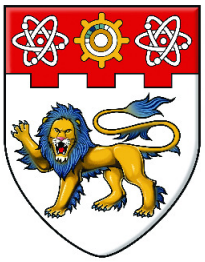

NANYANG
TECHNOLOGICAL
UNIVERSITY

# Distributed Classification with Variable Distributions

A Thesis Submitted to the School of Computer Engineering
of the Nanyang Technological University

by

**QUACH VINH THANH**

In Partial Fulfillment of the Requirements for the
Degree of Master of Engineering

2014

# Abstract

When the data at a location is insufficient, one may apply a naïve solution to gather data from other (remote) places and classify it using a centralized algorithm. Although this approach has good performance, it is often infeasible due to high communication overheads and lack of scalability of the centralized solution. These concerns have led to the emergence of distributed classification.

The promise of distributed classification is to improve the classification accuracy of a learning agent (called *party*) on its respective local data, using the knowledge of other parties in the distributed network. However, current explorations implicitly assume that all parties receive data from exactly the same distribution of data. We show that this is too simple a scenario, and that in reality, data across parties may be different from each other, in terms of both the data distribution of the inputs (observations) and/or the outputs (labels).

We remove the current simplifying assumption by allowing parties to draw data from arbitrary distributions, thus formalizing a new and challenging problem of distributed classification with variable data distributions. We show that this problem is difficult, because it does not admit state-of-the-art solutions in the context of (conventional) distributed classification.

After posing the problem and illustrating its difficulty, we present a list of remarkable research challenges (or sub-problems) that should be addressed in this challenging field. For each of those challenges, we provide some potential research directions.

Finally, as the first attempt on this new problem, we present a simple-to-implement, straightforward yet working algorithm called VarDist that efficiently solves the problem where the data distribution may vary over the participating parties. Although VarDist is not a complete and sophisticated solution, it does have low costs of communication, while providing a more accurate classifier (than local learning) by benefiting from the auxiliary classifiers from the other parties.

# Acknowledgments

My deepest gratitude first goes to my supervisor Dr. Ng Wee Keong, for his time, effort, support and guidance during my Master candidature. I would also like to express my gratitude to my co-supervisor, Dr. Vivekanand Gopalkrishnan, for his continuous encouragement on my research. Together, they have taught me a lot of research skills and essential knowledge that has significantly inspired and benefited me in my academic career.

I would also like to thank Alex and Ardian for their valuable suggestions, which have greatly assisted me in completing this report.

Last but not least, I am extremely grateful to my parents for their invaluable encouragement and support during my hard times. Without them, this work definitely could not have been accomplished.

# Contents

# List of Figures

# 1 Introduction

## 1.1 Background and Motivation

Classification refers to a prediction problem in which we learn to predict the label of some data instance. Specifically, in classification we are given a set of *observations* (or *inputs*) of data, accompanied by their *labels* (or *outputs*), and our goal is to use some *learning algorithm* and learn a highly accurate prediction rule (also called *classifier*) from such dataset. Afterwards this rule can be utilized to classify any future or unseen data instance. Together with regression, clustering and association rule mining, classification is typically placed among the most common problems in machine learning and data mining. Classification has attracted a significant amount of attention since its inception, and its popularity has been witnessed in a variety of practical applications, for example, letter recognition, image annotation, medical diagnosis, etc.

### 1.1.1 Distributed classification

Standard or traditional classification assumes the setting in which we work with a single dataset or database readily accessible to us. In classification (and machine learning in general), it has been well known that the amount of data for which we use to learn is significantly important and directly affects the learning accuracy. In fact, as learning theory shows [17], we should try to learn from as much data as possible in order to best approximate the true labelling function. In some domains, however, it is problematic to gather or generate enough data. With such issue of *data insufficiency*, standard classification techniques fail to produce a good classifier.

Great technology advances in information systems, however, have greatly impacted upon the way we collect or generate data. Such advent of information technology has led to many databases that are produced at different locations. Present-day datasets, even those of the same type and nature, may be geographically distributed across different regions or locations. For example, medical records of patients are currently stored by various hospitals or institutions from different places. In such cases, the amount of data stored in other distributed databases would be valuable in helping to increase the accuracy of our learning algorithm, thereby enhancing our knowledge discovery. In other words, we can compensate the issue of data insufficiency by trying to learn from a variety of information sources that are located geographically far away from us.

An obvious and naïve solution would be to collect every piece of data from all of those information sources, and then simply apply some standard classification algorithm on the newly combined data. This is commonly referred to as *centralized learning*. Since we would be able to have access to a much bigger amount of data in this way, such methodology would lead to better classification results than *local learning* (that is, learning with our local insufficient data source only). However, a number of limitations prevent centralized learning from being practically feasible.

(i) First, transferring the whole collection of data from different information places would result in a huge amount of network communication. In other words, learning algorithms that are built on a centralized database would incur exorbitant costs of communication overheads. As a consequence, such solutions are undesirably impractical.

(ii) Second, even if we were to ignore such significant cost of communication in data transfer (and somehow manage to centralize data successfully), the resulting large centralized database would still not be desirable: we might have to face an enormous combined dataset which traditional classification algorithms are not scalable enough to cope with.

These concerns have led to the emergence as well as popularity of *distributed learning*. As opposed to centralized learning, our methodology in distributed learning is to learn from different remote data sources in a *decentralized* manner, that is, *without* gathering all datasets to a single location. We will focus on the context of classification only, for which we have the so-called problem of *distributed classification*.

Throughout the rest of this thesis, we use the term *party* to refer to a particular data source in a distributed environment. Distributed classification then aims to extract knowledge from the data located at all other parties, and then use that knowledge to enhance the classification accuracy at the local party. This problem has received a significant amount of attention in the machine learning and data mining community, attracting a deluge of works and solutions in recent years.

### 1.1.2 Heterogeneous databases with various distributions

A significant amount of studies has been conducted on both theoretical and practical aspects of distributed classification. However, current efforts only assume the simplest

case, in which all of the participating parties have exactly the *same* distribution of data. As a consequence, in all of current works there always exists (at least implicitly) a notion of *global distribution*. Such a notion of some common, single distribution allows us to focus on improving the accuracy of the so-called *global classifier*, instead of trying to improve the classification performance at some particular party. The global distribution also allows the newly constructed global classifier to be applied in exactly the same way at every single party.

While the assumption of same distribution may seem reasonable in some contexts, it does not actually hold in many real-world situations. The main reason is that the advent of new information technology is currently providing us with heterogeneous remote databases that are no longer suitable for conventional distributed classification. In this subsection, we discuss several practical scenarios in which the produced or generated datasets from various information sources do not have the same distribution.

To elaborate on this, let us first make a digression and review some important concepts in the theory of classification, including those of an *input*, *output*, and data *distribution*. When encountering some particular data instance, a classifier makes a prediction by assigning some label/class (output) to the unlabeled part (input) of that data point. In machine learning theory, it is standard to assume some probability distribution on the data points. More specifically, each input is supposed to be distributed according to an *input distribution*. Similarly, given a specific input, there exists an *output distribution* that generates some output or label based on that input. By assuming such distributions of data, we are allowed to look 'outside' the dataset we learn from, so that we are able to make good predictions on new unseen data. In fact, the data distributions have been the main assumption of learning theory to provide strong guarantees on the *generalization* (or out-of-the-training-set) accuracy.

As we have already mentioned, there are likely to be some real-world scenarios in which a global distribution does not exist. The main reason is essentially that the input

distribution and/or the output distribution might differ across the participating parties. Let us consider such scenarios, as illustrated by the various distributed learning environments in Figure 1.1, Figure 1.2 and Figure 1.3. We first make a few explanations regarding the components of these figures.

- In these figures, each party has its local data represented by a circle, and its classification problem represented by a text box.
- For every party, the input space is depicted by the shapes within the circle, while the distribution of inputs is depicted by a particular arrangement of those shapes.
- On the other hand, the output space at every party is denoted by the classification problem in the box, and the distribution of outputs is denoted by a particular set of probability values.
- Meanwhile, distributed classification requires the parties to exchange some information between themselves. For example, this information might be in the form of extracted knowledge, models, or in some cases, data instances. We represent such various flows of information transferred across the network by multiple arrows crossing the dotted lines.

#### 1.1.2.1 Scenario 1 (Different input distributions, same output distribution)

In this scenario, the parties that together participate in distributed classification may have access to their local data that are generated from different input distributions. However, given some input observation, the output distribution is exactly the same for all parties. Figure 1.1 illustrates these cases, in which the distribution of inputs is changed across the parties, portrayed by different arrangements of the shapes.

This scenario is common in many practical applications, e.g., spam detection, speech recognition, etc. In such applications, the difference in the process of data collection might

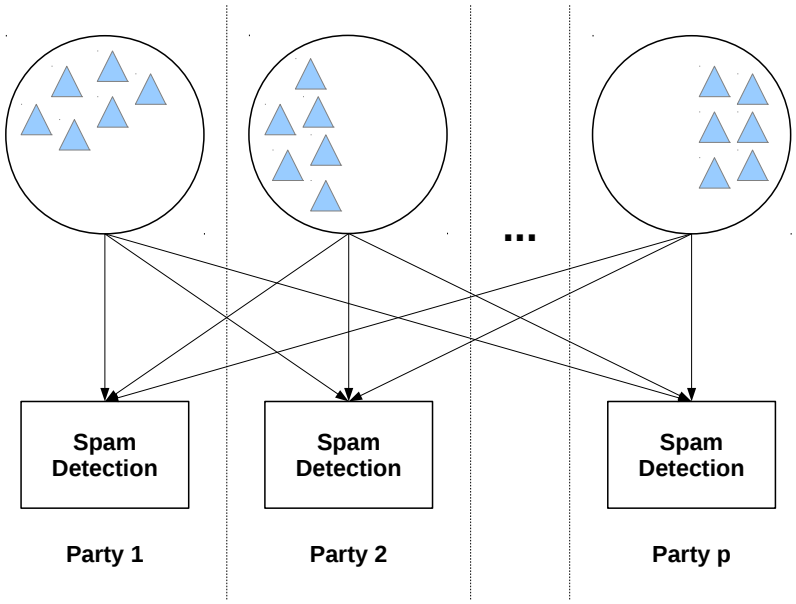


Figure 1.1: Scenario 1: Different input distributions, Same output distribution

often lead to variable input distributions among the parties. Let us take an example from spam detection, where the goal is to classify an input document (say, an email or some arbitrary text) into whether it is a spam or not. A common scenario is that different parties may collect data at different time periods. As a consequence, the distribution of the input documents at some party may be different from that at other parties. For instance, the first party may have access to some spam dataset collected three years ago and mostly focusing on cars, whereas the second party uses much more up-to-date spam data, whose topic is mainly about health and hygiene products. Another example of different input distributions can be seen in speech recognition. Here each party could represent a specific geographical region, and data at the various parties could contain voices of different accents [18], thus having different input distributions.

#### 1.1.2.2 Scenario 2 (Different output distributions)

In this scenario, local data at the various participating parties may have different distributions of the labels, regardless of whether their input distributions are the same or not.

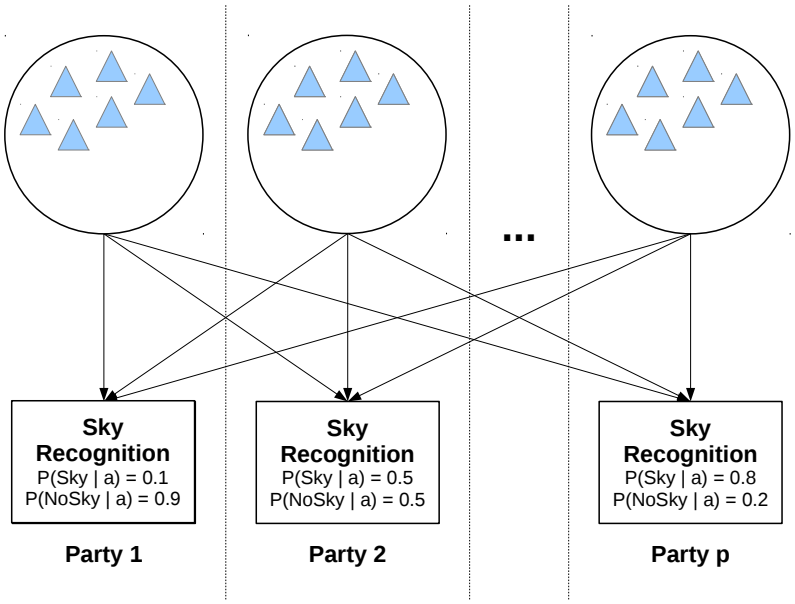


Figure 1.2: Scenario 2a: Same input distribution, Different output distributions

In other words, we do not care about whether the input observations at those datasets are generated from the same distribution; instead, we focus our attention to the potential difference in the conditional probability distribution that generate some label from an arbitrary observation. Figure 1.2 and Figure 1.3 together illustrate these cases.

