COMPARISON OF TWO MODELS IN DIFFERENTIALLY PRIVATE DISTRIBUTED LEARNING

BY LIYANG XIE

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Professor Anand D. Sarwate

and approved by

_____

_____

_____

New Brunswick, New Jersey

January, 2016

ABSTRACT OF THE THESIS

Comparison of two models in differentially private distributed learning

by Liyang Xie

Thesis Director: Professor Anand D. Sarwate

Designing medical systems that can automatically diagnose patient's conditions from test data can greatly improve healthcare systems. With the help of machine learning tools and differential privacy consideration, this system can be made more efficient and powerful. Empirical risk minimization is a common and useful technique with which we can obtain a good approximation of globally optimal classifier and thus give good statistical classification result. Firstly we introduce three models for medical data learning and two methods for distributed model. Then we compare a novel distributed classification method which we called the "feature method" with traditional averaging method on different real world data sets to gain an insight into their performance and properties. Next we give analysis on the performance of the feature method under non-private and differentially private conditions and conduct some experiments to draw several important conclusions from them. Finally we conclude that the distributed learning system we recommend achieve the best result among the three models.

Acknowledgements

I take this opportunity to express gratitude to my advisor, Professor Anand D. Sarwate, for sharing expertise, sincere and valuable guidance to me. I also thank professor Waheed U. Bajwa and professor Kristin J. Dana, for the support and attention. I am also grateful to my parents who encourage me through this time.

This work was supported in part by the National Science Foundation (NSF) under award CCF-1453432 and by the NIH under award 1R01DA040487-01A1.

Contents

List of Figures

Abbreviations

i.o.	**i**nfinitely **o**ften
ERM	**E**mpirical **R**isk **M**inimization
i.i.d	**i**ndependent **i**dentically **d**istributed
w.r.t	**w**ith **r**espect **t**o
w.p.	**w**ith **p**robability
w.h.p.	**w**ith **h**igh **p**robability
PCA	**P**rincipal **C**omponent **A**nalysis
SVM	**S**upport **V**ector **M**achine
s.c.	**s**trongly **c**onvex
MSE	**M**ean **S**quare **E**rror
W.l.o.g	**W**ithout **l**oss **o**f **g**enerality
p.d.	**p**ositive **d**efinite
s.t.	**s**uch **t**hat

Symbols

f	linear classifier
f_{priv}	differential private linear classifier
$f^* = \text{argmin} \int l(f^T x, y) dP$	a given global optimal classifier, act as a reference, fixed but unknown.
$f(\cdot)$ or $h(\cdot)$	predict function(not necessarily linear)
$\mathcal{F}$ or $\mathcal{H}$	function class
$l(\cdot)$	loss function
Λ	regularization parameter
ϵ	privacy parameter
δ	privacy parameter
x_i and y_i	a data point and its corresponding label
d	data dimension, integer
$\mathcal{X} \times \mathcal{Y}$	data and label class $\subset \mathcal{R}^d \times \{-1, +1\}$, unless otherwise specified.
D or D'	data set $= \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}, i = 1, 2, \cdots, n\}$
M	any matrix
S^n_{++}	symmetric p.d. $n \times n$ matrix
C_i (C or c_i or c)	constant
$I_A(x)$	indicator function
$R(f)$	regularization term
$J_R(f, D) = \frac{1}{m} \sum_{i=1}^m l(f^T x_i, y_i) + \Lambda R(f)$	regularized empirical loss using data site D, which has m data points.
$L(f) = E_{(x,y)\sim\mathcal{P}}[l(f(x), y)]$	expected loss $L(f)$ for a classifier f, given a distribution $\mathcal{P}$.

Chapter 1

Introduction

1.1 Our work and contribution

The contributions of this thesis are mainly in five parts:

Firstly, we introduced several existing differentially private empirical risk minimization algorithms. We implemented them and gave a short analysis on the basic properties of these algorithms.

Secondly, we introduced three models (local model, global model and distributed model) that we use in our experiments. Then we introduced two methods: a traditional aggregation method called average method and a novel method called "feature method" were introduced and we analyzed their privacy guarantee under the sense of differential privacy. We gave a theoretical analysis of the feature method in three cases: all non-private, public-private and fully-private. We showed that how privacy parameter ϵ, number of points at local sites and other parameters affect the system's performance.

Thirdly, we tested some interesting trade-off phenomena that widely exist in the distributed model for differential private empirical risk minimization (ERM) using the average method and the feature method and gave some explanations for them. These trade-offs are important because we can set related parameters according to them and improve system performance. Moreover, these trade-offs gave us an insight into the relations between the parameters and the system. We gave explanations for these results. Then we compared the average method and the feature method in distributed-setting under all non-private, public-private and fully-private conditions using MNIST[1] and

[1] http://yann.lecun.com/exdb/mnist/

Covertype[2] data sets. We reached several conclusions in each case about how the parameters of system (ϵ, number of points at the sites, etc.) and noise affect the final performance of these two methods.

Fourthly, we compared the performance among local model, global model and distributed model using the feature method and the average method. We tried to give intuitions about the benefits of distributed model over other two models by experiments.

Finally we gave final conclusion of our work and future works.

1.2 Application of statistical classification

1.2.1 Statistical classification

Our problem falls into the field of statistical classification. The classification problem in machine learning has been widely studied and applied in research and industry [1–4]. The goal of classification algorithm is to take an input vector $\mathbf{x} \in R^d$ for some integer d and assign it to one of K discrete classes $\mathcal{C}_k$ where $k = 1, \ldots, K$. The classes here are taken to be disjoint, so that each input is assigned to one and only one class. The input space is thereby divided into decision regions whose boundaries are called "decision boundaries" or "decision surfaces" [5]. In our research, we focus on binary classification – a formal model can be found in Shalev-Shwartz and Ben-David's paper [6].

1.2.2 Medical application

Nowadays diagnosis of diseases benefits more and more from data sharing techniques. Moreover, data from neuroimaging and genetics is very helpful in improving scientific reproducibility and in accelerating research progress [7].

But the problem of privacy and risk of re-identification arise when medical information is shared. Also it involves some ethical and legal aspects, in additional to technical or administrative issues. Data may not be shared for the following reasons: it may be easy to re-identify by linking it to public information, it may be some of the particular characteristics of the patient that make the re-identification easier or the data itself is not shareable due to laws or administrative regulations [8].

In order to solve these problems, we can use data use agreements (DUAs). However, it may be easy to handle the agreements but it is difficult and labor costly to fully carry out the agreements [8].

[2] http://archive.ics.uci.edu/ml/datasets/Covertype

Moreover, DUAs are impractical in some respects. For example, it often takes long time to negotiate an agreement. It is likely that the data that the researcher has found is not suitable for their research since the researcher does not know anything about the data before it becomes available. In addition, the disadvantages of DUAs are intensified in the case of the distributed systems that we will introduce. Researchers may be required to handle several DUAs simultaneously, which is however quite painful.

Another way is to share the data derivatives. For example, we can compute a function on the data set and distribute the function output. In the differentially private algorithms that we will introduce in Chapter 3, we use randomized algorithms which introduce noise to protect privacy at the expense of a loss in accuracy. The algorithm outputs the classifier (data derivatives) of the data points in training set.

Chapter 2

Background and privacy model

2.1 ϵ-differential privacy model

In this section the concept of differential privacy is introduced. Differential privacy provides a way to quantify privacy risk (using the difference between two probabilities) so that there is a trade-off between privacy and accuracy, which is different from the definition of privacy in common sense.

From now on we use $\|\cdot\|$ as l_2 norm unless otherwise noted.

2.1.1 Motivation

The work in this thesis is motivated by applications of distributed information processing to medical research. For example, consider the problem of learning from neuroimaging data held at multiple sites. If the input of our system is MRI images, each image can be viewed as a high dimensional vector. Here d is the total number of voxels in one image. In this case, the input database is modeled as a collection of N patients' records $D = (x_1, x_2, \ldots, x_N)$, where x_j is the jth person's image vector. If these images are labeled by expert or other methods, the input also contains corresponding labels $L = (l_1, l_2, \ldots, l_N)$, forming a **training set**. The training set is used to fit a model that can be used to predict a "response value" from the "predictors". The goal of machine learning is to use the training set to build a classifier to predict the label l of a future data point (image) x.

These labeled or unlabeled images are private data associated to patients and therefore are not directly shareable. One way to solve this problem is to use "ad hoc" solutions such as anonymization of patients' information. But this approach was proved to be

inadequate to protect against re-identification due to the presence of public side information; see Ganta et al. [9]. A famous case is the re-identification of certain users from an anonymized dataset published by Netflix, which is a provider of Internet streaming media [10]. The other way to solve this problem is to carry out some operations on the data before sharing them so that anyone who tries to use or share the output will not be able to identify individuals. These operations have the property that preserves the privacy of data, which means only the data derivatives are shared and thus the patients' information will not be disclosed to an adversary. Here we use the concept of differential privacy [11]. Differential privacy basically guarantees that an analyst observing the output does not learn too much about any individual's membership in the database. The analyst therefore makes few inferences about any individual after his seeing the output of algorithm. It is also natural to realize this goal by requiring that the adversary's prior and posterior views about an individual (i.e. before and after having access to the database) shouldn't be "too different" or that access to the database shouldn't change the adversary's views about any individual "too much" [12, 13]. If nothing is learned about an individual, then the individual cannot be harmed by the analysis [14]. Algorithms that guarantee differential privacy are randomized by adding noise before, during or after computing functions of data [14, 15]. An algorithm is differentially private if someone observing the output does not learn too much about any individual's membership in the database. To protect privacy, the true answer is perturbed by the addition of random noise generated according to a carefully chosen distribution, and this response (e.g. the true answer plus noise) are returned to the user.

Differentially private learning has been widely studied, with examples including Chaudhuri et al. [16], Jain and Thakurta [17] for kernel learning, Thakurta and Smith [18] for feature selection, Xiao et al. [19] for wavelet transform, Friedman and Schuster [20] for data mining, Li et al. [21] for optimizing linear counting queries, and boosting methods in Dwork's paper [22]. Also, there are a lot of interesting papers that one can refer to which considering the properties of differential privacy and explain it in various ways [23–33]. A summary of the application of differential privacy in health data is given in Dankar and El Emam's paper [34].

2.1.2 A simple example

Imagine an adversary observes some results from a computation performed on a data set D but does not have access to D itself. The adversary may think two possible cases: either D or $D^{'}$ was used to get this result. Suppose the result happens w.p. of 99.99% if we use D (i.e. $P(result|D) = 99.99\%$) but only 0.01% if we use $D^{'}$ (i.e. $P(result|D^{'}) = 0.01\%$). Then one would be very confident that data set D rather than

$D^{'}$ is used. Thus, if the adversary has complete knowledge of common part of D and $D^{'}$, he is able to infer more information about the remaining parts from the output of the algorithm. Differential privacy seeks to prevent such re-inferences.

2.1.3 Mathematical explanation

We denote an algorithm with privacy property by $A_p(\cdot)$. This algorithm is randomized so that re-identification of the data on the user side is very difficult. From the definition in Dwork's paper [11], algorithm $A_p(\cdot)$ is ϵ-differentially private if for any subset of outputs S:

$$e^{-\epsilon} \cdot P(A_p(D') \in S) \leq P(A_p(D) \in S) \leq e^{\epsilon} \cdot P(A_p(D') \in S)$$

for any databases D and D' differing in a single point. Here the $A_p(D)$ and $A_p(D')$ are the outputs of the algorithm on input database D and D', respectively, and $P(\cdot)$ is the randomness over noise in the algorithm.

Pathak et al. [35] points out that there are two proposed definitions for adjacent data sets with the stronger one based on deletion: D' contains one entry less than D, and with the weaker one based on substitution: one entry of D' differs in value from D. And here we use the weaker one for our research.

As said in Dwork et al. [15], this definition, if written in this way: $|\log(\frac{P(A_p(D)\in S)}{P(A_p(D')\in S}|) \leq \epsilon$, is much more stringent than statistical closeness. One can have a pair of distributions whose statistical difference is arbitrarily small, yet the ratio is infinite (by having a point where one distribution assigns probability zero and the other, non-zero).

This inequality with ϵ parameter shows that when one data point changes, the output will not change too much since it is bounded by the original probability (i.e. the output distribution is closed). More specifically, a user who sees the output and also knows the common part of these two databases is still uncertain about the remaining person's data. Since this holds for any two databases which differ in one data point, each individual in the database is given a guarantee of this protection [8]. The goal of differential privacy is that we can let others learn useful things from a group without learning to much about any individual in the group.

The goal of analyzing statistical databases is to learn something useful. For example, suppose we learn that smoking can cause cancer. So a man who smokes will have higher insurance premiums because he is also a smoker. This is true even if he is not in the database. That is, you can be affected even if you are not in the database. Here our

objective is to learn something from the database. Differential privacy limits the privacy risk by showing that you will be affected by the learning but will not be further affected by joining in the database. So this is why we say the output is essentially "equally likely". Lee and Clifton [36] discuss the choice of ϵ in detail.

Dwork et al. [15] and Wasserman and Zhou [37] give a stronger version of ϵ-differential privacy which indicates an algorithm $A(B)$ taking values in a set τ provides ϵ-differential privacy if $\sup_S \sup_{D,D'} \frac{\mu(S|B=D)}{\mu(S|B=D')} \leq e^{\epsilon_p}$ where the first supremum is over all measurable[1] $S \subseteq \tau$ and the second one is over all data sets D and D' differing in a single entry. Dwork's definition [15] can be derived from this since we let $\mu(S|B = D) = P(A(D) \in S)/P(B = D)$ (similar for D') and substitute $P(A(D) \in S)$ into Dwork's definition so that we can get the right inequality. Then we interchange D and D' so that the left inequality holds, too.

Also in another way, we can write the expression of definition of differential privacy in this form:

$$\sup_S \sup_{D,D'} \frac{\mu(f = f_1|B = D)}{\mu(f = f_2|B = D')} \leq e^{\epsilon_p}. \tag{2.1}$$

2.2 Empirical risk minimization

Returning to our motivation from neuroimaging, one approach is to use machine learning to detect illness. That is, we use previously classified images to build a classifier f that can be seen as a criterion to test further images. Classified images are those that have been diagnosed by an expert. These labeled images are called training set $D = \{(x_i, y_i) \in X \times Y : i = 1, 2, \ldots, n\}$ of n data-label pairs. Here x_i is the vector of image corresponding to ith patient and y_i is label to show whether he is ill ($y_i = 1$) or he is healthy ($y_i = -1$). We use D to find a function $f(x)$, we can use this f to find y_i by computing $f(x_i)$ where $i = n+1, \ldots$. So the goal of our system is: to build an accurate, stable classifier using a differentially private algorithm. Such an algorithm could replace or assist a human expert (e.g. a doctor).

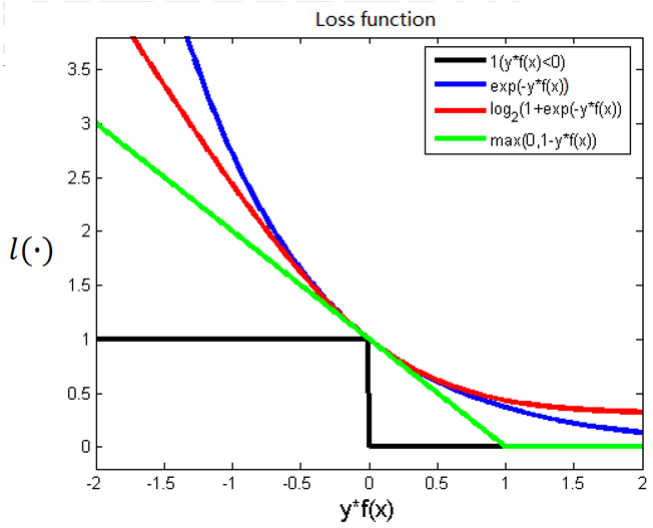
2.2.1 Loss function

To achieve this goal, we need to measure how good a classifier is. We firstly solve the problem of building a non-private classifier. To do this we measure the gap between

[1] If X is a set and F is the sigma-algebra of X. A is a subset of X, then A is called measurable if A is a member of F. We use "measurable" here because this is a technical consideration for most algorithms but more mathematically precise.

