would like to express my deepest gratitude to my committee chair, advisor and guide, Dr. Sethuraman Panchanathan for inculcating in me the desire to deliver a high quality research study as part of this thesis. In particular, I would like to thank him for his excellent support and patience in allowing me to choose and work on a study that interests me the most, a kind of freedom rarely given to Master of Science students.

I would also like to convey my sincere thanks to both my committee members and professors, Dr. Jieping Ye and Dr. Baoxin Li, under whom I had taken “CSE 591 - Machine Learning” and “CSE 598 - Multimedia Information Systems” respectively, for their invaluable guidance on both my coursework and research.

This work would not have been possible without the unflinching support I received from my mentor, guide and dear friend, Dr. Narayanan C. Krishnan (CK). From the inception of this research to its finish, he has seen it through its thick and thin, providing me with timely directions and the confidence to keep moving.

I would like to express my special thanks to Rita, Shayok, Dr. Gaurav, Dr. Vineeth and Prasanth for their invaluable advice and insights that helped me shape this work. I also thank all the members of the Center for Cognitive Ubiquitous Computing (CUbiC) at Arizona State University for having made me a part of it and for having created such a congenial environment for research in the lab.

Lastly, I would like to acknowledge the role of my friends and family for their help and understanding. Most importantly, I would like to offer my heartfelt thanks to Appa, Amma and Nandu for everything that I have today and hope to continue their belief in me by working in my chosen area of interest and making a valuable contribution to the society.

TABLE OF CONTENTS

Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

# CHAPTER 1

# INTRODUCTION AND MOTIVATION

Statistical machine learning frameworks aid approximating unknown functions based on data examples. Their utility in recognizing patterns in data makes them a core component in intelligent systems. Among the various challenges faced in building such systems, of particular interest is that of developing robust learning frameworks that function in real-world environments, characterized by non-stationary and continuously evolving data. Today when an enormous amount of data is being generated and collected every second, there is an ever increasing need for these systems and their learning frameworks to possess the ability to handle evolving data by adapting to newer circumstances and reliably recognize newer patterns. This chapter lays the groundwork for this thesis by giving an overview of the challenges real-world data pose with respect to learning a classification model and some of the known learning strategies used for tackling the problem. The foundation is used to broach upon the idea of transfer learning, delved in detail over the rest of the document.

## 1.1. Dataset Shift in Real-World Conditions

Deducing the output of unseen future data is impossible without making assumptions concerning the nature of the data. These assumptions, termed as *bias*, define the hypothesis space and enable algorithms to favor one particular generalization over others [1]. A classic assumption, many learning frameworks are known to make is to consider training and test data to be identically distributed (ID). This assumption, however, fails to hold in real-world conditions,

where data gets outdated frequently over time leading to poor performances of the trained classifier models. This problem need not be restricted to temporal sequences alone, but can be generalized to all forms of sequential data [2]. Examples of application domains where this phenomenon can be observed include signal processing, speech recognition, computer vision, system monitoring, financial forecasting, natural language processing and web mining.

### *2.1.1. Types of Dataset Shift*

The objective of a statistical learning framework can be generalized to the idea of learning a model that makes predictions, $P(y|x)$ for targets $y$ given data examples $x$. Given this formulation, the problem of *dataset shift* can be understood and differentiated based on the model to be learnt and the cause for the change between the training and test datasets. A brief explanation of the different qualitative categories of dataset shift, as mentioned in [3], is given below:

a. **Simple Covariate Shift:** Given a data distribution that can be modeled as $P(y|x)\ P(x)$, the change in data is termed as covariate shift, when there is a change in $P(x)$ as a causal effect of the change in the covariate distribution. A covariate can be defined as an explanatory or a control variable that may be used along with other variables of primary interest for predictive purposes. The covariate is not hidden and is a known component. The change in covariate distribution will not have any effect on the prediction $P(y|x)$.

b. **Prior probability shift:** If the data distribution is modeled in a target-conditioned fashion i.e. $P(x|y)P(y)$ and the prediction $P(y|x)$ is inferred using the Bayes rule, a change in data distribution between training and test

scenarios can be caused due to the change in the prior distribution $P(y)$, while $P(x|y)$ remains unchanged. This problem is termed as prior probability shift.

c. **Sample selection bias:** Sample selection bias occurs when the training data points $\{x_i\}$ do not accurately represent the distribution over test scenario due to a selection process for each item $i$. Similar to covariate shift, the selection process can be modeled to depend on a selection variable $v$. However, unlike covariate shift, in this problem, the selection variable $v$ has an influence over the label distribution $P(y)$.

d. **Imbalanced data:** The problem of imbalanced data arises when in a multiclass dataset one or more classes occur rarely, compared to the others. To avoid redundant training examples of one class, a class-balanced dataset is typically obtained by discarding samples belonging to the frequently occurring class samples or synthetically adding instances of minority class examples. This, leads to a difference in data distribution, between the training and test scenarios, and is termed as a shift *by design*. Given the change, the problem can be perceived as the case of a prior probability shift with a known $P(y)$ value.

e. **Domain Shift:** Domain shift refers to the shift in the value of data points $\{x_i\}$ as a function of a latent variable $x_0$ specific to a domain. A common way of looking at it is a change in the measurement system of $x_i$.

f. **Source component shift:** When data is made up of a number of different sources, each with its own characteristics, the proportions of the sources can vary between training and test scenarios. This change in data is referred to as

source component shift. Three further cases of this shift namely mixture component shift, factor component shift and mixing component shift are detailed in [3]. The difference between this problem and sample selection bias is explained on the basis of quality of the shift. While the shift here has more to do with the change in the underlying causes we may not have any control over, the change in sample selection bias has more to do with the way the data points have been sampled from a specific population – a factor that can be controlled.

A closely related dataset shift that often gets mentioned in the context of on-line incremental learning over data streams is the problem of *concept drift*. Concept drift refers to the change in the underlying classification function, which might be the result of a change in $P(y)$, $P(x|y)$ or $P(y|x)$. The core assumption with the notion of concept drift is the uncertainty with respect to the future samples (test instances), unlike the problems described above, where we are well aware of the presence of a shift [4].

### 1.2. Transfer Learning

In the presence of such data shifts, a straightforward approach to adapting to the changes would be to re-train a separate classifier model from scratch over the new training instances. The process, however, is expensive and can be cumbersome owing to the costs associated with collecting and labeling new training datasets. Still, in some scenarios, it may be possible to obtain very few labeled target instances, but insufficient for training a reliable classifier. In such circumstances, a viable alternative that has the potential to minimize the overall cost of building a

classifier for the newer instances is to transfer appropriate knowledge from the outdated data for use with the new data. The suggested approach is analogous to how humans generalize when learning a new task from extremely few examples. This learning technique is broadly termed as *Transfer Learning*.

The idea of transfer is a central component of learning in humans. As humans, we face a continuous stream of tasks to be learnt over our lifetime and h-andle it with our ability to build on existing knowledge or experience, acquired from tasks learnt in the past [1]. The transfer of knowledge in humans is instinctive, occurring without any conscious thought process, and involves distinguishing relevant and irrelevant knowledge across multiple tasks. This process allows us to generalize well and learn new tasks fast and better. For example, when faced with learning a skill as complex as driving a car, years of learning experience with basic motor skills, typical traffic patterns, communication, logical reasoning, language, etc., play a role in helping us learn. Learning relies so heavily on transfer that without it; the level of human intelligence would be substantially lower [5].

In the last couple of decades, significant research has been made in formulating methodologies for tackling the different kinds of shifts discussed above. A detailed discussion of all these methods would lie beyond the scope of this thesis. Nevertheless, a brief survey of some of the connected literature has been presented in Chapters 2 and 3 to provide the reader with sufficient background and understanding. A majority of the approaches are essentially modifications of established traditional machine learning techniques such as

neural networks, relational learning, on-line incremental learning and ensemble learning that incorporate the idea of knowledge transfer in them.

## 1.3. Motivation

The work presented in this thesis derives its motivation from two specific computational challenges that were faced in realizing automated activity recognition [6]. The challenges and an illustration of dataset shift in a real-world setting are discussed as part of the following case study.

### *2.1.2. Recognition of Cooking Activities: A Motivational Case Study*

In recent times, an increased interest in the fields of human computer interaction and pervasive computing has given rise to many challenging problems based on pattern classification. Among these, activity recognition has been an important problem that has found use in applications as diverse as gaming, surveillance, location recognition, social interaction and assistive and rehabilitative devices. The basic idea behind activity recognition is to recognize actions of one or a group of users by extracting and interpreting data captured using sensing devices (cameras, accelerometers, microphones, etc.,) that are carried by the user or present in the operating environment. For these systems to be robust and easily deployable, it is necessary they overcome challenges such as change in operating environments, addition of new sensing technologies and variations in activity traits across individuals [7].

In this case study, an activity recognition problem namely “Recognition of Cooking Activities” is reviewed. The task is a research problem that focuses on establishing a framework that can be used to support patients, suffering from

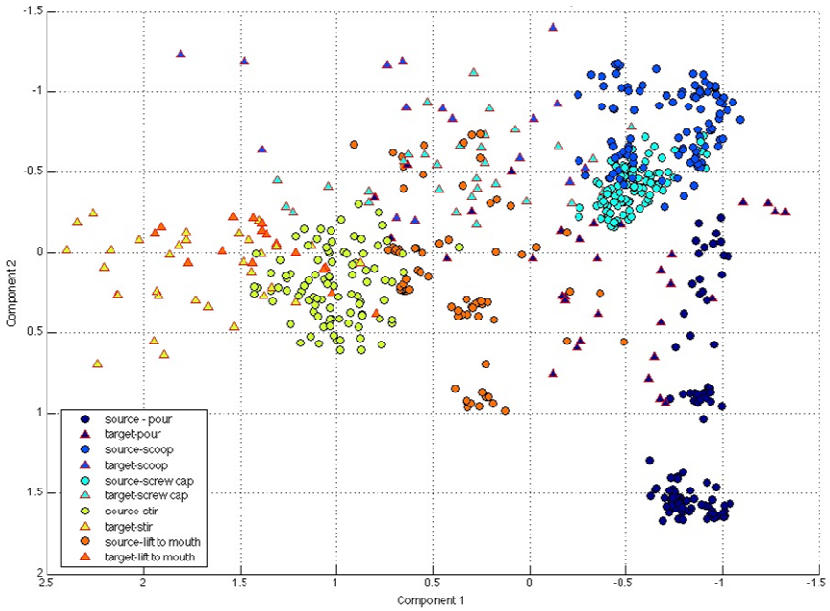
memory impairment, in their instrumental activities of daily living (IADL)[1]. Developing an activity recognition system that can be used for identifying the different stages of a cooking activity requires pattern classification of fine-grained tasks unlike the classification of ambulatory movements. With the objective of training a classifier in a supervised manner, motion data of 5 participants performing the activity of "making a drink and drinking it" were recorded in video and with accelerometers. In order to obtain sufficient training data, the data capture session had been designed to have users enact actions 20 times using mock objects (dummy objects). Relevant features were extracted and the points annotated with one of the 5 common hand gestures namely *pour*, *scoop*, *screw/unscrew cap*, *stir* and *lift to mouth.* 5-fold cross validations were run with SVM and AdaBoost such that in each fold, data points corresponding to 4 subjects were used to train the classifier and the data instances of the left-out subject were used for testing, to obtain high mean accuracies of 92% and 90% respectively.

The classifier models obtained were then tested for their ability to generalize over data points captured in a real-world setting. In this setting, no mock objects were used and the activities were actually performed instead of being enacted out like in training. The new data points captured activities of 4 participants who "made a glass of Tang and drank it". Each participant was asked to repeat the entire process 4 times, in order to obtain sufficient data points for testing. This time however, the average accuracies obtained from SVM and

[1] The Lawton Instrumental Activities of Daily Living (IADL) established in 1969 counts food preparation as one of the eight IADLs used for assessing independent living skills among old adults [19]

AdaBoost over 5 folds were 79.4% and 68.4% respectively, significantly lower than the earlier results. In this process of learning a classifier that classifies data points from a real-world setting with good accuracies, two computational challenges can be observed.

The first observation is the failure of the ID assumption to hold on the data considered. The PCA plot (Figure 1) shows signs of a dataset shift between the points of the two domains – mock data space and real-world space. The decrease in the classification accuracies correlates with this observation, indicating the occurrence of spatio-temporal variations in the movement patterns between different system contexts (e.g. operating environments, user traits, etc.,). Since the model was trained on a dataset that consisted only of instances captured from the mock data space, the problem here is a case of *sample selection bias*. This can be further affirmed by observing that the system context affects the label




of its real-

distribution $P(y)$ as well.

The second observation is the difficulty in collecting real-world training data. The difficulty is primarily because of the cumbersome process of collecting, cleaning and annotating real-world data. Besides the effort in preparation of the data, the process of getting participants involved in a real-life activity reduces the data point throughput during data capture sessions, for the same time and effort spent. Owing to this the costs associated with such processes tend to make it inefficient and preventive.

These challenges together with the application setting, offers ample scope for studying how "knowledge" from the labeled training dataset, captured in sufficient amounts from the mock data space, and the few labeled instances that were captured from the real-world space, can be exploited to build a classifier that shows an improved performance on the real-world data.

### *2.1.3. Problem Statement*

This thesis explores the idea of knowledge transfer between non-identical training (source) and test environments (target), by weighting instances based on their relevance in the test environment. The central problem addressed can be stated as "given a significant amount of source data, whose distribution is known to be different from that of the target data, and a small sample of labeled target data, is it possible to design a method that combines these different datasets to reliably classify new unseen data points from the target domain".

To solve this problem, a boosting based transfer learning framework is designed and evaluated. In the process, the following issues are discussed:

1. How can the relevance of instances in a source domain be measured with respect to that of a target domain?
2. How can the boosting algorithm be modified to incorporate the usefulness of source instances and develop a robust transfer learning technique?

### 1.4. Summary

The rest of this thesis is divided into the following chapters: Chapter 2 gives a brief background of transfer learning methodologies and reviews some of the prominent works published in the area. Chapter 3 discusses on using boosting for transfer learning by showcasing some of the known literature present. The properties of each technique are highlighted and their limitations mentioned. Chapter 4 describes the proposed methodology – cost sensitive boosting for transfer learning – and elaborates on the different boosting schemes, the cost estimation processes investigated as part of the thesis. In addition, a thorough description of the different real-world datasets, the algorithms were tested on, namely, gesture recognition, activity recognition and text categorization datasets is also included. Chapter 5 presents the results obtained from the different experiments conducted over the datasets and proceeds to analyze and interpret them. Chapter 6 summarizes the work presented in this thesis and concludes by highlighting the potential future directions of this research.

# CHAPTER 2

# BACKGROUND AND RELATED WORK

Even though "transfer" has been an actively studied phenomenon in cognitive literature, transfer learning as a research direction in computer science attained prominence only in the last decade. Literature published since then has given rise to a variety of transfer learning algorithms, referred using different titles such as: lifelong learning, multi-task learning, inductive transfer, domain adaptation, cross-domain transfer, context-sensitive learning, meta-learning and incremental learning [8]. In this chapter, an introductory account of transfer learning is given supported by a brief literature review done by the author as part of this thesis. The literature review is by no means exhaustive and is rather intended to provide the reader with a well organized overview on the subject. For further reading it is recommended to go through the following survey papers [9; 8; 10], conference proceedings [1] and books [3].

## 2.1. Definitions and Vocabulary

The NIPS Inductive Transfer Workshop 2005 defines transfer learning as "a transfer of knowledge across domains, tasks and distributions that are *similar but not the same*". In general, the training and test datasets involved in transfer learning can be described in terms of the domains they have been sampled from and the learning tasks they represent. A domain $\mathcal{D}$ is the marginal distribution $P(X)$ observed over an instance set $X$ in a specific feature space $\mathcal{X}$ (typically $\mathbb{R}^d$). Given an input space $\mathcal{X}$ and a label space $\mathcal{Y}$ ($\{-1, 1\}$ for binary classification

problems), a task $\mathcal{T}$ is equivalent to the unobserved classification function $f: \mathcal{X} \rightarrow \mathcal{Y}$ to be learnt, which can be stochastically modeled as $f(x) = P(y|x)$, where $y \in \mathcal{Y}$ is the corresponding label of an instance $x \in \mathcal{X}$. The term *source* is used to refer to the data from which knowledge is extracted, while the term *target* is used to refer to the data, over which a classifier model is to be learnt, under the support of the transferred knowledge. A formal definition of transfer learning, as stated by [8]**,** is given below. The definition takes only one source domain and one target domain into account. It can, however, be generalized to apply to cases dealing with multiple sources and multiple targets.

**Definition:** *Given a source domain $\mathcal{D}_s$ and a learning task $\mathcal{T}_s$, a target domain $\mathcal{D}_t$ and a learning task $\mathcal{T}_t$ transfer learning aims to help improve the learning of a target predictive function $f_t(.)$ in $\mathcal{D}_t$ using the knowledge in $\mathcal{D}_s$ and $\mathcal{T}_s$, where $\mathcal{D}_s \neq \mathcal{D}_t$, or $\mathcal{T}_s \neq \mathcal{T}_t$,*

In the above definition, the conditions $\mathcal{D}_s \neq \mathcal{D}_t$ and $\mathcal{T}_s \neq \mathcal{T}_t$ denote the differences between the source and target domains and tasks respectively. The difference between the domains can be explained as either a disparity in the feature spaces $\mathcal{X}_s \neq \mathcal{X}_t$, or a shift in the marginal distribution over the instances $P(X_s) \neq P(X_t)$. On the other hand, the differences between the tasks can be interpreted as either a change in the label space $\mathcal{Y}_s \neq \mathcal{Y}_t$, or that of the predictive function $f_s(.) \neq f_t(.)$. As seen in Section 1.1.1, covariate shift, sample selection bias, domain shifts or source component shifts are typical causes behind marginal distribution shifts, while imbalanced data or concept drift create label space differences. Table 1 gives examples illustrating these differences based on the

document classification problem. While these differences may allow justifying the use of transfer learning, a vital aspect that should not be overlooked is the *related*-ness between the source and target datasets. Similarity is central to transfer and is a fundamental rationale behind a successful transfer.