- In Figure 1.2, only the distribution of outputs or labels varies across the parties.
- In Figure 1.3, both the input and output distributions change over all of the participating parties. In this case, each party may have an arbitrarily different distribution of labels on the inputs (which are also likely to be distributed according to different input distributions).

As in the first scenario, this scenario is also likely to occur in reality. Let us take an example of medical diagnosis. Here each party could represent a hospital that desires to use distributed classification to enhance the process of knowledge discovery on its local data. The dataset consists of multiple patient records (which are essentially the inputs).

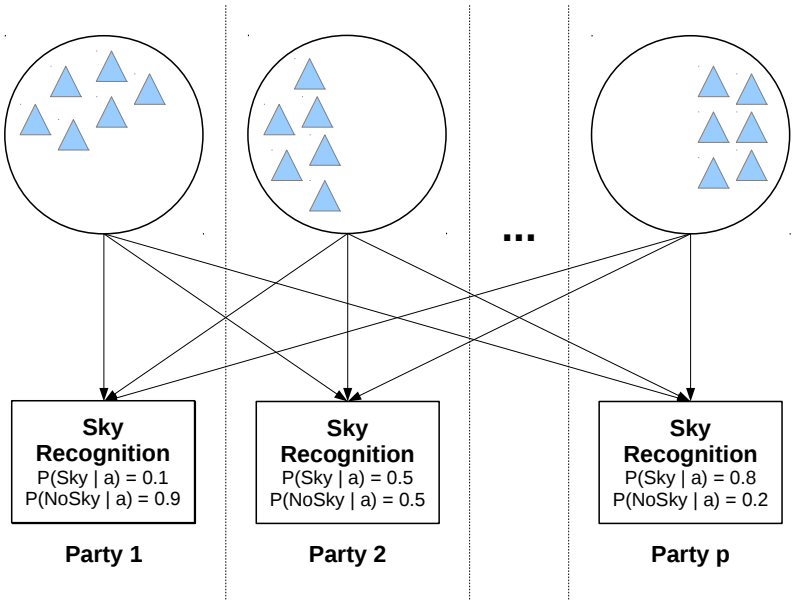


Figure 1.3: Scenario 2b: Different input distributions, Different output distributions

On examining a disease, the same patient could be diagnosed by different doctors (as well as examined using different medical methods) in different hospitals. In other words, the same input could be classified according to different label distributions at different parties.

### 1.1.3 Distributed classification with variable distributions

When the input and/or output distributions differ across the parties, distributed classification becomes much more difficult. The reason is that the notion of a global distribution among the participating parties (and correspondingly, the notion of a global classifier) no longer exists. A sceptical reader may then question the necessity of distributed classification. The reason is that standard classification is established based on the same distribution between the training and test sets, and therefore the data with different distributions might be harmful to the local learning process. In this case, he or she may even suppose that a party might as well use it own local data for training its own classifier.

However, distributed classification is still necessary (as opposed to local learning) for two reasons.

(i) First, the local data is not enough for the learning algorithm to produce an accurate classifier. This reason is essentially the motivation of distributed classification in the first place (see Section 1.1.1).

(ii) Second, it turns out that if the other distributions are somehow similar to the local distribution, the auxiliary data with such different distributions are not necessarily harmful and may actually *help* the learning process.

We now elaborate on the second reason. This is essentially the idea of a large and emerging field called *transfer learning* (and its close cousin, *domain adaptation*), which has received high and increasing amounts of attention over the last few years. Let us provide some examples to illustrate the usefulness of auxiliary data with different (but similar) distributions. One example is about text classification. In natural language processing, we may not have enough labeled data on some specific (target) domain, while there are abundant data on another (source) domain. Nevertheless, such auxiliary source data may still help to increase the classification accuracy with respect to the target domain, as can be seen in many works [4,7,8,19]. Another example is about video concept detection. Videos on different domains might have different distributions (for example, they might be from different TV channels or be recorded in different languages). However, the classified videos on one domain can help to classify ones on another domain better (see [43]). We can also see the usefulness of auxiliary data from a different domain in the problem of object detection or object categorization (see [44]). Overall, it can be seen that the data drawn from the other distributions may actually help the learning process with respect to the local data distribution, so the necessity of distributed classification is still justified.

We have shown that the problem of distributed classification is compulsory even when the participating parties have access to data which are distributed according to different distributions. This motivates the problem of *distributed classification with variable distributions*, which is the focus of this thesis. The overall goal is to enhance the classification accuracy at the local parties by combining the knowledge from other remote heterogeneous parties. However, this should be done (1) in a distributed fashion between multiple participating parties, while (2) simultaneously addressing the issue of potentially different data distributions among those parties. With the new problem in mind, in order to avoid confusion, we will call the old problem of distributed classification in Section 1.1.1 by the name of *conventional/standard distributed classification*.

It turns out that the newly posed problem is a more difficult one than conventional distributed classification. Let us attempt to resolve the new problem by applying the same idea (regarding the concepts of global distribution and global classifier) from conventional distributed classification. Specifically, since each party may be entitled to a particular input and/or output distribution, one may wonder whether it is still possible to construct a global distribution/classifier or not. Suppose we try to construct a global distribution by consolidating the data distributions from all of the parties. However, this is challenging because (1) we are uncertain about the data distribution at every party, and (2) we do not know the parameters in the mixture of those distributions during consolidation.

Suppose further that we could somehow manage to construct such a global distribution (say, by using some approximation methods), and then continue to obtain a global classifier right afterwards. As opposed to conventional distributed classification, this final classifier would be likely to have a poor performance in terms of generalization when being applied later at every party. The reason behind this argument is that such classifier is constructed to be good only with respect to the aforementioned mixture of data distributions. It is, however, not 'tuned' to be good for the specific data distribution at

some party (which is different from the mixture), and is thus not able to enhance the classification accuracy for that party (which is the promise of distributed classification).

## 1.2 Research contributions

In this thesis, we make two research contributions, including one major contribution and one minor contribution.

**Major contribution: The problem.** We make a major contribution in *posing the novel problem of distributed classification with variable distributions*. As seen from our study in the previous introductory sections, many real-world scenarios of learning in a distributed environment have not yet been adequately addressed. Because of this, the newly-posed problem of distributed classification in which the parties have potentially different data distributions is *important*, and yet *unresolved* by current approaches. To the best of our knowledge, we are the first to pose this exciting novel problem.

We have also given some discussion that distributed classification with variable data distributions is much more *difficult* than the original problem of conventional distributed classification. In the next chapters we will further highlight the difficulty of the new problem by presenting a detailed review of current approaches in conventional classification, together with their limitations in addressing this new problem.

Finally, we discuss several potential *challenges* of the new problem that make it difficult. They can be regarded as the *sub-problems* that might arise and need to be resolved during any attempt for distributed classification with variable distributions. For each of those challenges, we also provide several potential research directions on how to address it.

**Minor contribution: A preliminary algorithm.** As the first attempt to tackle the newly-posed problem, we provide a simple yet working solution called VarDist. The

principle of this algorithm is to use parameter transfer (in particular, transfer of the locally trained classifiers among the parties). The algorithm then tries to adapt those classifiers to some specific data distribution at the target party (i.e., the party that desires to seek knowledge from the other parties). The adaptation procedure is borrowed from the field of transfer learning. Our solution is highly efficient with respect to the communication costs. In addition, experimental results show that our solution is able to deal with the difference in distributions between the parties, and produce an accurate classifier with better performance than the locally learned classifier.

However we want to emphasize that we *do not* attempt to create a complete and sophisticated solution, as the new problem of distributed classification with variable distributions is highly challenging and may require a great deal of research and efforts from the community. Instead, the VarDist protocol that we propose is a straightforward, simple-to-use and simple-to-implement algorithm for classification in a distributed environment where participating parties have different data distributions.

## 1.3 Organization

The rest of this thesis is organized as follows.

- In Chapter 2, we formally introduce the aforementioned problem of distributed classification with variable data distributions. This problem differs from previous formulations in that the data distribution at each party is now arbitrary, and is not required to be coherent with that at the other parties. To make such distinction more clearly, we also introduce a formalized model of conventional distributed classification, which can be valuable in its own right.

- In Chapter 3, we analyze the incompetence of existing solutions for this new problem. We shall study the closest relevant techniques studied in the literature of

conventional distributed classification. We show that these approaches make a restricted assumption (of the same data distribution among the participating parties), and therefore deal with a much simpler problem. We then conclude that none of the current works can be applied (or at least be easily adapted) to solve the new problem (which essentially also helps to highlight its difficulty).

- In Chapter 4, we discuss four of the potential challenges associated with distributed classification when the parties have different data distributions. These challenges are highly relevant, and should be adequately addressed when attempting to resolve the problem. In addition, we discuss several potential research directions related to such challenges.

- In Chapter 5, we present the solution VarDist, together with some analysis on the communication costs of the algorithm. Experimental results regarding the communication costs and accuracy of VarDist are also provided here.

- Finally, in Chapter 6, we give a conclusion for this thesis, and discuss a few potential future works.

# 2
# Problem Definition

Conventional or standard distributed classification assumes the same data distribution among all participating parties. The problem of distributed classification with variable distributions is essentially conventional distributed classification in which a new constraint is imposed: different parties may have different data distributions.

The objective of this section is to formally define the goal of learning in distributed classification with variable distributions. In order to do this, we need to first formalize the setting of standard distributed classification, as described in Section 2.2. A revision of basic theory on classification is also necessary, and is presented in Section 2.1.

## 2.1 Standard classification

As introduced in Chapter 1, we focus our attention on a particular type of learning problems: *classification*. For this type of learning, a data instance is essentially composed of two parts: *input* (also called *observation* or *features*) and *output* (also called *label* or *class*). The input represents the (unlabeled) measurements of various attributes for

an instance, while the output represents the class or label of that instance. We now present some probabilistic formulation of the classification problem. This presentation is essentially standard and can be seen in many works.

We assume that inputs are modelled by a random variable $X$ that takes values in some space $\mathcal{X}$. $\mathcal{X}$ is typically known as the *input space* (also *feature space*). For each instance, each input $X$ is accompanied by an output, denoted by a random variable $Y$, that takes values in some space $\mathcal{Y}$. $\mathcal{Y}$ is typically known as the *output space*. We only focus on *binary classification*, in which there are only two possible labels. In such cases, we usually denote the output space as $\mathcal{Y} = \{-1, 1\}$.

We assume that each input $X$ is distributed according to a (fixed but unknown) marginal distribution function $P_X$ of $X$. In other words, we have $X \sim P_X$ (which should be understood as the random variable $X$ is distributed according to the distribution $P_X$). We also assume that given a particular input $X$, each output $Y$ is distributed according to a (fixed but unknown) conditional distribution $P_{Y|X}$ of $Y$ on $X$. In other words, $P(X, Y) = P(X)P(Y|X)$ is the joint distribution of $P_X$ and $P_{Y|X}$. Sometimes we also use the shorthand notation for the whole instance instead of the two parts $X$ and $Y$: we let each instance be modeled by a random variable $Z = (X, Y)$. The instance space is then denoted by $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, and the probability distribution according to which $Z$ is generated is $P(Z) = P(X, Y) = P(X)P(Y|X) = P_X \times P_{Y|X}$. We also denote this joint distribution by $P_Z$.