$f(x_i)$ and y_i in the training set by using a so called "loss function" which was introduced by Wald et al. [38]. A loss function is large when the difference between the prediction $f(x_i)$ and actual label is large and vice versa. So it can be seen that a loss function acts as penalty. Some typical loss functions are shown in Figure 2.1.

FIGURE 2.1: Some typical loss functions



The simplest loss function is the "0/1" loss function. It equals 1 when $f(x_i) \neq y_i$ and 0 otherwise. The "0/1" loss function is not widely used in practice because it is not convex and differentiable. Convex surrogates of the "0/1" loss function are highly preferred because of the computational and theoretical virtues that convexity brings in. This is of more importance if we consider smooth surrogates as witnessed by the fact that the smoothness is further beneficial both computationally – by attaining an optimal convergence rate for optimization, and in a statistical sense – by providing an improved optimistic rate for generalization bound [39]. Some interesting discussions about loss function can be found in Rosasco et al. [40] and Lin [41].

A popular loss function is the hinge loss $L = (1 - y_i f^T x_i)^+$. As long as $y_i f^T x_i \geq 1$ the point x is correctly classified by the classifier f with large confidence and the hinge loss is equal to 0 (now y_i and $f^T x_i$ have same sign). When $y_i f^T x_i \leq 1$, x is either correctly classified with small confidence $0 \leq y_i f^T x_i \leq 1$, or misclassified $y_i f^T x_i \leq 0$ (now y_i and $f^T x_i$ have different signs). In these cases the hinge loss is positive, and increases as $1 - y_i f^T x_i$.

2.2.2 Definition of ERM

In supervised learning problems, we have a space of objects X and corresponding label space Y. Our goal is to learn a hypothesis function $f : X \to Y$ which outputs an object

$y \in Y$ given $x \in X$. This function can be generalized to future cases $(X, Y) \sim P_{X,Y}$. Consequently, the output of f depends on the training data D_{train} and is a random variable. We assume that there is a joint probability distribution $P(x, y)$ over X and Y and that the training set consists of m instances $(x_1, y_1), \ldots, (x_m, y_m)$ drawn i.i.d. from $P(x, y)$. We constructed this model because it allows us to model uncertainty in predictions. The label y is not a deterministic function of x but a random variable with conditional distribution $P(y|x)$ for a fixed x. Based on this, the total "penalty" we pay for incorrect classification is

$$R(f) = \mathrm{E}[l(f(x), y)] = \int l(f(x), y) dP(X, Y),$$

which is also called **expected loss**. However, we do not know the prior distribution $P(X, Y)$. So we need an approximation to replace $R(f)$. Here we introduce the **empirical risk** $R_{emp} = \frac{1}{n} \sum_{i=1}^{n} l(f(x_i), y_i)$. So the whole problem is to choose a function $f(\cdot)$ that minimizes R_{emp}. This is called empirical risk minimization and was introduced in Vladimir Vapnik 's paper [42] and analyzed in his book [43]. Also, ERM has been widely studied in the literature [44–51].

Also, ERM can be used to compute M-estimators [52], a concept in robust statistics which is obtained as the minima of sums of functions of the data. If we add a regularization term $R(\cdot)$ on R_{emp} to prevent overfitting, we get regularized ERM.

Why does regularization prevent overfitting? Theorem 13.2 to Corollary 13.7 in Shalev-Shawartz's book [6] gives an answer to the convex ERM problem with Tikhonov regularization. It shows that intuitively, the regularization term reflects the complexity of hypothesis. The regularization term is seen as stabilizer of learning algorithm and it explains the phenomenon that changing a data point in the training set does not affect the performance of output classifier too much. This indicates how to control the trade-off between empirical risk and the difference between the true and empirical risk. Lagrange duality indicates that when we want to find linear classifier f that minimizes ERM with bounded norm $\|f\| \leq C$ for some constant C, we can find f by minimizing the regularized ERM for a suitable choice of Lagrange coefficient λ. We can see from the next section that we can control the trade-off between estimation error and approximation error by choosing λ instead of C.

2.2.3 Why does ERM make sense?

We need to evaluate the performance of function $f(\cdot)$ by using non-negative real-valued loss function: $L(\hat{y}, y)$, which measures how different the prediction $\hat{y}$ $(= f^T x$ in linear case) is from the true outcome y.

The **Bayes risk** is the smallest risk achievable by f: $R^* := \inf_{all\ possible\ f} R(f)$.

Since $P_{X,Y}$ is never known to us, we can estimate $R(f)$ by empirical risk. Our goal here is to find f_n^* that minimizes the empirical risk $R_n(f)$ over some class of functions $\mathcal{F}$. Before we use ERM as our methods to build classifier (function), we should know why it is a good approximation to the minimum of expected loss.

Basically, there are three parts that make the minimum of empirical risk deviate from the true minimum of expected loss. The first part is **approximation error**, which is denoted by $\inf_{f\in\mathcal{F}} R(f) - R^*$ (in some materials, $\inf_{f\in\mathcal{F}} R(f)$). Approximation error can be seen as a measurement of the richness of class $\mathcal{F}$ and relies only on the class of functions. The second part is **estimation error**, denoted by $R(f_n) - \inf_{f\in\mathcal{F}} R(f)$. Here f_n is a classifier that minimizes empirical risk. Estimation error depends on data and can be reduced by increasing the training size. The third part is $R_n(f_n) - R(f_n)$, which measures the effect of empirical approximation. It relies on data, too. When the family of functions, optimization accuracy and number of training sample increase, Table 1 in Bousquet and Bottou [53] show how it affects these three errors. This kind of decomposition is introduced in Bousquet and Bottou [53] and analysis in Shalev-Shwartz and Srebro [54]. Also, there is a trade-off between the estimation error and the approximation error by choosing a different class $\mathcal{F}$.

The next step is to bound $P\{\sup_{f\in\mathcal{F}} |R_n(f) - R(f)| \geq \epsilon\}$. The explanation below follows from Yoonkyung Lee's lecture notes [55].

Firstly, we can establish a probabilistic bound of the estimation error of the form: $P(R(f_n^*) \leq \inf_{f\in\mathcal{F}} R(f) + \epsilon) \geq 1 - \delta$. Therefore we can determine how much data we need to guarantee that the risk of the empirically optimal classifier f_n^* is within ϵ-bound of the minimum risk of classifier f in F with confidence $1-\delta$ without any distributional assumption on $(\mathcal{X}, \mathcal{Y})$.

Now we examine the tail probability of $R_n(f) - R(f)$. We want to show that given enough data points $(n \to \infty)$ $P(\sup_{f\in\mathcal{F}} |R_n(f) - R(f)| \geq \epsilon) = 0$. This shows the empirical risk is a good approximation to real risk at certain level.

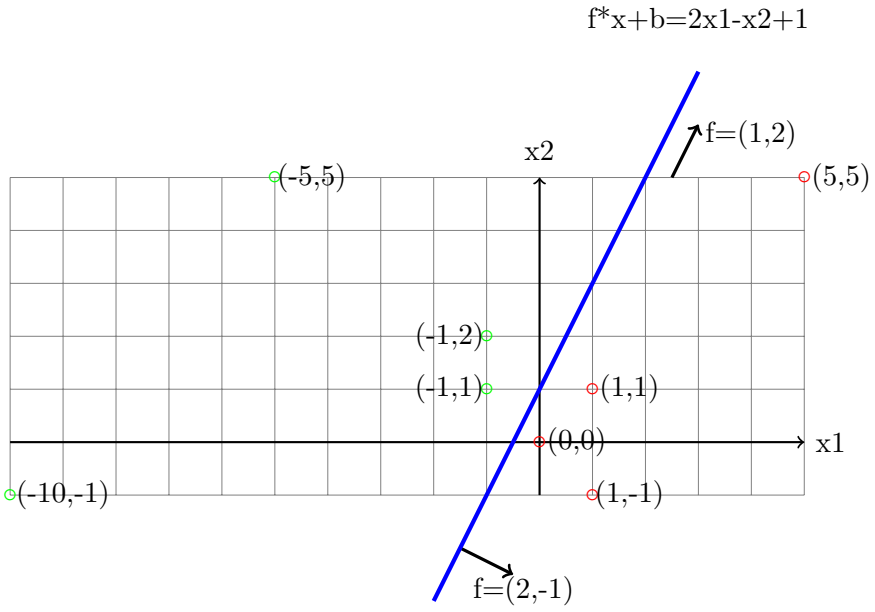
The problem is separated into two cases: the function class $\mathcal{F}$ contains a finite number of elements or an infinite number of elements. The analysis of these two cases are shown in Lee's lecture notes [55]. The idea is to use VC dimension to construct a bound of $P(\sup_{f\in\mathcal{F}} |R_n(f) - R(f)| \geq \epsilon)$.

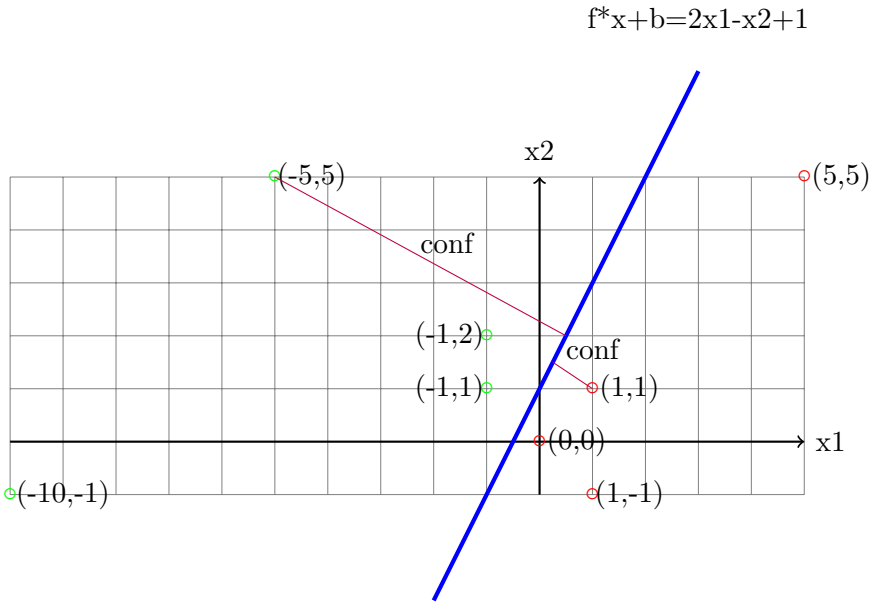
2.3 Geometric explanation of binary classification

2.3.1 A simple example

From now on we focus on the linear classifier ($\in \mathrm{R}^d$ for some integer d). A question arises why the class of linear classifier is appropriate for our classification problem. VC theory indicates that although the class of linear classifier contains an infinite number of elements, it has finite VC dimension, which is $d+1$ for a classifier of dimension d. So according to the fundamental theorem of statistical learning [6] (Theorem 6.7), we know that the class of linear classifiers is PAC learnable and the ERM rule can apply to it.

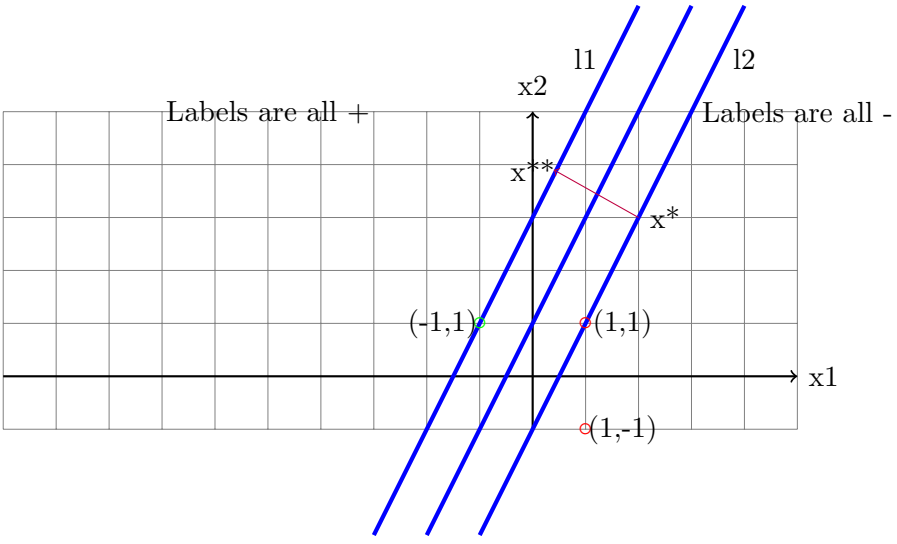
One way to solve the high dimensional binary linear classification problem is to use the support vector machine algorithm. There are many good descriptions of SVM [56–58]. Below we just give a simple example of how to solve binary classification. We consider the $d = 2$ case, and fixed classifier $f = (2, -1)$, and several points $x_i = (x_{i1}, x_{i2}), b = 1$. In Figure 2.3.1 we show these data points and corresponding coordinates. All the points on the right part are labeled $+1$ and -1 on the left part. The first question is: how to measure the extent of separation? That is, we want to define a value to measure our confidence that some points are well classified. Here we use $(f^T x + b)y$ as measurement. We can see in the right part the confidence of point $(0, 0)$, $(1, 1)$, $(1, -1)$, $(5, -5)$ are 1, 2, 4, 16 respectively. The points in the left part $(-1, 1)$, $(-1, 2)$, $(-5, 5)$, $(-10, -1)$ have confidence 2, 3, 14, 20 respectively, which has a positive correlation with the distance between data point and separation line. So this quantity satisfies our requirement.





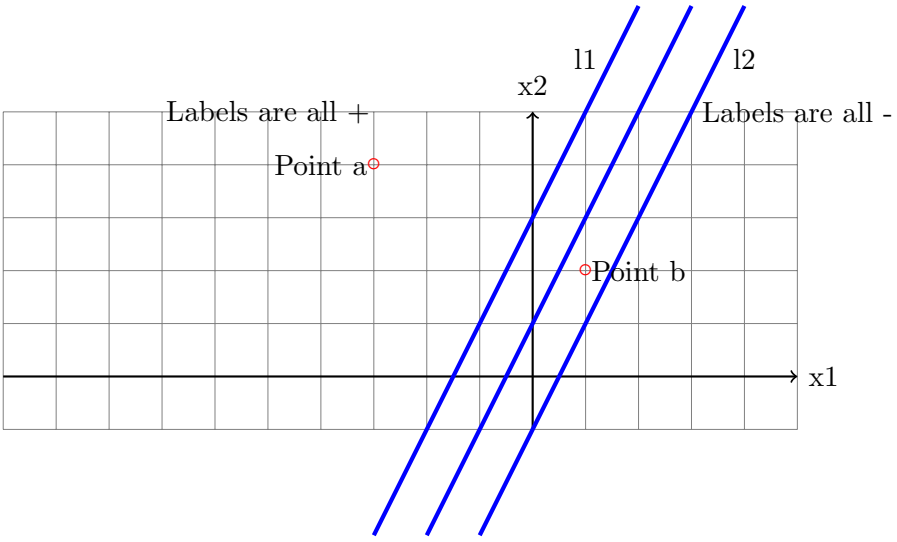
Now we want to find a line (in higher dimension, plane or hyperplane) to separate these points. But there are a lot of lines that separate the data given in Figure 2.3.1. We also need a quantity to measure whether a line is more qualified than other line as the separation line. One such measurement is $\max_{f,b} \min_{i} (f^T x_i + b) y_i$. The value of f and b is uniquely determined by the dataset. Firstly we compute min(conf) among all data points, then we compute max(min(conf)) among all the f and b (See Figure 2.3.1).