Table 1 Source and Target Differences illustrated in the case of Document Classification.

| **Difference** | **Document Classification Example** |
|---|---|
| $\mathcal{X}_s \neq \mathcal{X}_t$ | Documents in source domains may be in English, while documents in target domains are in Chinese. |
| $P(X_s) \neq P(X_t)$ | Term frequencies for the documents are distributed differently in the two domains. |
| $\mathcal{Y}_s \neq \mathcal{Y}_t$ | Training domain has binary document classes and test domain has multiple document classes. |
| $f_s(.) \neq f_t(.)$ | When the document classes are balanced in training environment and imbalanced in the test environment. |

## 2.2. Categorizing Transfer Learning Algorithms

It is useful to categorize transfer learning algorithms in order to be able to separate out the concerns and capabilities of systems incorporating these. Like in

humans, transfer learning algorithms facilitate in improving the target task performance, learning speed or, sometimes, both. Which one can be achieved depends on the availability of adequate training data to learn [9]. For example, the objective of the transfer technique used in a speech recognition system that can adapt to new speakers would be different for a dictation system from that of an interactive voice response system. For a dictation system, it might be acceptable to expect a new speaker to train a system for 30 to 40 minutes, as the speaker may eventually go on to use the system for years. On the other hand, a recognition framework that is used as part of an interactive voice response system can only count on a few seconds of unsupervised speech [10] and should learn fast.

A set of distinctions in transfer [9], can be made based on whether the algorithms retain the source task accuracy, after learning the target task or focus exclusively on learning the target task alone. Algorithms that belong to the former group are termed as *sequential transfer* and those that go by the latter approach are termed as *non-sequential transfer*. Further distinctions can be made in the case of sequential transfer algorithms based on whether the source and target tasks are learnt simultaneously or separate in time. Algorithms that adopt the first approach are generically termed as *functional transfer*, while other algorithms that learn the tasks one at a time by carrying an explicit representation from one task to the other are known as *representational transfer*. *Multi-task learning* [11] is a commonly cited technique that falls under the category of functional learners.

Transfer learning settings can be characterized by the availability of labeled data and the variation in the domain or task distributions across the source and target domains. Founded on these learning settings and similar to the categorization of traditional machine learning algorithms, transfer settings can be conveniently categorized [8] as:

- **Inductive transfer:** In this setting, the target task is different from the source task and has very less labeled data to obtain the required classification performance. Similar to inductive learning, the labeled data can be used to obtain a weak target inductive bias. The bias can then be corrected based on the knowledge derived from the source tasks. Here, the source data may or may not be labeled.
- **Transductive transfer:** The objective of transduction is to label the unlabeled data seen during training. Following this, in a transductive transfer setting, the target data is unlabeled and available, while the source data is labeled and available in abundance. The difference in the data, between the target and the source tasks, is generally modeled as a difference in their feature space or domains.
- **Unsupervised transfer:** Here, the target tasks are different from, but related to, the source tasks. However, just as in unsupervised learning, both the source and target data are unlabeled. Common unsupervised techniques such as clustering, dimensionality reduction and density estimation are typically used to make sense of the target data.

It is important that the knowledge to be transferred is well represented. The knowledge may be specific only to certain source domains or may be common across many domains. A good representation makes this information easily identifiable. The knowledge can be modeled as a set of instances, a group of features, model parameters or a relational map. Based on this, transfer algorithms can be classified into the following categories [8]:

- **Instance-based transfer:** reuses training instances from the source domain to augment the training instances observed in the target domain typically by re-weighting or re-sampling.
- **Feature-based transfer:** aims at finding an alternate feature space for the target domain. Common approaches include feature selection and vector space transformations.
- **Model-based transfer:** uses components such as model parameters, of previously learnt source models to influence learning the target task. Approaches vary from plain superimposing of model shape constraints to partitioning of the parameter-space.
- **Relation-based transfer:** works with the idea of spotting and capitalizing on the structural or relational similarity between the source data and the target data. Suitable statistical relational learning techniques are generally applied for the purpose.

The literature review presented from Section 2.4 onwards is organized based on the above categorization.

### 2.3. Relatedness and Negative Transfer

The effectiveness of transfer depends on the source task and how related it is to the target task [12]. When the source and target tasks are strongly related, it would be worthwhile for a transfer algorithm to take advantage of it, to improve the performance on the target tasks significantly. Here, the transfer is termed to be *positive*. However, when the source tasks are not sufficiently similar or if the algorithm itself fails to exploit the existing knowledge in the source tasks, the performance over the target tasks may not only fail to improve, but may actually decrease. This phenomenon is called *negative transfer*. The problem of avoiding negative transfer is an open research issue and can be viewed as the problem of "when to transfer".

Proper selection of the source knowledge can be the difference between positive and negative transfer [5]. Many of the current algorithms assume that the given source tasks are relevant to the target task. These algorithms separate out the process of selection from the transfer framework and assume that the source tasks have been *manually selected*, for transfer, by human experts using heuristics or domain knowledge. On the contrary, it would be more suited for use in real-world applications, if the selection is *automatic* and embedded into the transfer framework. Automatic selection would entail the computation of an *"a priori"* measure of task relatedness before training, instead of evaluating the performance of the classifier retrospectively. The problem of automatic selection is difficult to solve owing to the missing target domain information. Often, the limitation is partially overcome by structuring the abundant source data into a hierarchy of

similar source tasks. Selective transfer helps improving the results in cases where only few support tasks are relevant. Known literature explain this notion of relatedness based on two ideologies (1) Task based similarity and (2) Domain based similarity. Task based similarities are generally computed and applied in an inductive transfer setting. The latter finds more application in transductive transfer settings such as domain adaptation and assumes the predictive functions to be constant across the two domains.

*Task based similarity* refers to a measure that quantifies the difference between a source task $P(\mathcal{X}_s, \mathcal{Y}_s)$ and a target task $P(\mathcal{X}_t, \mathcal{Y}_t)$. Ben-David and Schuller [13] define relatedness in the case of a data generation model. They term tasks as $\mathcal{F}$-related, where $\mathcal{F}$ is a set of transformations $f: \mathcal{X} \to \mathcal{X}$, if for some fixed probability distribution over $\mathcal{X} \times \mathcal{Y}$, the data in each of these tasks is generated by applying some $f \in \mathcal{F}$ to this fixed distribution.

Showcasing a more practical approach, Thrun and O'Sullivan's [14] propose a Task Clustering algorithm that groups learning tasks into classes of mutually related tasks, by using a *globally weighted Euclidean distance metric* to measure the proximity between data points. The distance metric is learnt by minimizing the average inter-cluster similarity and maximizing the intra-cluster similarity. Similarity between tasks is then computed using cross-validated predictive accuracies of *k-nearest-neighbor* classifiers, learning one task using the distance metric of another. Under test conditions, the target tasks observed are matched with source task clusters and the appropriate distance metrics are transferred for classification.

Eaton [5] defines the concept of *transferability* as the transfer relationship between two tasks as the change in performance between learning with and without transfer. The measure's viability is demonstrated as part of a boosting based instance transfer [15] and a relational transfer framework [16]. The instance transfer algorithm titled *TransferBoost* is discussed in the next chapter in more detail. The relational transfer framework is modeled on the same lines as the task clustering algorithm mentioned above with the transferability measure used instead of the Euclidean distance metric.

*Domain based similarity* refers to a measure that quantifies the difference between the source domain $P(\mathcal{X}_s)$ and the target domain $P(\mathcal{X}_t)$. Kifer et al., introduce the concept of $\mathcal{A}$-distance in [17]. For a given domain $\mathcal{X}$ and a collection $\mathcal{A}$ of subsets of $\mathcal{X}$ and probability distributions $\mathcal{D}$ and $\mathcal{D}'$ over $\mathcal{X}$, such that every set in $\mathcal{A}$ is measurable with respect to both distributions, the $\mathcal{A}$-distance between the distributions is theoretically defined as

$$d_{\mathcal{A}}(\mathcal{D}, \mathcal{D}') = 2 \sup_{A \in \mathcal{A}} | \Pr_{\mathcal{D}}[A] - \Pr_{\mathcal{D}'}[A]) |$$

$\mathcal{A}$-distance is closely related to learning a classifier that discriminates between points sampled from different domains. It can be implemented [18] easily by associating a positive label with the source data and a negative label with the target data, thereby modeling this into a binary classification problem. For two dataset samples $\mathcal{S}_s$ and $\mathcal{S}_t$, each of size $m$, the error of a classifier $h$, $\mathcal{A}$-distance is theoretical proven to be,

$$d_{\mathcal{A}}(\mathcal{S}_s, \mathcal{S}_t) = 2(1 - 2 \min_{h \in \mathcal{H}} err(h))$$

Though, some of the above methods can be helpful in detecting change in data and task distributions, in practice, the goals of avoiding negative transfer and facilitating a positive transfer is difficult to realize. Often, algorithms that have safeguards to avoid negative transfer have a reduced effect from positive transfer due to the extra caution [12]. On the other hand, approaches that transfer aggressively might transfer better, but may lack protection from negative transfer. In addition, one cannot discount the inevitable bias actual applications would face when predicting negative transfer with very less information in hand.

### 2.4. Instance-based Transfer

In instance-based transfer, individual data instances are selected from the source domains to help train a classifier for the target domain. When the source and target tasks can be represented in the same instance space, an instance-based transfer may be sufficient for generalizing over the target domain. The training objective for an instance-based transfer is to minimize an error function over target instances and the selected source instances. Instance reweighting and importance sampling are two popular methodologies applied to realize instance-based transfer.

Jiang and Zhai [17] linearly combine several adaptation heuristics using instance-level and global coefficients, into a unified objective function. They tackle domain adaptation using a three step strategy over a probabilistic model of the data, which includes, (1) removing “misleading” training instances in the source domain, (2) assigning more weights to labeled target instances than labeled

source instances and finally (3) augmenting training instances using target instances with predicted labels.

Wu and Dietterich [18] use source domain instances, referred to as "auxiliary data", to improve the classification accuracy of support vector machines (SVM) and identify support vectors that are applicable to a target task. Liao et al.[19] propose an active learning method to select the unlabeled data in a target domain to be labeled with the help of source domain data. They realize this with the help of auxiliary variables and a Fisher information matrix.

A popular framework that finds use as an instance weighting solution for domain adaptation [22] is that of *empirical risk minimization*. The objective of this method is to learn an optimal model $\theta^* \in \Theta$ in a model family, such that expected risk, expressed in terms of a loss function$l(x, y, \theta)$, is minimized. The objective function can be written as,

$$\theta^* = \arg\min_{\theta \in \Theta} \mathbb{E}_{(x,y) \in \mathcal{X} \times \mathcal{Y}}[l(x, y, \theta)]$$

For the setting of domain adaptation, the idea is to obtain an optimal model for the target domain and minimize the expected loss over the target distribution. This can be expressed as,

$$\theta_t^* = \arg\min_{\theta \in \Theta} \sum_{(x,y) \in P(D_t)} P(\mathcal{D}_t).\, l(x, y, \theta)$$

The problem then is reduced to approximating $P(\mathcal{D}_t)$ utilizing the labeled instances picked from the source domain $D_s$. The above problem can be then rewritten as

$$\theta_t^* = \arg\min_{\theta \in \Theta} \sum_{(x,y) \in P(\mathcal{D}_s)} \frac{P(\mathcal{D}_t)}{P(\mathcal{D}_s)} . l(x, y, \theta)$$

$$\approx \arg\min_{\theta \in \Theta} \sum_{i=1}^{n_s} \beta . l(x_{s_i})$$

where $\beta = \frac{P\left(x_{t_i}, y_{t_i}\right)}{P\left(x_{s_i}, y_{s_i}\right)}$. Since it is assumed that the predictive functions are constant, $\beta$ can be estimated as $\frac{P\left(x_{t_i}\right)}{P\left(x_{s_i}\right)}$. This process of estimating the properties of the target distribution with samples generated from source distribution, different from the target distribution, is termed as *importance sampling*.

Huang et al., [23] propose a kernel mean matching algorithm to learn $\beta$ by matching the means between the source and target domain data in a reproducing kernel Hilbert space (RKHS). An advantage of using KMM is that it avoids performing density estimation of either $P(x_{s_i})$ or $P\ (x_{t_i})$, which is difficult when the size of the dataset is small.

Sugiyama et al., [24] propose an algorithm named *Kullback-Leibler Importance Estimation Procedure* (KLIEP) to estimate $\frac{P\left(x_{s_i}\right)}{P\left(x_{t_i}\right)}$ directly, based on the minimization of the Kullback-Leibler divergence measure. KLIEP can be integrated with cross validation to perform model selection automatically in two steps: (1) estimating the weights of the source domain data; (2) training models on the reweighted data.

### 2.5. Feature-based Transfer

Argyriou et al., [25] propose a method for learning a low-dimensional feature representation which is shared across a set of multiple related tasks. Building upon the 1-norm regularization problem, they use a new $(2,1)$-norm regularizer to come up with a non-convex optimization problem, which attempts to simultaneously select a low dimensional feature representation and learn them. They proceed on to formulate an equivalent convex optimization problem and use an iterative algorithm to solve the problem. The algorithm alternately performs a supervised and unsupervised step, where the first step independently learns the parameters of the tasks' regression or classification functions and the latter step converges towards a low-dimensional representation for these task parameters in an unsupervised manner. The optimization problem can be written in this context of TL as given below:

$$\arg\min_{A,U} \sum_{t\in\{T,S\}} \sum_{i=1}^{n_t} L\left(y_t, < a_t, U^T x_{t_i} >\right) + \gamma \left|\left|A\right|\right|_{2,1}^2$$

In this equation, S and T denote the tasks in the source domain and target domain, respectively. $A = [a_s, a_T] \in R^{d\times 2}$ is a matrix of parameters. $U$ is a $d \times d$ orthogonal matrix for mapping the original high-dimensional data to low dimensional representations($U^T X_t$ and $U^T X_s$). The $(2,1)$ norm of A is defined as $\left|\left|A\right|\right|_{r,p} \coloneqq \left(\sum_{i=1}^{d} \left|\left|a^i\right|\right|_r^p\right)^{\frac{1}{p}}$.

Blitzer et al., [26] focus on using unlabeled data from both the source and target domains to learn a common feature representation that is meaningful across

both the domains. The authors term this method as *Structural Correspondence Learning* (SCL). The first step of SCL is to define a set of *pivot* features on the unlabeled data from both domains. These are features that behave in the same way for discriminative learning across both domains. After having selected the pivot features, these are removed from the data and treated as a new label vector. Thus, $m$ binary classification problems can be constructed, where $m$ is the number of pivot features. These classification problems are then trained from the unlabeled data and solved using a linear classifier, $f_{l(x)} = sign(w_l^T.x),\ l = 1,\ldots,m$ to learn a parameter matrix $W = [w_1 w_2 \ldots w_m]$. After obtaining $W$, singular value decomposition (SVD) is applied on it. Let $W = UDV^T$, then $\theta = U^T_{[1:h,:]}$, where $h$ is the number of shared features, is the matrix whose rows are the top left singular vectors of $W$. In the final step, standard discriminative algorithms can be applied to the augmented feature vector to build models. The augmented feature vector contains all the original feature $x_i$ appended with the new shared features $\theta x_i$. Though it has been shown experimentally, that SCL can reduce the difference betwen domains, selecting the pivot features is difficult and domain-dependent. In this paper, Blitzer et al. have used a heuristic method to select pivot features for natural language processing (NLP) problems, such as POS tagging.

Pan et al., [27] exploit a dimensionality reduction method named, *Maximum Mean Discrepancy Embedding* (MMDE) to learn a shared low dimensional latent feature space, such that the distributions between the source and target domain data are the same or close to each other. The theory of Maximum Mean Discrepancy states that the distance between distributions of two

samples is equivalent to the distance between the means of the two samples mapped into a Reproducible Kernel Hilbert Space (RKHS). By capitalizing on this theory, MMDE converts the problem of minimizing a distance function in feature space into a semidefinite program in RKHS to identify a lantent set of features. Post this, supervised and semi-supervised learning approaches are used to train a model for a mapping between the tasks and data across both the domains. The method, however, is computationally inefficient.

### 2.6. Model-based Transfer

Most model-based transfer techniques can be categorized into: (1) approaches that partition the parameter space of a conventional learning algorithm into task-specific parameters and general (cross-task) parameters and (2) approaches that learn shape constraints, which are superimposed when learning a new function.

Evgeniou and Pontil [28] propose an SVM based parameter transfer approach, where the parameters of SVMs for the source and target domain, $w_s$ and $w_t$ share a common parameter, $w_o$. Thus, $w_s = w_o + v_s$ and $w_t = w_o + v_t$. An optimization framework is then formulated for determining the parameters, $w_o, v_t, v_s$.

Raina et al., [29] present an algorithm for constructing the covariance matrix, $\Sigma \in \mathbb{R}^{d \times d}$ for an informative Gaussian prior, $\mathcal{N}(o, \Sigma)$, to learn and classify documents observed in a specific target domain, when the available training data from the target domain is scarce ($n << d$). The algorithm uses

other "similar" learning problems to learn a good underlying mapping from word pair features to word parameter covariances.

Lawrence and Platt [30] propose an efficient algorithm known as MT-IVM (Multi-task, Informative vector machine), which is based on Gaussian Processes (GP), to handle the multi-task learning case. MT-IVM tries to learn parameters of a Gaussian Process over multiple tasks by sharing the same GP prior.

Gao et al., [31] observe that several classification models may be available in a training domain, either sourced from a set of relevant tasks or learnt using different classifiers. No single model may help in summarizing the target task as such. Thus, they propose a locally weighted ensemble in order to additively combine the predictions of multiple source models. The weights for the models are computed by clustering the different tasks into graphs and estimating the similarity of the neighborhood of test instances in these graphs.