A *prediction function*, or a *hypothesis* $f$ is expressed in term of a mapping from $\mathcal{X}$ to $\mathcal{Y}$, i.e. $f : \mathcal{X} \to \mathcal{Y}$. Since we only focus on classification, we also call $f$ a *classifier*. The term 'hypothesis' is standard in learning theory, and we will use the two terms 'hypothesis' and 'classifier' interchangeably.

Normally, in order to obtain the final classifier (as the result of learning), we need to select it among a *hypothesis space* $\mathcal{F}$. Because of this, we need some measurement

of goodness for each hypothesis $f \in \mathcal{F}$. Usually, we consider a loss function $\ell$, where $\ell(f, z) = \ell(f, x, y)$ measures the cost that $f$ makes upon encountering the instance $Z = z = (x, y)$. The expectation of this loss function over the unknown distribution $P_Z$ is defined to be the *risk* of $f$, commonly used as a criterion of selection for $f$ among hypotheses in $\mathcal{F}$,

$$R(f) = \mathbb{E}_{Z \sim P_Z} \left[ \ell(f, Z) \right]. \tag{2.1}$$

In classification, it is common that $\ell$ is the 0-1 loss function, i.e.,

$$\ell(f, z) = \ell(f, x, y) = 1_{f(x) \neq y}. \tag{2.2}$$

so that the risk of a classifier $f$ becomes

$$R(f) = \mathbb{E}_{Z \sim P_Z} \left[ \ell(f, Z) \right] = \mathbb{E}_{Z \sim P_Z} \left[ 1_{f(X) \neq Y} \right] = \Pr(f(X) \neq Y), \tag{2.3}$$

which we also call by other names such as the *generalization error*, *true error* or simply *error*. We will use the terms 'risk' and 'error' interchangeably.

**Definition 2.1 (True error)** *The true error (also called 'error' or 'generalization error') of a classifier $f \in \mathcal{F}$ is the probability that $f$ misclassifies an instance $Z$ generated according to the distribution $P_Z$,*

$$\text{err}(f) = R(f) = \Pr(f(X) \neq Y). \tag{2.4}$$

The goal of classification is to select a hypothesis $f \in \mathcal{F}$ with minimal risk. However, since the distribution $P_Z$ is unknown, we cannot minimize the risk functional $R(f)$ directly. Instead, we rely on an *empirical training set* that is given to us. In learning theory, it is common that we consider $S$ as an *independent and identically distributed* (I.I.D.) sample set of $n$ examples generated from $P_Z = P_X \times P_{Y|X}$,

$$S = \{(X_1, Y_1), \ldots, (X_n, Y_n)\} \sim (P_X \times P_{Y|X})^n, \tag{2.5}$$

then the problem of classification is defined as follows.

**Definition 2.2 (Classification)** *In the problem of classification, one is supplied with an i.i.d. training set $S$ of labeled data, and the objective is to learn a classifier $g$ with as small risk as possible.*

Such a minimal-risk classifier $g$ will accurately map the unlabeled input $X \in \mathcal{X}$ to the output $Y \in \mathcal{Y}$. This final classifier $g$ can be the result of many standard learning algorithms (for example, Naïve Bayes, Decision tree, Logistic Regression, Support Vector Machines, etc.)

## 2.2 Conventional distributed classification

The problem of conventional distributed classification differs from traditional classification in several aspects.

(i) First, there are $M$ parties, denoted by $P_1, \ldots, P_M$, in a distributed learning environment. The objective of each party $P_m$, $m = 1, \ldots, M$, is to collaborate and exchange information with the rest of the parties, with the purpose of enhancing the performance (that is, reducing the generalization error) of its locally constructed classifier.

(ii) Second, each party can observe only its own local dataset (instead of the whole amount of data in the distributed network). However, data at every party is generated according to the same distribution across all parties. We use $S_m$ to denote the local dataset at the party $P_m$, $m = 1, \ldots, M$, then

$$S_m = \{(X_1, Y_1), \ldots, (X_{n_m}, Y_{n_m})\} \sim (P_X \times P_{Y|X})^{n_m} \tag{2.6}$$

where $n_m$ is the size of the local dataset located at the party $P_m$. In this manner, the whole data available is the union of all datasets located at the parties, and this

has the same role as the entire data $S$ in standard learning. In other words,

$$S = \bigcup_{m=1}^{M} S_m = \bigcup_{m=1}^{M} \{(X_1, Y_1), \ldots, (X_{n_m}, Y_{n_m})\} \tag{2.7}$$

However, unlike standard classification, the set $S$ is now not 'viewable' (i.e., not readily accessible) by all parties.

(iii) Third, the environment imposes the communication constraint on the learning algorithms, that is, an appropriate learning solution for conventional distributed classification needs to have low communication costs.

(iv) Finally, suppose the party $P_m$, $m = 1, \ldots, M$, has applied some standard classification algorithms on its local dataset $S_m$, and constructed a 'locally optimal' classifier, denoted by $f_m$. In distributed classification, the goal of a party $P_m$ is to (1) attempt to extract knowledge from all of the other parties $P_{m'}$ (where $m' \in \{1, \ldots, M\} \setminus \{m\}$) in a decentralized manner, and then (2) construct a new classifier that has greater accuracy than the locally optimal classifier. In other words, if we denote this new classifier by $g_m$, then $g_m$ must have greater accuracy than $f_m$,

$$\mathrm{err}(g_m) < \mathrm{err}(f_m). \tag{2.8}$$

Since every party has the same distribution of data, $P_X \times P_{Y|X}$, there exists the notions of 'global distribution' and 'global classifier'. We denote by $g$ the global classifier, then this classifier is the same for all participating parties,

$$g \equiv g_m, \quad m = 1, \ldots, M. \tag{2.9}$$

The problem of distributed classification has a new constraint in comparison to standard learning: the *communication constraint*. Let us elaborate on this by first letting

$\mathsf{comm}(\cdot)$ denote the communication cost function. For example, $\mathsf{comm}(c)$ is the communication cost that is incurred upon computing the set of computations $c$. Likewise, $\mathsf{comm}(S_m)$ is the communication cost that is incurred upon sending the dataset $S_m$ across the network.

Distributed classification demands that we must minimize the communication cost of constructing the final classifier $g$. Ideally, the optimal cost should be as small as approaching zero, i.e.,

$$\mathsf{comm}(g_m) \to 0.$$

However, this condition is not practically feasible in distributed classification. Therefore, we can relax the condition and instead use an alternative requirement: the communication cost of constructing $g$ should be much less than that of data centralization. Since the cost of centralization is essentially the cost of sending the whole dataset $S$ over the network, we have the requirement

$$\mathsf{comm}(g_m) \ll \mathsf{comm}(S).$$

We are now ready to state the property that a solution of distributed classification must satisfy.

**Property 2.1 (Communication constraint)** *Given a party $P_m$, $m = 1, \ldots, M$, the communication cost $\mathsf{comm}(g_m)$ of learning a classifier $g_m$ should be kept optimal, such that*

$$\mathsf{comm}(g_m) \to 0 \quad \text{or} \quad \mathsf{comm}(g_m) \ll \mathsf{comm}(S), \tag{2.10}$$

*where $\mathsf{comm}(S)$ is the communication cost of centralizing the whole data available from all parties.*

## 2.3 Distributed classification with variable data distributions

By being aware of the potential variation in data distributions among the parties, the problem of distributed classification with variable distributions differs from conventional distributed classification in several aspects.

(i) First, in the previously described distributed learning environment, the $M$ parties may now have different distributions of data. In particular, each party $P_m$, $m = 1, \ldots, M$, has its own input distribution, denoted by $P_X^{(m)}$, and its own output distribution, denoted by $P_{Y|X}^{(m)}$.

(ii) Second, the local dataset $S_m$ located at the party $P_m$ is generated according to its own data distribution, that is,

$$S_m = \{(X_1, Y_1), \ldots, (X_{n_m}, Y_{n_m})\} \sim \left(P_X^{(m)} \times P_{Y|X}^{(m)}\right)^{n_m}. \tag{2.11}$$

(iii) The goal of the party $P_m$ (where $m = 1, \ldots, M$) is to (1) attempt to extract knowledge from the other parties $P_{m'}$ (where $m' \in \{1, \ldots, M\} \setminus \{m\}$) in a decentralized manner, and then (2) output the final classifier, now denoted by $g_m$, such that $g_m$ has greater accuracy than the locally optimal classifier $f_m$, *with respect to its own data distribution* $P_X^{(m)} \times P_{Y|X}^{(m)}$,

$$\text{err}\left(g_m \middle| P_X^{(m)} \times P_{Y|X}^{(m)}\right) < \text{err}\left(f_m \middle| P_X^{(m)} \times P_{Y|X}^{(m)}\right). \tag{2.12}$$

Here we use the notation $\text{err}(f|D)$ to indicate the error of a classifier $f$ with respect to some distribution $D$.

In this new setting, the union of multiple datasets from various distributions is

$$S = \bigcup_{m=1}^{M} S_m = \bigcup_{m=1}^{M} \{(X_1, Y_1), \ldots, (X_{n_i}, Y_{n_i})\}, \tag{2.13}$$

which is the analogue of the entire dataset $S$ in standard learning. However, note that this collection is now *no longer useful* to learning. In addition, the concept of a global classifier no longer exists. Therefore, the final classifier $g_m$ produced at every party $P_m$ is not necessarily the same across all parties as in conventional distributed classification (hence the appearance of the subscript $_m$ in $g_m$).

The new problem has introduced us a second and equally important constraint in distributed classification, which is the (potential) variation in data distributions. More specifically, we refer to the setting where the input and output distributions are arbitrary among the parties.

**Property 2.2 (Variation in data distributions)** *$P_X^{(m)}$ and $P_{Y|X}^{(m)}$ are arbitrary for all $m = 1, \ldots, M$. Specifically, this allows any of the following situations:*

$$\forall m \neq m' \text{ in } \{1, \ldots, M\} \ , \ P_X^{(m)} \equiv P_X^{(m')} \text{ and } P_{Y|X}^{(m)} \equiv P_{Y|X}^{(m')}. \tag{2.14}$$

$$\forall m \neq m' \text{ in } \{1, \ldots, M\} \ , \ P_X^{(m)} \neq P_X^{(m')} \text{ or } P_{Y|X}^{(m)} \neq P_{Y|X}^{(m')}. \tag{2.15}$$

We are now ready to integrate the two aforementioned properties to formally define the problem of distributed classification with variable data distributions:

**Definition 2.3 (Distributed classification with variable data distributions)** *Given a distributed system with $M$ parties, for each party $P_m$, $m = 1, \ldots, M$, the problem is to learn a classifier $g_m$ (1) by using information from all of the other parties $P_{m'}$ (where $m' \in \{1, \ldots, M\} \setminus \{m\}$) and (2) in a decentralized manner, such that*

$$\text{err}\left(g_m \middle| P_X^{(m)} \times P_{Y|X}^{(m)}\right) < \text{err}\left(f_m \middle| P_X^{(m)} \times P_{Y|X}^{(m)}\right), \quad \forall m = 1, \ldots, M \tag{2.16}$$

*while satisfying Property 2.1 and Property 2.2.*

The problem allows the communication constraint to be formally imposed, while also permitting the participating parties to have arbitrary data distributions. Since it can model different scenarios of real-world distributed settings, this problem can be regarded as the general version of conventional distributed classification.

We make a few notes on terminology. The main difference between distributed classification with variable data distributions and its conventional counterpart is that everything (including the data distribution and the final classifier) is viewed with respect to a specific party. In other words, we focus on one party only, and the goal is to learn the knowledge from the other parties to enhance the classification accuracy with respect to the local data distribution. Because of this, throughout the rest of this thesis we will use special names to differentiate between the party in consideration and the other parties. Specifically, we will use the term *target party* to indicate the party that desires to learn from others. Meanwhile, the rest of the parties that help the target party during learning are referred to by the term *source parties*.