More specifically, let a be an arbitrary point and let a' be its orthogonal projection on the separating line in Figure 2.3.1. We can find through analytic geometry that $a = a' + mf$ where m is some real number and f is the normal vector of the separating line (in our case, $m < 0$).



Suppose now we have a data set and the points in the data set are linearly separable. The point on the left part has label $+1$ and the point on the right has label -1. There

are two parallel lines which we call l_1 and l_2 (See Figure 2.3.1). These two lines go through two data points separately (the green one and the red one) and there is no data point between them. It is clear that any line between them can separate these data points (not necessarily parallel w.r.t the two lines). Suppose that without loss of generality, the expression of l_1 is $f^T x + b = +1$ and l_2 is $f^T x + b = -1$ and data point x^* is on l_1 and data point x^{**} is on l_2. The joint line from x^* to x^{**} is orthogonal to l_1 (also l_2). From last paragraph we can see that $x^{**} = x^* + mf$. since x^{**} is on l_1, we have: $f^T x^{**} + b = +1$. Plugging the last equation into it we have $f^T(x^* + mf) + b = +1$. Thus $f^T x^* + b + mf^T f = +1$ and x^* is on l_2. So $m = \frac{2}{f^T f}$. And now we define the distance between l_1 and l_2 the "margin" and let's denote it by r. We have: $r = \|mf\| = \|\frac{2*f}{f^T f}\| = \frac{2}{\sqrt{f^T f}}$. So our purpose is to maximize $\frac{2}{\sqrt{f^T f}}$, which is equivalent to minimizing $f^T f$ under the constraint of $((f^T x + b)y \geq 1$ if $y = +1$ and $(f^T x + b)y \leq -1$ if $y = -1)$. We call the data points on the lines "support vectors" because the lines are determined by these points (See Figure 2.3.1). The method we use here is called "support vector machine". SVM is a supervised learning models with associated learning algorithms that analyze data and recognize patterns. There are a lot of literatures about the application of SVM (and also its variants) for classification problems [59–64] as well as applications in medical research [65–67].



The discussion above is the case when the data points are linearly separable. What about the case of non-linearly separable? Here we introduce "slack" variables ξ_i. This case is called "Soft SVM" [6]. There are two conditions in this case, the first condition is "misclassification". In this case $\xi_i \geq 1$, like point a in Figure 2.3.1. The second case is "margin violation", which means the data point is not misclassified but is too close to the support line (l_2), like point b in Figure 2.3.1. In this case, $0 \leq \xi_i \leq 1$. So the optimization problem changes to minimizing $f^T f + C\sum_i \xi_i$ w.r.t f, b, ξ_i under the constraint $((f^T x_i + b)y_i \geq 1 - \xi_i$ and $\xi_i \geq 0$.

Taking hinge loss function ($l = \max(0, 1 - t * y)^+$) and homogeneous case (b = 0) as an example here. As long as $yf^Tx \geq 1$, the data point x is correctly classified by the function (classifier) f with large confidence and the hinge loss is equal to 0. When $yf^Tx < 1$, x is either correctly classified with small confidence ($0 \leq yf^Tx \leq 1$), or misclassified ($yf^Tx < 0$). In these cases the hinge loss is positive, and increases w.r.t $1 - yf^Tx$. SVM computes a linear classifier with a large margin and small average hinge loss on the training set. If a point (x_i, y_i) is correctly classified by f with large confidence, then $(1 - y_i f^T x_i))^+ = 0$. If this is not the case, then $(1 - y_i f^T x_i))^+ > 0$ increases with the distance from x to the correct half-space of large confidence.

Finally, SVM requires both large margin (i.e. small $\|f\|$) and small misclassification rate on the testing dataset, so we have to solve the minimization problem:

$$\underset{f}{\operatorname{argmin}} \frac{1}{2}\|f\|^2 + C\sum_{i=1}^{n} l(y_i, f^T x_i), \tag{2.2}$$

here C is a parameter to have a trade-off between regularization term and error rate.

This is a convex optimization problem and can be solved by the method of Lagrange multipliers in Chapter 5 in Boyd's book [68].

2.4 Logistic regression

Logistic regression is another prediction method used for in binary classification. The goal of logistic regression is to model the conditional probability $P(Y = 1|X = x)$ as a function of x. Let $p(x) = P(Y = 1|X = x)$. The idea is that we need to design a function g that takes the result of linear classification as domain: $p(x) = g(f^Tx + b) \rightarrow g^{-1}(p(x)) = f^Tx + b$. We can see that the domain of g should be unbounded and the range should belong to (0,1). Instead of choosing g, we try to choose a suitable g^{-1}. The Logit transformation $\log \frac{p}{1-p}$ turns out to be a good choice of g^{-1}. So letting $g^{-1}(p(x)) = \log \frac{p(x)}{1-p(x)}$, we have the following:

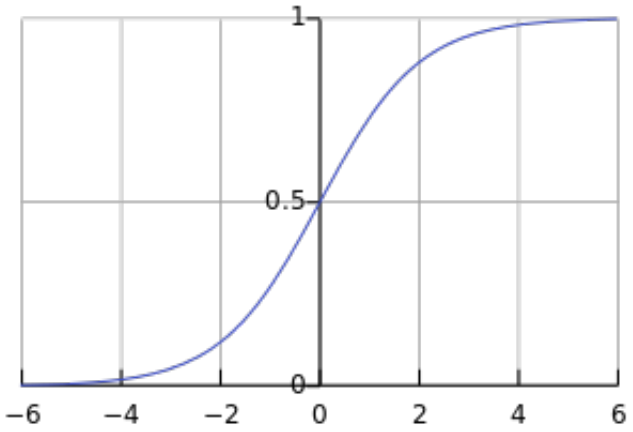
Definition 2.1. *(Logistic regression in binary linear classification)*

$$p(x; f, b) = \frac{1}{1 + e^{-(f^Tx+b)}} \tag{2.3}$$

The logistic function $\frac{1}{1+e^{-t}}$ is shown in Figure 2.2.

It can be seen from Figure 2.2 that to minimize the misclassification rate, we should predict $Y = 1$ when $p \geq 0.5$ and $Y = 0$ when $p < 0.5$. This means guessing 1 whenever

FIGURE 2.2: Logistic function



$f^T x+b$ is non-negative and 0 otherwise. So logistic regression produces a linear classifier. The decision boundary that separates the two predicted classes is the solution of $f^T x + b = 0$.

But another question is: how can we apply the logistic regression to our binary classification problem? Taking the homogeneous case (b=0) as an example, we have

$$\begin{aligned} P(y=1|f,x) &= \frac{e^{(f^T x)}}{1+e^{(f^T x)}}, \\ P(y=0|f,x) &= \frac{1}{1+e^{(f^T x)}}. \end{aligned} \tag{2.4}$$

Rewrite the equation (2.4) in this way:

$$\begin{aligned} P(y=1|f,x) &= \frac{e^{(f^T x)}}{1+e^{(f^T x)}} \\ &= \frac{1}{\frac{1}{e^{(f^T x)}}+1} \\ &= \frac{1}{e^{(-f^T x)}+1} \\ &= \frac{1}{e^{(-yf^T x)}+1}\Big|_{y=1} \end{aligned} \tag{2.5}$$

and similar for $P(y=0|f,x)$. So we unified the two forms to one single equation by integrating the label with the classifier:

$$P(y=\pm 1|f,x) = \frac{1}{1+e^{(-yf^T x)}}. \tag{2.6}$$

and this is the form that we use in the following chapters.

Using the analysis in Section 4.4 in Hastie et al. [69], we want to solve the following optimization problem in order to find the optimal classifier f

$$maximize \quad l(f) = \sum_i \log P(y_i|f, x_i) = -\sum_i \log(1 + e^{(-y_i f^T x_i)}). \tag{2.7}$$

This is an ERM problem with following loss function:

$$P(z) = \log(1 + e^{-z}). \tag{2.8}$$

And we call this function the "logistic loss", which is a convex and differentiable approximation to 0-1 loss.

2.5 Optimization method

Mathematical optimization (mathematical programming) is the procedure that choose the best solution to math problem under certain constrains. The theory of optimization is becoming a more and more important mathematical as well as interdisciplinary area, especially in the interplay between mathematics and many other fields like biology, computer science, engineering and economics [70].

Mathematical programming contains a lot of aspects such as: convex programming, integer programming, nonlinear programming and stochastic programming. Antoniou and Lu's book [71] provides a lot of practical algorithms to solve these problems. The problem we deal with here is called "convex optimization", which is the most important issue in mathematical optimization due to its wide applications. Convex optimization deals with the problem of minimizing real, convex functions over convex set. We will see that the convexity of objective function guarantees that any local minimum must be a global minimum and thus guarantees the unique solution. For a comprehensive introduction to convex optimization, please refer to Boyd and Vandenberghe's book [68].

Usually there is no analytical formula for the solution of convex optimization problems, but there are very effective methods for solving them like interior-point methods. We often use software package to get numerical solution. The optimization solver that we use in Scipy function "minimize" is Nelder–Mead method, which is a nonlinear well-defined numerical method that does not use derivatives. This method was proposed in Nelder and Mead [72] and its converge rate is analyzed in Lagarias et al. [73].

We cannot yet claim that solving general convex optimization problems is a mature technology, like solving least-squares or linear programming problems. Research on general nonlinear convex optimization is still a very active research area. But it is

reasonable to expect that solving general convex optimization problems will become a technology in the near future usable by non-specialists.

Chapter 3

Algorithms

In this chapter we describe three typical differential private ERM algorithms and some comparisons between them to show how differential privacy affects the performance of these algorithms. We describe privacy guarantees (these algorithms indeed protect user's privacy in a differential private way) and utility guarantees (they are non-trivial even adding noise and controllable through parameters). Alvim et al. [74] proposed an information-theoretic framework to reason about both information leakage and utility. They also proved that ϵ-differential privacy implies a tight bound on both the information leakage and utility. Some other ideas of differentially private perturbation methods are mentioned in Sarwate and Chaudhuri [75].

We also implemented algorithms in papers like Jain and Thakurta [17], Kakade et al. [76] . We just give references here rather than introducing them all. All python code can be found in Github link `https://github.com/xieliy`.

3.1 Output perturbation

Output perturbation is the simplest algorithm we used here, which was introduced in Chaudhuri et al. [77] and proof details can be found in Chaudhuri et al. [78]. The algorithm is shown below:

The intuition of Output perturbation 1 is well understood: "private classifier = classifier + noise". An easily understandable example is that imagine you walk on a street and listen to the people around you talking and laughing. If you hear that they speak in English, you are much more confident that you are walking in a street in US rather than in Japan. The reason is that P(people talk in English|in US) = 99.99% and P(people talk in English|in Japan) = 0.01%. So given the fact that what you heard is

Algorithm 1 Output perturbation [78])

Require: $\|x_i\| \leq 1$, $y_i \in \{-1, +1\}$. Data set D is drawn i.i.d according to P
Inputs: D, ϵ_p, Λ.
Steps: 1.Draw a noise vector b w.r.t density $\frac{1}{\alpha}e^{-\beta\|b\|}$ with parameter $\beta = \frac{n\Lambda\epsilon_p}{2}$ (see Section 3.3 for details)
2.Compute $f_{priv} = \text{argmin}_f\, J_R(f, D) + b$
Output: f_{priv}

English (output of algorithm), you can immediately decide the database it correspondent to (country). If we add noise on the words before they come to your ear, that is, "what you heard = talk + noise", you are not that sure which language they use. Now the situation may become: P(people talk in ???|in US) = P(people talk in ???|in Japan) = 50%, which means you can hardly figure out which country you are in and thus hide the information of database. Note that this property are also affect little even some of the people remove from the database (become silent in street).

Theorem 3.1. (Privacy guarantee, Theorem 6 in Chaudhuri et al. [78]) *If $R(\cdot)$ is differentiable, and 1-strongly convex, and $l(z)$ is convex and differentiable, with $|l'(z)| \leq 1$ for all z, then Output perturbation 1 is ϵ_p-differentially private.*

Theorem 3.2. (Utility guarantee, Theorem 15 in Chaudhuri et al. [78]) *Let $R(f) = \frac{1}{2}\|f\|^2$, and let $L(f_0) = \min_f L(f) = L^*$, and let $\delta > 0$. If $l(z)$ is convex and differentiable with $|l'(z)| \leq 1$ and $l'(z)$ is c-Lipschitz. Then there exists a constant C s.t. if the number of training samples satisfies $n > C \max\left(\frac{\|f_0\|^2 \log(\frac{1}{\delta})}{\epsilon_g^2}, \frac{\|f_0\| \log(\frac{d}{\delta}) d}{\epsilon_g \epsilon_p}, \frac{\|f_0\|^2 \log(\frac{d}{\delta}) c^{\frac{1}{2}} d}{\epsilon_g^{\frac{3}{2}} \epsilon_p}\right)$, then the output of Output perturbation 1 satisfies $P(L(f_{priv}) \leq L^* + \epsilon_g) \geq 1 - 2\delta$.*

3.2 Objective perturbation

Objective perturbation is introduced in Chaudhuri et al. [78]. It has a significant improvement in performance compared with Output perturbation 1. The algorithm is shown below:

Theorem 3.3. (Privacy guarantee, Theorem 9 in Chaudhuri et al. [78]) *If $R(f)$ is doubly differentiable and 1-strongly convex. If $l(z)$ is convex and doubly differentiable with $|l'(z)| \leq 1$ and $|l''(z)| \leq c$ for some constant c. Objective perturbation method 2 is ϵ_p-differentially private.*

Theorem 3.4. (Utility guarantee, Theorem 18 in Chaudhuri et al. [78]) *Let $R(f) = \frac{1}{2}\|f\|^2$, and let $L(f_0) = \min_f L(f) = L^*$. If $l(z)$ is convex and doubly differentiable with $|l'(z)| \leq 1$ and $|l''(z)| \leq c$ for some constant c, then there exists a constant C s.t. if the*

Algorithm 2 Objective perturbation [78])

Require: $\|x_i\| \leq 1$, $y_i \in \{-1, +1\}$. Data set D is drawn i.i.d according to $\mathcal{P}$
Inputs: $D, \epsilon_p, \Lambda, c$
Steps: 1.Set parameters: $\epsilon_p' = \epsilon_p - 2\log(1 + \frac{c}{n\Lambda})$
If $\epsilon_p' > 0$, then $\Delta = 0$, else $\Delta = \frac{c}{n(e^{\frac{\epsilon_p}{4}} - 1)} - \Lambda$ and $\epsilon_p' = \epsilon_p/2$.
2.Draw a vector b with density $\nu(b) = \frac{1}{\alpha}e^{-\beta\|b\|}$ with $\beta = \epsilon_p'/2$ (see Section 3.3 for details).
3.Compute $f_{priv} = \text{argmin}_f \left(J_R(f, D) + \frac{1}{n}b^T f + \frac{1}{2}\Delta\|f\|^2\right)$.
Output: f_{priv}

number of training samples satisfies: $n > C \max\left(\frac{\|f_0\|^2 \log(\frac{1}{\delta})}{\epsilon_g^2}, \frac{\|f_0\|^2 c}{\epsilon_g \epsilon_p}, \frac{\|f_0\| \log(\frac{d}{\delta}) d}{\epsilon_g \epsilon_p}\right)$, *then the output of Objective perturbation 2 satisfies* $P(L(f_{priv}) \leq L^* + \epsilon_g) \geq 1 - 2\delta$.

The proof details can be found in the paper.