### 2.7. Relation-based Transfer

Different from the other three contexts, the relational knowledge transfer approach deals with transfer learning problems in relational domains, where the data are non-id and can be represented by multiple relations, such as networked data and social network data. This approach does not assume that the data drawn from each domain be independent and identically distributed as traditionally assumed. It tries to transfer the relationship among data from a source domain to target domain.

Mihalkova and Mooney [32] perform transfer between Markov Logic Networks (MLN). An MLN consists of a set of first-order logic formulae, each with a weight attached, and provides a model for the joint distribution of a set of variables. Given a learned MLN for a source task an MLN is learnt for a related target task by starting with the source-task one and diagnosing each formula, adjusting ones that are too general or too specific in the target domain. The hypothesis space for the target task is therefore defined in relation to the source task MLN by the operators that generalize or specify formulas.

Dai et al., [33] present a general transfer learning framework called *EigenTransfer*. Their idea is to construct a task graph to represent the transfer learning tasks and model the relations between the target data and the auxiliary data. Instances, features and labels are represented as nodes in the task graph, while the edges are set based on the relations between the end nodes, connecting the target and auxiliary data in a unified graph structure. By computing the eigenvectors of the graphs, the tasks can be represented in a spectral feature space, reflecting the intrinsic structure of the target data, auxiliary data and the relations between them. Knowledge transfer from the auxiliary data is then done in this new feature space, to help learning the target data.

Dai et al., [34] investigate the concept of *translated learning*, where knowledge transfer is performed between two entirely different feature spaces, in this case, text and images. Their algorithm combines feature translation and the nearest neighbor into a unified model by making use of a language model, which is represented using Markov Chains. They adopt the Risk Minimization

framework and formulate a problem that minimizes the risk $R(c, x_t)$ of misclassifying $x_t$ to the category $c$. Assuming no prior difference among all the classes, the risk is simplified to represent the distance between the feature and task space, measured using Kullback-Leibler divergence measure. They make the actual transfer with the help of a translator function $\phi(y_t, y_s) \propto p(y_t|y_s)$.

In the domain of activity recognition, Kasteren et al., [35] present a framework that allows to transfer knowledge of activity recognition from one context to the next. They use wireless binary sensing nodes that can be used to capture activities anywhere in a household, such as measuring a door being opened, a toilet being flushed or the temperature of a stove rising. In this work, they describe a method which uses unlabeled data captured from house A together with labeled data from house B, to learn the parameters of model for activity recognition in house A. The difference in the domains appears in the form of the difference in the layout of the houses and thereby difference in the location of the sensors and the properties they measure. To solve this problem, the authors use a set of manual mapping operations namely Intersect, Duplicate and Union to get the final feature set over which a semi-supervised learning algorithm is used.

Rashidi and Cook [36] propose an unsupervised approach for mapping the sensor and layouts of different living spaces for transferring the activity information from a set of source living spaces to a target living space.

# CHAPTER 3

# BOOSTING AND TRANSFER LEARNING

The central challenge in transfer learning lies in formulating an approach that makes most of the available auxiliary data. In instance transfer algorithms; this problem gets manifested into that of identifying the relevant instance points that would be useful in helping learn a tuned classifier that classifies target domain data points correctly. AdaBoost, short for "Adaptive Boosting", is a well-established algorithm that boosts a weak learning algorithm into a strong one by calling it repeatedly so that the cumulative error of the strong classifier is reduced. It is essentially a greedy algorithm that incrementally alters the distribution of the training data points, used for training the weak-learning algorithm at each iteration. This process allows identifying important examples in the training dataset. However, similar to other traditional learning algorithms, it assumes that the training and test data are sampled from the same instance space. Recent research efforts have looked to extend the boosting principle to work with auxiliary data. Typical challenges faced in modifying AdaBoost for the purpose of transfer learning include:

- Formulation of a similarity measure between cross-domain samples
- Design of weight update factors and obtaining the corresponding optimal value of the parameter $\alpha$ that minimizes the training error bound in the target domain.
- Definition of an appropriate loss function.

- Linear combination of the weak classifiers to obtain a strong classifier that generalizes well over the target domain.

This chapter gives a brief background of the boosting theory and continues to provide a small survey of novel boosting based algorithms that have been adapted for transferring knowledge from source instances to a target domain.

### 3.1. AdaBoost: An Overview

The basic idea of boosting is to learn a "strong" classifier by combining simple classifiers known as "weak learners", which would do at least slightly better than

**Algorithm 2** AdaBoost

**Input** Given a set of $N$ labeled examples $\{(x_1, y_1), \ldots, (x_N, y_N)\}$,distribution $D$ over the $N$ examples, an integer $T$ specifying maximum number of iterations and a weak learner **Weak Learn**
**Initialize** the initial weight vector: $w_i^1 = D(i)$ for $i = 1, \ldots, N$.
**For** $t = 1, \ldots, T$

1. Set $p^t = \frac{w^t}{\left(\sum_{i=1}^{N} w_i^t\right)}$.
2. Call **Weak Learn**, providing it with the distribution $p^t$. Then, get back a hypothesis $h_t : X \to Y$(or $[0, 1]$).
3. Calculate the error of $h_t$:

$$\epsilon_t = \sum_{i=1}^{N} p_i^t \left| h_t(x_i) - y_i \right|. \tag{5.1}$$

4. Set $\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$.
5. Update the new weight vector:

$$w_i^{t+1} = w_i^t \exp\left(-\alpha_t y_i h_t(x_i)\right) \tag{5.2}$$

**Output** the final classifier

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{5.3}$$

Figure 2: The AdaBoost algorithm as described by Freund et al [37]

chance. AdaBoost [37], as proposed in the seminal work of Freund et al., is probably the most popular boosting algorithm. It maintains a distribution or a set of weights over the training set and presents the weak learner with the important examples from the set to obtain a "weak hypothesis". The goodness of a weak hypothesis, as measured by the error over the distribution$D$, is used to update the weights of the training points. In a boosting iteration, the weights of the correctly classified instances are reduced, while the incorrectly classified points are increased. As a result, the weak learner for the subsequent iteration focuses on learning a model that correctly classifies the incorrectly classified instances of the previous iteration. The objective of the algorithm is to find a strong hypothesis, by linearly combining the set of weak hypotheses, with a low cumulative error relative to a given distribution. The pseudo-code for the algorithm is given in Figure 2. In the context of transfer learning, AdaBoost implicitly focuses on the small amount of target domain training data if they are incorrectly classified at any given iteration. It uses the rest of the source data to learn a model that classifies this set of target domain data.

### 3.2. Instance-based Transfer: TrAdaBoost and TransferBoost

The foremost boosting based algorithm that was proposed for the purpose of transfer learning is Dai et al.'s *TrAdaBoost* [38]. TrAdaBoost considers the target and source data separately by applying different weight update schemes on them. The weights of misclassified target data points are increased, as in AdaBoost, using the weight update factor $\alpha_t$ computed from the error $\epsilon_t$ over the target data. The weights of source data points are, however, decreased, similar to the weight-

Table 2 : TrAdaBoost algorithm from [38]

**Algorithm** TrAdaBoost

**Input**: the two labeled datasets $T_d$ and $T_s$, the unlabelled dataset $S$, a base learning algorithm Weak Learner, the maximum number of iterations, C, a vector consisting of cost factors associated with every sample in $T_d$.

**Initialize**: the initial weight vector $w^1$=$\{w_1^1, w_2^1, \cdots, w_{n+m}^1\}$.

For $t = 1, \ldots, N$

1. Set $p^t = {w^t}/{\sum_{i=1}^{m+n} w^t(i)}$
2. Call the weak learner, providing it with $p^t$ and the combined training set of $T_d$ and $T_s$ along with the cost factors for $T_d$ C. Get back the hypothesis $h_t : X \to Y; Y \in \{-1, 1\}$.
3. Calculate the weighted error of $h_t$ on $T_s$
$$\varepsilon_t = \sum_{i=m+1}^{m+n} \frac{w^t(i) \cdot |h_t(x_i) - y_i|}{\sum_{j=m+1}^{m+n} w^t(j)}$$
4. Set $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$ and $\beta = 1/(1 + \sqrt{2 \log n / N})$. Note that $\varepsilon_t$has to be less than $1/2$
5. Update the new weight vector
$$w^{t+1}(i) = \begin{cases} w_i^t \beta^{|h^t(x_i) - y_i|}, & 1 \le i \le n \\ w_i^t \beta_t^{-|h^t(x_i) - y_i|}, & n+1 \le i \le m+n \end{cases}$$

Output: The final hypothesis
$$h_f(x) = \begin{cases} 1, & \prod_{t=N/2}^{N} \beta_t^{-h_t(x)} \ge \prod_{t=N/2}^{N} \beta_t^{-1/2} \\ 0, \text{Otherwise} \end{cases}$$

ed majority algorithm, using a constant factor $\alpha$ that has been set according to Littlestone and Warmuth [39]. The concept of similarity for transfer is implicit and assumes misclassified source instances to be the most dissimilar to the target instances. The weights of the misclassified source instances are decreased to weaken their impact on the weak learner at a given iteration. Thus, source domain

**Algorithm 3.2** TransferBoost

**Input:** source tasks $S_1, \ldots, S_k$, where $S_i = \{(x_j, y_j)\}_{j=1}^{|S_i|}$,
target training examples $T = \{(x_t, y_t)\}_{t=1}^{n}$, and
the number of iterations $K$.

1: Merge the source and target training data
$D = S_1 \bigcup \ldots \bigcup S_k \bigcup T$.

2: Initialize $w_1(x_i) = 1/|D|$, for $(x_i, y_i) \in D$. Let $\mathbf{w}_1(D)$ be the weight vector for all instances in $D$. (Optionally, these initial weights could be specified by the user.)

3: **for** $t = 1, \ldots, K$ **do**

4: Train model $h_t : X \rightarrow Y$ on $D$ with weights $\mathbf{w}_t(D)$.

5: **for** $i = 1, \ldots, k$ **do**

6: Choose $\alpha_t^i \in \mathbb{R}$.

7: **end for**

8: Choose $\beta_t \in \mathbb{R}$.

9: Update the weights for all $(x_j, y_j) \in D$:

$$w_{t+1}(x_j) = \begin{cases} \frac{1}{Z_t} w_t(x_j) \exp\left(-\beta_t y_j h_t(x_j) + \alpha_t^i\right) & (x_j, y_j) \in S_i \\ \frac{1}{Z_t} w_t(x_j) \exp\left(-\beta_t y_j h_t(x_j)\right) & (x_j, y_j) \in T \end{cases},$$

where $Z_t$ normalizes $\mathbf{w}_{t+1}(D)$ to be a distribution.

10: **end for**

**Output:** the hypothesis

$$H(x) = sign\left(\sum_{t=1}^{K} \beta_t h_t(x)\right) .$$

Figure 3: TransferBoost Algorithm from [15]

instances that are similar to the target domain instances will have large training weights, while the source domain instances that are not so similar will have lower weights. Dai et al. provide a theoretical analysis of the algorithm and derive the training and generalization error bounds and show that the average weighted training loss on the source data converges to zero from the $\left\lceil \frac{T}{2} \right\rceil^{th}$ to the $T^{th}$ iteration. Hence, only the weak hypotheses between the $\left\lceil \frac{T}{2} \right\rceil^{th}$ and the $T^{th}$ iterations are linearly combined to obtain the strong hypothesis.

*TransferBoost* [15] uses a hierarchical weight updating scheme which boosts both individual instances and a set of instances corresponding to a source task. Sufficient information is assumed to be available from a collection of source tasks, $S_1, \ldots, S_k$, each characterized by different $\mathcal{X} \rightarrow \mathcal{Y}$ mappings. On boosting iteration $t$, each source task $S_i$ is assigned a weight $\alpha_t^i$ based on a notion of transferability from the source task to the target task. These weights denote the contribution made by each source tasks to learn a target tasks based their relatedness. Transferability, as previously mentioned, is essentially a greedy measure, defined as the change in classification performance on the target task between learning with and without transfer. To compute transferability, a classifier $\widehat{h_t}$is first trained on the target data $T$ with distribution $\frac{\boldsymbol{w}(T)}{||\boldsymbol{w}(T)||_1}$. Another classifier $\widetilde{h_t^i}$ is then trained on $S_i \cup T$ with distribution $\frac{\boldsymbol{w}(S_i \cup T)}{||\boldsymbol{w}(S_i \cup T)||_1}$. Based on the individual performance of these classifiers, transferability is given by $\alpha_t^i = \widehat{\epsilon_t} - \widetilde{\epsilon_t^i}$, where $\epsilon$ is the weighted error of classifier $h$ on $T$. The weighting scheme for individual instances follows from AdaBoost, increasing the weights of misclassified instances disregarding whether they belong to the source or the target domain. The algorithm is given in Figure 3.

### 3.3. Feature-based Transfer: Joint Boosting

Torralba et al., [40] present a multi-class boosting procedure, used for object recognition in images, which learns an array of strong classifiers $(x, c)$ , that can classify different object classes $c \in \mathcal{C}$ by finding a shared feature space for the

classes, instead of separately training binary classifiers. At each boosting round, various subsets $S \subseteq \mathcal{C}$ of classes are examined and a weak classifier is learnt to distinguish the subset from the background. The subset learner that maximally reduces the weighted error on the training set for all the classes is added to the strong learner for that class. Instead of iterating through an exhaustive list of $(2^{|\mathcal{C}|} - 1)$ subsets, the authors use forward selection of the best features for recognizing a class. By using a decision stump for a weak learner, which can be viewed as a feature selection process, the algorithm, in a way, becomes equivalent to functioning within a manifold. The transfer here is probably not so obvious as the other algorithms, and can be readily seen as related to multitask learning when each object class is considered as a task.

### 3.4. Model-based Transfer: TaskTrAdaBoost

Yao and Doretto [41] extend the boosting algorithm to transfer knowledge from multiple source tasks to learn a specific target task. With the assumption that closely related tasks are likely to share some parameters, the framework works on transferring suitable parameters from multiple source tasks to a target task in two phases. In the first phase, standard AdaBoost is used to learn the each source task and obtain a collection of candidate weak classifiers $\mathcal{H}$. A regularizing threshold $\gamma$ is utilized to constraint the coefficient $\alpha$ to selecting the best of weak classifiers to be included in the set. In the second phase, another boosting algorithm is run to select the best of the weak classifiers in the set $\mathcal{H}$, with respect to the target data. At each round of the boosting iterations, a weak classifier $h \in \mathcal{H}$ is chosen such

that it gives the lowest weighted error on the target training data, ensuring the harder examples are learnt. Intuitively, the strong classifier can be seen as a linear combination of selected source task classifiers. In some ways, this method is remarkably similar to the Locally Weighted Ensemble algorithm [31] mentioned in the previous chapter.

# CHAPTER 4

# COST-SENSITIVE BOOSTING

This chapter presents different schemes for employing a cost-sensitive framework for transfer learning, by extending the original AdaBoost [37] framework proposed by Freund and Schapire. In the proposed extension, the boosting framework is applied separately to source and target domain data. The boosting updates for the source domain data is modified to take into account the cost factors that represent the relevance of the source domain samples with respect to target domain data. This ensures that weights of instances in source domain data that are not relevant to target domain data are slowly decreased to reduce its impact on learning, while maintaining the weights of the relevant samples.

## 4.1. Notation

Formally, the labeled source and target training data samples are referred to as *diff-distribution and same-distribution* training data, following the notation in [38], while underlining the difference in them. Thus, let $T_d = \{(x_i^d, y_i^d)\}_{i=1}^n$ represent diff-distribution samples, $T_s = \{(x_i^s, y_i^s)\}_{i=1}^m$represent the same-distribution training samples and $S$ refer to the set of unlabeled test data taken from the target domain. The objective of the algorithm is to learn a target classifier (Figure 4) that classifies the test data $S$ with minimum error, by training on the same-distribution dataset $T_s$ supplemented by the relevant instances picked out from the diff-distribution source dataset $T_d$. The approach for solving the problem is centered around two main heuristics namely (1) attaching weights or

cost items to based on the estimated relevance with and (2) applying separate boosting schemes on and .

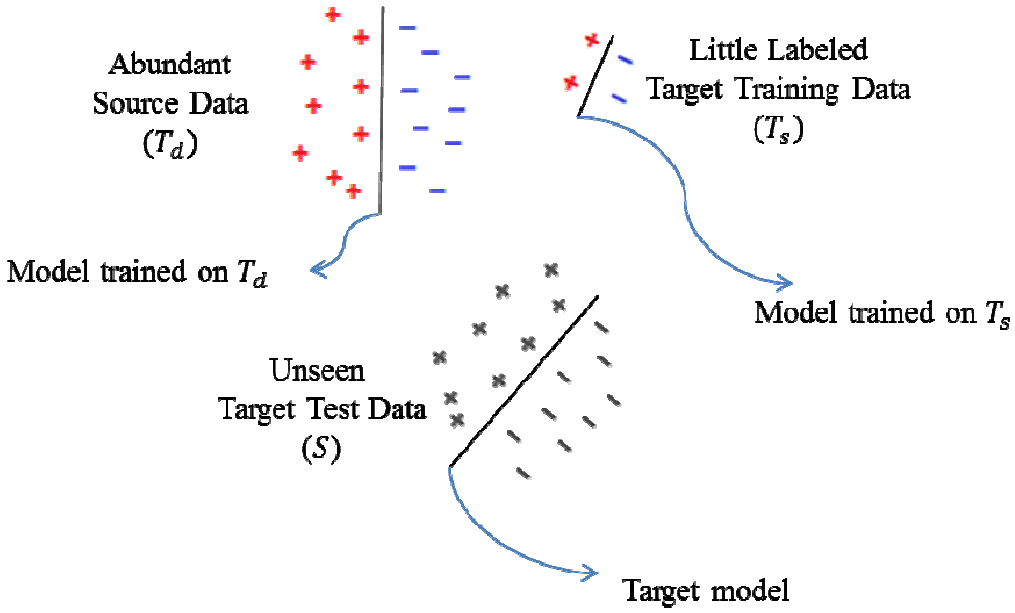



Figure 4 An illustration of the problem statement

### 4.2. Cost-Sensitive Boosting Framework for Transfer Learning

The Cost-Sensitive Boosting framework is a result of an empirical exploration of building a fast instance-based transfer algorithm on top of AdaBoost [37], for learning a task observed in a target domain with the aid of a small set of labeled target training instances and sufficient set of labeled source training instances . The algorithm design is based on three straightforward principles:

- to compute source relevance, right at the instance-level (rather than a task-level) and to keep it independent from the boosting algorithm (against encapsulating it with the boosting algorithm),

- to keep the interests of the boosting schemes applied over same-distribution data and the diff-distribution data separate (instead of having a unified boosting scheme) and
- to train a common weak learner, on each boosting iteration, from a sample set of the most relevant and hard to learn instances for that iteration (target instances are relevant by default).