# 3

# Conventional Distributed Classification & Limitations

In this chapter, we will give the detailed review of current works in conventional distributed classification. As discussed in chapter 1, these current approaches make an (unrealistic ) assumption that all participating parties have the same data distribution. Specifically, there exists a global distribution of inputs $P_X$ as well as a global distribution of outputs $P_{Y|X}$ at all of the parties. In other words, we have

$$P_X \equiv P_X^{(m)}, \quad m = 1, \ldots, M,$$

and

$$P_{Y|X} \equiv P_{Y|X}^{(m)}, \quad m = 1, \ldots, M.$$

Because of this, the local classifiers produced by learning at all parties are equivalent, that is,

$$g \equiv g_m, \quad m = 1, \ldots, M.$$

Therefore, there exists the notion of a global classifier, and the problem of learning the final local classifier at each party can be directly mapped to the problem of learning this global classifier. The current approaches in conventional distributed classification can be categorized into two methodologies:

(i) *Single Classifier Systems*: solutions for which all of the parties collaboratively work together to construct a *single* classifier. This (global) classifier will be broadcast to all participating parties.

(ii) *Multiple Classifier Systems*: solutions for which there are *multiple* classifiers produced at the participating parties. These classifiers will be then used together to make new predictions.

In the next two sections, we will respectively present the previous works regarding these two methodologies in details. In addition, we will also discuss their limitations in addressing the new problem of distributed classification with variable distributions.

## 3.1 Single Classifier Systems

A single classifier system constructs and uses only a single model for the classification tasks. Specifically, these approaches distribute or parallelize the computational tasks required to construct a single model over multiple parties, with the purpose of improving the generalization accuracy by learning from more data. Initially motivated by the need to improve scalability for learning on large datasets, these approaches have over time evolved for other purposes, and can now be used to resolve the problem of (conventional) distributed classification. The current approaches can be further categorized into two methodologies: *distributed optimization approaches* and *distributed non-optimization approaches*. We will respectively present the details of these two categories in the two subsections 3.1.1 and 3.1.2. The limitations of single classifier systems in general are then discussed in the subsection 3.1.3.

### 3.1.1 Distributed non-optimization approaches

These approaches work collaboratively to parallelize the computation for learning the global classifier. However, there are no mathematical optimization problems involved during computation. Moreover, the parallelization is often conducted in a delicate and intelligent manner that is suitable for some specific learning algorithms only.

One of the oldest and most common learning algorithms is decision tree [37]. [6] provides a distributed decision tree induction approach, where parties continuously propagate and collect data statistics to induce their local decision tree which eventually converges to a global model.

Support Vector Machines (SVM) is a famous learning algorithm [15,41], and is generally considered to be among the best ones. One method regarding SVMs that is particularly fitted to distributed classification is the cascade SVM approach, which is proposed in [26]. In this algorithm, the SVM models are recursively constructed from the local data of parties and the resultant support vectors of these models, and this process is repeated until the final model converges. The works [1, 2] extend this approach by using Reduced SVM to minimize the number of support vectors. As a consequence, the amount of computation and communication is reduced, making such solutions suitable for deployment in distributed environments.

Some approaches construct the global classifier by the use of *averaging*. Specifically, each of the parties first builds its local classifier using some learning algorithm, and then all of the local models are averaged to obtain the final classifier. One example is [33], which computes the mixture of multiple weight vectors for conditional maximum entropy models. Another example is [35], which uses parameter mixing to collaboratively learn the global structured perceptron classifier.

### 3.1.2 Distributed optimization approaches

In these approaches, there is a single global mathematical optimization problem to be solved. The parties together solve the problem by executing multiple local computations while collaboratively exchanging information among each other in a communication-efficient manner. Distributed optimization is an emerging trend of distributed learning, and has attracted a deluge of works over the last few years.

Distributed optimization approaches often aim to parallelize the computation workload during solving the optimization problem. Each of the parties executes some local computations, and exchanges computed values with the other parties. The most common methodology is distributed convex optimization [5]. [45] extends this to create the first parallel stochastic gradient descent algorithm. In one of the recent works, [10] discusses the well-known method of Alternating Direction Method of Multipliers (ADDM) together with its application in statistical learning. Parallelized computation of the gradients in gradient descent is also proposed in [40]. [22] works on distributed constrained convex optimization by introducing a simple subgradient approach called the 'dual averaging subgradient method'.

### 3.1.3 Limitations

Whether the approaches are optimization-based or non-optimization-based, single classifier systems focus on collecting the knowledge from all parties and use them to train the global classifier. However, the concept of such a global classifier is meaningful only when there is a global distribution across all participating parties, i.e., when the parties have the same data distribution.

In the new problem of distributed classification with variable distributions, the global distribution no longer exists, and therefore such single classifier solutions are no longer applicable. If we insist on applying the approaches to address the new problem, the

final classifier may behave in an erratic and unpredictable manner, since each party now has a distinct data distribution. Note that the final classifier can also be thought of as being built with respect to some (unknown) mixture of distributions from all parties. Because of this, the final classifier will have low error for data distributed according to this mixture, rather than for the data distribution at the target party. If we were to use the final classifier at the target party, it may have unpredictably high error exceeding that of the locally trained classifier.

Overall, we have argued that single classifier systems are not suitable to solve the problem of distributed classification with variable distributions. These approaches may lead to a uncontrollably bad classifier with respect to the target party, so that distributed classification may be worse than local learning.

## 3.2 Multiple Classifier Systems

Contrary to the single classifier systems, multiple classifier systems build multiple classifiers and then perform prediction by combining the outputs of all (or a selected subset of) classifiers. These approaches are very close in spirit to a famous field in machine learning called *ensemble learning*. Because of this, we sometimes call the solutions in this category by the name of *ensemble approaches*.

An explanation on the success and popularity of ensemble learning (and thus the associated ensemble approaches) is so-called *model diversification* (i.e., the multiple models being produced are diverse from each other). The hope is that if those classifiers make errors on different data instances, a combination of them will reduce the total error (and the problem is how to quantify a particular measure of diversity). As [12] shows, utilizing model diversification and focusing on how to make the models diverse can help us to gain improvements on accuracy.

Boosting [38] is one of the most famous ensemble learning methods, and [25] extends traditional boosting in order to build an ensemble of boosting classifiers in parallel. [31] uses an efficient approach to collect the prediction votes from the local Ivote models (proposed by [11]) of distributed parties. Stacked Generalization [42] uses a second level meta-classifier trained on the distributed ensemble of base classifiers for combining their outputs.

### 3.2.1 Limitations

Suppose there are multiple classifiers $\{f_1, f_2, \dots\}$ being produced during computation in a multiple classifier solution. The diversity between those classifiers is only meaningful, however, if they are built on datasets that come from a single distribution. When the classifiers are trained using data distributed according to different distributions, intuitively the divergence between those distributions is the main factor to affect this diversity. The greater the divergence is, the more diverse the classifiers are, and the less meaningful it is to conduct distributed classification. Therefore, when the data distributions at the participating source parties are too different from that at the target parties, the multiple local classifiers being produced might be harmful to the target party, and might degrade its generalization accuracy.

We use the following result to illustrate such situation. One of the most common methods for ensemble learning is majority voting, and we will prove that majority voting of locally optimal classifiers from all participating parties does not necessarily lead to good predictions (as opposed to the conventional case where all parties have the same data distribution).

**Theorem 3.1** *Majority voting of optimal classifiers from all parties (with different data distributions) does not guarantee an optimal solution for distributed classification.*

**Proof:** We prove the theorem by contradiction. Suppose the majority voting of optimal classifiers always results in an optimal classifier. Formally, if we consider a system with $M$ parties, and denote their optimal classifiers $\{f_1^*, \ldots, f_M^*\}$, i.e.,

$$\mathrm{err}(f_i^*) = 0 \text{ with respect to } P_X^{(i)} \text{ and } P_{Y|X}^{(i)}, \tag{3.1}$$

and consider a particular party $P_i$ which wants to learn from other parties $P_j$, for all $j \neq i$, then

$$g_i = \sum_{j \neq i}^{M} w_j f_j^*$$

is optimal, i.e., $\mathrm{err}(g_i) = 0$ with respect to $P_X^{(i)}$ and $P_{Y|X}^{(i)}$. (Note that here we use the subscripts $_i$ and $_j$, since the subscript $_m$ that has been used in Section 2 will be used later in this proof.)

We will prove that such statement is wrong. Let us consider a simple case where only $P_X$ changes over all parties, and every party has the same output distribution $P_{Y|X}$. Assume a target function $t(\cdot)$ where $t(a) = 1$ and $t(b) = 0$. The loss function $\ell$ is the absolute loss, i.e., $\ell(x, y) = |x - y|$.

Without loss of generality, suppose in $k$ out of $M - 1$ parties $P_j$, for all $j \neq i$, the local distribution $P_X$ outputs *a almost surely*, i.e.,

$$\Pr(X = a) = 1.$$

Denote the collection of these $k$ parties as $K$. In the rest of those $M - 1$ parties, whose collection is denoted by $\bar{K}$, $P_X$ always produces $b$ almost surely, i.e.,

$$\Pr(X = b) = 1.$$

Let $h_0$ be the classifier that always outputs 0, i.e., $h_0(x) = 0$ for all $x \in \mathcal{X}$, and $h_1$ be the classifier that always outputs 1, i.e., $h_1(x) = 1$ for all $x \in \mathcal{X}$. It is obvious that,

$$\mathbb{E}\,\ell(h_0) = 0 \text{ at every party in } K, \tag{3.2}$$

$$\mathbb{E}\,\ell(h_1) = 0 \text{ at every party in } \bar{K}. \tag{3.3}$$

Therefore, at every party in $K$, $h_0$ is the optimal classifier to be outputted. Similarly, at every party in $\bar{K}$, $h_1$ is the optimal classifier to be outputted.

Now consider the party $P_i$, whose local distribution $P_X^{(i)}$ puts uniform probability on all instances, i.e., $\Pr(X = a) = \Pr(X = b) = 1/2$. The expected loss of $g_i$ with respect to $P_X^{(i)}$ is

$$\begin{aligned} E\ell(g_i) &= \mathbb{E}\left[\sum_{P_m \in K} w_m f_m^* + \sum_{P_n \in \bar{K}} w_n f_n^* + w_i f_i^*\right] \\ &= \mathbb{E}\left[\sum_{P_n \in \bar{K}} w_n + w_i f_i^*\right]. \end{aligned}$$

The expectation is taken over $X = a$ and $X = b$. Since $t(a) = 1$ and $t(b) = 0$,

$$\ell(g_i|X = a) = \left|1 - \sum_{P_n \in \bar{K}} w_n - w_i f_i^*(a)\right|, \tag{3.4}$$

$$\ell(g_i|X = b) = \left|\sum_{P_n \in \bar{K}} w_n - w_i f_i^*(b)\right|. \tag{3.5}$$

Simple algebra gives us

$$\text{err}(g_i) = \mathbb{E}\,\ell(g_I) = \frac{1}{2} + \frac{w_i}{2}(f_i^*(b) - f_i^*(a)). \tag{3.6}$$

If there is shortage of data at party $P_i$ so that $f_i^*(b) = 1$ and $f_i^*(a) = 0$ after local training, or if $w_i < 1$, we all end up with $\text{err}(g_i) > 0$, which contradicts the above assumption that $\text{err}(g_i) = 0$. By contradiction, we complete the proof of the Theorem.

We see that the combination of optimal classifiers $\{f_1, \ldots, f_M\}$ may cause the final classifier $g_i$ to perform very poorly on $P_X^{(i)}$ (with expected loss greater than $1/2$ in some cases). This is because $\{P_X^{(1)}, \ldots, P_X^{(M)}\}$ are too diverse from each other; in particular, $\Pr(X = a) = 1$ for $k$ parties, $\Pr(X = a) = \Pr(X = b) = 1/2$ for $P_i$, and $\Pr(X = b) = 1$ for the rest of the parties.