3.3 Noise generation

One tricky part of implementing of above algorithms (and possibly will be frequently encountered later) is the generation of noise from the distribution

$$\nu(b) = \frac{1}{\alpha}e^{-\beta\|b\|}. \tag{3.1}$$

Let's firstly look at the properties of this density function. We observe that if b, b' are two vectors with $\|b\| = \|b'\|$, then the densities at b and b' are the same. Therefore, we firstly give the direction according to uniform distribution and then sample the norm $\|b\|$. These two procedures are independent with each other.

Firstly we generate a vector U which direction is uniformly distributed, in which $U = \text{Uniform}(x)$. In order to normalize the norm, the result should be like $U = x/\|x\|$. Now the problem is to find the distribution of x. Because we want the direction to be uniformly distributed, the shape of density function should be symmetric when rotating w.r.t to its axis. This reminds us of the multivariate Gaussian distribution. So we generate a vector r i.i.d w.r.t $N(0, 1)$. The joint distribution of r_i (let's denote this by $X = (r_0, \ldots, r_n)$ is a multidimensional normal distribution with $X \sim N(0, I_n)$.The density of X is as follow:

$$f(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{||x||^2}{2}}. \tag{3.2}$$

It can be seen that this distribution function makes sure that the density is uniquely determined by norm and the directions are uniformly distributed. We can normalize the vector to make the norm 1 after generating the data.

Then the second step is to generate the norm. According to Lemma 17 and the discussion at the end of Lemma 16 in Chaudhuri et al. [78], the distribution of $\|b\|$ is a Gamma distribution:

$$\Gamma\left(d, \frac{1}{\beta}\right)(x) = \frac{x^{d-1}e^{-x\beta}}{(1/\beta)^d\Gamma(d)}, \tag{3.3}$$

here d is the shape parameter and $\frac{1}{\beta}$ is the scale parameter.

Practically, we can also draw the norm from Erlang distribution instead of Gamma distribution

$$\text{Erlang}\,(d, \beta)\,(x) = \frac{\beta^d x^{d-1}e^{-\beta x}}{(d-1)!}, \tag{3.4}$$

with parameter d and β. Note that the Erlang distribution is the sum of d exponential distribution random variables.

Let $\|b\| = b$, we have $p(b) = \frac{b^{d-1}e^{-b\beta}}{(1/\beta)^d\Gamma(d)}$. d is the dimension of data. The parameter β in Algorithm 1 and 2 in paper Chaudhuri et al. [78] is a function of other parameters.

In the implementation of algorithm, we generated a vector according to the first step. Then we generated the norm according to second step. We multiply them to get the noise vector.

3.4 Experiments and analysis

Before we run the Algorithm 1 and 2 on data set, we use PCA to reduce the dimension of data and running time of the algorithm while keeping the information of original data.[1] For details of PCA, please refer to Appendix A.

We use the Python programming language to implement these two algorithms (and all the experiments that follows).

Firstly, we compare error rate w.r.t privacy parameter ϵ. The result is shown in Figure 3.1:

Analysis: From Figure 3.1 it can be seen that the performance in non-private case is better than objective perturbation, which is better than output perturbation. The first

[1] Empirically, running time of algorithm is proportional to the dimension and number of data point.

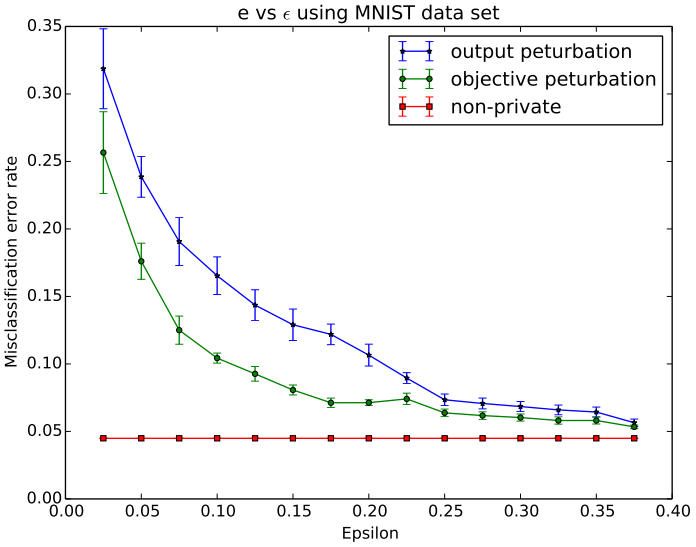


FIGURE 3.1: DP-ERM with logistic loss, e($\epsilon, m = 8678, \Lambda = 0.01$) vs ϵ, ϵ is from 0.025 to 0.375 with step 0.025, run the whole system 20 times for fixed ϵ and plot with error bar.

inequality is easy to see because adding noise makes the output classifier deviate from the optimal classifier computed from non-private case. This optimum is unique for any data set due to the fact that the problem is strongly convex (see Definition B.2). The second inequality comes from the comparison between Theorem 3.2 and Theorem 3.4. Dividing the third term

$$\frac{\|f_0\|^2 \log(\frac{d}{\delta}) c^{\frac{1}{2}} d}{\epsilon_g^{\frac{3}{2}} \epsilon_p}$$

of n in Theorem 3.2 by the second term $(\frac{\|f_0\|^2 c}{\epsilon_g \epsilon_p})$ in Theorem 3.4 we get the result

$$\frac{\log(\frac{d}{\delta}) d}{\epsilon_g^{\frac{1}{2}} c^{\frac{1}{2}}}.$$

Here the parameters $\delta \in (0, 1)$ and $\epsilon_g < 1$ are based on the assumption in Chaudhuri et al. [78]. Parameter $c < 1$ because we have the assumption that $|l'(z)| \leq 1$ (see Definition B.4). So the value of expression $\frac{\log(\frac{d}{\delta}) d}{\epsilon_g^{\frac{1}{2}} c^{\frac{1}{2}}}$ is much greater than 1. On the assumption that dominating term in the sample requirement for objective perturbation has a better dependence on $\|f_0\|$ and $\frac{1}{\epsilon_g}$, we have the conclusion that output perturbation needs more data points to achieve same performance as objective perturbation. Since using manpower to gather and correctly label data points can be very expensive [79], reduction in the size of training set is a great improvement in medical systems and other commercial applications. Intuitively we can say that Algorithm 2 performs better than

Algorithm 1 because Algorithm 1 adds noise directly to the result, whereas Algorithm 2 optimizes after adding noise.

The Figure 3.1 also shows the most significant characteristic of a differentially private system: error rate and noise variance decrease with the increasing of the privacy parameter ϵ. This is clear because we generate the norm of noise vector according to $\Gamma(d, \frac{1}{\beta})$ and β is always proportional to ϵ. So large ϵ means higher probability (the shape of distribution is more concentrated around its mean: 0), the noise is small and thus the classifier is less affected, resulting in lower error rate. From the definition of differential privacy, larger ϵ means the restriction on the database imposed by neighbor database is small. When ϵ is very large, the restriction on the database imposed by neighbor database is out of the range $0 \leq P(A(D)) \leq 1$ which is trivial. Note that as long as there is noise, it will not be possible that the performance of private learning can be the same as the non-private case.

We also implemented the algorithm with the Huber loss function, an approximation to SVM. Huber loss is defined as follows:

$$Huber(z) = \begin{cases} 0 & \text{if} \quad z > 1 + h \\ \frac{1}{4h}(1 + h - z)^2 & \text{if} \quad |1 - z| \leq h \\ 1 - z & \text{if} \quad z < 1 - h. \end{cases} \tag{3.5}$$

Here h is the "Huber constant", which controls the shape of function. Observe that the Huber loss is convex and differentiable, and piecewise doubly-differentiable but not globally doubly differentiable. A class of Huber loss functions with different Huber constant is shown in Figure 3.3. One experience indicates that if other factors remain unchanged, the smoother the loss function is, the lower the error rate will be. This is only an empirical claim, we can see that for two convex functions $f_1(x), f_2(x)$ with β-smooth constant β_1 and β_2, respectively (See Definition B.5). If $\beta_1 > \beta_2$ then with same change in their domain Δx, $\Delta(\nabla f_1(x)) > \Delta(\nabla f_2(x))$, which means if image the shape of $f_1(x), f_2(x)$ as like a bowl, $f_1(x)$ has a sharper change. If $f_1(x), f_2(x)$ represent our two objective functions with $h_1 < h_2$, we can see that with sharper shape (gradient changes more rapidly), the minimization procedure has more difficulty in finding the minimizer with same searching steps (accuracy). So this is the reason why the claim only holds most of the time.

This experiment's parameter settings are the same as those for Figure 3.1. The result is shown in Figure 3.2.

Then we compare error rate w.r.t number of points in training set. The result is shown in Figure 3.4:

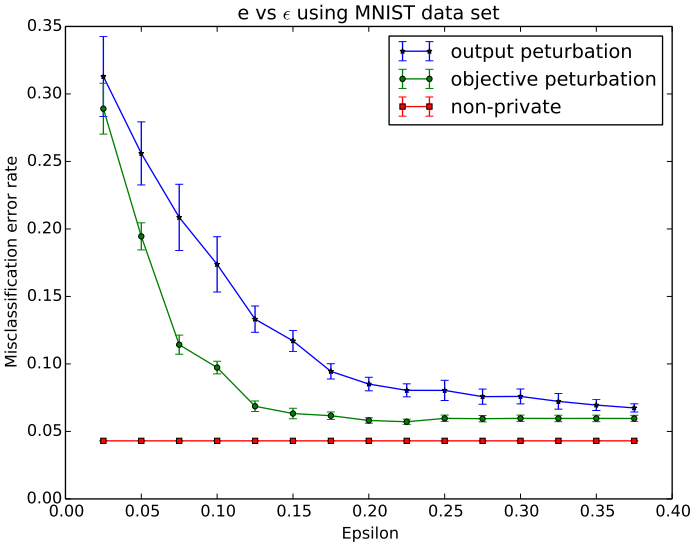


FIGURE 3.2: DP-ERM with Huber loss, $\mathrm{e}(\epsilon, m = 8678, \Lambda = 0.01, h = 0.5)$ vs ϵ, ϵ is from 0.025 to 0.375 with step 0.025, run the whole system 20 times for fixed ϵ and plot with error bar.

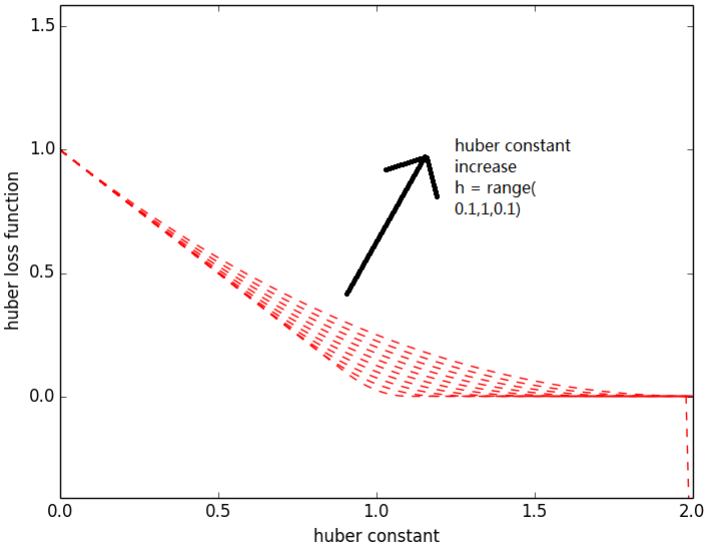


FIGURE 3.3: Huber loss

Analysis: From Figure 3.4 it can be seen that the performance is generally improved when we have more data samples. This is true because we have a minor generalized error $R(f)$ and w.h.p. we should also have a low error rate on testing set.

Finally we conducted an experiment on how regularization term λ affects the performance. In Section 2.2.3 we have argued that there is a trade-off between estimation error and approximation error. If we enrich the function class $\mathcal{F}$ (like increasing the

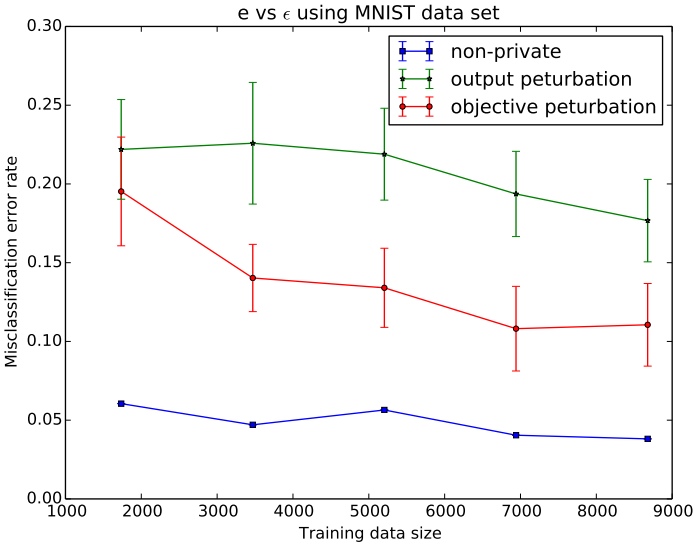


FIGURE 3.4: DP-ERM, logistic loss, e$(m, \epsilon = 0.1, \Lambda = 0.01, h = 0.5)$ vs m, m is from 1735 to 8677 with step 1735, run the whole system 20 times for fixed m and plot with error bar.

radius of Euclidean ball from which our classifier comes), the approximation error decreases (hence estimation error increases) since we have more functions to choose from. Note that in regularized empirical risk minimization, approximation error depends on the regularization term λ since we cannot control the size and shape of constraint directly. This parameter plays a role similar to that of the complexity of the hypothesis class: decreasing λ can reduce the approximation error.

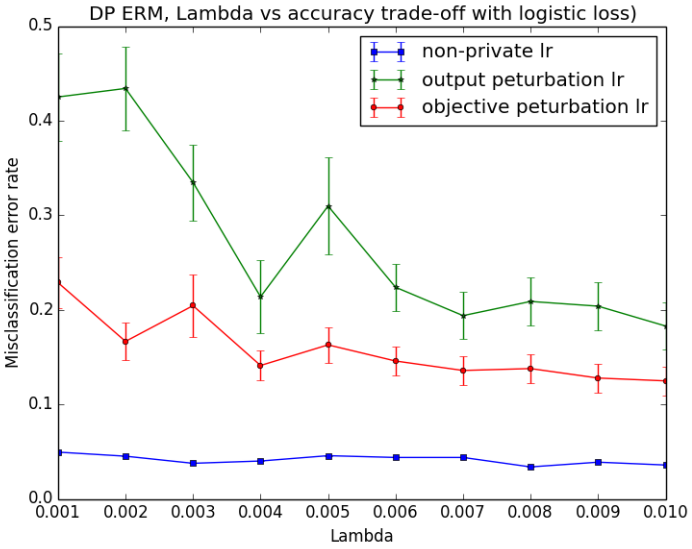


FIGURE 3.5: DP-ERM, logistic regression, e$(\Lambda, m = 8678, \epsilon = 0.1)$ vs Λ using '37' in MNIST dataset, Λ is from 100 to 10^{-13} with ratio 0.1, run the whole system 20 times for fixed Λ and plot with error bar.

Analysis: from Figure 3.5 it can be seen that when the regularization parameter λ increases, the error rate comes down and becomes steady after 0.007. So that is the reason why we make λ be 0.01 (corresponding to the lowest error rate) in experiments using MNIST data set. This gives us the hint that we need to choose a good λ value before we use regularized ERM to conduct any experiment.

3.5 Generalized objective perturbation

Kifer et al. [80] also discussed a privacy preserving objective perturbation. This method guarantees (ϵ, δ)-differential privacy (Gaussian noise) and can be used for function with weaker properties than Algorithm 1 and Algorithm 2.