The algorithm attaches a relevance indicating cost factor on to every instance in $T_d$, determining how useful learning from that instance would be. This is motivated by an intuition that, given the cost of misclassifying a source domain instance that holds a good probability of occurring as part of a target task, an existing robust target classifier model would be expected to perform with a minimal classification error over the target domain data, and at the same time manage to classify the source domain data with a reduced net cost of misclassification. Framing such a dual objective helps reducing the chances of

**Algorithm 1** Cost-sensitive Boosting for Concept Drift

**Given:** Labeled datasets $T_d$ and $T_s$ and the number of iterations $T$.

1. Compute a cost item $C_i \in [0,1]$ for each instance $(x_i^d, y_i^d) \in T_d$.
2. Initialize weight vector $D^1(x^{s,d}) = 1/(n+m)$.
3. For $t = 1, ..., T$:
   4. Train base learner using distribution $D^t$.
   5. Obtain hypothesis $h_t : X \to Y; Y \in \{-1, 1\}$.
   6. Calculate weighted errors, $\epsilon_t^s$ and $\epsilon_t^d$ on $T_s$ and $T_d$ respectively.
   7. Choose $\alpha_t^d$ (refer Table 1).
   8. Update weight vectors $D^{t+1}(x^d)$ augmented by the cost items $C_i$ (refer Table 1).
   9. Set $\alpha_t^s = \frac{1}{2} log(\frac{1-\epsilon_t^s}{\epsilon_t^s})$.
   10. Update weight vectors $D^{t+1}(x_j^s) = \frac{D^t(x_j^s) exp(-\alpha_t^s h_t(x_j^s) y_j^s)}{Z_t}$, where $Z_t$ normalizes $D^{t+1}$ to a distribution.

**Output:** the hypothesis $H(x) = sign(\sum_{t=1}^{T} \alpha_t^s h_t(x))$.

Figure 5 Generalized Cost-Sensitive Boosting Algorithm

learning a classifier that may over fit over $T_s$, considering that it is made up of far few target samples than required for generalization.

A formal description of the framework is presented in Algorithm described in Figure 5 . As seen in the algorithm, regular AdaBoost is directly used for minimizing the training error over $T_s$. The boosting coefficients $\alpha_t^s$ allow the weak learners to focus on the hard target domain examples. On the other hand, the samples in $T_d$ are weighted based on the influence they have in predicting instances in the target domain. This boosting scheme combines a prior cost factor with the classification error based weight update factor, to give a different boosting coefficient $\alpha_t^d$ for every chosen instance in the source domain. Thus, for a given iteration, if an instance that is considered to be irrelevant is misclassified, the factor by which the instance weights are increased is significantly low when compared to that of a more relevant misclassified instance. Similarly, the weight of a correctly classified source domain sample with low relevance is made lower than a correctly classified source domain sample with a higher cost. Thus, the weight update curves are more gradual and hence, natural based on the relevance of a source domain instance.

The boosting schemes for samples in $T_d$ are responsible for the actual knowledge transfer and have been adapted from the cost-sensitive boosting framework proposed by Sun et al. for dealing with imbalanced classes [42]. The proposed approach in this thesis, in contrast to the boosting framework described in [42], applies cost-sensitivity selectively to samples in $T_d$ alone. The weight update coefficient $\alpha_t^d$ is derived using one of the three algorithm schemes namely

Table 3: Weight update equations for the different Boosting schemes [42]

| Algorithm | $\alpha_t^d$ | $D^{t+1}(x_i^d)$ |
|---|---|---|
| *AdaC1* | $\frac{1}{2}log\frac{1+\sum_t y_t^d h_t(x_t^d)C_t D^t(x_t^d)}{1-\sum_t y_t^d h_t(x_t^d)C_t D^t(x_t^d)}$ | $\frac{D^t(x_t^d)exp(-\alpha_t^d C_t h_t(x_t^d)y_t^d)}{Z_t}$ |
| *AdaC2* | $\frac{1}{2}log\frac{\sum_{t,y_t^d=h_t(x_t^d)} C_t D^t(x_t^d)}{\sum_{t,y_t^d\neq h_t(x_t^d)} C_t D^t(x_t^d)}$ | $\frac{C_t D^t(x_t^d)exp(-\alpha_t^d h_t(x_t^d)y_t^d)}{Z_t}$ |
| *AdaC3* | $\frac{1}{2}log\frac{\sum_t C_t D^t(x_t^d)+\sum_t y_t^d h_t(x_t^d)C_t^2 D^t(x_t^d)}{\sum_t C_t D^t(x_t^d)-\sum_t y_t^d h_t(x_t^d)C_t^2 D^t(x_t^d)}$ | $\frac{C_t D^t(x_t^d)exp(-\alpha_t^d C_t h_t(x_t^d)y_t^d)}{Z_t}$ |

*AdaC1*, *AdaC2* and *AdaC3* summarized in Table 33. A brief analysis on the impact of these different weight update equations is given below:

- *AdaC1*: the weights of the $T_d$ samples that are incorrectly classified are reduced by a factor of $exp(C)$. Among these, samples that have higher relevance as indicated by the cost, tend to be reduced by a smaller amount compared to samples with lower cost. However, the difference is expressed in exponential terms.
- *AdaC2:* the weight updates are impacted directly by the cost factor. Thus the weight change is directly related to the relevance of the sample. Even though the weights of the samples in $T_d$ decrease over iterations, the change in the weight is conservative for samples that are more relevant in comparison to samples that are less relevant. This is the weight update model described in the algorithm.
- *AdaC3:* the sample weights are updated by the combinational results of AdaC1 and AdaC2. Due to the complicated situation of training error and cost setups, it is difficult to decide how AdaC3 changes the weights for samples in $T_d$ according to the cost factors.

The next subsection elaborates on the approaches adopted in this work for estimating costs.

### 4.3. Cost Estimation

The role of the cost items is primarily to associate the relevance of source domain samples $T_d$ with respect to the target domain samples $T_s$. This can be computed in a supervised or unsupervised manner. We select techniques from both of these approaches to study their impact on the end result. Following are the different approaches employed in this work:

- **Instance Pruning based cost estimate (IP):** This is a supervised approach and follows the technique proposed by Jiang et al., [19] for pruning misleading different domain instances. Their approach involves learning a classifier model using the few labeled target domain sample set $T_s$ and using this model to select instances from the source domain that are correctly classified. Instead of eliminating all the instances that are incorrectly classified, we use the probability of correct classification associated with each sample as the cost factor. Thus in the process, samples in $T_d$ with high probability of correct classification have higher cost items compared to the samples in $T_d$ with low probability of correct classification. Since the estimated values are probabilities, the cost thus computed is already normalized between $[0, 1]$. Since we have primarily used SVM as our base classifier, we have adopted the probability estimation as proposed by [19] for determining the costs of samples in $T_d$. For a binary class problem, a simple

Platt's scaling through logistic sigmoid function is used. The parameters of this function are learnt from the classification margins of the training sample. For the multi-class scenario, we use a one vs one multi-class SVM that learns the decision boundaries between all pairs of classes. The margins from each of these classifiers are converted to probability values using Platt's scaling. In the second step, an optimization procedure is employed to learn the true classification probabilities from these pair wise probabilities. This technique is implemented in the popularly used LIBSVM package [43].

- **Relevance Measure based on a distance metric(ED):** This too is a supervised technique and consists of two steps. In the first step, the pair-wise distance between all the samples (source and target domain) are computed. In the second step, a ratio between the sums of the distances of the $i^{th}$ instance in $T_d$ from all the samples in $T_s$ that belong to different and same class respectively.

$$c_i = \frac{dist(x_i^d, x_j^s)}{dist(x_i^d, x_j^s)}$$

This measure can be considered as a relevance measure. It measures how similar a sample from a particular class is with respect to a target set. Different distance metrics can be employed depending on the dataset. For the experiments conducted in this work, we used a Euclidean distance on datasets of low dimension and the cosine distance for datasets with high dimensions.

- **KLIEP based cost estimate:** Kullback-Liebler Importance Estimation Procedure [24] is a technique that is used for transductive instance transfer

learning that involves estimation of weights for source domain samples through the minimization of the KL-divergence measure between the probability densities of the source and target domain data. The basic idea behind the technique is to compute an importance estimate (that is considered as the cost factor by our algorithm) such that the KL-divergence from the true test input density to its estimate is minimized. The algorithm carries out this minimization without explicitly modeling the training and test data densities. The optimization problem for KLIEP is convex, so the true global solution can be obtained. A cross validation approach is typically used for model selection process of the minimization procedure in KLIEP. A Gaussian kernel is used during the minimization procedure.

- **Concept Feature Vector Distance(CFVD)**: Concept feature vector is a term that is used in the context of detecting concept drift in a data stream. The sequential data is divided into batches. Concept feature vectors describing the data in each batch are then determined. The distance between concept feature vectors of consecutive "batches" of data is calculated. Concept drift is detected if this distance is greater than a certain threshold. In the current context of determining the similarity between the source and target domain, we define the batches to be the source and target domain data itself. Concept feature vectors are determined for samples belonging to a particular class of the source and target domain respectively. The distance between these concept feature vectors of the source and target domain is treated as the cost of the source domain samples. Formally, let $X_s^i$ and $X_t^i$ be the source and target

samples belonging to class $i$. Then the concept feature vector for source set is defined as

$$C_s^i = \frac{1}{|X_s^i|} \sum_{k=1}^{X_s^i} x_{sk}^i$$

The concept feature vector for class of the target domain $C_t^i$ also follows similarly. The concept feature vector distance is the distance between $C_s^i$ and $C_t^i$. These distances are calculated for each of the classes separately, which is then normalized. The cost factors associated with every sample belonging to a particular class is then determined as difference between 1 and the normalized concept feature vector distance associated with that class. Note that the cost factors of all the samples belonging to a particular class will be identical.

$$c^i = 1 - \frac{dist(C_s^i - C_t^i)}{\sum_j dist(C_s^j - C_t^j)}$$

Thus the cost of samples belonging to "very different" source and target domain class will be lower than the samples belonging to "similar" source and target domain classes. We use Euclidean distance metric to compute the distance between the concept feature vectors for datasets with low feature dimension and use cosine distance metric for datasets with very large feature dimension.

### 4.4. Dynamic Cost Update

The cost factors associated with $T_d$ can be static, in the sense remain the same across boosting iterations. However during the course of the boosting process, the weights to samples $T_s$ change. Thus, at certain iteration it is possible to observe a

higher weight to a subset of samples in $T_s$. This essentially means that the algorithm is finding it difficult to learn this particular subset of $T_s$. Thus, the distribution of samples in $T_s$ changes, reflecting their ability to be learned. To take into account the changes in the weights of $T_s$ samples, we ensure that the $T_d$ cost factors are also updated. Thus, relevance to $T_d$ samples is determined based on harder $T_s$ samples.

A new SVM model is learned, at every iteration; from the target domain labeled samples drawn according to the distribution of these samples for that particular iteration. Samples in $T_s$ that have higher weights influence the decision boundary of this model. This SVM model is then used to classify all the source domain samples. The hypothesis is that the newly computed cost factors of the $T_d$ samples reflects the importance of these samples with respect to the new distribution of the $T_s$ samples.

### 4.5. Comparing with other boosting based transfer-learning approaches

While this is not the first boosting approach for transfer learning, a discussion centered on the similarity between the proposed approach presented in this work and other boosting based transfer-learning approaches. The most commonly cited boosting based transfer learning approach is the TrAdaBoost algorithm of Dai et al., [38]. Both, Cost-sensitive boosting and TrAdaBoost, employ the original boosting based approach for updating the weights of labeled target domain samples. The main difference between the two algorithms is in the manner in which the source domain samples are handled. TrAdaBoost uses the weighted

majority algorithm to adjust the weights, repeatedly decreasing the weight of incorrectly predicted source domain sample by a constant factor

$\beta = 1/\left(1 + \sqrt{2\ln\frac{n}{N}}\right)$ where, $\alpha = -\log\left(\frac{1}{\beta}\right)$. It also notes that since the error on the labeled target domain samples converges to 0 only after half of the total number of iterations, the TrAdaBoost algorithm considers only the weak hypothesis learned in the second half of the boosting iterations to arrive at the final strong classifier. Intuitively the weak hypothesis learned during the initial rounds of boosting fit a majority of data, with the focus on the harder examples during the later rounds. If the harder examples represent outliers in the $T_s$ data, then TrAdaBoost has a tendency to over fit the same-distribution training data.

Another important difference between TrAdaBoost and Cost-sensitive boosting is the manner in which weight updates are performed on the source domain training data. In TrAdaBoost, the weights of $T_d$ samples either decrease or remain constant between successive iterations. There is no way in which the weight of a relevant sample can be increased, once decreased during the previous rounds of boosting. When the weights of these relevant samples become very low, their influence on learning a good weak hypothesis becomes negligible. In contrast, the Cost-sensitive boosting algorithm allows for increase in weights for the target domain samples. However it ensures that weight increase is proportional to the relevance of the sample with respect to the majority of the labeled target domain samples. Thus during the later rounds of boosting, Cost-

sensitive boosting has a higher potential to retain relevant source domain samples for learning the weak hypothesis compared to TrAdaBoost.

The set based boosting for instance level transfer (TransferBoost) proposed by Eaton et al., [15] is another algorithm that uses boosting for transfer learning. Transferboost uses a set based weight-updating scheme. It breaks the source domain instances into task-based sets. Instead of updating the weights of individual instances, it updates the weights of instances in a set in a similar manner. The scheme adopted for updating the weights of the labeled target domain data is similar to the Cost-sensitive boosting algorithm. Furthermore, it can be noted that the weight update in TransferBoost for the source domain instances is a special case of the Cost-sensitive boosting algorithm. When the parameter $\alpha_t$ , in the TransferBoost algorithm that represents the transferability of set is made a constant at the individual instance level, then TransferBoost boils down to Cost-sensitive boosting.

## 4.6. Datasets

The proposed methodology was experimented on various real-world and synthetic datasets. Each of the chosen datasets have unique characteristics and can be described by properties such as number of instances, number of attributes, number of class labels and class imbalance. A brief description of each of the dataset is given below.

### *5.6.1. Activity Gesture Dataset (act-ges)*

The activity gesture dataset is a multiclass real-world dataset of motion data collected for learning to recognize the different gestures used in the activity of

"making a drink and drinking it". Each instance can be associated with one of the 5 class labels namely *pour, scoop, screw/unscrew cap, stir* and *lift to mouth.* The data was collected from two domains, over which, data and task distribution were observed to vary.

The first domain refers to data motion data collected from 5 users enacting out the different activity gestures with the help of dummy objects. The motion data was captured using three tri-axial accelerometers placed in the user's dominant wrist, elbow and non-dominant wrist respectively. Out of these only the data captured from the dominant wrist and elbow were retained for their discriminative properties. The activity gestures were enacted 20 times to be sufficient for training. The participants were given explicit instructions on how to perform the activity gesture. The data obtained was then manually segmented and annotated with the help of a synchronized video of the activities performed. The second domain corresponds to data captured similarly as described above, but in a more realistic setting. In this setting, 4 users were asked to make a glass of Gatorade and drink it, instead of enacting the different gestures using dummy objects. The entire activity was repeated 4 times by each user. The mock and realistic scenarios are taken as the source and target domain respectively in our experiments. A brief description for both the mock and realistic scenarios is presented in Table 4. The objective is to recognize gestures performed in a realistic scenario, using the data from the mock scenario with a small number of labeled samples from the realistic scenario.