# 4

# Potential Challenges & Research Directions

We have already presented the limitations of existing approaches from conventional distributed classification in Section 3. As previously discussed, current works cannot be used or extended to solve distributed classification with variable data distributions, which confirms the difficulty of the newly created problem.

In this chapter, we want to highlight the exciting *challenges* of the new problem that make it difficult. Essentially, these challenges can also be thought of as the various *sub-problems* that might arise and need to be resolved during any attempt for distributed classification with variable distributions. We will identify and present four of such potential challenges/sub-problems that emerge themselves as important within the main problem. Moreover, we will discuss several potential research directions on how to address each of those challenges. The four sub-problems, which are (1) divergence, (2) knowledge representation, (3) transferability, and (4) adaptation & integration, will be presented sequentially in the next few sections.

## 4.1 Divergence

The most intuitive challenge that can be raised with different distributions is that, even when a classifier $f_m$ is supposed to be locally optimal at some party $P_m$, it may still have poor generalization accuracy at another party $P_{m'}$ (where $m' \in \{1, \ldots, M\} \setminus \{m\}$). The reason for this behavior is because the distributions of data at the two parties $P_m$ and $P_{m'}$ may be too diverse or different from each other.

Such divergence is significantly important. In fact, if we managed to approximate the degree of divergence between the data distributions at $P_m$ and $P_{m'}$, it would be highly likely that we could properly estimate the accuracy of the optimal classifier $f_m$, which is trained according to its own distribution at $P_m$, on the 'new' distribution at $P_{m'}$. This mandates the problem of quantifying the difference or divergence between two distributions:

**Sub-Problem 4.1 (Divergence)** *The problem is to derive a measure of divergence* $\mathsf{Div}(\cdot)$ *between two probability distributions* $A$ *and* $B$ *such that*

(i) $\mathsf{Div}(A, B) = 0$ *if* $A \equiv B$,

(ii) $\mathsf{Div}(A, B) = \mathsf{Div}(B, A)$, *and*

(iii) $\mathsf{Div}(A, B) + \mathsf{Div}(B, C) > \mathsf{Div}(A, C)$ *where* $C$ *is another probability distribution.*

*In other words,* $Div(\cdot)$ *is a metric.*

Here we can think of $A$ and $B$ respectively as the distributions of data at any two parties $P_m$ and $P_{m'}$.

Obviously, an intuitive property of $\mathsf{Div}(\cdot)$ is that it should be *sensitive to the actual variation in the distributions*. For example, suppose the input distribution $P_X^{(1)}$ at party $P_1$ is only slightly more different than that at party $P_2$, denoted by $P_X^{(2)}$. Then when

they are both compared to the input distribution $P_X^{(3)}$ at party $P_3$, the measure should be aware of this slight difference, and encapsulate the following inequality

$$\mathsf{Div}\left(P_X^{(1)}, P_X^{(3)}\right) > \mathsf{Div}\left(P_X^{(2)}, P_X^{(3)}\right). \tag{4.1}$$

In this manner, some standard distances such as $L_1$ and $L_2$ norms are not acceptable because they seem to too large even for favorable adaptation situations [29].

In addition, another desired property is that it needs to be *measurable with consideration of communication constraints*. In this manner, the problem becomes quite difficult. Again, $L_1$ and $L_2$ norms are not suitable candidates since they cannot be measured from finite samples. Several solutions of distance between distributions have also been introduced in statistics, transfer learning, and domain adaptation; for example, the $\mathcal{A}$-distance, the discrepancy distance, the Kullback-Leibler (KL) divergence, the Maximal Mean Discrepancy (MMD), etc. However, these divergence measures make a requirement that data should be gathered at one place for their computation, which is not practically feasible in our problem.

To resolve this challenge, one might seek to derive a new divergence $\mathsf{Div}(\cdot)$ by extending some of the existing distance measures (e.g., the $\mathcal{A}$-distance [29], the discrepancy distance [34], or the MMD) to the context of distributed environments, in which the amount of communication incurred for the computation should be minimized as much as possible.

## 4.2 Knowledge representation

In Section 4.1, we have introduced the sub-problem of estimating the difference or divergence between data distributions at various parties. As mentioned above, some works in transfer learning have been proposed to accomplish this task. However, they need the entire collection of data from all participating parties to be centralized into one location,

and this requirement is prohibited by the communication constraint (see Property 2.1) imposed in distributed classification.

Besides estimation of the divergence function, we emphasize that this argument also applies for other types of computation. Whenever we need data from the other parties during the construction of our final classifier, it will involve high costs of data transfer over the network. Therefore, this provides a motivation for some alternatives to transferring the entire datasets from the other parties during our learning. One requirement is that such alternatives should be both *concise* and *accurate* so that they can be considered as a good representation for the data at each party.

**Sub-Problem 4.2 (Knowledge representation)** *For each party $P_m$, $m = 1, \ldots, M$, the problem is to create a representation*

$$\mathrm{Rep}_m = \mathsf{Approximate}\left(P_X^{(m)}, P_{Y|X}^{(m)}\right),$$

*such that*

$$\mathsf{comm}(\mathrm{Rep}_m) \ll \mathsf{comm}(S_m). \tag{4.2}$$

Clearly, another desired property of such representation is that it *should be sensitive to any possible change in the distribution of data*. In other words, $\mathrm{Rep}_m$ should reflect even a slight variation in $P_X^{(m)}$ (or $P_{Y|X}^{(m)}$). In this manner, the divergence between the data distributions from any two parties $P_m$ and $P_{m'}$ can be accurately estimated by approximating the difference in their knowledge representations instead. That is, we can have

$$\mathsf{Div}\left(\mathrm{Rep}_m, \mathrm{Rep}_{m'}\right) \approx \mathsf{Div}\left(P_X^{(m)}, P_X^{(m')}\right) \text{ or } \mathsf{Div}\left(P_{Y|X}^{(m)}, P_{Y|X}^{(m')}\right). \tag{4.3}$$

From this observation, during the derivation of knowledge representations, attention should be paid regarding the difference between any two representations. This difference

is very important in helping us resolve the challenge in Section 4.1, and therefore should be made to be practically computable.

As for potential research directions, one potential idea for the construction of such representations is to utilize several data summarization techniques such as clustering, Gaussian mixture models, minimum enclosing balls, etc. In addition, sometimes we can consider some simple and intuitive representations of data rather than deriving a much more complex one. For example, we may use a small and specific subset of data (e.g., through careful sampling), the optimal classifier trained on the data, or the centroids obtained by clustering the data, etc.

## 4.3 Transferability

Intuitively, learning across various distributions is not useful when the input distributions have a tendency to vary greatly among the parties. In a similar argument, if the output distributions at the participating parties are too different or diverse from each other, distributed classification may even degrade the generalization accuracy of the locally trained classifier. For both situations, auxiliary knowledge extracted from the other parties may be harmful to our learning, in which case distributed classification is in fact not meaningful to conduct (and we should consider local learning instead).

The above argument motivates the importance of realizing the effect of learning across different distributions. Specifically, we need to make a proper decision on whether or not the utilization of knowledge collected from all of the other parties will have a negative effect on the learning process at some particular party. In other words, we should make clear judgements in whether the generalization accuracy at the local party is made worse during distributed classification, since the data distributions from the other parties may be too different from ours. This issue can be formally stated as follows:

**Sub-Problem 4.3 (Transferability)** *Suppose we are given two parties $P_m$ and $P_{m'}$ with unknown divergence between the data distributions. The problem is to decide whether or not the party $P_m$ after utilizing the knowledge extracted from the party $P_{m'}$ will produce the final classifier $g_m$ for which*

$$\mathrm{err}(g_m) > \mathrm{err}(f_m).$$

*Again, the error values are measured with respect to the local data distribution at the party $P_m$.*

In short, we need to discover whether there is a so-called *negative transfer*. Here we have borrowed both of the terms 'transferability' and 'negative transfer' from the problem of transfer learning. Transfer learning (or its close cousin, domain adaptation) refers to an emerging field in which one aims to learn across distributions. We will provide a more detailed review regarding transfer learning and domain adaptation in Chapter 5 when we introduce the solution VarDist.

A few potential research directions may be suggested here. One method could be to first cluster all of the participating parties (in which is the divergence between the data distributions can be chosen as the distance function). The parties are then sorted out according to their closeness in distributions to each other. The top source parties that have the most similar distributions to the target party may be picked, since the rest of the source parties may have too diverse distributions and lead to negative transfer.

Another approach is to define some measure of transferability based on the previously defined distance or divergence measures. For example, one viable route is that we have a systematic selection methodology of a threshold $\delta$, such that whenever the divergence between any two parties $P_m$ and $P_{m'}$ exceeds this value, i.e.,

$$\mathsf{Div}\left(P_X^{(m)}, P_X^{(m')}\right) \geq \delta \text{ or } \mathsf{Div}\left(P_{Y|X}^{(m)}, P_{Y|X}^{(m')}\right) \geq \delta$$

it actually indicates negative transfer, in which case we are advised against the harmful collaborative learning of knowledge between the two parties $P_m$ and $P_{m'}$.

## 4.4 Adaptation and Integration

The final challenge is special in that it should be dealt with only after we have already tackled the first three sub-problems. Suppose that:

- We have managed to measure (at least to a certain degree of accuracy) the empirical divergence between two distributions of the two parties $P_m$ and $P_{m'}$.
- We have also discovered that the collaboration between such two parties will actually provide a boost to the generalization accuracy of locally trained classifiers. In other words, we are highly certain about a positive transfer.
- Finally, we have successfully constructed a simplified yet accurate model of representation knowledge, which can then be used as a representation for any particular party.

To address the problem of distributed classification, we need to accomplish the last task (which is arguably the most important step): utilizing such beneficial knowledge extracted from the party $P_{m'}$ in order to gain an improvement to the local generalization accuracy at the party $P_m$.

- The first step would be attempting to bridge the potential gap between data distributions of the two parties. We call this step *adaptation*, which is a term borrowed from the problem of domain adaptation.
- Suppose we have already decided to select some adaptation method $\mathsf{Adapt}(\cdot)$, and we are now ready to adapt any new knowledge from other data distributions to the target party. Since the target party also has some (insufficient) data itself, we need a procedure $\mathsf{Integrate}(\cdot)$ to combine the previous result of adaptation with the process of local learning on the available local dataset. We call this step *integration* since it facilitates integration of auxiliary knowledge into the target party.

These two steps can be formalized in the following sub-problem:

**Sub-Problem 4.4 (Adaptation and Integration)** *Given the target party $P_m$, the problem is to derive two methods. The first method is*

$$\mathsf{Adapt}\left(f_{m'}, P_X^{(m)}, P_{Y|X}^{(m)}\right), \quad \forall m' \neq m. \tag{4.4}$$

*In other words, other party classifiers must be able to adapt to $P_X^{(m)}$ or $P_{Y|X}^{(m)}$ be using the function* $\mathsf{Adapt}(\cdot)$*. The second method is*

$$g_m = \mathsf{Integrate}\left(\left\{\mathsf{Adapt}\left(f_{m'}, P_X^{(m)}, P_{Y|X}^{(m)}\right) \middle| \forall m' \neq m\right\}, f_m, S_m\right). \tag{4.5}$$

*In other words, the previous result of adaptation must be able to be integrated (through the function* $\mathsf{Integrate}(\cdot)$*) to either (1) the locally trained classifier $f_m$ or (2) the local dataset $S_m$ at the party $P_m$. Overall, the combination of* $\mathsf{Adapt}(\cdot)$ *and* $\mathsf{Integrate}(\cdot)$ *should lead to the final classifier $g_m$ such that*

$$\mathrm{err}\left(g_m \;\middle|\; P_X^{(m)}, P_{Y|X}^{(m)}\right) \leq \mathrm{err}\left(f_m \;\middle|\; P_X^{(m)}, P_{Y|X}^{(m)}\right). \tag{4.6}$$

This sub-problem is essentially the main focus of transfer learning and domain adaptation (in which we try to adapt from some source domains to a target domain to improve the accuracy of the target classifier). However, the sub-problem is much more challenging due to the communication constraint. Specifically, while attempting to resolve the tasks of adaptation and integration, a desired algorithm must also be highly aware of the amount of communication overheads. This is ultimately necessary: we have mentioned above that any solution, no matter how good it is at boosting the generalization accuracy at the target party, would not be considered practically useful if it simply assumed the setting of centralized learning and ignored the cost of information exchange. With this criterion in mind, many works in transfer learning and domain adaptation (e.g., [28]) are

not applicable since they require full access to all datasets (including those at the source parties) during training.