Algorithm 3 Generalized objective perturbation [80]

Require: Problem domain $F \subseteq R^d$ is closed and convex. r: convex. $l(\cdot,\cdot)$: convex with continuous Hessian matrix whose eigenvalue is less than λ and $\|\nabla l(f,(x,y))\| \leq \zeta$ for all f and (x,y) belong to domain. $\|x_i\| \leq 1$, $y_i \in \{-1,+1\}$, data D are drawn i.i.d according to some unknown probability distribution P.
Inputs: $D, \Delta \geq \frac{2\lambda}{\epsilon}, \epsilon, \delta$.
Steps:
If require ϵ-differential privacy **then:**
draw noise vector $b \in R^d$ w.r.t density $\nu(b;\epsilon,\zeta) = \frac{1}{\alpha} e^{-\epsilon \frac{\|b\|}{2\zeta}}$.
else if require (ϵ,δ)-differential privacy **then:**
draw noise vector $b \in R^d$ w.r.t density $\nu(b;\epsilon,\delta,\zeta) = \mathcal{N}\left(0, \frac{\zeta^2(8log\frac{2}{\delta}+4\epsilon)}{\epsilon^2} I_{d\times d}\right)$.
end if
Output: $f_{priv} = \text{argmin}_f \hat{\mathcal{L}}(f,D) + \frac{1}{n}r + \frac{\Delta}{2n}\|f\|_2^2 + \frac{1}{n}b^T f$ where $\hat{\mathcal{L}}(f,D) = \frac{1}{n}\sum_{i=1}^{n} l(f(x_i),y_i)$.

Theorem 3.5. (Privacy guarantee, Theorem 2 in Kifer et al. [80]) *The first case of Algorithm 3 is $(\epsilon,0)$-differentially private and the second case is (ϵ,δ)-differentially private if they meet the requirements.*

Theorem 3.6. (Utility guarantee, Theorem 4 in Kifer et al. [80]) *Under the requirements of Algorithm 3, we have for the first case of algorithm, setting $\Delta = \Theta\left(\frac{\zeta d \log d}{\epsilon\|\hat{f}\|}\right)$, we have*

$$\mathrm{E}\left[\hat{J}(f_{priv};D) - \hat{J}(\hat{f};D)\right] = \mathcal{O}\left(\frac{\zeta\|\hat{f}\| d\log d}{\epsilon n}\right). \tag{3.6}$$

For the second case, setting $\Delta = \Theta\left(\frac{\sqrt{\zeta^2 d\log(\frac{1}{\delta})}}{\epsilon\|\hat{f}\|}\right)$,

$$\mathrm{E}\left[\hat{J}(f_{priv};D) - \hat{J}(\hat{f};D)\right] = \mathcal{O}\left(\frac{\zeta\|\hat{f}\|\sqrt{\zeta^2 d\log(\frac{1}{\delta})}}{\epsilon n}\right), \tag{3.7}$$

where $\hat{J} = \hat{\mathcal{L}}(f, D) + \frac{1}{n}r$ *and* $\hat{f} = \operatorname{argmin} \hat{J}$.

We compare error rate w.r.t privacy parameter ϵ. The result is shown in Figure 3.6:

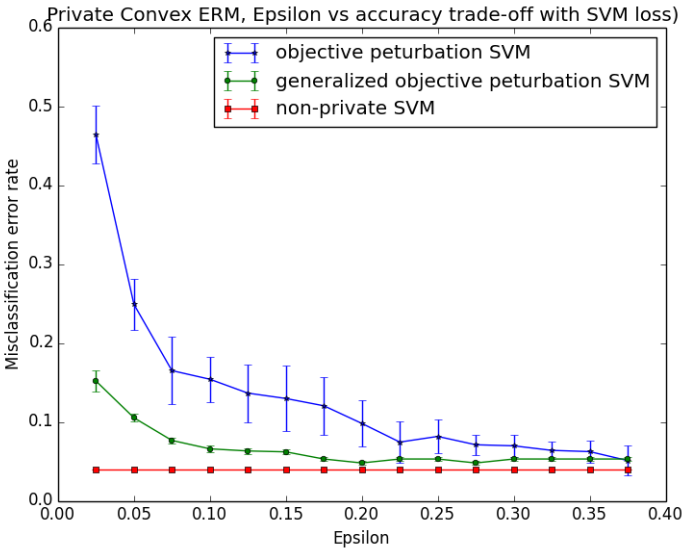


FIGURE 3.6: Private Convex ERM, e($\epsilon, m = 8678, \Lambda = 0.01$) vs ϵ using '37' in MNIST dataset, ϵ is from 0.025 to 0.375 with step 0.025, run the whole system 20 times for fixed ϵ and plot with error bar.

Analysis: In our implementation we need to first compute upper bound of norm of gradient ζ and upper bound on the eigenvalues of Hessian λ. We have

$$\begin{aligned} \|\nabla l(f,(x,y))\| &= |l'(f,(x,y))|\|xy\| \\ &\leq |l'(f,(x,y))| \\ &\leq 1 \end{aligned}$$

since we use Huber loss function. So we set $\zeta = 1$. For Hessian, the (i,j) off-diagonal entry is $l''(f,(x,y))x_i x_j y^2$, whose norm $|l''(f,(x,y))x_i x_j y^2| \leq \frac{1}{2h}$. Diagonal entry is $l''(f,(x,y))x_i^2 y^2 + l'(f,(x,y))y$. Thus is absolute value is bounded by $\frac{1}{2h} + 1$. So using the fact that $\lambda \leq \sqrt{tr(H^T H)} = \sqrt{\sum_{i,j} |H_{i,j}|^2}$, we have

$$\lambda \leq \sqrt{(d^2 - d)(\frac{1}{2h})^2 + d(\frac{1}{2h} + 1)^2} \tag{3.8}$$

and we set $\lambda = \sqrt{\frac{d^2}{4h^2} + d(1 + \frac{1}{h})}$. The result demonstrates Theorem 3.6. Seeing from right part of these two equalities in Theorem 3.6, case 1 is:

$$\frac{\frac{\zeta\|\hat{f}\| d \log d}{\epsilon n}}{\frac{\zeta\|\hat{f}\|\sqrt{\zeta^2 d \log(\frac{1}{\delta})}}{\epsilon n}} = \sqrt{\frac{d}{\log \frac{1}{\delta}}} \log d,$$

which is an amount that is usually greater than 1 considering the practical case. So the performance of generalized case is better than normal case. The same as case 2.

Now we have introduced three typical differentially private ERM algorithms. We can see that all of them are designed to use only one data set, which is a disadvantage when we need more data to get better result. So we start considering how to construct a learning system that can merge the results from multiple data site. We can either combine the data from different sets in some way or combine the data derivatives (for example, classifier). Figure 3.7 shows us a general distributed learning system. A distributed learning system can be seen as a "two level" learning system. The second level of learning try to find the optimal global result by combining all the information from local sites and ancillary information. In Chapter 4 we will see two kinds of combination method that can take advantages of the information from local data sets.

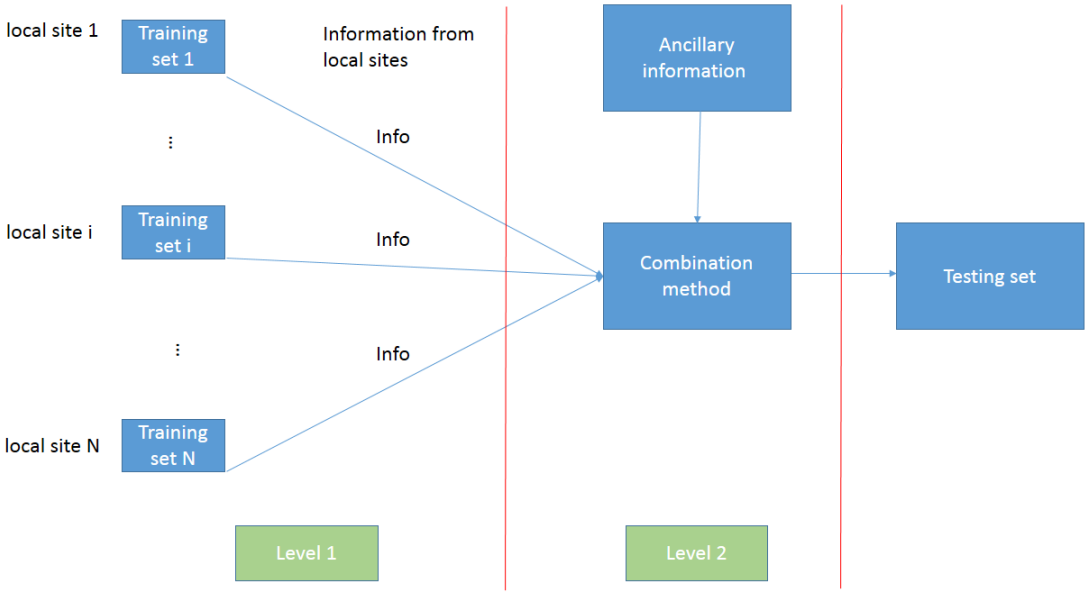


FIGURE 3.7: Two level learning system

Chapter 4

Distributed learning in ERM

4.1 Introduction

In this chapter we turn to the distributed setting, in which our goal is to learn from multiple databases. Our motivating application here is to learn from medical data, which is often distributed among different hospitals and research institutes around the world. Researchers want to take advantage of these data to enable better predictions. For example, they may want to have better performance in classifying healthy and sick patients. Due to privacy concerns and legal restrictions, we are not allowed to store and analyze the data on a single machine. Collaborative algorithms have been developed in order to satisfy this growing appetite for distributed processing, and parallel machine learning has been well studied (see Bekkerman et al. [81]). Several technologies such as Map-Reduce [82], Hadoop [83] and Spark [84] have become popular standards for big data analysis. Here, we show that aggregating private computation across many sites may lead to more accuracy than learning from a single local site under the same conditions (privacy level, regularization term, etc), even though each local computation is made less accurate to protect privacy. Although guaranteeing no privacy at local sites is incorrect in practice, it provides a baseline against which we can measure. It can be seen that the non-private case is an extreme case of private case (as the noise goes to 0). **We refer to the individual medical centers as "local site(s)" and the center attempting to learn or combine the classifiers from local sites as the "aggregation site"**. The distributed setting we use here was also studied in Sarwate et al. [8] and Potluru et al. [85]. Similar distributed optimization models, not incorporating privacy, have been studied by several authors such as Shamir et al. [86], including Zhang and Xiao [87] based on an inexact Newton method, Zhang et al. [88] for parametric smooth convex optimization problems, McDonald et al. [89] based on the conditional

maximum entropy model, Zinkevich et al. [90] and Han et al. [91] for gradient descent, Duchi et al. [92] and Zhang et al. [93] in a information theoretic framework and Huang et al. [94] and Ji et al. [95] incorporated differential privacy concern in distributed logistic regression (see also paper Zhang et al. [96], Zhang et al. [97], Rosenblatt and Nadler [98], Tsianos et al. [99] and Seeger et al. [100]).

In this chapter we firstly introduce three models: local model, global model and distributed model. Then we give an introduction to two aggregation methods: a traditional model named "average method" and a new approach that we called the "feature method", which was described in Sarwate et al. [8]. Next we show the privacy guarantees of these two methods in distributed model. Finally, we conduct mathematical analysis of the feature method.

4.2 Three models

Here we introduce three models: local model, global model and distributed model, which are used in the following sections and chapters and are also widely encountered in distributed learning problems.

The Local Model, shown in Figure 4.1, represents the case that we use the data only in our hand and do not borrow any data or data derivatives from other sites. The advantages are: the local model is economical and there is no need to deal with outside information. But the disadvantage of it is also clear: a good performance can be guaranteed only if the data set is large enough.

The Global Model, shown in Figure 4.2, is the case that there is no privacy issue. Researchers can feel free to merge all the data they can get and treat them as a whole. The global model is an ideal case since algorithms can access to all data directly. This is sometimes called a "pooled analysis." However, the global model is not practical due to privacy considerations.

The Distributed Model, shown in Figure 4.3, represents the most common situation we are confronted with. The aggregation site wants to take the advantages of data from other sites. Each local site computes its own classifier (possibly using different algorithms) and the aggregation site combines them by using some methods. One way for aggregation is to simply average the local classifiers, in this case, we ignore the aggregation site or treat it as a local site. Another way to aggregate classifiers is to use local classifiers to transform the aggregation set into a new set. Then we use this new set to compute a classifier so as to classify the testing set. These two ways are called

FIGURE 4.1: Local model

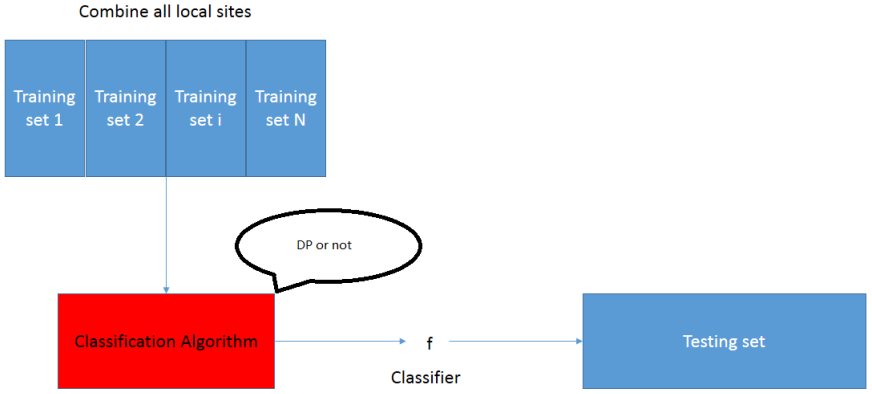


FIGURE 4.2: Global model

the "average method" and "feature method" respectively. We will see how they work in the following sections.

4.3 Average method and feature method

In the last section, we mentioned two aggregation methods that we are going to apply in distributed model. These methods are described in Algorithm 4 and 5. Their concepts are very similar to boosting, which takes the advantages of weaker classifiers [101, 102].

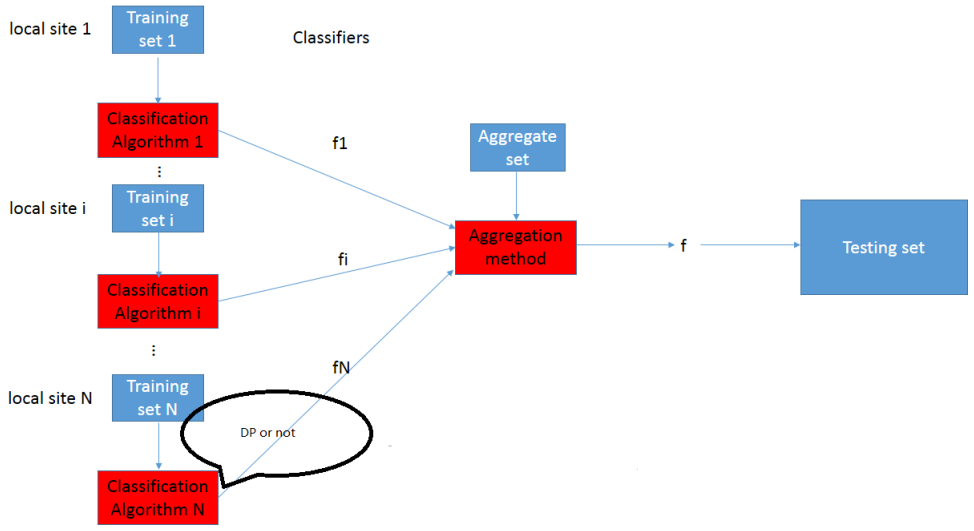


FIGURE 4.3: Distributed model

The average method 4 is commonly used in distributed learning. It can be seen as the best unbiased estimator of f^* since local classifiers $f_1, \cdots, f_N$ are i.i.d if all variables are the same from site to site (since data points are i.i.d) and $\mathrm{E}[\bar{f}] = \frac{1}{N}\sum_{i=1}^{N}\mathrm{E}[f_i] = \mathrm{E}[f_i]$ (a close analogy of this is the scalar random variable). Intuitively, the average method treats the importance of each local site equally.