Table 4 Description of the Activity Gestures

| Activity Gesture | Mock Scenario | Realistic Scenario |
|---|---|---|
| Pour | Take the glass that is full and pour its contents into the empty glass. Pour a small quantity every time. | Pour the water from the glass. |
| Scoop | Use a spoon to scoop contents from the glass that is full into the empty glass | Use two scoops of powder for making the drink. |
| Unscrew Cap | Unscrew the lid of the water bottle. Pause for a couple of seconds. Screw on the lid on the bottle | Open the powder drink jar, and close it after you finish using it |
| Stir | Take the spoon and stir the contents of the glass for 30 seconds | Ensure the powdered drink has dissolved by stirring the mixture |
| Lift to Mouth | Take an empty glass and pretend that you are drinking water from the glass by taking several short sips. | Drink the glass of beverage that was prepared. |

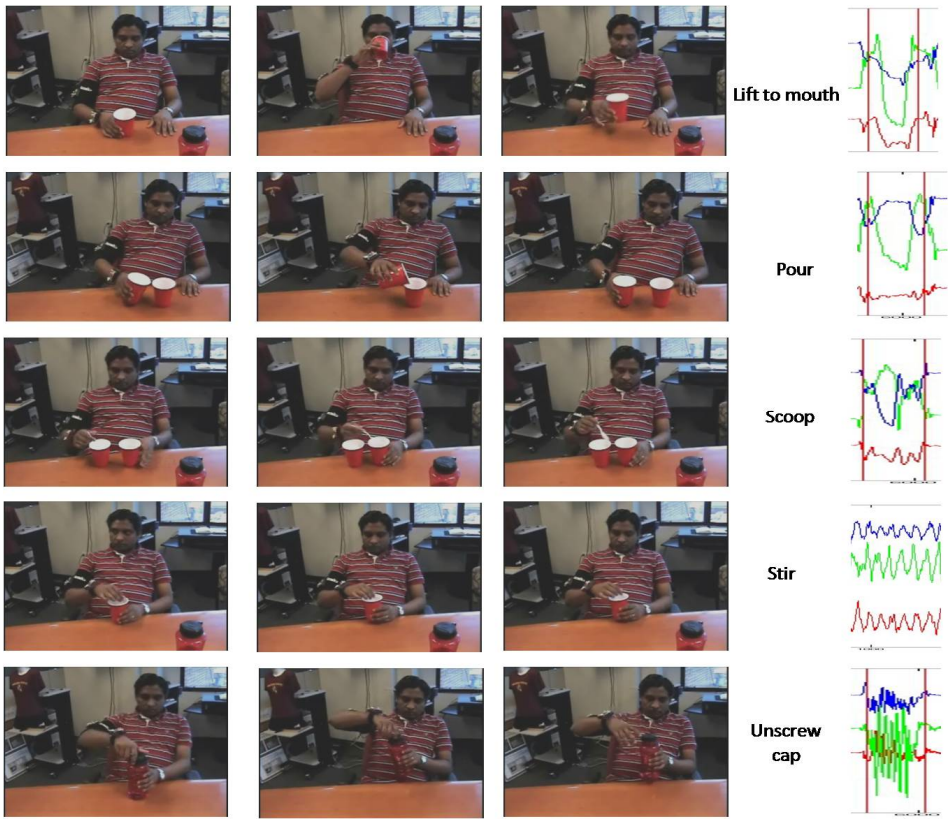



Figure 6: Capturing activity gesture datastreams

A distinct pattern can be spotted for each of the activity gestures in the data stream sample captured using the accelerometer placed on the wrist in Figure 6. Intuitively, the individual characteristics of each of the gestures can be tracked in the patterns seen. For example, the gesture unscrew cap can be defined by a number of rapid repetitive movements of the unscrew action with the initial effort to loosen the cap, while stir can be represented by a more relaxed and relatively slower set of mechanical movement. A dip in the z-axis acceleration appears for the gestures, scoop and lift to mouth, but the y-axis values increase for scoop and falls for lift to mouth. On the basis of these observations discriminative features that either aggregated over the temporal and frequency characteristics of each of

the axes or combined data across multiple axes as done by correlation coefficients, were extracted to obtain instance points in a 44-dimensional space.

### *5.6.2. WSU Smart Home Dataset[2] (act-rec)*

The WSU Smart Home dataset is a multi-class dataset that evaluates learning general models of activities by abstracting over different environments and residents. The dataset is being used in an on-going research project in the Center for Advanced Studies in Adaptive Systems (CASAS), Washington State University and has been collected from 7 smart environment testbeds, each consisting of a variety of sensors that include motion, door, temperature, light, item, etc., embedded on The dataset contains sensor events related to a set of eleven ADL activities namely - Cooking, Eating, Sleeping, Relaxing, Working, Bed-to-toilet, Enter Home, Leave Home, Taking Medicine, Personal Hygiene and Bathing. Pre-segmented sequences of sensor events corresponding to an activity were used to form a feature vector that represents the start and end time, duration, frequencies of different sensor firings within this duration and the preceding activity. All the sensor IDs were mapped onto labels corresponding to the room in which the sensor resided including: Kitchen, Kitchen Cabinet, Medicine Cabinet, Front Door, Lounge Chair Bedroom, Living Room, Dining Room, Bathroom, Hallway, Bathtub, etc., Each of the apartments had different layouts and the number of people and pets who resided in them also varied. The layouts of the different apartments and the different sensors present in them are shown in Figure 7. For the experiments conducted, the activity samples corresponding to one

[2] Most of the dataset is available at http://ailab.wsu.edu/casas/datasets.html


Figure 7: Sensor layout for the seven CASAS smart environment testbeds [49]

apartment was taken to be the target data while the samples from all the other

apartments were considered to be the source domain. Since data for all the activities was not present in every apartment, only those source domain activity samples whose labels were present in the target domain were used.

### *5.6.3. 20Newsgroups*

The 20 Newsgroups dataset is a collection of approximately 20,000 newsgroup documents, partitioned across 20 different newsgroups. It primarily consists of 7 top-level categories, with each category consisting of multiple sub-categories totaling to 20. Its hierarchical structure facilitates in modeling text categorization datasets into resembling transfer learning scenarios such as sample selection bias and has de facto become a dataset used prominently for comparative studies of transfer learning algorithms. For the experiments, two different sets of transfer learning datasets were extracted from the 20Newsgroups corpus (1) Newsgroups1 - one modeled for transfer from *single source task to a single target task*, and (2) Newsgroups2 - modeled for transfer from *multiple source tasks to a single target task*. Apart from this, experiments were also run on a readily available dataset extracted from the 20Newsgroups corpus, named Usenet1 that simulated class imbalance. Though the original corpus is a real-world dataset, the datasets worked with have all been simulated and can thus be considered to be synthetic, nevertheless applicable to a real-world scenario. All of the learning problems formulated on these datasets are binary classification problems. Some more information on the sub-category distributions and pre-processing steps are explained below.

• **Newsgroups1:** the task to be learnt is that of categorizing documents into two parent categories. The original feature space of the unprocessed dataset was reduced to that of 45000 by removing the common list of 526 stop words and having a document frequency threshold value of 2. The feature space was also converted into a binary space such that Data points in the source domain and target domain are drawn from different subcategories that belong to the parent category, thus inflicting a change in the space in which data is distributed. The division of the subcategories for the source and target domain is described. Six different binary class datasets were generated for the experiments based on these divisions. Due to computational constraints, we randomly pick 1000 samples for the source and target domain dataset.

• **Newsgroups2:** this dataset was created to evaluate the performance of the proposed approach in multi-source domain scenarios. The dataset is adapted from the work done by Eaton et al., For each domain, a set of binary task was generated to distinguish one class from a set of negative classes, ensuring that each task had unique negative examples and equal class priors. The first newsgroup in each major category was used as negative examples for the tasks given by the 13 remaining newsgroups. These negative examples are drawn from the following newsgroups: alt.atheism, comp.graphics, misc.forsale, rec.autos, sci,crypt. For each dataset, there was one target task and the other tasks from the same domain serve as the source tasks. The original 20newsgroup dataset was represented as a binary vector of the 100 most discriminating words determined by Weka's string

to work vector filter. Only 5% of the original dataset is used for the experiments, since the originals are very large.

**Usenet1:** This dataset [44] is based on the newsgroup collection. Simulated streams of messages from different newsgroups are sequentially presented to a user, who then labels them as interesting or junk according to his/her personal interests. The messages are presented to the user in batches. The user switches between his/her choices of junk for every batch. As a result, there is a complete reversal in the class labels as we move across each batch. The challenge here is to classify the user choices for a particular batch, using the training samples available from the previous batch and a few samples from the current batch. The description of the dataset in terms of its size and feature space is presented. We consider this to be hardest dataset, due to the reversal of labels across batches, which also leads to a class imbalance problem.

# CHAPTER 5

# RESULTS AND DISCUSSION

Various experiments were conducted on the datasets mentioned in the previous chapter to understand the properties of the data, evaluate the performance of the proposed approach under static and dynamic configurations, spot the effect of cost computation in knowledge transfer, empirically note the effect of multiple source transfer and make comparative studies with related algorithms. This chapter presents the results of these experiments and analyzes the various aspects observed during the study.

## 5.1. Properties of Data

Principle component analysis was used for visualizing and understanding the data and task distributions over the training and test domain of the act-gest and act-rec

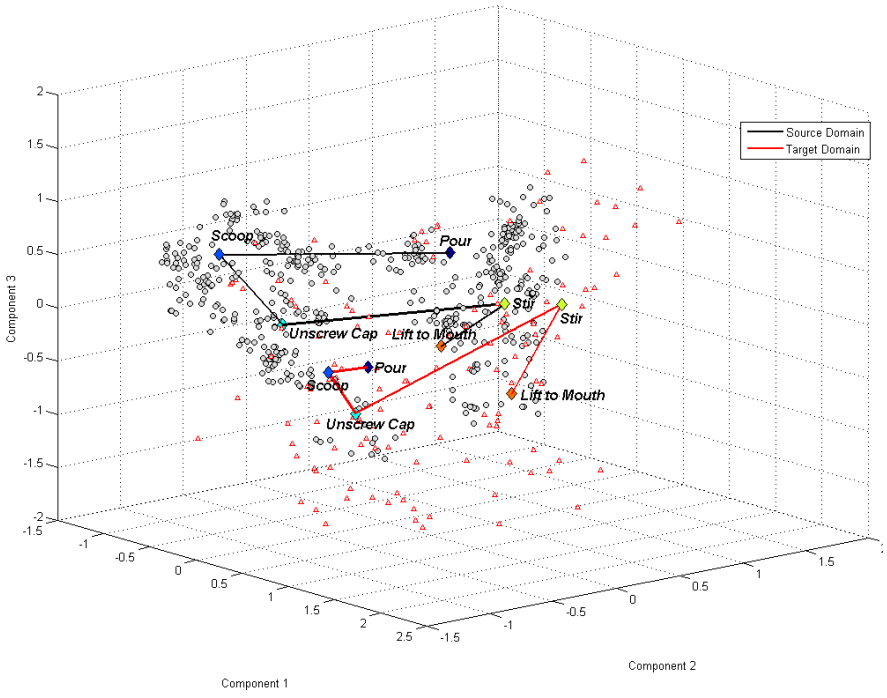



Figure 8: Source and target distribution of act-gest dataset

datasets. This exercise was done in order to establish an empirical basis for the appropriateness of applying transfer learning to a real-world problem and therefore does not distinguish between training and test data points. Data points belonging to both source and target domains were projected onto the space defined by the first three principle component vectors. Figures 8 and 9 plot the data points of act-gest and act-rec datasets respectively in their corresponding PCA vector space. Figure 8 uses all source and target instances in the act-gest dataset. The changes in task distributions between source and target domains were earlier illustrated in Figure 1 over two basis vectors. Here, the visualization presented indicates signs of the data suffering from *domain shift*, defined essentially as a change in the measurement system of the new data points $x_i$. A pretty much uniform and uni-directed translation of patterns is noticed between the source and target domain, plotted by a path connecting the centers of the cluster of points associated with the different activity labels *pour, scoop, unscrew cap, stir* and *lift-to-mouth*, in the source and target domains respectively. Among the five activity labels, the one that has a rather skewed displacement (not very noticeable in the Figure) is *pour*, which may be a result of latent and uncontrollable factors such as user traits, weight and shape of the container being held, etc.,

Figure 9 plots the data points of Apartment_B dataset, split into 8 different plots showing the shift in task distributions across the source and target domains. Act_rec_B was chosen as it contains data points associated with all 11 activities. Here only 8 of the 11 are chosen for illustration purposes. The plots capture

interesting patterns connected with the characteristics of the activities performed by different residents in smart home test-beds under different sensor settings and home layouts. Despite these differences, activities such as *eating*, *enter home* and *leave home* show little variations owing to the basic nature of these tasks. These activities cannot be performed very differently either by different residents or under different settings. Dataset shift cannot be always visualized in this manner particularly over datasets with high dimensionality. Sometimes, the change in the distribution may be perceivable only in higher dimensions (> 3). This is expectedly the case with the 20Newsgroups datasets, which possess high dimensions and are sparse, as they showed no reliable difference between the source and the target instances despite having synthetically made to have a change in the dataset distribution.

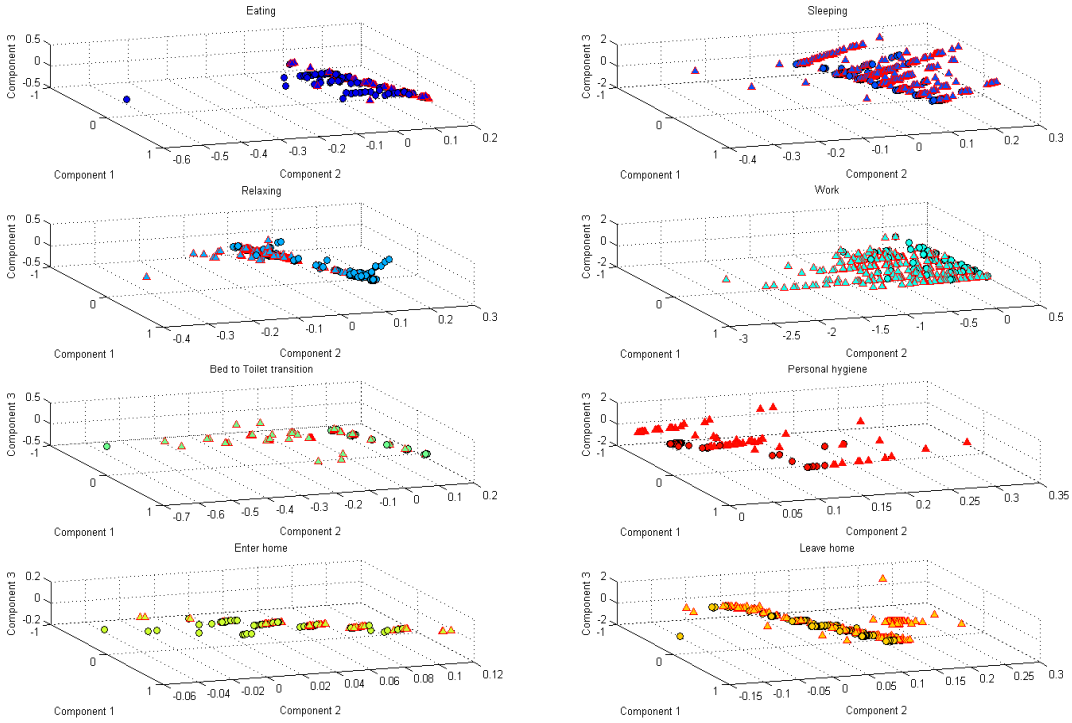



Figure 9: Source and Target data distribution of act_rec dataset

## 5.2. Performance Evaluation

All experiments were conducted using SVM as the base classifier. The toolbox by Chang and Lin [43] – LibSVM was used for conducting experiments with SVM. This toolbox supports multi-class SVMs through pairwise coupling. It constructs $k(k-1)/2$ number of binary classifiers for a $k$-class problem and combines the probability of classification obtained from each classifier through Platt's scaling to obtain the final probability of a sample belonging to a particular class. SVM was run using a linear kernel. The penalty factor C for act-gest and 20Newsgroups datasets it was set to 1 and for act_rec it was set to 100. The parameters were selected based on the best classification accuracy obtained from running a 5-fold cross validation over the source data. The maximum number of boosting iterations was set to 100 for all the boosting variants including AdaBoost, TrAdaBoost and the Cost-sensitive boosting framework. At every boosting round, the training dataset was sampled, such that all class labels were represented and uniformly distributed. Comparison of performance over the various experiments were done using an average generalized accuracy obtained over 5-fold cross validations. The various approaches were implemented using well integrated programs in MATLAB and will be shortly made available for public use.

### *5.6.4. Comparison of Classification Accuracies*

Table 5 : Comparison of Performance at 1% of the Target Training Data

| Dataset | Svm$T_s$ | Svm$T_d$ | Svm$T_{ds}$ | Ada | Trada | *Adac1* | *Adac2* | *Adac3* |
|---|---|---|---|---|---|---|---|---|
| User 1 | 0.77 | 0.56 | 0.79 | 0.85 | 0.82 | 0.85 | **0.88** | 0.85 |
| User 2 | 0.84 | 0.64 | **0.98** | 0.93 | **0.98** | 0.97 | **0.98** | **0.98** |
| User 3 | 0.54 | 0.33 | 0.71 | 0.67 | 0.65 | 0.70 | **0.75** | 0.74 |

| Dataset | Svm$T_s$ | Svm$T_d$ | Svm$T_{ds}$ | Ada | Trada | *Adac1* | *Adac2* | *Adac3* |
|---|---|---|---|---|---|---|---|---|
| User 4 | 0.44 | 0.61 | 0.77 | 0.73 | 0.75 | 0.76 | 0.79 | **0.80** |
| Apartment-A | 0.71 | 0.67 | 0.71 | 0.78 | 0.63 | 0.80 | **0.82** | 0.75 |
| Apartment-B | 0.67 | 0.62 | 0.68 | 0.72 | 0.57 | 0.79 | **0.80** | 0.76 |
| Apartment-C | 0.79 | 0.37 | 0.81 | 0.76 | 0.49 | 0.79 | **0.83** | 0.78 |
| Apartment-D | 0.76 | 0.34 | 0.77 | 0.82 | 0.52 | **0.83** | 0.81 | 0.81 |
| Apartment-E | 0.29 | 0.04 | 0.45 | 0.46 | **0.70** | 0.46 | 0.48 | 0.49 |
| Apartment-F | 0.58 | 0.20 | 0.60 | 0.62 | 0.40 | 0.67 | **0.68** | 0.67 |
| Apartment-G | 0.52 | 0.44 | 0.55 | 0.53 | 0.46 | **0.59** | **0.59** | 0.58 |
| Rec vs Talk | 0.68 | 0.72 | 0.75 | 0.72 | 0.73 | 0.71 | **0.83** | 0.72 |
| Rec vs Sci | 0.63 | 0.70 | 0.69 | 0.69 | 0.69 | 0.70 | **0.77** | 0.69 |
| Sci vs Talk | 0.60 | 0.64 | 0.67 | 0.64 | 0.70 | 0.67 | **0.74** | 0.68 |
| Comp vs Rec | 0.80 | 0.73 | 0.85 | 0.83 | 0.72 | 0.82 | **0.86** | 0.84 |
| Comp vs Sci | 0.62 | 0.64 | 0.67 | 0.68 | 0.58 | 0.69 | **0.76** | 0.69 |
| Comp Vs Talk | 0.86 | 0.68 | 0.87 | 0.87 | 0.73 | 0.88 | **0.89** | 0.88 |

Table 5 compares the classification accuracy given by algorithms when trained on 1% of the target training data and supplementary source data. In most cases, one of the three cost-sensitive boosting procedures is seen to perform better than the other algorithms, with *AdaC2* performing the best among the three and consistently better than TrAdaBoost. Some of the individual trends that arise owing to the properties of each dataset is further discussed below:

**Act_gest:** The difference in the performance of the three cost-sensitive boosting approaches looks marginal. Retrospectively, having assessed the results of all the datasets, a possible reason that can be associated with this trend is the relatively low number of test samples available for each of the user. It is interesting to note that, while Svm$T_s$ and Svm$T_d$ separately yield poor performances, just combining the source and target domain data results in an increased performance over the target domain. This can be attributed to two particular properties of this dataset.