One research direction that we might want to consider is similar to [32]. This work proposes a consensus regularization method for transfer learning from multiple source domains, and focuses particularly on an efficient protocol for distributed environments. However, it doesn't take advantage of the availability of labeled data at the target party. This limitation can be hopefully addressed by some works on domain adaptation in which target labeled data are delicately exploited [28]. However, we highlight that combination of the two approaches (or many of their variations) might not be trivial, requiring careful and extensive research.

# 5

# VarDist: A Feasible Approach

In this section we aim to make a preliminary attempt, through an algorithm called VarDist, to address the problem of distributed classification with variable distributions. Note that multiple challenges of the new problem essentially lie in the fact data distributions vary among the participating parties (see the sub-problems 4.1, 4.3 and 4.4 in Chapter4). As a consequence, we need to investigate some methodologies that can learn across distributions.

One such methodology is from a field called *transfer learning* (and its cousin, *domain adaptation*), and we will first have a brief overview on such fields in Section 5.1. The idea of VarDist is essentially built based on transfer learning/domain adaptation, and is discussed in Section 5.2. In Section 5.3, we present the full algorithm, which is a communication-efficient protocol among the participating parties. Finally, in Section 5.4 we provide two results for the performance of VarDist:

(i) We first provide an analysis on the *communication cost* of the protocol.

(ii) We then give many experimental results regarding the *generalization accuracy* of VarDist.

From these results we will see that VarDist essentially satisfies the communication constraint (property 2.1), while leading to a gain in accuracy for the target party when the parties have different data distributions (property 2.2). Together, VarDist fulfils the requirement specified in the problem 2.3 (stated in Chapter 2), and therefore can be regarded as an initial (preliminary) solution to the problem of distributed classification with variable distributions.

## 5.1 Transfer Learning & Domain Adaptation

Transfer learning is a popular and emerging field in recent years, and has attracted increasing attention in recent years. We are given two different learning environments $A$ (which we call the *source domain*) and $B$ (which we call the *target domain*). By 'different', we mean that the two domains have different data distributions. Transfer learning and domain adaptation sets out to analyze how data in the source domain $A$ (or the knowledge extracted from such data) can be *'transferred'* or *'adapted'* to the target domain $B$, in order to help improve the learning performance in $B$. It can be seen that the problem of distributed classification with variable distributions is closely related to the fields of transfer learning and domain adaptation. Our goal is to borrow the idea from these two fields in solving our new problem.

The wide applicability of transfer learning has allowed it to attract a deluge of works over the last few years. Since our intention is to study the applicability of transfer learning works in the distributed environment, we present only some of the most representative works here. We *do not* attempt to make a comprehensive literature review of transfer learning and domain adaptation. An interested reader may want to find out more details by referring to [36] for an excellent survey on transfer learning.

A common class of approaches in transfer learning and domain adaptation is *reweighting*. Specifically, we establish the difference in data distributions as a ratio, and then re-weight the source data accordingly so that it can be used on the target data. Huang et al. [27] estimated this ratio by matching the means of the data of the source and target domains in a reproducing kernel Hilbert space (RKHS). [39] proposed another approach for estimating the ratio by minimizing the Kullback-Leibler divergence. TrAdaBoost [16] extended the Adaboost algorithm to train an ensemble of classifier on the weighted source and target data. The weights of the models and instances (of both source and target domains) are dynamically determined by the accuracy of the trained models on the target domain data.

Another methodology is the so-called *multi-task learning*, for which we learn from a set of multiple related tasks. [3] aims to minimize the error by solving the convex optimization problem, which leads to a set of common (and sparse) features that can be used for all tasks. [30] modeled the relevance of features to the prediction task, called meta-priors, by introducing a set of feature weights and model hyper-parameters, which can be obtained using standard convex optimization computations. [9] investigated multi-task learning in the context of Gaussian Processes (GP), modeling the similarities between different tasks using a shared covariance function and a 'free-form' covariance matrix.

Note that current transfer learning techniques assume that data in both source and target domains are readily available at our disposal. This is not a practical assumption in the context of distributed classification, where communication constraints restrict data centralization. Therefore, *most* works in transfer learning, including those presented above, fail to be applicable in solving our new problem. In the next section, we present a few class of transfer learning approaches (as categorized by [36]), and further investigate why they fail in addressing the new problem. We will see that there is one category that is appropriately suitable for distributed classification with variable distributions (although not originally intended to be so).

## 5.2 Principle of VarDist: Parameter Transfer

In the excellent survey, Pan et al. [36] classifies the current transfer learning techniques into four different categories, including *instance transfer*, *feature representation transfer*, *parameter transfer*, and *relational knowledge transfer*. Understanding these categories is highly important for the principle of VarDist, so we will give a short overview of them below.

(i) **Instance transfer**: In these approaches, we re-weight the data in the source domain to make it 'closer' to the target domain. This essentially corresponds to the reweighting method described in Section 5.1.

(ii) **Feature representation transfer**: In these approaches, we find a 'good' feature representation with the purpose of minimizing the divergence between the source and target domains.

(iii) **Parameter transfer**: In these approaches, we assume the source and target domains have similar or related parameters, so that the parameters extracted on one can be used for others. Most techniques belonging to this approach are closely related to multi-task learning (described in Section 5.1).

(iv) **Relational knowledge transfer**: In this approach, the data from each domain (1) are not assumed to be independent and identically distributed, and (2) can be described using relations (e.g., social networks). Our goal is to transfer these relations within the data to another domain.

The relational-knowledge-transfer approaches make the assumption that each domain data are not independent and identically distributed, and therefore they are not appropriately applicable according to our problem definition 2.3 in Chapter 2. Meanwhile, the instance-transfer approaches requires access to both source and target datasets in

order to learn the weights for the source domain. This is also not desirable since the source and target data are located at different parties in our problem. Similarly, the feature-representation-transfer approaches also need data from both domains to learn an appropriate feature representation, and therefore not applicable to distributed settings.

The last class of techniques, i.e., parameter-transfer approaches, deserves our attention. A large portion of this class involves multi-task learning, which uses data from both domains to learn the shared or related parameters, and therefore undesirable. However, the idea of parameter transfer can be applied to distributed learning, since the communication cost of transferring the parameters over the network is much less than that of transferring the whole dataset itself. As a consequence, we can use this idea to resolve our problem, and parameter transfer is in fact the main principle behind our solution VarDist.

## 5.3 VarDist

Before continuing with the construction of the main algorithm, we first make some notation suitable for the representation of VarDist. We assume that there are $K$ source parties, and denote them by

$$P_1^{(S)}, \ldots, P_K^{(S)},$$

where the superscript $^{(S)}$ is used to indicate a source party. Similarly, the target party is denoted by $P^{(T)}$ (and the superscript $^{(T)}$ indicates the target party).

In order to construct the algorithm VarDist, we need to first address the following issues that are closely related to the principle of parameter transfer:

(i) What parameters should we transfer?

(ii) How do we use the transferred parameters?

We will discuss the viable solution for these two questions in the next subsections.

### 5.3.1 Parameters to transfer

To answer the first question, one particular intuitive choice of parameters that we can transfer is the classifier that are locally trained using the local datasets at the source parties. In other words, we aim to use the so-called *auxiliary classifiers*. Since these models come from training on the data at source parties, we also call them *source classifiers*. We denote by $f_k^{(S)}$ the source classifier that is transferred from the source party $P_k^{(S)}$, where $k = 1, \ldots, K$.

However, we emphasize that the classification algorithm used for building each source classifier $f_k^{(S)}$ is also important. For example, if we were to use decision tree algorithms, the size of the entire output tree may be so significantly high that it cannot be efficiently transferred over the network. Another algorithm that one might be interested in for (its high accuracy) is Support Vector Machines (SVM). However, if standard SVM is used (with kernels), the resulting number of support vectors may be big, in which case sending such a classifier to the target party essentially means that we have to transfer a large portion of data to the target party.

Because of this, we decide to use linear classifiers in VarDist. The main advantage of linear classification is that the amount of communication is limited to only one array whose size is the number of features (or data columns). As a consequence, transferring linear classifiers makes VarDist highly efficient in terms of communication costs.

The classifier that is used in VarDist is the L2-regularized L2-loss linear support vector classification (SVC). This learning algorithm solves the following mathematical optimization problem:

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\left(\max(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i)\right)^2, \tag{5.1}$$

where $\mathbf{w}$ is the weight vector of the linear classification, and $l$ is the size of data at the source party in consideration. The main reason why we use Linear SVC in VarDist is that

such algorithm is known to have high performance (in terms of both speed and accuracy) on sparse document datasets, which are often encountered in transfer learning and domain adaptation. In VarDist, we use the library Liblinear [23] for the implementation of L2-regularized L2-loss linear support vector classification.

### 5.3.2 Adaptation of the transferred parameters

We have decided to use auxiliary linear SVC classifiers as the parameters to transfer to the target party. The next question is how to integrate these models into the training process on the available target data. For such adaptive learning with help from auxiliary classifiers, there have been a few works in domain adaptation that are particularly useful to be applied here. A few examples can be listed as [20,21,43]. However, both [20] and [21] assume (and make use of) the unlabeled dataset in the target domain. Therefore, such works are not applicable to our setting where only labeled data is given at the target party.

For this reason, we choose A-SVM (short for Adaptive-SVM) in [43] as the adaptation algorithm to solve our problem. A-SVM extends the standard SVM algorithm by trying to solve the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i\sum_{k=1}^{K} t_k f_k^{(S)}(\mathbf{x}_i) + y_i(\mathbf{w}^T\phi(\mathbf{x}_i)+b) \geq 1-\xi_i, i=1,\ldots,l, \\ & \xi_i \geq 0, i=1,\ldots,l. \end{aligned} \tag{5.2}$$

where $t_k$, $k = 1,\ldots,K$, is the weight for the source party $P_k^{(S)}$, and $\sum_{k=1}^{K} t_k = 1$. Once we have solved this problem and obtained the optimal values $\mathbf{w}^*$ and $b^*$, then the adapted classifier is defined as follows:

$$f(\mathbf{x}) = \sum_{k=1}^{K} t_k f_k^{(S)}(\mathbf{x}) + \mathbf{w}^{*T}\phi(\mathbf{x}_i) + b^* \tag{5.3}$$

and the decision function is

$$\text{sign}(f(\mathbf{x})).$$

### 5.3.3 The VarDist algorithm

We are now ready to present the solution VarDist. The algorithm is actually a protocol between the source parties and the target party. There are two phases. In the first phase (called the *source training* phase), each source party trains a linear classifier by using the L2-regularized L2-loss linear SVC algorithm. Next, all of the source parties transfer the weight vectors of their locally trained classifiers to the target party. In the second phase (called the *target adaptation* phase), the target party then tries to solve the adaptation problem (5.2) with the received auxiliary classifiers. The whole procedure is presented in Algorithm 1.

**Algorithm 1:** The VarDist Algorithm

**Input**: Source parties $P_1^{(S)}, \ldots, P_K^{(S)}$, Target party $P^{(T)}$
**Output**: Final (adapted) classifier $g$

**SOURCE TRAINING PHASE:**
**for** $k = 1, \ldots, K$ **do**
    Source party $P_k^{(S)}$ constructs the source classifier $f_k^{(S)}$ by solving the problem (5.1).
    Source party $P_k^{(S)}$ sends $f_k^{(S)}$ to Target party $P^{(T)}$.
**end**
**TARGET ADAPTATION PHASE:**
Target party $P^{(T)}$ solves the problem (5.2) and obtains $\mathbf{w}^*$ and $b^*$.
Target party $P^{(T)}$ outputs $g$ as specified in (5.3).