The feature method 5 describes a new way to deal with distributed learning. It uses a "two-level" structure (local sites-aggregation site) to learn the overall output classifier. Instead of treating the local classifiers equally (average method), the feature method decides the importance of each classifier according to the data points at aggregate site. We will see this mechanism in the following sections and in Chapter 5. It reflects the character of distribution of data points at each local site (they are from the same unknown distribution $\mathcal{P}$, though), so that is the reason why we call this method the "feature method".

Algorithm 4 Average method in distributed model

Inputs: Data sets D_i and corresponding algorithms Alg_i, $i = 1, \cdots, N$.
Steps: 1. Compute classifiers of local sites: $f_i = Alg_i(D_i)$, $i = 1, \cdots, N$
2. Average these classifiers: $\bar{f} = \frac{1}{N}\sum_{i=1}^{N} f_i$
Output: Combined classifier $\bar{f}$.

Note that in Algorithm 4 and Algorithm 5, the algorithms (Alg_i) that used at local sites or aggregation site can be non-private ERM, output perturbation and objective perturbation we mentioned in Chapter 3 or any other classification algorithms, as long as they all compute linear classifiers with same dimension in $\mathcal{R}^d$. Because the aggregation

Algorithm 5 Feature method in distributed model

Inputs: Data sets D_i and corresponding algorithms Alg_i, $i = 1, \cdots, N$, aggregation site D_0 and corresponding ERM objective function $J_R(\omega, \cdot)$.
Steps: 1. Compute classifier of each local site: $f_i = Alg_i(D_i)$
2. Stack these classifiers into a matrix M_f whose ith row is the classifier f_i^T
3. Transform each data point at the aggregation site D_0 to form a new aggregation site $D_0^{'}$, where each point $x_i^{'}$ at $D_0^{'}$ is: $x_i^{'} = M_f x_i$.
4. Compute classifier using the new aggregation site: $\omega_{\sum} = \text{argmin}_{\omega} J_R(\omega, D_0^{'})$
Output: Classifier $f_{\sum} = M_f^T \omega_{\sum}$.

site does not need to access to the original data, the local sites may also use different methods to train their classifiers.

4.4 Privacy guarantees for the two methods

In this section, we discuss privacy guarantees in distributed learning. We focus our attention on different ways of adding noise – the average method and feature method – and how they protects privacy.

4.4.1 Average method

There are two ways of adding noise in the average method. The first one adds noise at local sites only and then perform the averaged classifier on the testing set directly. The other one trains classifiers non-privately at local sites and then adds noise to the output classifier before evaluation on testing set. These systems are shown in Figure 4.4 and Figure 4.5.

Our conclusion here is that these two scenarios preserve differential privacy w.r.t each local site. Nissim et al. [103] mentioned sample and aggregate framework and the average method and the feature method can be seen as the implementation of this framework with aggregation function equal to averaging function and ERM procedure, respectively.

For the first case, we use post-processing property of differential privacy.

Proposition 4.1. (Proposition 2.1 in Dwork and Roth [14]) *Let $\mathcal{M}$: $N^{|X|} \to R$ be a randomized algorithm that guarantees (ϵ, δ)-differential privacy. Let $f : R \to R^{'}$ be an arbitrary randomized mapping. Then $f \circ \mathcal{M} : N^{|X|} \to R^{'}$ is (ϵ, δ)-differentially private.*

Notice that $|X|$ here can be seen as the number of images that possibly appear. Take the MNIST data set as an example, a data point in this data set is represented by a

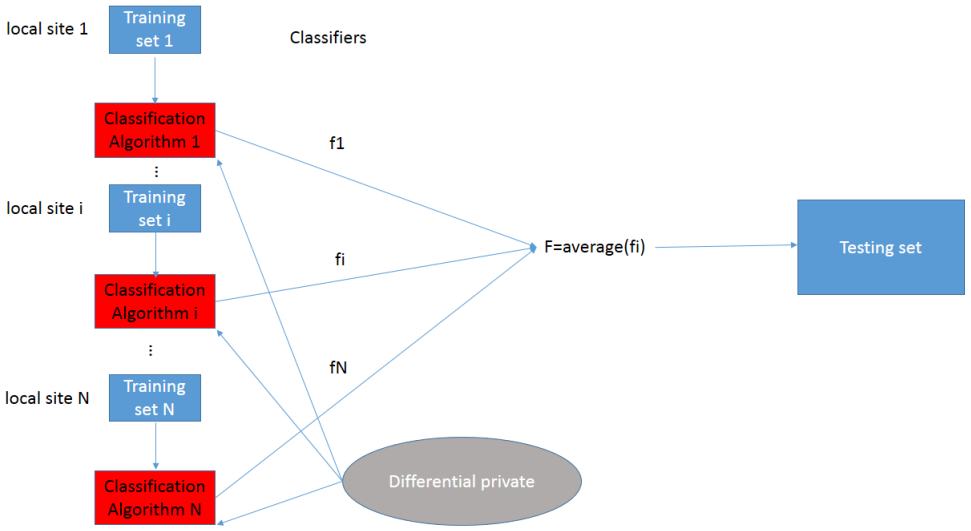


FIGURE 4.4: Average method, scenario 1

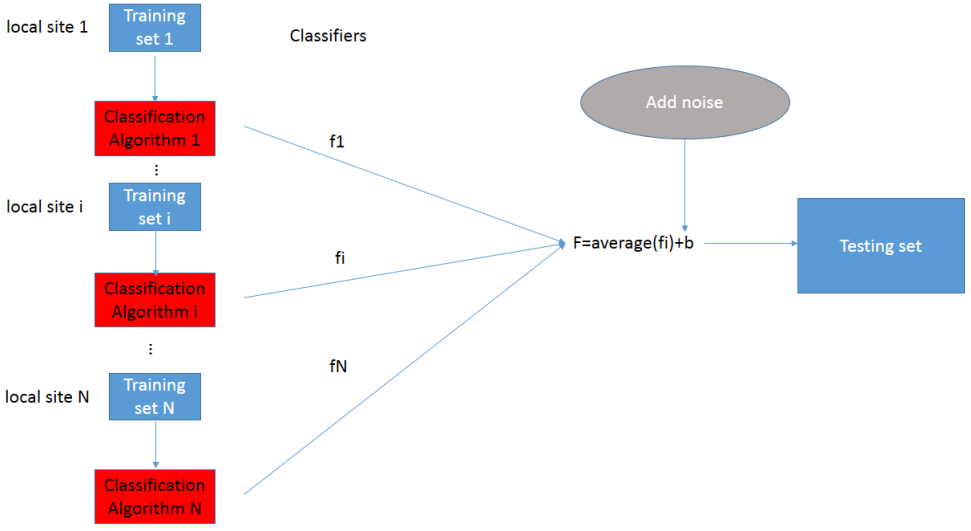


FIGURE 4.5: Average method, scenario 2

$28 \times 28 = 784$ pixels image where the depth of each pixel is 256 (unsigned type). This means $|X| = 256^{784}$. So $N^{|X|}$ represents the data set as a histogram. For any data set $x \in N^{|X|}$, x_i represents the number of points equal to a specific image i, where $i = 1, \cdots, 256^{784}$.

Using the notation in the average method 4, $Alg_i(D_i)$ represents the procedure that applies differentially private algorithm i at local site i ($\mathcal{M}$ in Proposition 4.1), and f is a deterministic mapping $Alg_i(D_i) \to \frac{1}{N}(\sum_{j \neq i} f_j) + Alg_i(D_i))$, using Proposition 4.1 it can be seen that the result preserves differential privacy. Then if we choose a deterministic mapping w.r.t every local site, the result will preserve privacy w.r.t each of them.

For the second case, if we use output perturbation method 1 after averaging the classifiers, then the analysis here is the same as Theorem 6 in Chaudhuri et al. [78]. One thing needing our attention is that our output is $\bar{f} = \frac{1}{N}\sum_{i=1}^{N} f_i + b$, where $f_i = Alg_i(D_i)$, is the non-private classifier i computed from local site i. Using the analysis from Theorem 6 in Chaudhuri et al. [78] we have

$$\|b\| - \|b^{'}\| \leq \|b - b^{'}\| = \frac{1}{N}\|\bar{f}^{'} - \bar{f}\| = \frac{1}{N}\|f_i^{'} - f_i\| \leq \frac{2}{Nn_i\Lambda}. \tag{4.1}$$

Here $f_i^{'}$ is the non-private output of local site i which changes one data point and n_i is the number of data points at local site i. In order to protect this site's privacy we should have

$$\frac{P(f|D_i)}{P(f|D_i^{'})} = \frac{v(b)}{v(b^{'})} = e^{-\frac{Nn_i\Lambda\epsilon_p}{2}(\|b\|-\|b^{'}\|)} \leq e^{\epsilon_p}. \tag{4.2}$$

So we should generate noise with parameter $\beta = \frac{Nn_i\Lambda\epsilon_p}{2}$ instead of $\beta = \frac{n_i\Lambda\epsilon_p}{2}$, which means we need less noise to protect privacy here. Thus in this case, for noise with same magnitude, average method in distributed setting provides more privacy for each local site than the local model.

In a scenario where we average classifiers learned non-privately at each site (the all non-private case), if we change any data point at any site, the output is almost surely different. Therefore the adversary will know which data set we have used based on the results. Using the Definition 2 in Chaudhuri et al. [78] we have $\frac{P(f|D)}{P(f|D^{'})} = \frac{1}{0} = \infty \geq e^{\epsilon_p}$ for any fixed ϵ_p.

4.4.2 Feature method

The feature method is mentioned in Sarwate et al. [8] and compared empirically with the local model both in public-private and fully-private scenarios. The feature method can be seen as a form of ensemble weighting [104] based on treating classifiers learned from local data as new features.

Similarly, there are two ways of preserving privacy on the feature method. The first one is to use objective perturbation 2 at local sites and non-private ERM at the aggregation site. The second one is to train classifiers non-privately at all local sites and using objective perturbation at the aggregation site. The fully-private scenario can be seen as the combination of these two cases. These systems are shown in Figure 4.6 and Figure 4.7.

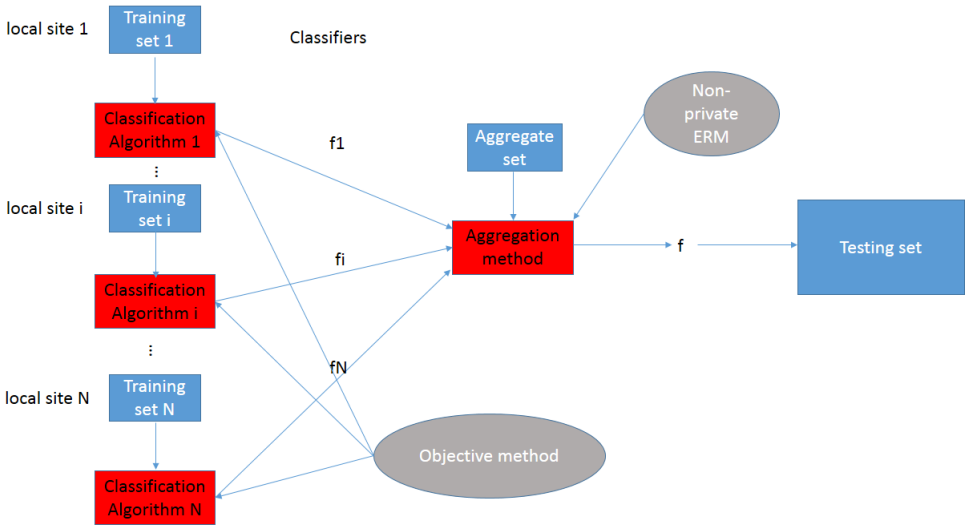


FIGURE 4.6: Feature method, scenario 1

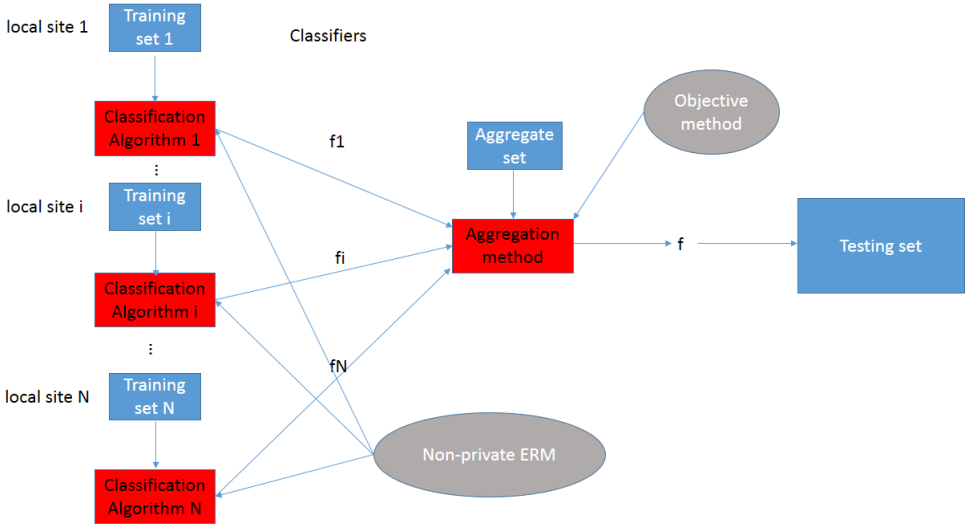


FIGURE 4.7: Feature method, scenario 2

Our conclusions here are that the first scenario protects differential privacy w.r.t each local site only and the second scenario protects differential privacy at the aggregation site and local sites. The first conclusion is easy to see and the analysis of the second conclusion uses the differential privacy of objective perturbation and post-processing property of differential privacy.

The second scenario not only protects the privacy of aggregation site but also protects local sites. To see how it provides privacy for local sites, we need to look at the feature method 5 first. Changing one data point at local site i means changing the ith row of transformation matrix M_f, which is equivalent to changing ith coordinate of every data

point at the aggregation site. This indicates that in order to protect the privacy, a lot of noise is needed.

4.5 Utility analysis for the feature method

The non-private average method has been analyzed in previous work by Shamir et al. [86] and Zhang et al. [88]. They show that under a reasonable set of conditions, the average method (distributed model) performs better than the local model that only learns from one site's data.

While the method works well in practice Sarwate et al. [8], very little is known about its theoretical properties. We would like to better understand the feature method.

We first give an analysis of the mechanism of the feature method without privacy consideration. This **all non-private case** can be seen as a baseline to which we can add the differential privacy property.

First, let us consider the regularized ERM objective function at each local site:

$$J_R(f, D_{local}) = \frac{1}{m}\sum_{i=1}^{m} l(f^T x_i, y_i) + \Lambda R(f), \tag{4.3}$$

and at aggregation site:

$$J_R(\omega, D_0) = \frac{1}{m_0}\sum_{i=1}^{m_0} l(\omega^T(x_{trans,i}), y_i) + \Lambda R(\omega). \tag{4.4}$$

Also, we define $D_{local,i}$ as the ith local site and the unique optimal output classifier of it as:

$$f_i = \underset{f\in\mathcal{R}^d}{\operatorname{argmin}} J_R(f, D_{local,i}), \tag{4.5}$$

and unique optimal output classifier at aggregation site as:

$$\omega_{min} = \underset{\omega\in\mathcal{R}^N}{\operatorname{argmin}} J_R(\omega, D_0), \tag{4.6}$$

From now on we use D_{local} for data set at any local site and D_0 for data set at the aggregation site unless otherwise noted. Note that the parameters and functions in expression (4.3) and (4.4) can be different from site to site. Here we choose ω as variable to represent the "weights" that the aggregation site gives to the different local sites. The parameter m_0 is the number of data points at the aggregation site and ω is the parameter

we want to learn[1]. The transformed data point is $x_{trans,i} = A_{trans}x_i$ where A_{trans} is the stack of classifiers of local sites (the matrix M_f in the feature method 5):

$$A_{trans} = \begin{bmatrix} \cdots & f_1^T & \cdots \\ \cdots & f_2^T & \cdots \\ & \vdots & \\ \cdots & f_N^T & \cdots \end{bmatrix}.$$

Here we impose some assumptions on the parameters and functions. These conditions are standard in classical statistical analysis.