The first one being that the data points belonging to each class label are well separable in their common feature space and respective domains. This has been pointed out previously in [6]. Secondly, since it is suspected that the dataset suffers from *domain shift* , the separability of data points facilitate knowledge transfer by either extrapolating or translating the source model in the feature space. Thus, an addition of labeled target data, as few as 1 instance per class, to the source data might improve the performance by a good deal. Nevertheless, the cost-sensitive boosting schemes, *AdaC2* and *AdaC3* further improve the performance over SVM$T_{ds}$.

**Act_rec**: Once again, the results obtained from these datasets also show just a marginal difference between the accuracies of *AdaC1*, *AdaC2* and *AdaC3*, with *AdaC2* giving the best performance most of the time. Though the cost-sensitive boosting algorithms have an upper hand in the performance in almost all the cases, the dataset pertaining to Apartment-E has TrAdaBoost giving an improvement of over 20 percentage points from the next best performing technique. It was noticed that the number of target domain training examples available for Apartment-E was very low (equivalent to having one instance per class) in contrast to the other apartments. This low sample size of $T_s$ could have resulted in incorrect cost estimation, leading to a poor performance of the cost-sensitive schemes. Unlike the act_gest dataset, no straightforward correlation is found to exist between the performance of SVM$T_{ds}$, SVM$T_s$ and SVM$T_d$. Furthermore, the performance of AdaBoost is noticed to be greater than that of SVM$T_{ds}$ indicating that the expected correlations may be captured by using a

boosting framework that focuses on a specific subset of samples during each iteration.

**20Newsgroups1:** Among the three datasets, it is only in this dataset that AdaC2 shows a significantly better performance among the three cost-sensitive boosting algorithms. In contrast to the observation made in act_rec datasets, it can be seen that adding a small amount of labeled target domain data to the source domain helps in improving the performance of the model that is trained only on the source data as inferred from the classification performances of SVM$T_d$ and SVM$T_{ds}$, indicating the complementary nature of the source data in learning the target tasks.

### *5.6.5. Advantage of AdaC2 over AdaC1 and AdaC3*

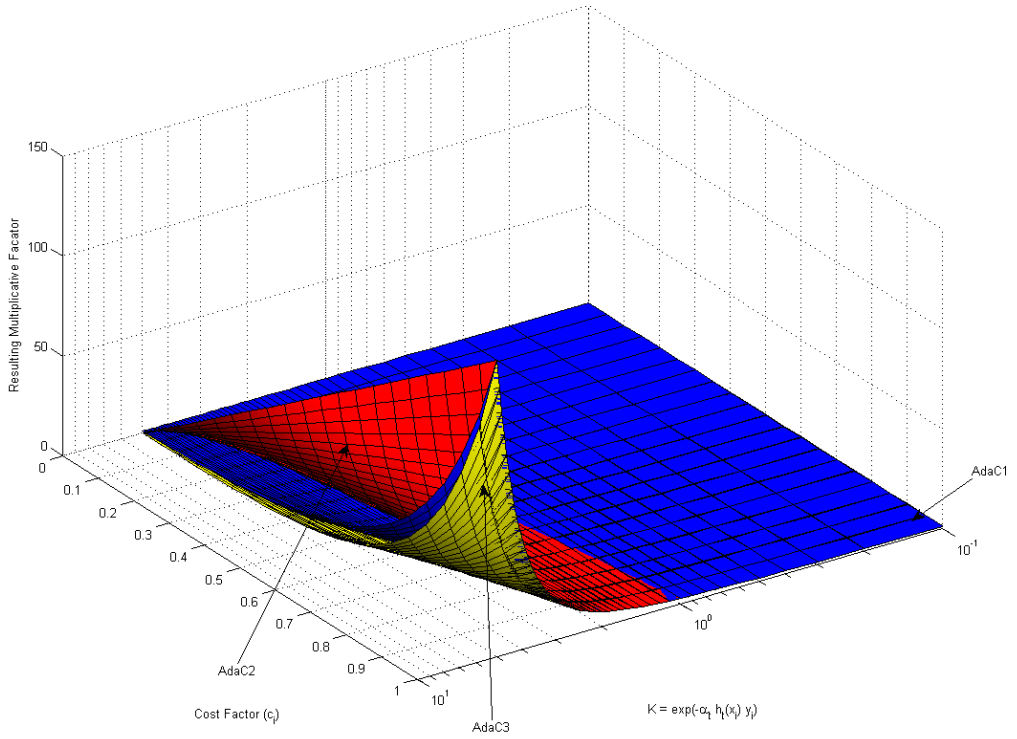
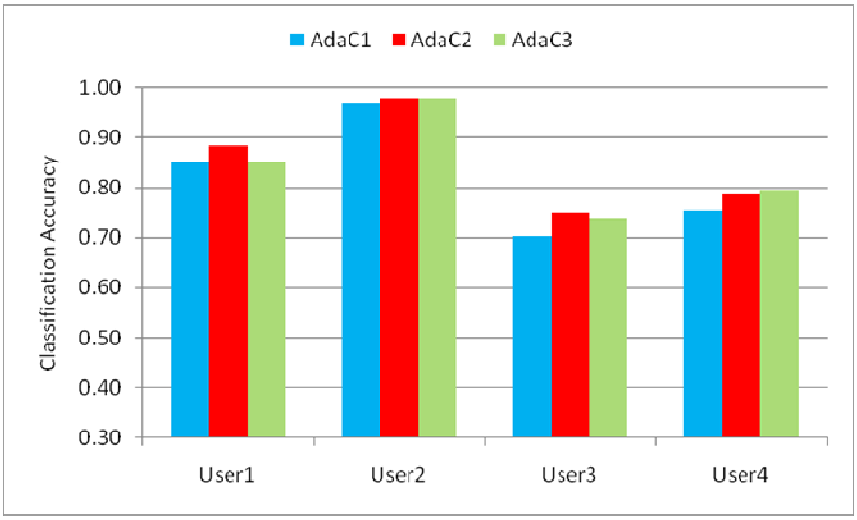


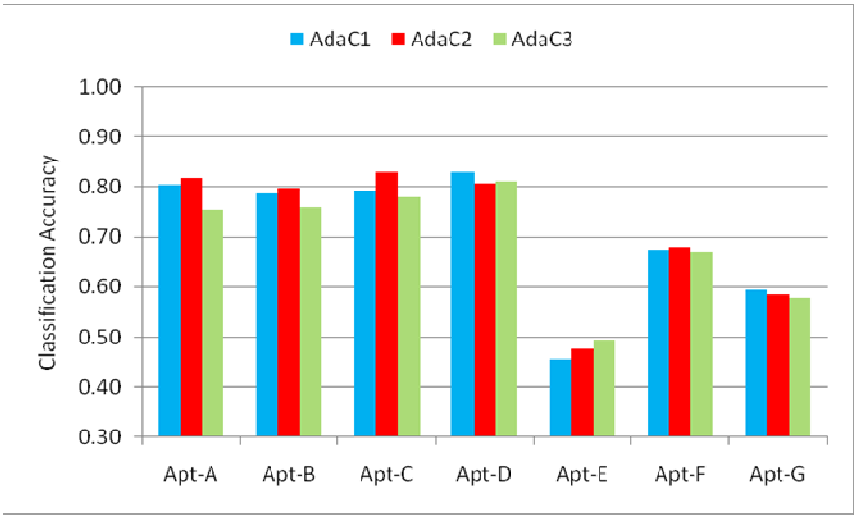

Figure 10 : Comparison of the weight update factors of AdaC1, AdaC2 and AdaC3

*AdaC1*, *AdaC2* and *AdaC3* all boost more weights on relevant samples that are misclassified than irrelevant samples that are misclassified. Similarly, they decrease weights more on relevant samples that are classified correctly than less relevant samples classified correctly. However, there is a marked difference in the way these weighting equations have an effect over a specific instance based on its relevance. Figure 10 pictorially shows the effect of *AdaC2* weighing each sample by its associated cost directly against *AdaC1* which attaches the cost factor in the exponential term thereby having a diminished role. The result is *AdaC1* ends up conserving the weights of less relevant samples and very reactive towards highly relevant samples. On the other hand *AdaC2* reacts to relevance in a smoother fashion thereby resulting in its tending towards conserving weights of relevant instances. Though effectively, *AdaC3* is a combinatorial result of *AdaC2* and *AdaC1*, the result of attaching the cost in the exponential component of the equation makes it act similar to *AdaC1* when the variable K in the graph is high and like *AdaC2* when K is low. More noticeably it acts closer to *AdaC1* when handling samples on the basis of its misclassifications. Figure 11 presents the plots obtained of the experimental results of the three algorithms on all the datasets. It is very evident that *AdaC2* has a much more effective influence on utilizing a cost measure for reliable knowledge transfer. From here, all further analyses would use only *AdaC2* for their study.

### *5.6.6. Correlation Between the Performances of $SVMT_d$, $SVMT_{ds}$ and AdaC2*

An interesting pattern that establishes the idea of relatedness emerges from the results obtained in Table 5 is of a subtle correlation between the classification



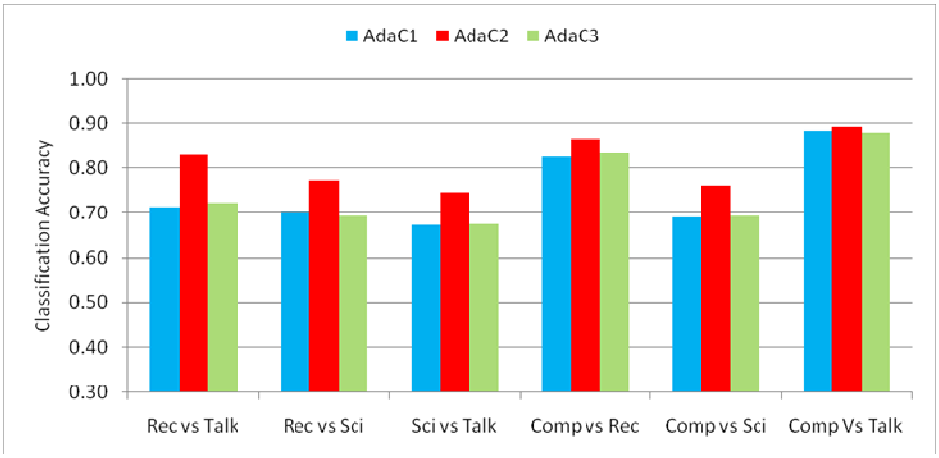
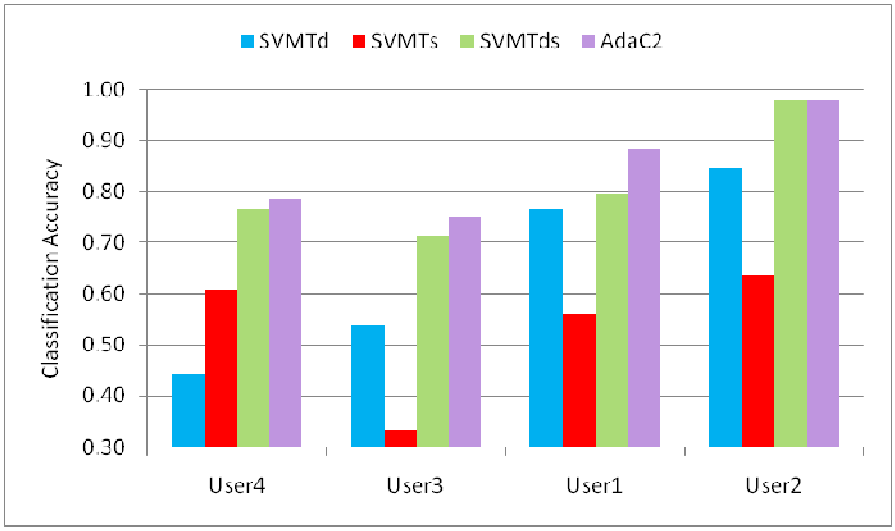
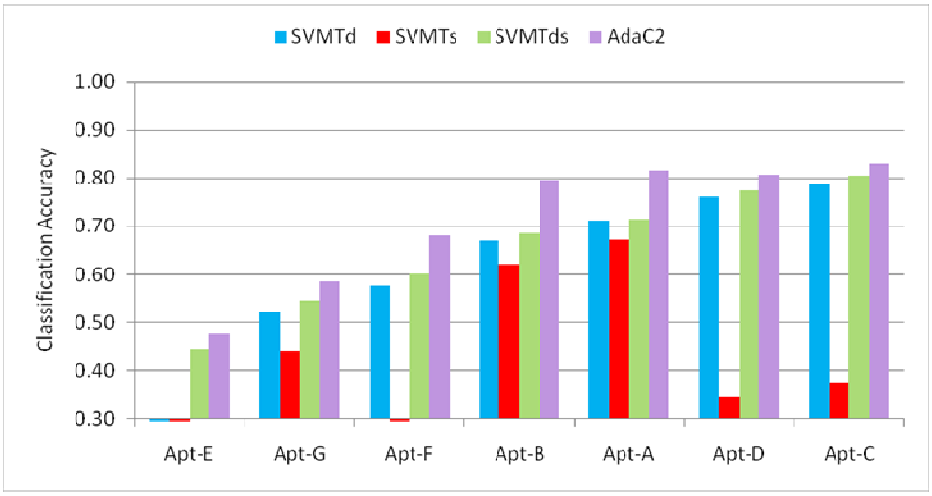




Figure 11 : Comparison of Cost-sensitive boosting results on (a) act_gest (b) act_rec (c) 20Newsgroups1 datasets, having trained on 1% of the target training data. AdaC2 can be observed to give better results among the three.

accuracies of baseline $SVMT_d$ and the improvement in performance shown by AdaC2 over $SVMT_{ds}$. The relationship noticed is that of an increase in the classification accuracies over SVM$T_{ds}$ when the source training data alone is able to classify target domain samples to a reasonable extent. For the purpose of illustrating this idea, plots were generated from the obtained results of SVM$T_d$, SVM$T_s$, SVM$T_{ds}$ and *AdaC2* by sorting the datasets in the increasing order of accuracies of SVM$T_d$ on the target dataset. This is shown in Figure 12. It can be observed that the highest difference between *AdaC2* and SVM$T_{ds}$ is obtained in the middle of this sequence. For the act_gest datasets, User 4 and User 2 show little to no improvement, while User 3 and User 1 show significant changes, while act_rec datasets, apartments F, B and A show improvement close to 10% points in accuracies while the other apartments seem to step up by around 4-5% points alone. Similarly, in the case of 20Newsgroups1, datasets Comp vs Rec and Comp vs Talk seem to give the best results over SVM$T_{ds}$.

The difference in the accuracies seem to follow a bell curve of sorts, with no or very low improvements for very similar and dissimilar task distributions, as inferable from the target domain accuracies of SVM$T_d$. This once again takes us back to the idea of "different, but related" datasets. If the target domain dataset was very similar or identical to the source domain, no transfer is needed and an improvement may not be expected out of transfer. On the other hand, if the target domain is vastly different from the source domain, then a transfer may not be possible and instead learning may need to be done from scratch. It should be noted here that though this pattern may be a useful indicator in deciding whether



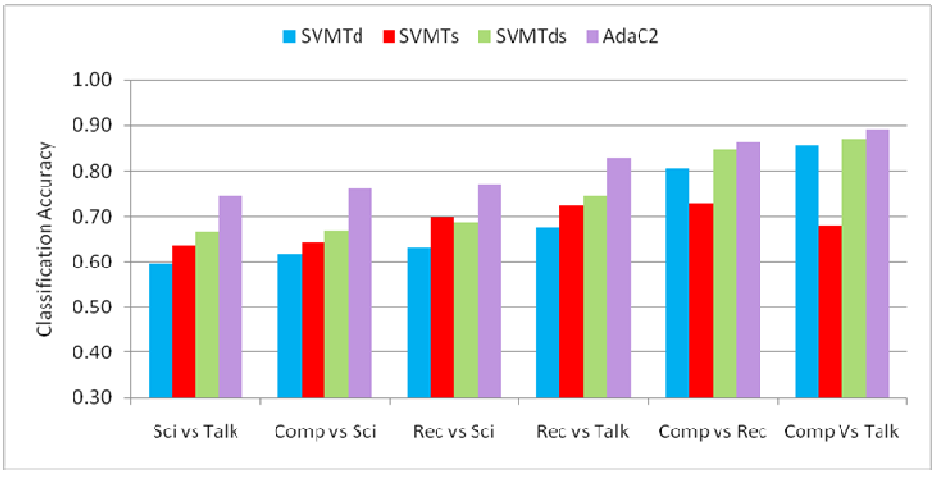




Figure 12 : Plots illustrating the correlation between the increase in performance of *AdaC2* over SVM$T_{ds}$ and SVM$T_{d}$.

"to transfer or not", the measure is still done only retrospectively and across different datsets, over a scale of relative similarity and dissimilarity

### *5.6.7. Classification Accuracy vs. Size of Target Training Data*

Figures 13(i) – 13(vi) illustrate the change in the accuracies across SVM$T_s$, SVM$T_{ds}$, AdaBoost, TrAdaBoost and *AdaC2* when 1%, 5% and 10% of the target training data is used for training on the act_rec datasets along with the auxiliary source domain data. When 5% of the target training examples are used, applying cost-sensitive boosting for transfer learning continues to be fruitful. However, the difference in the performance of *AdaC2* and SVM$T_d$ or *AdaC2* and SVM$T_s$ reduces with the increase in target training data, indicating that the target data is moving towards becoming sufficient for learning a reliable classifier without the need for auxiliary data.