The dual optimization problem associated with (5.2) is:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{l}(1-\lambda_i)\alpha - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j) \\ \text{subject to} \quad & \sum_{i=1}^{l}\alpha_i y_i = 0, \\ & 0 \le \alpha_i \le C, i = 1,\ldots,l. \end{aligned} \tag{5.4}$$

where

$$\lambda_i = y_i \sum_{k=1}^{K} t_k f_k^{(S)}(\mathbf{x}_i).$$

In order to solve the adaptation problem (5.4), one may follow [43] to obtain the solution. However, we make a slight modification here in that we do not use the modified SMO strategy in [43]. Instead, in the VarDist algorithm we use the second-order working set selection proposed by [24]. The first reason why we use this method is that it is highly efficient and has been successfully used to solve standard SVM (resulting in one of the fastest SVM solvers – LibSVM [13]).

By rewriting the dual optimization problem (5.4) as

$$\begin{aligned} \min_{\alpha} \quad & f(\alpha) \\ \text{subject to} \quad & y^T\alpha = 0, \\ & 0 \le \alpha_i \le C, i = 1,\ldots,l, \end{aligned} \tag{5.5}$$

where

$$f(\alpha) = \frac{1}{2}\alpha^T Q\alpha + p^T\alpha,$$

and

$$p_i = (\lambda_i - 1)\alpha_i, \quad i = 1,\ldots,n,$$

we can cast the problem into the suitable form that is ready to be solved by LibSVM [13]. In this manner, VarDist uses LibSVM as one of its components (though we solve a different optimization problem here).

After we have solved the dual problem (5.4) and obtained the optimal value $\alpha^*$, the adapted classifier can be rewritten as follows:

$$f(\mathbf{x}) = \sum_{k=1}^{K} t_k f_k^{(S)}(\mathbf{x}) + \sum_{i=1}^{n} \alpha_i^* y_i K(\mathbf{x}_i, \mathbf{x}) + b^* \tag{5.6}$$

while the decision function is still $\text{sign}(f(\mathbf{x}))$.

## 5.4 Evaluation

### 5.4.1 Analysis of Communication cost

It can be seen that the VarDist algorithm requires only one-time transfer of an array of weights between a source party and the target party. Specifically, for each source party, only a weight vector $w^*$ and one value $b^*$ are sent, and therefore the amount of communication between one source party and the target party is

$$O(d+1),$$

where $d$ is the number of features in the dataset. Note that this communication cost is the same for all of the source parties. As a consequence, the total amount of network communication incurred during executing VarDist is

$$\mathsf{comm}(\mathsf{VarDist}) = O\left((d+1) \times K\right).$$

Meanwhile, the cost of communication when we transfer the whole dataset $S_k$ from a source party $P_k$:

$$O\left(n_k \times d\right),$$

where $n_k$ is the size of the dataset $S_k$. The total amount of communication incurred for data centralization is therefore

$$\mathsf{comm}(\mathsf{Centralize}) = O\left(d \times \sum_{k=1}^{K} n_k\right).$$

It can be seen that this amount is significantly higher than the communication cost required for VarDist. We come to a conclusion that VarDist is communication-efficient, and satisfies Property 2.1 regarding the communication requirement (specified in Chapter 2).

### 5.4.2 Experiments on Accuracy

In this section, we provide experimental results regarding the classification accuracy of VarDist. We compare between 3 different algorithms, including VarDist, SourceOnly, and TargetOnly. The SourceOnly algorithm learns a classifier using only the source data (that is, the union of data from all of the source parties), while the TargetOnly algorithm learns a classifier using only the target data (at the target party). It is clear that the SourceOnly algorithm incurs high amounts of data transfer over the network and is not practically desirable.

We experiment with 2 text datasets: (1) *20 Newsgroups* and (2) *Reuters-21578*. Both datasets have a hierarchy in which under the top categories there is a number of subcategories. We divide the datasets in exactly the same manner as [16]: data is split into source and target datasets based on the subcategories.

- There are 5 datasets within in the 20 Newsgroups dataset, namely $<$*comp vs sci*$>$, $<$*comp vs talk*$>$, $<$*rec vs sci*$>$, $<$*rec vs talk*$>$, and $<$*sci vs talk*$>$. Here the datset $<A$ *vs* $B>$ has positive instances in $A$ and negative instances in $B$. For each dataset $<A$ *vs* $B>$, we have a source dataset and a target dataset, namely $<A$ *vs* $B>$.*src* and $<A$ *vs* $B>$.*tar* respectively.

- Meanwhile, the Reuters-21578 dataset also contains 3 datasets within it: $<$*orgs vs people*$>$, $<$*orgs vs places*$>$, $<$*people vs places*$>$. How the datasets are obtained can be explained in the same way as the 20 Newsgroups dataset.

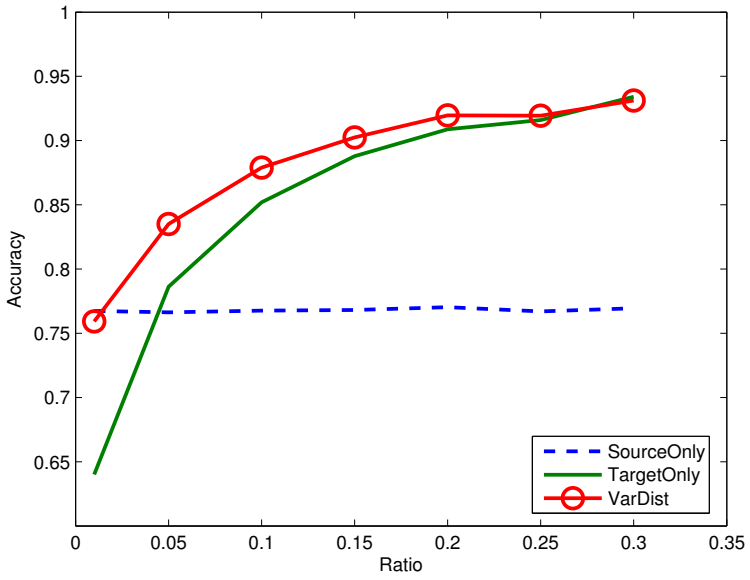


Figure 5.1: accuracy of sci vs talk, number of source = 1

All datasets can be downloaded from http://www.cse.ust.hk/TL/index.html.

The number of source parties is taken as an input, and the total source data is divided into multiple datasets, one for each source party. Another input is the ratio of source data size versus target data size, i.e.,

$$\text{ratio} = \frac{\text{target data size}}{\text{source data size}},$$

which indicates how small the target data is in comparison with the source data. Typically this ratio should be small; otherwise, the number of target data instances would be big enough for good classification results (and thus there is no need to conduct distributed classification). In the experiments we vary the ratio from 0.01 to 0.3.

We show some selected and interesting results here, and other combinations yield qualitatively analogous outcome. It can be seen that in most of the datasets, VarDist always outperforms the other two algorithms. The margin by which VarDist outperforms TargetOnly depends highly on the closeness between the source and the target domains, illustrated by the performance of SourceOnly.

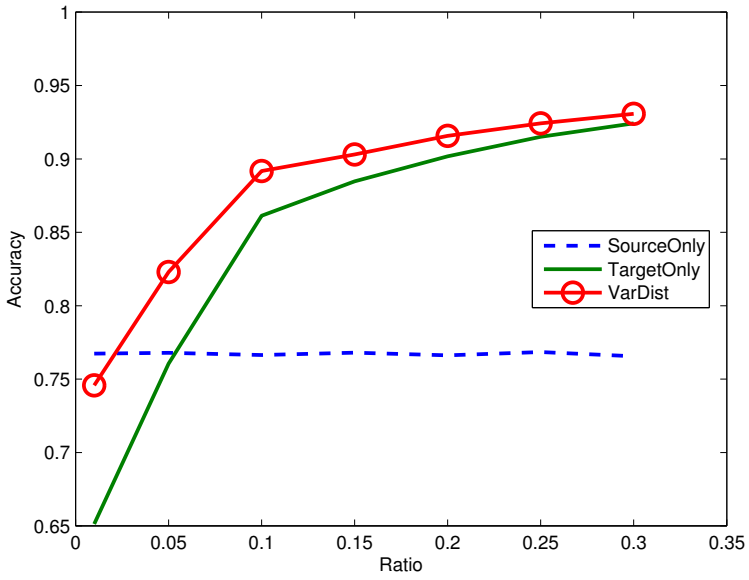


Figure 5.2: accuracy of sci vs talk, number of source = 2

- When SourceOnly has high accuracy on the target domain (e.g., <*sci vs talk*>, <*orgs vs people*>), learning with auxiliary source classifiers helps us boost the accuracy of VarDist over TargetOnly by a certain margin.

- When SourceOnly has moderate performance on the target domain (e.g., <*comp vs sci*>), learning with help from the source parties does not actually enhance our locally trained classifier. Therefore, the performance of VarDist and TargetOnly are almost similar, and distributed classification is neither useful nor useless.

- The most interesting dataset is <*people vs places*>, where the SourceOnly algorithm has very bad performance on the target domain (only slightly better than random guess). In this case, using the source auxiliary classifiers is harmful to learning, resulting in the worse accuracy of VarDist than TargetOnly. This indicates negative transfer.

In addition, consider the three plots with the <*sci vs talk*> dataset. Here we vary

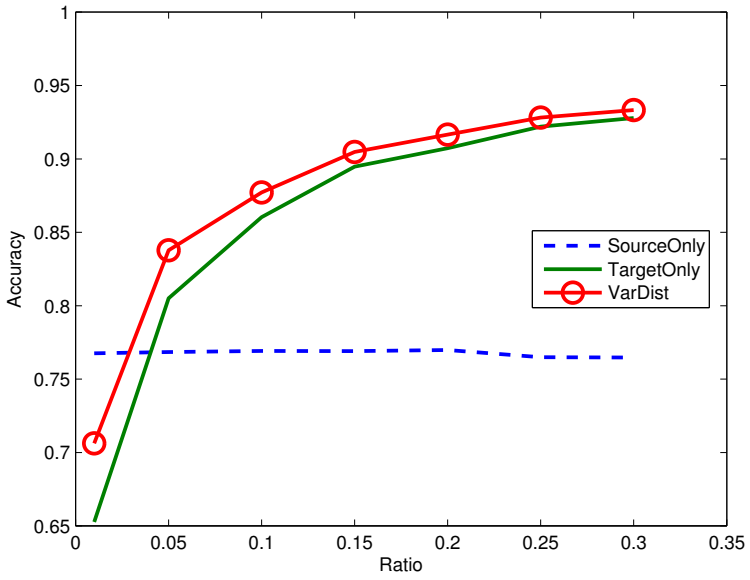


Figure 5.3: accuracy of sci vs talk, number of source = 10

the number of source parties (1, 2, and 10). It can be seen that the number of parties have little impact on the performance of VarDist. Again, experiments with other datasets when varying the number of sources yield similar results. All of the results show that VarDist have good generalization accuracy in comparison with the two baselines where we use either the source data or the target data only. The reason is that VarDist tries to *adapt* the classifiers from the source parties to the target party.

Overall, we conclude that VarDist is able to (1) be aware of the difference in the distributions between the source parties and the target party, and (2) provide an adaptive solution that helps to increase the accuracy at the party by a certain margin. As a consequence, VarDist can be regarded as an appropriate (preliminary) solution for the problem of distributed classification with variable distributions.