Definition 4.2. Let $span\{f_i\}$ be the subspace that is spanned by the output classifiers of local sites.

Definition 4.2 provides the fundamental setting of our learning problem. It shows that the domain of learning procedure in the aggregation site lies in a subspace[2].

Notice that the data training procedure at the aggregation site can be seen as training a linear combination of classifiers computed from local sites. The output is the combination of this local classifiers. To see why, rewrite the objective function in the aggregation site as

$$J_R(\omega, D_0) = \frac{1}{m_0}\sum_{i=1}^{m_0} l(\underbrace{\omega^T A_{trans}}\, x_i, y_i) + \Lambda R(\omega). \tag{4.7}$$

We can see now that $\omega^T A_{trans}$ is just a linear combination of f_i using ω_i and our output is ω_{min}. Thus to classify a new point $x \in \mathcal{R}^d$, the system predicts the label by using $\hat{y} = sign(\omega_{min}^T A_{trans} x)$, which shows that the overall output classifier is a linear combination of classifiers from local sites. The feature method lets the data determine the importance of each classifier rather than weights them equally as the average method.

Since the output classifier is a linear combination of local classifiers, we should pay attention to the question as to whether the given global optimal classifier f^* belongs to the span of $\{f_i\}$. This question is generally hard to answer since we can assume little about the data distribution $\mathcal{P}$ and f^*. If $f^* \notin span\{f_i\}$, then no matter what we do (tune parameter, add data points or improve the algorithm) at the aggregation site, our result can not reach f^*. This situation is essentially not learnable (see the definition of agnostic learnability in Shalev-Shwartz and Ben-David [6](Definition 3.3)). Regardless of the relationship between N and d, f^* can either belong or not belong to $span\{f_i\}$.

[1] We use $\omega \in \mathcal{R}^N$ to distinguish from $f \in \mathcal{R}^d$.

[2] Subspace of $\mathcal{R}^d$ is always convex.

But there is still an ideal case when we have $N \geq d$ local sites and d of them are linearly independent (thus form a basis of $\mathcal{R}^d$), now $span\{f_i\} = \mathcal{R}^d$ and we guarantee that $f^* \in span\{f_i\}$. But we can only guarantee this if $N \gg d$.

Instead of learning the given global optimal classifier f^*, we switch our goal to find another classifier f^*_{span} that achieves the best generalization error within the subspace that is spanned by the local classifiers $\{f_i\}$. That is, we have the following assumption:

Definition 4.3. *(Best classifier)* Let

$$f^*_{span} = \underset{f \in span\{f_i\}}{\operatorname{argmin}} E_{(x,y)\sim\mathcal{P}}[l(f^T x, y)] \tag{4.8}$$

be the best classifier (fixed but unknown) that we can achieve in $span\{f_i\}$.

Note that when $l(\cdot,\cdot)$ is λ_1 s.c. or strictly convex, function $L(f) = E_{(x,y)\sim\mathcal{P}}[l(f(x), y)]$ is λ_1 s.c. or strictly convex. So finding f^*_{span} is a convex optimization problem and thus guarantee the uniqueness of it. For the case when $l(\cdot,\cdot)$ is only convex, we fix one of its optimal solution among the optimal set as a reference. Like f^*, we assume that f^*_{span} exists is finite. f^*_{span} is fixed if $\{f_i\}$ is fixed, otherwise we treat it as a random vector whose randomness comes from local data. When the error between f^*_{span} and f^* is not reducible because $f^* \notin span\{f_i\}$, it becomes a fourth type of error in addition to the three errors we discussed in Section 2.2.3. From linear algebra we know that there exists either unique or infinite ω^* s.t. $A^T_{trans}\omega^* = f^*_{span}$[3]. In the case of infinity, ω^* forms an affine space in $\mathcal{R}^N$. This affine space is the solution set of $A^T_{trans}\omega^* = \bar{0}$ plus a displace of each point in it by a particular solution of $A^T_{trans}\omega^* = f^*_{span}$. So from now on we use ω^* to denote a reference point in this affine space.

We make Assumption 4.3.1 about the convexity of objective function. Convexity is a common and important condition in machine learning which makes learning problem feasible and bounded.

Assumption 4.3.1. (Convexity) *$l(\cdot,\cdot)$ is a λ_1 s.c. (convex or strictly convex, see Appendix B.1) loss function w.r.t ω, where $\omega \in \mathcal{R}^N$ (or w.r.t f, where $f \in \mathcal{R}^d$) and $R(\cdot)$ is a λ_2 s.c. regularization term. Here λ_1 and λ_2 are the strong convexity parameters (always positive).*

Note that for convex or strictly convex loss function, we simply replace the result in Lemma 4.11 and then set all $\lambda_1 = 0$. We only analyze the s.c. case here.

Assumption 4.3.2. (Bounded gradient of loss function) *There exists a constant C_1 s.t.* $\mathrm{E}\left[\|\nabla l(\cdot,\cdot)\|^2\right] \leq C_1^2$.

[3]These corresponding to the two cases that $\{f_i\}$ is linearly independent or linearly dependent

Assumption 4.3.2 point out a property of loss function: it has bounded gradient. Most of the common loss function we encounter satisfy this property. The reason that we use C_1^2 here is to avoid a square root, which we will see in sequel.

Assumption 4.3.3. *Assume that the regularization term $R(\cdot)$ is μ_1-smooth (see Definition B.5) at all local sites.*

From Assumption 4.3.3 we can bound $\mathrm{E}\left[\|\nabla R(f^*_{span})\|^2\right]$. Using the μ_1-smooth property of $R(\cdot)$ at local sites we have (for s.c. $l(\cdot,\cdot)$):

$$\begin{aligned}
\|\nabla R(f^*_{span}) - \nabla R(f^*)\| &\leq \mu_1 \|f^*_{span} - f^*\| \\
\|\nabla R(f^*_{span})\|^2 &\leq \|\nabla R(f^*)\|^2 + \mu_1^2 \|f^*_{span} - f^*\|^2 + 2\mu_1 \|\nabla R(f^*)\| \|f^*_{span} - f^*\| \\
\mathrm{E}\left[\|\nabla R(f^*_{span})\|^2\right] &\leq \|\nabla R(f^*)\|^2 + \frac{2\mu_1^2}{\lambda_1}(L(\bar{0}) - L(f^*)) \\
&\quad + 2\mu_1 \|\nabla R(f^*)\| \sqrt{\frac{2}{\lambda_1}(L(\bar{0}) - L(f^*))} \\
&= C_2. (\text{take expectation in both sides and using Lemma 4.11})
\end{aligned} \tag{4.9}$$

Assumption 4.3.4. *There exist constant C_3 s.t. $\mathrm{E}\left[\|\nabla R(\omega^*)\|^2\right] \leq C_3$.*

Note that we set $N = 10$ in our experiments, which is much smaller than data dimension[4], so usually local classifiers are linearly independent with each other and this is indeed true in our experiments. So we may assume that the rows of A_{trans} are linearly independent and hence ω^* is unique for fixed f^*_{span}. The intuition of this Assumption is similar to inequality (4.9). When we have finite data and regularization term, if we want to learn a good classifier, that is, $\mathrm{E}\left[\nabla J_R(\omega^*, D_0)\right] \approx \bar{0}$. We need $\frac{1}{m_0}\sum_{i=1}^{m_0} \mathrm{E}[\nabla l(\omega^{*T}(x_{trans,i}), y_i)] + \Lambda \mathrm{E}[\nabla R(\omega^*)] \approx \bar{0}$ according to equation 4.7. So a small $\|\nabla R(\omega^*)\|$ will make things better.

Assumption 4.3.5. *Let A^* be the matrix that each column is the transpose of best classifier f^*_{span},*

$$(A^*)^T = \begin{bmatrix} \cdots & {f^*_{span}}^T & \cdots \\ \cdots & {f^*_{span}}^T & \cdots \\ & \vdots & \\ \cdots & {f^*_{span}}^T & \cdots \end{bmatrix}$$

we can immediately bound $\mathrm{E}\left[|||A^*|||_{op}^2\right]$. Using inequality 4.19 we have: $\mathrm{E}\left[|||A^*|||_{op}^2\right] \leq N\mathrm{E}\left[\|f^*_{span}\|^2\right]$. Then $\|f^*_{span}\|^2 = \|f^*_{span} - f^* + f^*\|^2 \leq (\|f^*_{span} - f^*\| + \|f^*\|)^2$. Now using

[4]Indeed, the number of data sets that we can borrow from others is usually small in real world. So we almost always have $N < d$.

the result in Lemma 4.11 and the fact that $(\|x\| + \|y\|)^2 \leq 2\|x\|^2 + 2\|y\|^2$ *for any vector* x *and* $y \in \mathcal{R}^d$*, we have (for s.c.* $l(\cdot,\cdot)$*):*

$$\mathrm{E}\left[|||A^*|||_{op}^2\right] \leq \frac{4N}{\lambda_1}(L(\bar{0}) - L(f^*)) + 2N\|f^*\|^2 \tag{4.10}$$

The final goal of this chapter is to bound the MSE between the global optimal classifier f^* and $f_{\omega_{min}}$, where $f_{\omega_{min}}$ is the output combined classifier using ω_{min}. We give an outline of our proof and then work through it step by step.

The proof outline:

• Firstly, we use the following decomposition to bound MSE between $f_{\omega_{min}}$ and the best classifier f^*_{span}:

$$\begin{aligned}\mathrm{E}\left[\|f_{\omega_{min}} - f^*_{span}\|^2\right] \leq\ & \mathrm{E}\left[|||A^T_{trans}|||_{op}^2\right] \mathrm{E}\left[\|\omega_{min} - \omega^*\|^2\right] \\ & + \mathrm{Cov}\left[|||A^T_{trans}|||_{op}^2, \|\omega_{min} - \omega^*\|^2\right],\end{aligned}$$

then we deal with $\mathrm{E}\left[|||A^T_{trans}|||_{op}^2\right]$, $\mathrm{E}\left[\|\omega_{min} - \omega^*\|^2\right]$ and the covariance parts separately.

• Secondly, we bound $\mathrm{E}\left[\|\omega_{min} - \omega^*\|^2\right]$. Here we reduce the problem of bounding $\mathrm{E}\left[\|\omega_{min} - \omega^*\|^2\right]$ to the problem of bounding $\mathrm{E}\left[\|\nabla J_R(\omega^*, D_0)\|^2\right]$.

• Thirdly, $\mathrm{E}\left[\|\nabla J_R(\omega^*, D_0)\|^2\right]$ can be bounded by two parts:

$$\mathrm{E}\left[\|\nabla J_R(\omega^*, D_0)\|^2\right] \leq 2\mathrm{E}\left[\|\nabla J(\omega^*, D_0)\|^2\right] + 2\Lambda^2\mathrm{E}\left[\|\nabla R(\omega^*)\|^2\right] \tag{4.11}$$

and focus on bounding $\mathrm{E}\left[\|\nabla J(\omega^*, D_0)\|^2\right]$, which is the un-regularized empirical risk.

• Fourthly, we deal with $\mathrm{E}\left[|||A^T_{trans}|||_{op}^2\right]$ part. Similarly, we have the decomposition:

$$\mathrm{E}\left[|||A^T_{trans}|||_{op}^2\right] \leq 2\mathrm{E}\left[|||A^*|||_{op}^2\right] + 2\mathrm{E}\left[|||A^T_{trans} - A^*|||_{op}^2\right], \tag{4.12}$$

then we focus on bounding $\mathrm{E}\left[|||A^T_{trans} - A^*|||_{op}^2\right]$, which is solved by analyzing $\mathrm{E}[\|f_i - f^*_{span}\|^2]$.

• Fifthly, we show that the covariance part $\mathrm{Cov}(X, Y)$ can be bounded by $\mathrm{E}(X)\mathrm{E}(Y)$ and thus can be solved by using the results in previous steps.

• Finally, we bound $\mathrm{E}\left[\|f^*_{span} - f^*\|^2\right]$ and give our final conclusion about feature method by merging the results in steps 2 to 5 and substitute them into the decomposition in step 1.

Furthermore, we give other two bounds under the conditions of public-private case and fully-private case.

Now, we start working step by step.

In the first step, we bound the MSE $\mathrm{E}\left[\|f_{\omega_{min}} - f^*_{span}\|^2\right]$ by two parts:

$$\begin{aligned}\mathrm{E}\left[\|f_{\omega_{min}} - f^*_{span}\|^2\right] &\leq \mathrm{E}\left[|||A^T_{trans}|||^2_{op}\|\omega_{min} - \omega^*\|^2\right] \\ &= \mathrm{E}\left[|||A^T_{trans}|||^2_{op}\right]\mathrm{E}\left[\|\omega_{min} - \omega^*\|^2\right] \\ &+ \mathrm{Cov}\left[|||A^T_{trans}|||^2_{op}\,, \|\omega_{min} - \omega^*\|^2\right].\end{aligned} \tag{4.13}$$

Here, the first inequality comes from the well known relation $\|Av\| \leq |||A|||_{op}\|v\|$ for matrix A and vector v. The second equality is the application of $\mathrm{E}[X \cdot Y] = \mathrm{E}[X] \cdot \mathrm{E}[Y] + \mathrm{Cov}[X, Y]$ for two possibly dependent random variables X and Y. We decompose in this way because it also reflects the principle behind feature method, where $\mathrm{E}\left[|||A^T_{trans}|||^2_{op}\right]$ is the term related to training at all N local sites and $\mathrm{E}\left[\|\omega_{min} - \omega^*\|^2\right]$ part comes from the training procedure at the aggregation site using the classifiers computed from local sites.

The next step is to bound $\mathrm{E}\left[\|\omega_{min} - \omega^*\|^2\right]$ term.