It had been earlier mentioned, how 1% of the target training data proved to be far too insufficient for computing the cost of source instances in the case of Apartment-E, leading to a drop in the performance of *AdaC2*, while TrAdaBoost gave a good performance. On increasing the percentage of the available target training data to 5%, the performance of *AdaC2* is seen to improve. However, on a closer observation, the performance of *AdaC2* seems to correlate with that of SVM$T_s$. Comparatively, the increase in target training data does not seem to evoke an equally significant response from SVM$T_{ds}$. Given this, and the fact that SVM$T_d$ gives low accuracies for this apartment, the target domain clearly seems to be so much more different from the source domain. In this case TrAdaBoost

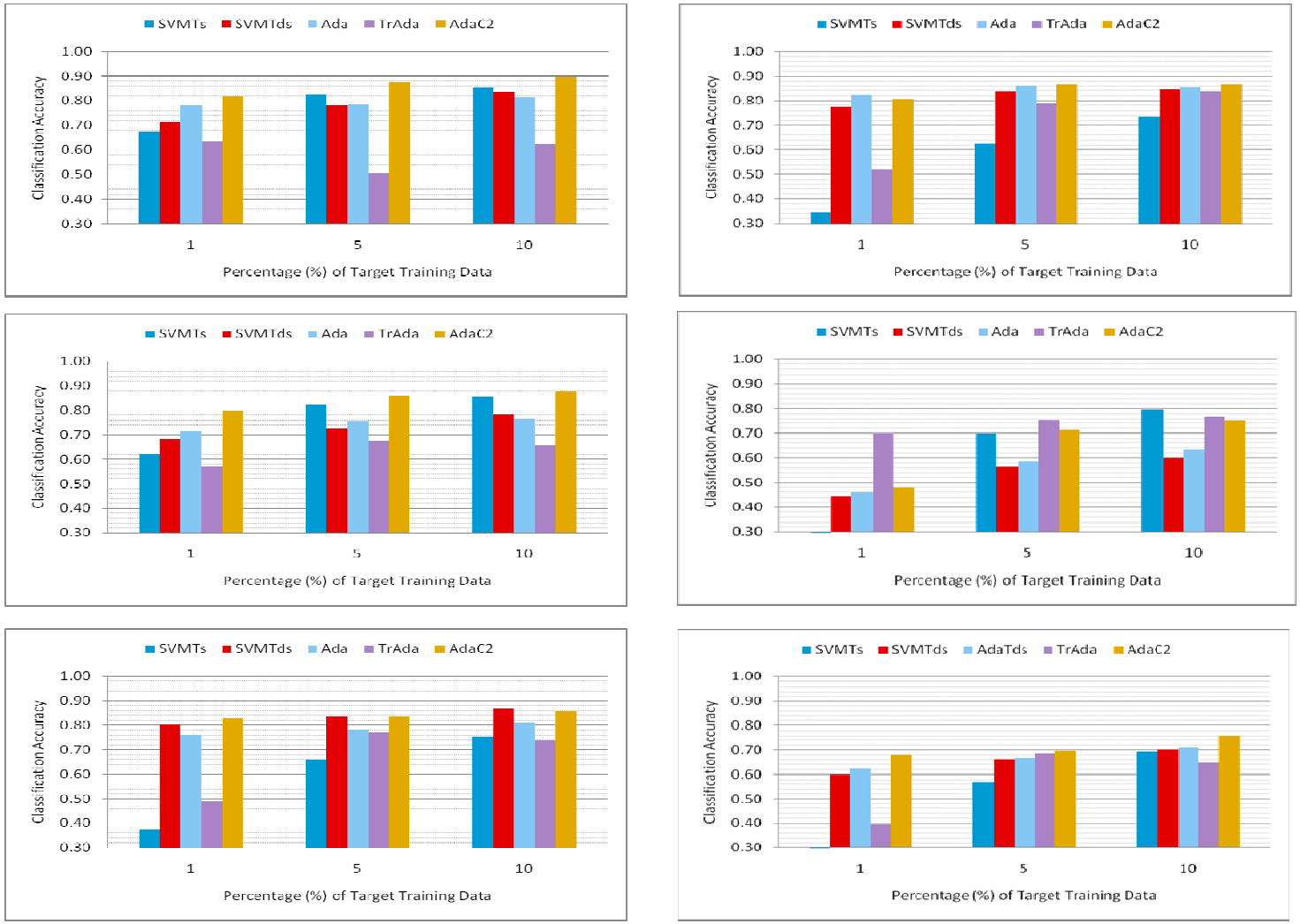



Figure 13 : Plots illustrating the variation in classification accuracies over act_rec dataset against 1%, 5% and 10% of Target Training Data. The datasets shown here include (i) Apt-A, (ii) Apt-B, (iii) Apt-C, (iv) Apt-D, (v) Apt-E and (vi) Apt-F.

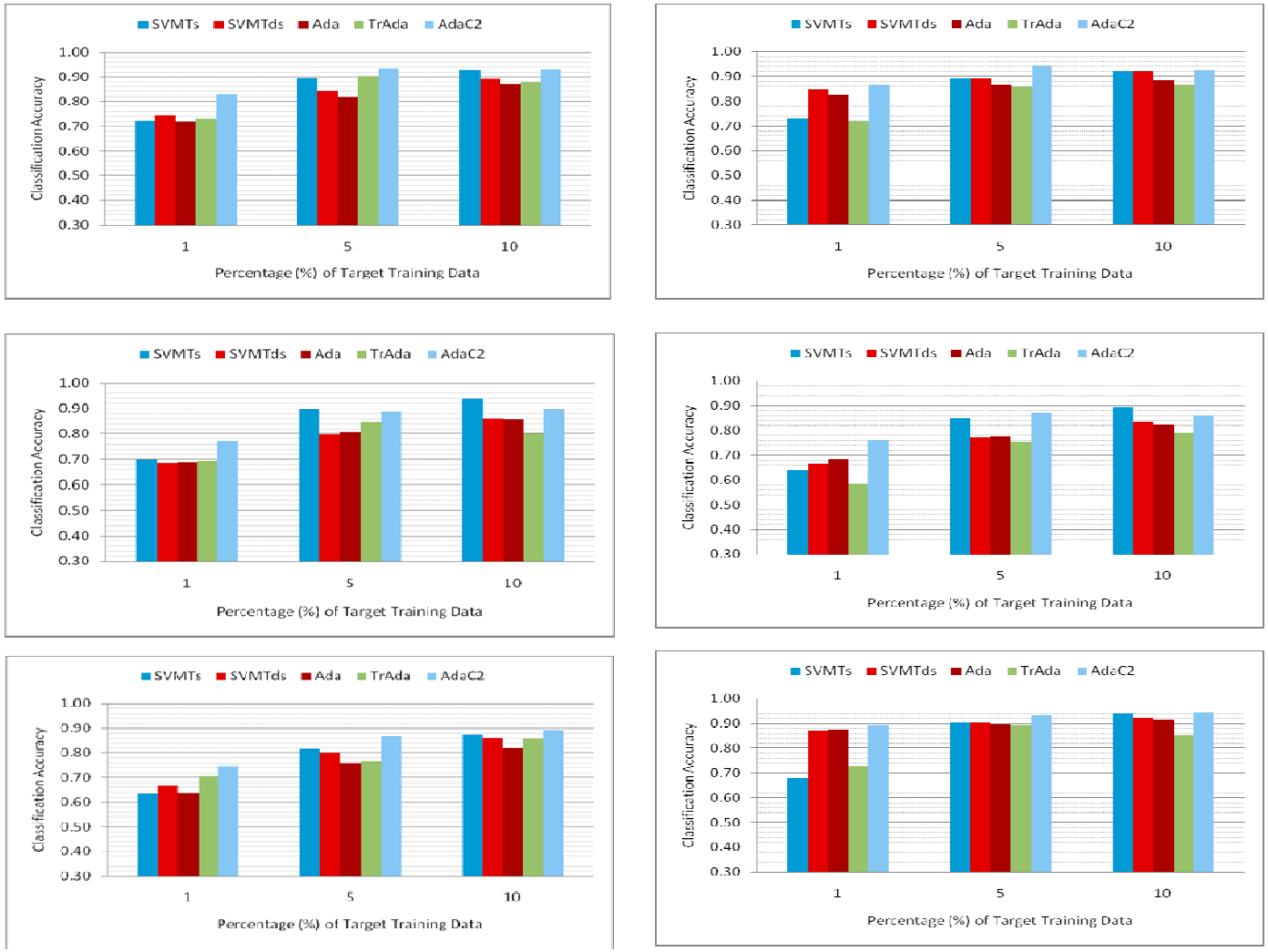


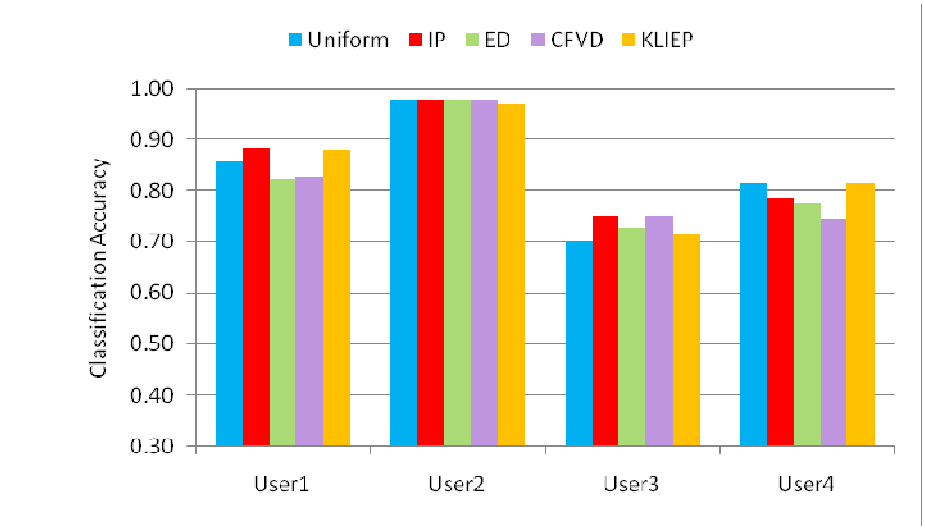

Figure 14 : Plots illustrating the variation in classification accuracies over 20Newsgroups1 dataset against 1%, 5% and 10% of Target Training Data. The datasets shown here include (i) Rec vs Talk, (ii) Rec vs Sci, (iii) Sci vs Talk, (iv) Comp vs Rec, (v) Comp vs Sci and (vi) Comp vs Talk.

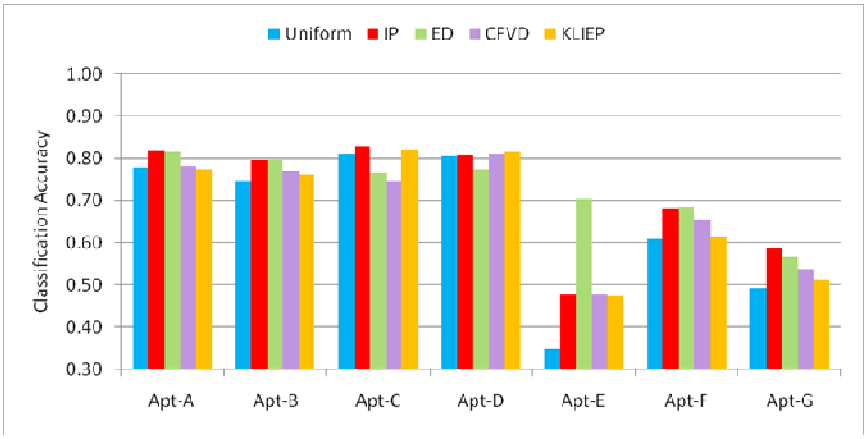
seems to have a clear sign of advantage over *AdaC2*. The reason is not very clear due to the correlation between SVM$T_s$ and *AdaC2*, which does away with the possibility of 1% target training data representing the target domain data sufficiently.

Similar trends can be observed on the 20Newsgroup1 dataset as well as illustrated by Figures 14(i) to 14(vi). It can be noticed that the performance of SVM$T_s$ increases significantly as we progress from 1% to 5%. However, it remains marginally lower than *AdaC2* transfer learning approach. The performance of the other techniques at 5% is lower than that of SVM$T_s$ or *AdaC2*. At 10%, SVMTs alone is sufficient to reliably classify the target domain unlabeled data. The change over AdaC2 becomes very marginal. However it is still worthwhile to note that while the performance of *AdaC2* and SVM$T_s$ is comparable, it still is significantly better than TrAdaBoost and AdaBoost.

### 5.3. Effect of Cost

The four different similarity measures mentioned in Chapter 4 were used for computing the cost factors and the performance of *AdaC2* was evaluated in each case to check how the classification accuracies varied with the different cost estimation techniques. To understand if the weighting instances using such measures had any effect at all in the first place, the algorithm's performance was measured over the base of a uniform cost. Running *AdaC2* with uniform costs is equivalent to running AdaBoost but with separate weight updates for samples in



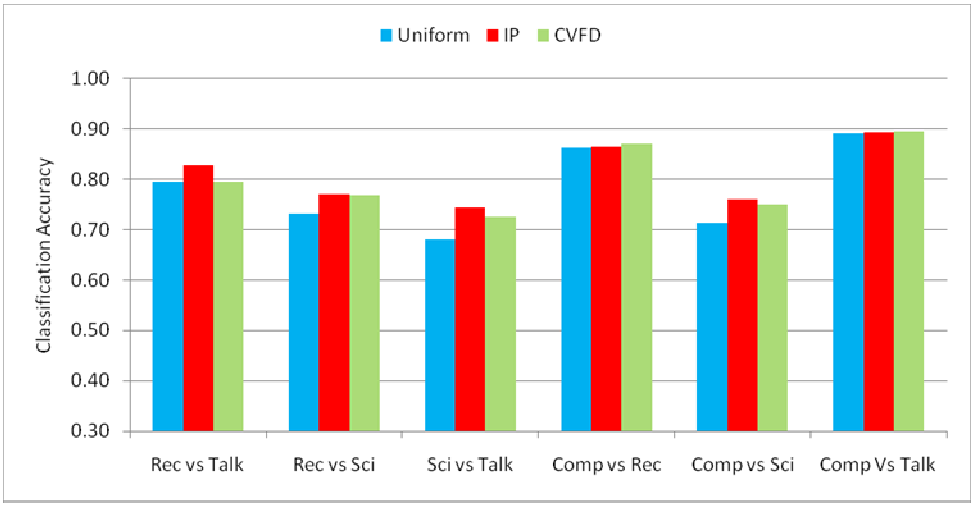




Figure 15 : Comparison of the classification accuracies of *AdaC2* using different cost estimation techniques.

$T_d$ and $T_s$. Figure 15 illustrates the varied performances obtained and the prevalence of instance pruning is briefly explained below.

There is a no clear trend that is noticeable from the results obtained from act_gest dataset, though there are changes in the accuracy values across the different users. It can be seen that Euclidean distance based relevance measure (ED) performs at par with the other techniques, when the similarity between the source and target domain is the highest, in this case that being data of user 2. In fact, for user 2, all the cost estimating procedures result in very similar performance. ED based cost performs poorer than the other approaches for the other users. The performance of the algorithm takes a marginal dip, when we assign uniform cost to all the samples in the source domain data. IP based cost procedure performs almost at par of sometimes even better with respect to all the other techniques. The CFVD based cost performs significantly poorer than IP based method for two of the 4 users and is at par with IP for the remaining 2 users. Overall, no straightforward conclusions can be made from the performance of the different cost estimation procedures on this dataset.

The act_rec dataset presents more clear trends in terms of the performance of the different cost estimation procedures. The first thing to be noticed is the sharp increase in the performance of the ED based cost for Apartment E. The performance increase is nearly 30%. It is interesting to note this result in the context that the target domain data for apartment E is most dissimilar to its source data. This probably implies that ED cost is able to clearly differentiate between

very different datasets. In contrast, the performance of ED drops significantly, when the source and target activities are similar as illustrated by its performance on Apartments C and D. The second thing to be noticed here is that invariably all cost estimation approaches perform yield a better performance over uniform cost, except for Apartments C and D. Even in this case, IP based cost performs marginally better than uniform cost. The similarity between the source and target domain samples is highest for these two datasets. Hence the results of IP and uniform cost appear to be similar. This implies that there is merit in considering cost estimation process to determine the relevance of source domain samples with respect to target domain. IP based cost appears to be yielding better results on this dataset. It can also be noted that the performance of KLIEP is at par with that of uniform cost. This could be attributed to the low number of target domain samples available for the importance estimation procedure. Incorporating the label information of the small amount of target domain training data does help to estimate better costs.

The most significant change in the plot pertaining to 20Newsgroups dataset is the absence of KLIEP and ER based cost estimation processes. We observed that the KLIEP process does not converge on this dataset. The primary reason for this is the significantly large dimension of this dataset (of the order 40,000). Furthermore the sparse nature of this dataset could also contribute to the lack of convergence of the KLIEP procedure. Euclidean distance based cost estimation on such a high dimensional dataset is also not worthwhile to consider,

as the output of such a process will be very similar to a uniform cost. IP based cost appears to be performing better than the other two approaches, even though the difference is only marginal. Similar to the observation made on the activity recognition dataset, uniform cost performs at par with the other two approaches when the similarity between the source and target domain is relatively higher, as illustrated by the performance on Comp vs Rec and Comp vs Talk datasets.

Overall the results from this set of experiments seem to indicate that the merit of IP based cost process over the other approaches. Furthermore, considering that the performance of this approach does not vary significantly over a uniform cost suggests that the division of the boosting approach with separate weight update equations for the source and target domain also helps in improving the performance. The performance of the cost based methods is nominal considering that the number of labeled target domain training samples is very less.