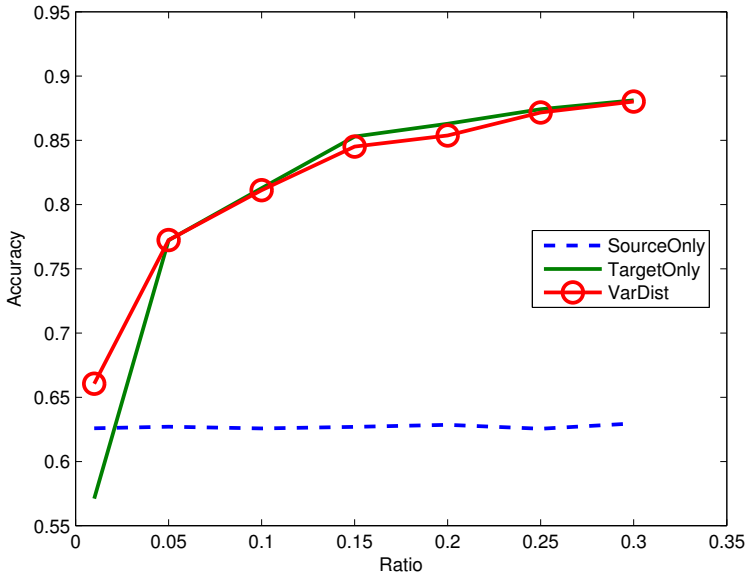


Figure 5.4: accuracy of comp vs sci, number of source = 2

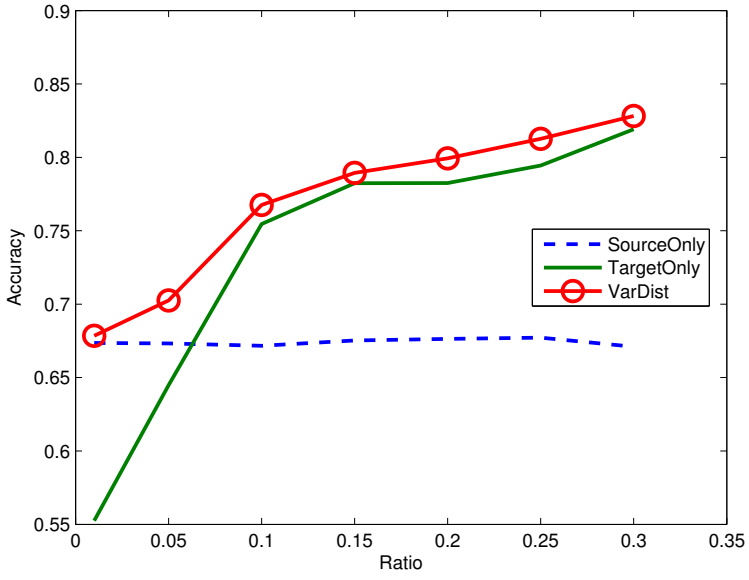


Figure 5.5: accuracy of orgs vs people, number of source = 2

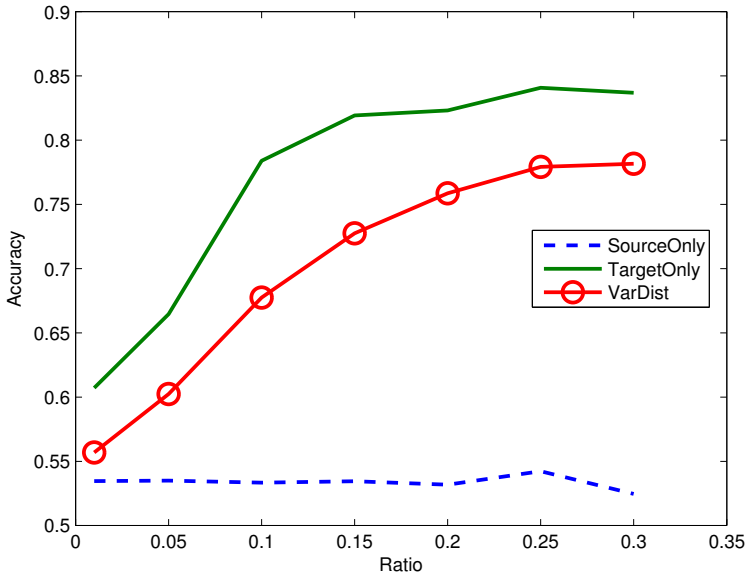


Figure 5.6: accuracy of people vs places, number of source = 2

# 6
# Conclusion

Although distributed classification has attracted growing attention from the community, current approaches in this field make a impractical assumption that the participating parties have the same data distribution. Through a variety of real-world examples, we introduce multiple scenarios that lead to various input distributions and/or various output distributions among the parties. The problem of distributed classification in these cases is difficult because the concepts of a global distribution and a global classifier (which are at the heart of conventional distributed classification) no longer exist.

In this paper, we formally define the problem of distributed classification with variable data distributions. This newly-posed problem requires solutions to be aware of the various distributions across the parties and, like conventional distributed classification, of the communication constraint in distributed environments. We illustrated the non-triviality of the new problem by showing the incompetence of existing approaches in conventional distributed classification; in particular, we showed that current techniques are not suitable to address the variation in distributions between the parties. We then discussed four

potential challenges/sub-problems that may arise within the main problem. For each of those sub-problems, we also suggested some potential research directions that help to resolve it.

As an initial attempt to address distributed classification with variable data distributions, we proposed an affirmative solution called VarDist. The main principle behind VarDist is that of parameter transfer, and the idea is to transfer the classifiers that are trained locally on the source parties. The adaptation phase is executed at the target party, and VarDist utilizes the idea of Adaptive-SVM to learn using those auxiliary classifiers. Experimental results show that our algorithm is able to provide a more accurate classifier by learning from the source parties, even when they have a different data distribution from the target party. In addition, the cost of communication in VarDist is significantly smaller than that of centralizing all data sources. VarDist serves as a preliminary solution for anyone who is interested in conducting distributed classification on parties where data distributions are potentially different.

There are two directions that can be considered as future work for this thesis.

(i) The first is detailed investigation of the transferability issue. Specifically, we need to examine whether the source auxiliary classifier might result in negative transfer or not. This would help us in deciding which classifiers should be used for helping to gain the accuracy at the target party, and which ones are harmful and should be discarded.

(ii) The second is consideration of adding privacy in distributed classification. Privacy is a highly important topic that has received increasing attention over the last decade. Privacy should be a huge concern especially in distributed classification, since a party might not be comfortable with sharing knowledge with the other parties. An emerging trend is differential privacy, and the work [14] has proposed

some algorithms on how to add differential privacy to empirical risk minimizers (including logistic regression, linear SVMs, etc.) Since we already have the auxiliary classifiers in the form of weights, this work would be an interesting direction for further exploration.

# 7 List of Author's Publications

## 7.1 Accepted Paper

(i) Quach Vinh Thanh, Vivekanand Gopalkrishnan, and Hock Hee Ang. Distributed classification on peers with variable data spaces and distributions. In *ICDM Workshops*, 2010.

# References


[1] Hock Hee Ang, Vivekanand Gopalkrishnan, Steven C. H. Hoi, and Wee Keong Ng. Cascade rsvm in peer-to-peer networks. In *ECML/PKDD (1)*, pages 55–70, 2008.

[2] Hock Hee Ang, Vivekanand Gopalkrishnan, Steven C. H. Hoi, and Wee Keong Ng. Classification in P2P networks with cascade support vector machines. *TKDD*, 7(4):20, 2013.

[3] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems*, pages 41–48, 2006.

[4] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175, 2010.

[5] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and distributed computation.* Prentice Hall, 1989.

[6] Kanishka Bhaduri, Ran Wolff, Chris Giannella, and Hillol Kargupta. Distributed decision-tree induction in peer-to-peer systems. *Statistical Analysis and Data Mining*, 1(2):85–103, 2008.

[7] John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *ACL 2007,*

*Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*, 2007.

[8] John Blitzer, Ryan T. McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *EMNLP 2007, Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, 22-23 July 2006, Sydney, Australia*, pages 120–128, 2006.

[9] Edwin V. Bonilla, Kian Ming Adam Chai, and Christopher K. I. Williams. Multi-task gaussian process prediction. In *Advances in Neural Information Processing Systems*, 2007.

[10] Stephen P. Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[11] Leo Breiman. Pasting small votes for classification in large databases and on-line. *Machine Learning*, 36(1-2):85–103, 1999.

[12] Gavin Brown. *Diversity in neural network ensembles*. PhD thesis, School of Computer Science, University of Birmingham, 2004.

[13] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM TIST*, 2(3):27, 2011.

[14] Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12:1069–1109, 2011.

[15] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[16] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *ICML*, pages 193–200, 2007.

[17] Luc Devroye, László Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition (Stochastic Modelling and Applied Probability)*, volume 31 of *Applications of Mathematics*. Springer Verlag, 1996.

[18] Chuong B. Do and Andrew Y. Ng. Transfer learning for text classification. In *Advances in Neural Information Processing Systems*, 2005.

[19] Mark Dredze, Alex Kulesza, and Koby Crammer. Multi-domain learning by confidence-weighted parameter combination. *Machine Learning*, 79(1-2):123–149, 2010.

[20] Lixin Duan, Ivor W. Tsang, Dong Xu, and Tat-Seng Chua. Domain adaptation from multiple sources via auxiliary classifiers. In *ICML*, page 37, 2009.

[21] Lixin Duan, Dong Xu, and Ivor Wai-Hung Tsang. Domain adaptation from multiple sources: A domain-dependent regularization approach. *IEEE Trans. Neural Netw. Learning Syst.*, 23(3):504–518, 2012.

[22] John C. Duchi, Alekh Agarwal, and Martin J. Wainwright. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Trans. Automat. Contr.*, 57(3):592–606, 2012.

[23] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[24] Rong-En Fan, Pai-Hsuen Chen, and Chih-Jen Lin. Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, 6:1889–1918, 2005.

[25] Wei Fan, Salvatore J. Stolfo, and Junxin Zhang. The application of adaboost for distributed, scalable and on-line learning. In *KDD*, pages 362–366, 1999.

[26] Hans Peter Graf, Eric Cosatto, Léon Bottou, Igor Durdanovic, and Vladimir Vapnik. Parallel support vector machines: The cascade SVM. In *Advances in Neural Information Processing Systems*, 2004.

[27] Jiayuan Huang, Alexander J. Smola, Arthur Gretton, Karsten M. Borgwardt, and Bernhard Schölkopf. Correcting sample selection bias by unlabeled data. In *Advances in Neural Information Processing Systems*, pages 601–608, 2006.

[28] Hal Daumé III. Frustratingly easy domain adaptation. In *ACL*, 2007.

[29] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *VLDB*, pages 180–191, 2004.

[30] Su-In Lee, Vassil Chatalbashev, David Vickrey, and Daphne Koller. Learning a meta-level prior for feature relevance from multiple related tasks. In *ICML*, pages 489–496, 2007.

[31] Ping Luo, Hui Xiong, Kevin Lü, and Zhongzhi Shi. Distributed classification in peer-to-peer networks. In *KDD*, pages 968–976, 2007.

[32] Ping Luo, Fuzhen Zhuang, Hui Xiong, Yuhong Xiong, and Qing He. Transfer learning from multiple source domains via consensus regularization. In *CIKM*, pages 103–112, 2008.

[33] Gideon Mann, Ryan T. McDonald, Mehryar Mohri, Nathan Silberman, and Dan Walker. Efficient large-scale distributed training of conditional maximum entropy models. In *Advances in Neural Information Processing Systems*, pages 1231–1239, 2009.

[34] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. In *COLT*, 2009.

[35] Ryan T. McDonald, Keith B. Hall, and Gideon Mann. Distributed training strategies for the structured perceptron. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4, 2010, Los Angeles, California, USA*, pages 456–464, 2010.

[36] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.*, 22(10):1345–1359, 2010.

[37] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[38] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

[39] Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in Neural Information Processing Systems*, 2007.

[40] Choon Hui Teo, Alex J. Smola, S. V. N. Vishwanathan, and Quoc V. Le. A scalable modular convex solver for regularized risk minimization. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007*, pages 727–736, 2007.

[41] Vladimir Vapnik. *Statistical learning theory*. Wiley, 1998.

[42] David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

[43] Jun Yang, Rong Yan, and Alexander G. Hauptmann. Cross-domain video concept detection using adaptive svms. In *ACM Multimedia*, pages 188–197, 2007.

[44] Yi Yao and Gianfranco Doretto. Boosting for transfer learning with multiple sources. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, pages 1855–1862, 2010.

[45] Martin Zinkevich, Markus Weimer, Alexander J. Smola, and Lihong Li. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 2595–2603, 2010.