### 5.4. Dynamic Cost Update

Table 6 : Comparison between *AdaC2* and *DAdaC2*

| Dataset | *AdaC2* | *DAdaC2* |
|---|---|---|
| User1 | **0.88** | 0.87 |
| User2 | **0.98** | **0.98** |
| User3 | **0.75** | 0.71 |
| User4 | 0.79 | **0.80** |
| Apt - A | **0.82** | **0.82** |
| Apt - B | **0.80** | 0.74 |
| Apt - C | **0.83** | 0.80 |
| Apt - D | **0.81** | 0.77 |
| Apt - E | **0.48** | **0.48** |
| Apt - F | 0.68 | **0.69** |
| Apt - G | 0.59 | **0.60** |
| Rec vs Talk | 0.83 | **0.84** |

| Dataset | *AdaC2* | *DAdaC2* |
|---|---|---|
| Rec vs Sci | **0.77** | **0.77** |
| Sci vs Talk | **0.74** | **0.74** |
| Rec vs Comp | 0.86 | **0.89** |
| Comp vs Sci | **0.76** | 0.75 |
| Comp vs Talk | 0.89 | **0.90** |

Table 6 compares the results obtained from Dynamic Cost-sensitive Boosting (*DAdaC2*) and the *AdaC2* scheme of Cost-sensitive Boosting algorithm over the three datasets, act_gest, act_rec and 20Newsgroups1, to verify if a dynamic cost update has any advantage over a static cost factor or not. The idea of *DAdaC2* is to adapt to the changes in the weights of $T_s$ samples by keeping the cost factor constantly updated. However, as the results indicate, there does not seem to be much of difference between the two results save for the few marginal points up and down. This is once again caused due to the small size of the available target training data (1% in this case). Due to the size of $T_s$ the cost computer over every iteration in *DAdaC2* is pretty much the same as the initial cost computed. If the size of $T_s$ was bigger, then there might be a better a chance for the distribution of hard examples to vary over every iteration and perhaps even oscillate. In such scenarios *DAdaC2* may be more useful.

### 5.5. Comparison with Multi-Source Transfer

Table 7 : Comparison of performance between *AdaC2* and TransferBoost

| Dataset | TrAda | AdaC2 | Transfer Boost |
|---|---|---|---|
| Apt-A | 0.63 | **0.82** | 0.71 |
| Apt-B | 0.57 | **0.80** | 0.69 |
| Apt-C | 0.49 | **0.83** | 0.79 |
| Apt-D | 0.52 | **0.81** | 0.78 |

| Dataset | TrAda | AdaC2 | Transfer Boost |
|---|---|---|---|
| Apt-E | **0.70** | 0.48 | 0.37 |
| Apt-F | 0.40 | **0.68** | 0.61 |
| Apt-G | 0.46 | **0.59** | 0.56 |
| baseball | 0.46 | **0.78** | 0.54 |
| electronics | **0.65** | 0.64 | 0.54 |
| med | 0.52 | **0.67** | 0.51 |
| mideast | 0.39 | **0.54** | 0.48 |
| misc | 0.47 | 0.51 | **0.53** |
| pchardware | 0.63 | **0.69** | 0.53 |
| windowsx | 0.64 | **0.66** | 0.57 |

Table 8 : Comparison between cost based ranking and error on SVM$T_d$

| Source | Cost Rank | Err SVM$T_d$ |
|---|---|---|
| **Apt-A** | | |
| Apt-F | 0.529517 | 0.431791 |
| **Apt-C** | **0.923077** | **0.452148** |
| **Apt-B** | **1** | **0.490467** |
| Apt-G | 0.488372 | 0.530998 |
| Apt-D | 0.423971 | 0.656869 |
| Apt-E | 0.547406 | 0.892347 |
| **comp.sys.ibm.pc.hardware** | | |
| **comp.sys.mac.hardware** | **0.913907** | **0.268817** |
| **comp.windows.x** | **0.900662** | **0.283871** |
| sci.electronics | 0.84106 | 0.32043 |
| **windows.misc** | **0.900662** | **0.337634** |
| sci.med | 0.84106 | 0.427957 |
| rec.sport.hockey | 0.834437 | 0.477419 |
| **rec.sport.baseball** | **0.960265** | **0.483871** |
| **rec.motorcycles** | **1** | **0.494624** |
| talk.politics.mideast | 0.662252 | 0.56129 |
| talk.politics.misc | 0.304636 | 0.582796 |
| talk.religion.misc | 0 | 0.589247 |
| **sci.space** | **0.940397** | **0.619355** |

Tables 7 shows the performance of AdaC2 on multisource datasets act_rec and 20Newsgroups2, with the training done on 1% target training data and the multi-

source auxiliary datasets. The results are compared with that of TrAdaBoost and TransferBoost. The better performance of AdaC2 in a multisource dataset can be explained based on the advantage a lower level measure of similarity might gain over a higher-level measure of similarity. In this case the lower level similarity is measured using instance pruning between each source instance and the target instances while the higher-level similarity is measured using transferability between a source task and a target task. Computing similarity at the lowest level allows choosing similar instances that might be spread in a disparate fashion across the different source dataset. Another factor by which the cost computation in *AdaC2* might be superior to that of TransferBoost is its indirect relation to source–target macro-level similarity which can be checked by computing a score for each source dataset by normalizing the total sum of cost computed over a particular source. This score is presented in Table 8 along with the error of a model trained on $T_d$ over the entire target domain dataset $S$. It is to be noted that the cost computed has only $T_s$ available from the target domain and would in variably suffer from bias at one point or the other. Nevertheless, the similarities in the patterns are pretty striking. Referring to the actual dataset properties, it can easily be seen that Apartment – A shares a similar layout and same number of residents with Apartment – B and C, while documents related to “pc hardware” would typically be similar to “comp.sys.mac.hardware” and ”comp.windows.x”. Thus, the failure of TransferBoost can thus be attributed to the failure of a top-down cost estimate for the datasets considered here. Of course it cannot be

guaranteed that a bottom-top approach would always work. A best solution would be to incorporate both top-bottom and bottom-top knowledge for computing similarity.

# CHAPTER 6

# CONCULSION AND FUTURE WORK

This thesis explores the idea of knowledge transfer between non-identical training and test environments, devising an instance-based transfer technique that integrates a measure of relatedness onto a boosting framework. The motivation for this work stemmed from drawing a parallel between lifelong learning in humans and intelligent systems that function in real-world environments, with particular focus on the research on building a robust and adaptable accelerometer based gesture recognition system that can automatically scale up to handling varying user traits and environmental factors affecting the input signal given to the system. A system that incorporates a transfer learning framework as suggested would be able to successfully transfer knowledge between stock training data and the new target domain to classify gestures reliably. The proposed method conforms to the boosting theory that supports the convergence of AdaBoost and is shown to provide empirical success over different real-world datasets - Accelerometer based 3D Gesture Recognition and Smart Home Activity Recognition, and synthetic datasets generated from the 20Newsgroups document corpus. Despite a successful showcase, much future work remains to be done to understand and perfect such a technique of knowledge transfer along with a set of open questions about transfer learning.

## 6.1. Summary of the Work

A summary of the different contributions made in this thesis is listed below:

1. Three different variants for cost-sensitive boosting were investigated for improving the generalized performance of activity and gesture recognition. The algorithms were compared with that of baseline results and TrAdaBoost, a well cited boosting based instance-transfer algorithm. Among the variants, *AdaC2* was seen to work the best, with useful theoretical properties and promising empirical results.
2. The effect of using four different relatedness measures (cost factors) were studied and compared against each other. There is certainly a merit in determining cost as it always performs better than a uniform measure. Invariably instance pruning was found to give the best results. The correlation of these measures along with the actual posteriori measures were analyzed and spotted.
3. The Cost-sensitive boosting algorithm was modified to include an adaptive cost estimated based on the changing distribution of the target training data. This, however, did not result in a significant change in performance and at times lead to overfitting.
4. The equivalence of computing cost over multiple source domains bundled together against training seperately over the most related source domains was analyzed using two multisource datasets and it was found that instance level similarity can very well propogate into task based similarity. The performance of *AdaC2* was further compared with the performance of a recently proposed

boosting based multi-source transfer algorithm named TransferBoost with positive results.

### 6.2. Future Work

Some of the future directions founded on the limitations of this work, its application and some open research questions in the field of transfer learning are discussed below:

- **Estimating Relatedness:** An often-discussed issue in this thesis has been the idea of measuring relatedness to decide whether to transfer or not. Can relatedness be measured at all with only very few target training instances? Is it correct to call two tasks related just because they help each other when trained together? Sometimes injecting noise improves generalization. This does not mean that noise task is related to the target task [11]. Measuring the relatedness *a priori* helps automating knowledge transfer in intelligent systems. On a different note, the different cost factors estimated in this thesis is not exhaustive. There always lies the scope of modeling relatedness as an optimization problem similar to structural risk minimization.
- **Target Domain Instance Selection:** Besides faced with the problem of insufficient quantity of target training data, a parallel issue relates with the quality of the data. Given, that the approach suggested here is an instance-based transfer technique, it is all the more important that the target domain training data that is available reflects the unseen target domain data points or target tasks. It might be possible in some applications where an Active

learning methodology of collaborating with an expert might help select target data of good quality.

- **Discovering Structure:** Many-a-times, an instance-based transfer approach that uses just the low level similarities between data may not be very helpful for learning. Or even if it is, more success might be obtained by making use of an underlying structure in the dataset. In such cases, the cost factors can be computed probably using a linear combination of the various structural properties of the data and the tasks. This would facilitate better learning.
- **System Integration:** An important challenge that gets overlooked in such research is that of building a system based on the algorithms. To successfully deploy transfer-based systems, many factors must be taken into account such as how relevant source tasks and target tasks can be captured under reduced costs, how much of the source task information requires to be stored into an efficient database and how well the framework interact with other required learning frameworks already present in the system.

REFERENCES

[1] Thrun, S., "Lifelong learning algorithms." s.l. : Kluwer Academic Publishers, 1998. Learning to learn. pp. 181-209.

[2] Bishop, Christopher M., *Pattern Recognition and Machine Learning.* s.l. : Springer, 2006.

[3] Quionero-Candela, J., et al., *Dataset shift in Machine Learning.* s.l. : The MIT Press, 2009.

[4] Zliobaite, I., "Learning under concept drift: an overview." s.l. : Tech. rep., Vilnius University, Faculty of Mathematics and Informatics, 2009.

[5] Eaton, E.R., "Selective knowledge transfer for machine learning." s.l. : UNIVERSITY OF MARYLAND, BALTIMORE COUNTY, 2009.

[6] Krishnan, N.C., "A Computational Framework for Wearable Accelerometer-Based Activity and Gesture Recognition." s.l. : Arizona State University, December 2010.

[7] Krishnan, N.C., *Scalable Activity Recognition.* s.l. : NSF Panel Workshop - Pervasive Computing at Scale, 2011.

[8] Pan, S.J. and Yang, Q., "A Survey on Transfer Learning." IEEE Transactions on Knowledge and Data Engineering, s.l. : Published by the IEEE Computer Society, 2009.

[9] Pratt, L. and Jennings, B., "A Survey of Connectionist Network Reuse Through Transfer." Learning to learn, s.l. : Kluwer Academic Pub, 1998, p. 19.

[10] Kuhn, R., et al., "Rapid speaker adaptation in eigenvoice space." Speech and Audio Processing, IEEE Transactions on, s.l. : IEEE, 2000, Issue 6, Vol. 8, pp. 695-707.

[11] Caruana, R., "Multitask learning." Machine Learning, s.l. : Springer, 1997, Issue 1, Vol. 28, pp. 41-75.

[12] Torrey, L. and Shavlik, J., "Transfer Learning." Handbook of Research on Machine Learning Applications.IGI Global, s.l. : Citeseer, 2009, Vol. 3, pp. 17-35.

[13] Ben-David, S. and Schuller, R., "Exploiting task relatedness for multiple task learning." Lecture notes in computer science, 2003, pp. 567-580.

[14] Thrun, S. and O'Sullivan, J., "Discovering structure in multiple learning tasks: The TC algorithm." 1996. Proceedings of the Thirteenth International Conference on Machine Learning. pp. 489-497.

[15] Eaton, E. and Desjardins., "Set-Based Boosting for Instance-level Transfer." s.l. : IEEE, 2009. 2009 IEEE International Conference on Data Mining Workshops.

[16] Eaton, E., Desjardins, M. and Lane, T., "Modeling transfer relationships between learning tasks for improved inductive transfer." Machine Learning and Knowledge Discovery in Databases, s.l. : Springer, 2008, pp. 317-332.

[17] Kifer, D., Ben-David, S. and Gehrke, J., "Detecting change in data streams." s.l. : VLDB Endowment, 2004. Proceedings of the Thirtieth international conference on Very large data bases. pp. 180-191.

[18] Ben-David, S., et al., "Analysis of representations for domain adaptation." s.l. : The MIT Press, 2007. Advances in Neural Information Processing Systems 19. p. 137.

[19] Jiang, J. and Zhai, C.X., "Instance weighting for domain adaptation in NLP." 2007. Annual Meeting-Association For Computational Linguistics. Vol. 45, p. 264.

[20] Wu, P. and Dietterich, T.G., "Improving SVM accuracy by training on auxiliary data sources." 2004. Proceedings of the twenty-first international conference on Machine learning. p. 110.

[21] Liao, X., Xue, Y. and Carin, L., "Logistic regression with an auxiliary data source." 2005. Proceedings of the 22nd international conference on Machine learning. pp. 505-512.

[22] Jiang, J., "A literature survey on domain adaptation of statistical classifiers." 2008.

[23] Huang, J., et al., "Correcting sample selection bias by unlabeled data." Advances in neural information processing systems, 2007, Vol. 19, p. 601.

[24] Sugiyama, M., et al., "Direct importance estimation with model selection and its application to covariate shift adaptation." Advances in Neural Information Processing Systems, 2008, Vol. 20, pp. 1433–1440.

[25] Argyriou, A., Evgeniou, T. and Pontil, M., "Multi-task feature learning." Advances in neural information processing systems, 2007, Vol. 19, p. 41.

[26] Blitzer, J., McDonald, R. and Pereira, F., "Domain adaptation with structural correspondence learning." s.l. : Association for Computational Linguistics, 2006. Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing. pp. 120-128.

[27] Pan, S.J., Kwok, J.T. and Yang, Q., "Transfer Learning via Dimensionality Reduction." 2008. Proceedings of the 23rd national conference on Artificial intelligence. pp. 677-682.

[28] Evgeniou, T. and Pontil, M., "Regularized multi--task learning." 109-117 : ACM, 2004. Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 109-117.

[29] Raina, R., Ng, A.Y. and Koller, D., "Constructing informative priors using transfer learning." s.l. : ACM, 2006. Proceedings of the 23rd international conference on Machine learning. pp. 713-720.

[30] Lawrence, N.D. and Platt, J.C., "Learning to learn with the informative vector machine." s.l. : ACM, 2004. Proceedings of the twenty-first international conference on Machine learning. p. 65.

[31] Gao, J., et al., "Knowledge transfer via multiple model local structure mapping." s.l. : ACM, 2008. Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 283-291.

[32] Mihalkova, L. and Mooney, R.J., "Transfer learning with Markov logic networks." 2006. Proceedings of the ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning.

[33] Dai, W., et al., "Eigentransfer: a unified framework for transfer learning." s.l. : ACM, 2009. Proceedings of the 26th Annual International Conference on Machine Learning. pp. 193-200.

[34] Dai, W., et al., "Translated learning: Transfer learning across different feature spaces." 2008. NIPS. pp. 353-360.

[35] van Kasteren, T., Englebienne, G. and Kröse, B., "Recognizing activities in multiple contexts using transfer learning." 2008. AAAI AI in Eldercare Symposium.

[36] Rashidi, P. and Cook, D.J., "Multi home transfer learning for resident activity discovery and recognition." 2009. Proceedings of the 4th International workshop on Knowledge Discovery from Sensor Data.

[37] Freund, Y., Schapire, R. and Abe, N., "A short introduction to boosting." JournalL-Japanese Society for Artificial Intelligence, 1999, pp. 771-780.

[38] Dai, W., et al., "Boosting for transfer learning." Proceedings of the 24th International Conference on Machine learning, 2007, pp. 193-200.

[39] Littlestone, N. and Warmuth, M.K., "The weighted majority algorithm." s.l. : IEEE Computer Society Press, 1989.

[40] Torralba, A., Murphy, K.P. and Freeman, W.T., "Sharing features: efficient boosting procedures for multiclass object detection." s.l. : IEEE Computer Society Press, 2004.

[41] Yao, Y. and Doretto, G., "Boosting for transfer learning with multiple sources." s.l. : IEEE, 2010.

[42] Sun, Y. and Kamel, M.S. and Wong, A.K.C. and Wang, Y., "Cost-sensitive boosting for classification of imbalanced data." 2007, pp. 3358-3378.

[43] Chang, C.C. and Lin, C.J., "LIBSVM: a library for support vector machines." 2001.

[44] Katakis, I. and Tsoumakas, G. and Vlahavas, I., "An ensemble of classifiers for coping with recurring contexts in data streams." 2008. Proceeding of the 2008 conference on ECAI. pp. 763-764.

[45] Lawton, M. P. and Brody, E. M., "Assessment of older people: self-maintaining and instrumental activities of daily living." The Gerontologist, 1969, Vol. 9, p. 179.

[46] Evgeniou, Theodoros and Pontil, Massimiliano., "Regularized multi--task learning." s.l. : ACM, 2004. pp. 109-117.

[47] Lawrence, Neil D. and Platt, John C., "Learning to learn with the informative vector machine." 2004. p. 65.

[48] Kasteren, Tim van, Englebienne, Gwenn and Kr\"{o}se, Ben., "Recognizing Activities in Multiple Contexts using Transfer Learning." 2008.

[49] Ben-David, S., et al., "Analysis of representations for domain adaptation." s.l. : The MIT Press, 2007. Advances in Neural Information Processing Systems 19. p. 137.

[50] Cook, D., "Learning Setting-Generalized Activity Models for Smart Spaces." *IEEE Intelligent Systems.* 2010.