Lemma 4.4. (Bound on $\mathrm{E}\left[\|\omega_{min} - \omega^*\|^2\right]$) *Under the Assumption 4.3.1 (s.c. case), we have:*

$$\mathrm{E}\left[\|\omega_{min} - \omega^*\|^2\right] \leq \frac{4}{(\lambda_1 + \Lambda\lambda_2)^2}\mathrm{E}\left[\|\nabla J_R(\omega^*, D_0)\|^2\right].$$

Proof. It can be easily seen that $J_R(\omega, D_0)$ is $(\lambda_1 + \Lambda\lambda_2)$ s.c. on $\mathcal{R}^N$. Using Definition B.3, for any $\omega \in \mathcal{R}^N$, we have

$$J_R(\omega, D_0) \geq J_R(\omega^*, D_0) + \langle \nabla J_R(\omega^*, D_0), \omega - \omega^* \rangle + \frac{\lambda_1 + \Lambda\lambda_2}{2}\|\omega - \omega^*\|^2.$$

Rewrite it:

$$\begin{aligned}\|\omega - \omega^*\|^2 &\leq \frac{2}{\lambda_1 + \Lambda\lambda_2}\left[J_R(\omega, D_0) - J_R(\omega^*, D_0) + \langle \nabla J_R(\omega^*, D_0), \omega^* - \omega \rangle\right] \\ &\leq \frac{2}{\lambda_1 + \Lambda\lambda_2}\left[J_R(\omega, D_0) - J_R(\omega^*, D_0) + \|\nabla J_R(\omega^*, D_0)\|\|\omega^* - \omega\|\right].\end{aligned}$$

Now divide both sides by $\|\omega - \omega^*\|$:

$$\|\omega - \omega^*\| \leq \frac{2(J_R(\omega, D_0) - J_R(\omega^*, D_0))}{(\lambda_1 + \Lambda\lambda_2)\|\omega - \omega^*\|} + \frac{2\|\nabla J_R(\omega^*, D_0)\|}{\lambda_1 + \Lambda\lambda_2}.$$

Since the strong convexity holds globally, we can set $\omega = \omega_{min}$. Therefore,

$$\|\omega_{min} - \omega^*\| \leq \frac{2(J_R(\omega_{min}, D_0) - J_R(\omega^*, D_0))}{(\lambda_1 + \Lambda\lambda_2)\|\omega_{min} - \omega^*\|} + \frac{2\|\nabla J_R(\omega^*, D_0)\|}{\lambda_1 + \Lambda\lambda_2}.$$

Using the definition of ω_{min} 4.6 we have $J_R(\omega_{min}, D_0) < J_R(\omega^*, D_0)$. So ignoring the first part of last inequality, squaring both side and take expectation w.r.t data distribution, we get

$$\mathrm{E}\left[\|\omega_{min} - \omega^*\|^2\right] \leq \frac{4}{(\lambda_1 + \Lambda\lambda_2)^2}\mathrm{E}\left[\|\nabla J_R(\omega^*, D_0)\|^2\right]. \tag{4.14}$$

□

The intuition of the result is clear: when the minimum of $J_R(\omega, D_0)$ (i.e., ω_{min}) is quite near to ω^*, that is, the left part of last inequality is small, the derivative of $J_R(\omega, D_0)$ at ω^* should be a small quantity, too.

Now our next step is to bound $\mathrm{E}\left[\|\nabla J_R(\omega^*, D_0)\|^2\right]$ where the expectation is w.r.t randomness of data at local sites and the aggregation site.

Using the inequality (4.11), we need to bound $\mathrm{E}\left[\|\nabla J(\omega^*, D_0)\|^2\right]$.

We need a technical lemma:

Lemma 4.5. *For $x, y \geq 0, n \geq 1, (x+y)^n \leq 2^{n-1}(x^n + y^n)$.*

Proof. Applying Jensen's inequality to the convex function x^n for $n > 1$ and $x \geq 0$, we have

$$\left(\frac{x+y}{2}\right)^n \leq \frac{x^n + y^n}{2}.$$

□

Now let $X = \|\nabla J(\omega^*, D_0)\| \geq 0$ and $k > 1$ be some integer, then

$$\begin{aligned}
\mathrm{E}\left[X^k\right] &= \mathrm{E}\left[|X|^k\right] \\
&= \mathrm{E}\left[|X - \mathrm{E}[X] + \mathrm{E}[X]|^k\right] \\
&\leq \mathrm{E}\left[2^{k-1}\left(|X - \mathrm{E}[X]|^k + |\mathrm{E}[X]|^k\right)\right] \qquad \text{(by Lemma 4.5)} \\
&= 2^{k-1}\mathrm{E}\left[|X - \mathrm{E}[X]|^k\right] + 2^{k-1}\mathrm{E}[X]^k.
\end{aligned} \tag{4.15}$$

Lemma 4.6. (Bound on $\mathrm{E}\left[\|\nabla J(\omega^*, D_0)\|\right]$) *Under the Assumption 4.3.2, we have*

$$\mathrm{E}\left[\|\nabla J(\omega^*, D_0)\|\right] \leq C_1. \tag{4.16}$$

Proof. We know that $\mathrm{Var}[X] = \mathrm{E}[X^2] - \mathrm{E}[X]^2 \geq 0$, then $\mathrm{E}[X] \leq \mathrm{E}[X^2]^{\frac{1}{2}}$, let $Z_i = \nabla l(\omega^{*T} x_{trans,i}, y_i)$, we have:

$$\begin{aligned}
\bar{Z} &= \frac{1}{m_0}\sum_{i=1}^{m_0} Z_i = \nabla J(\omega^*, D_0) \\
\|\bar{Z}\| &\leq \frac{1}{m_0}\sum_{i=1}^{m_0} \|Z_i\| && \text{(triangle inequality)} \\
\mathrm{E}\left[\|\bar{Z}\|\right] &\leq \frac{1}{m_0}\sum_{i=1}^{m_0} \mathrm{E}\left[\|Z_i\|^2\right]^{\frac{1}{2}} \\
&\leq C_1.
\end{aligned}$$

The third inequality is based on the Assumption 4.3.2. So we have $\mathrm{E}\left[\|\nabla J(\omega^*, D_0)\|\right] \leq C_1$. □

Now we need to do some further analysis of $\mathrm{E}\left[\|\nabla J(\omega^*, D_0)\|^2\right]$. The following Lemma 4.7 is used to bound $\mathrm{E}\left[\|\nabla J(\omega^*, D_0)\|^2\right]$ in other way:

Lemma 4.7. (Theorem 2.1 in De Acosta [105]) *Let $k \geq 2$ and X_i be a sequence of independent random vectors in a separable Banach space with norm $\|\cdot\|$ and $\mathrm{E}[\|X_i\|^k] < \infty$. There exists a finite constant C_4 s.t.*

$$\mathrm{E}\left[\left|\|\sum_{i=1}^{n} X_i\| - \mathrm{E}[\|\sum_{i=1}^{n} X_i\|]\right|^k\right] \leq C_4\left[(\sum_{i=1}^{n}\mathrm{E}[\|X_i\|^2])^{\frac{k}{2}} + \sum_{i=1}^{n}\mathrm{E}[\|X_i\|^k]\right].$$

Note that in previous result (4.15): $\mathrm{E}[X^k] \leq 2^{k-1}\mathrm{E}\left[|X - \mathrm{E}[X]|^k\right] + 2^{k-1}\mathrm{E}[X]^k$. Let $k = 2$ and substitute $X = \|\nabla J(\omega^*, D_0)\|$:

$$
\begin{aligned}
\mathrm{E}\left[\|\nabla J(\omega^*, D_0)\|^2\right] &\leq 2\mathrm{E}\left[|\|\nabla J(\omega^*, D_0)\| - \mathrm{E}[\|\nabla J(\omega^*, D_0)\|]|^2\right] \\
&\quad + 2\mathrm{E}\left[\|\nabla J(\omega^*, D_0)\|\right]^2 \\
&= 2\mathrm{E}\left[\left|\left\|\sum_{i=1}^{m_0}\frac{1}{m_0}\nabla l(\omega^{*T}x_{trans,i}, y_i)\right\| - \mathrm{E}[\left\|\sum_{i=1}^{m_0}\frac{1}{m_0}\nabla l(\omega^{*T}x_{trans,i}, y_i)\right\|]\right|^2\right] + 2C_1^2 \\
&\leq 2C_4\left[(\sum_{i=1}^{m_0}\mathrm{E}[\|\frac{1}{m_0}\nabla l(\omega^{*T}x_{trans,i}, y_i)\|^2])^{\frac{2}{2}} + \sum_{i=1}^{m_0}\mathrm{E}[\|\frac{1}{m_0}\nabla l(\omega^{*T}x_{trans,i}, y_i)\|^2]\right] \\
&\quad + 2C_1^2 \\
&\leq 4C_4\sum_{i=1}^{m_0}\mathrm{E}[\|\frac{1}{m_0}\nabla l(\omega^{*T}x_{trans,i}, y_i)\|^2] + 2C_1^2 \\
&\leq \frac{4C_4C_1^2m_0}{m_0^2} + 2C_1^2,
\end{aligned}
$$

where in the last inequality, we use the Assumption 4.3.2 and Lemma 4.6, so we have

$$
\mathrm{E}\left[\|\nabla J(\omega^*, D_0)\|^2\right] \leq \frac{4C_4C_1^2}{m_0} + 2C_1^2. \tag{4.17}
$$

The intuition behind this result is that when number of data points at the aggregation site becomes large ($m_0 \to \infty$), we can guarantee that on average, the global minimizer becomes better at the aggregation site. This means we indeed can learn a good classifier by adding more data points.

Now go back to the decomposition 4.11, we are ready for the following lemma:

Lemma 4.8. *Under the Lemma 4.4, conclusion (4.17) and Assumption 4.3.4, we have*

$$
\mathrm{E}\left[\|\omega_{min} - \omega^*\|^2\right] \leq \frac{4}{(\lambda_1 + \Lambda\lambda_2)^2}\left(\frac{8C_4C_1^2}{m_0} + 4C_1^2 + 2\Lambda^2C_3\right),
$$

Proof. Plugging conclusion (4.17) and Assumption 4.3.4 into inequality 4.11, then using Lemma 4.4 we will get the result. □

It is quite interesting to notice that in the above result, even if the number of data points m_0 at the aggregation site tends to infinity, we cannot guarantee that we can learn an optimal weighted parameter ω^* due to the term $2\Lambda^2C_3 + 4C_1^2$. This is due to three reasons: the first reason is that we need an non-zero value of Λ^2 to prevent over-fitting at the cost of accuracy. The second reason is the constant $C_3 = \mathrm{E}\left[\|\nabla R(\omega^*)\|^2\right]$. This value is determined by $\nabla R(\cdot)$, which is the restriction in the learning procedure at the aggregation site, and ω^*, which is determined by the classifiers from local sites. So this

term can be seen as a connection between two level of learning procedures. Since there is no information communication between this two level, we can not reduce C_3. The third one is C_1, which can be significantly reduced with the help of large m_0. So there are three ways we can do to get a better result: the first is to enrich the parameter family by relaxing the regularization ($\Lambda \to 0$)[5], the second one to enlarge data set ($m_0 \to \infty$), the third way is to exchange information between two learning level. With these three tricks we hope that the algorithm can output something close to the optimal weighted parameter ω^*.

Now the fourth step is to bound $\mathrm{E}\left[|||A^T_{trans}|||^2_{op}\right]$ term.

Using the decomposition $A^T_{trans} = A^* + A^T_{trans} - A^*$, the triangle inequality of operator norm, we have

$$\mathrm{E}\left[|||A^T_{trans}|||^2_{op}\right] \leq 2\mathrm{E}\left[|||A^*|||^2_{op}\right] + 2\mathrm{E}\left[|||A^T_{trans} - A^*|||^2_{op}\right]. \tag{4.18}$$

Clearly our randomness here only comes from the data point at the local sites. So our work now consists at bounding $\mathrm{E}[|||A^T_{trans} - A^*|||^2_{op}]$.

Note that from the definition of operator norm on matrix, $|||M|||_{op} = \sup_v\{\|Mv\| : v \in V$ with $\|v\| \leq 1\}$, we have

$$\begin{aligned}
\mathrm{E}\left[|||A^T_{trans} - A^*|||^2_{op}\right] &= \mathrm{E}\left[|||(A^T_{trans} - A^*)^T|||^2_{op}\right] \\
&= \mathrm{E}\left[(\sup_v\{\|(A^T_{trans} - A^*)^T v\| : \|v\| \leq 1\})^2\right] \\
&= \mathrm{E}\left[(\sup_v\{\sqrt{\sum_{i=1}^N ((A^T_{trans} - A^*)^T_{row,i} v)^2} : \|v\| \leq 1\})^2\right] \\
&\leq \mathrm{E}\left[(\sup_v\{\sqrt{\sum_{i=1}^N (\|(A^T_{trans} - A^*)^T_{row,i}\|\|v\|)^2} : \|v\| \leq 1\})^2\right] \\
&\leq \mathrm{E}\left[(\sup_v\{\sqrt{\sum_{i=1}^N (\|(A^T_{trans} - A^*)^T_{row,i}\|)^2}\})^2\right] \\
&= \mathrm{E}\left[\sum_{i=1}^N (\|A^T_{trans} - A^*)^T_{row,i}\|)^2\right] \\
&= N\mathrm{E}\left[\|(A^T_{trans} - A^*)^T_{row,i}\|^2\right] \\
&= N\mathrm{E}\left[\|f_i - f^*_{span}\|^2\right],
\end{aligned}$$

[5] Usually we have $\Lambda = \mathcal{O}(\frac{1}{n})$, where n is the number of data points in training set.

where the first inequality we use the Cauchy–Schwarz inequality, the second inequality we use the fact that $\|v\| \leq 1$. W.l.o.g, we assume f_i is i.i.d and hence the fourth equality is hold.

So our problem now is to bound $\mathrm{E}\left[\|f_i - f^*_{span}\|^2\right]$. Notice that this expression is similar to $\mathrm{E}[\|\omega_{min} - \omega^*\|^2]$ with only difference being that we need to consider everything in $\mathcal{R}^d$ rather than in $\mathcal{R}^N$. We have the following lemma:

Lemma 4.9. (Bound on $\mathrm{E}\left[\|f_i - f^*_{span}\|^2\right]$) *Under Assumptions 4.3.2, Assumptions 4.3.3 and Lemma 4.7, we have*

$$\mathrm{E}\left[\|f_i - f^*_{span}\|^2\right] \leq \frac{4}{(\lambda_1 + \Lambda\lambda_2)^2}\left(\frac{8C_4C_1^2}{m_i} + 2\Lambda^2 C_2 + 4C_1^2\right), \tag{4.19}$$

where m_i is the number of data point at local site i.

Now go back to the decomposition (4.12), we have:

Lemma 4.10. (Bound on $\mathrm{E}\left[|||A^T_{trans}|||^2_{op}\right]$) *Using decomposition (4.12), inequality (4.10), inequality 4.19 and Lemma 4.9 with the same notations in Lemma 4.9, we have*

$$\begin{aligned}\mathrm{E}\left[|||A^T_{trans}|||^2_{op}\right] \leq{}& \frac{8N}{(\lambda_1 + \Lambda\lambda_2)^2}\left(\frac{8C_4C_1^2}{m} + 2\Lambda^2 C_2 + 4C_1^2\right)\\ &+ \frac{8N}{\lambda_1}(L(\bar{0}) - L(f^*)) + 4N\|f^*\|^2,\end{aligned}$$

Proof. Plugging Lemma 4.9 into inequality 4.19, substitute the result and inequality (4.10) into decomposition (4.12). □

If the number of data point at each local site are not same, we need to sum up over all the m_i instead of using $N \times m$. W.l.o.g, we assume that all local sites have the same number of data point ms.

The intuition of Lemma 4.9 is clear, too. If data sets at all local sites are large enough ($m \to \infty$) while keeping a large domain ($\Lambda \to 0$), we are sure that our classifiers from local sites will be close to f^*_{span}. From Lemma 4.10, this means $\mathrm{E}\left[|||A^T_{trans}|||^2_{op}\right] \to |||A^*|||^2_{op}$. Also, a stronger version of this is $f_i \to f^*_{span}$ for $i = 1, \cdots, N$ which is illustrated in Lemma 4.9.

The second-last step is to deal with the covariance term: $\mathrm{Cov}\left[|||A_{trans}^T|||_{op}^2, \|\omega_{min} - \omega^*\|^2\right]$. We use the relation between two random variables X and Y

$$\begin{aligned}\mathrm{Cov}(X,Y) &\leq \sqrt{\mathrm{Var}(X)\mathrm{Var}(Y)} \\ &= \sqrt{(\mathrm{E}(X^2) - \mathrm{E}(X)^2)(\mathrm{E}(Y^2) - \mathrm{E}(Y)^2)} \\ &\leq \sqrt{\mathrm{E}(X)^2\mathrm{E}(Y)^2} \\ &= \mathrm{E}(X)\mathrm{E}(Y). \end{aligned} \tag{4.20}$$

So we have

$$\mathrm{Cov}(|||A_{trans}^T|||_{op}^2, \|\omega_{min} - \omega^*\|^2) \leq \mathrm{E}(|||A_{trans}^T|||_{op}^2)\mathrm{E}(\|\omega_{min} - \omega^*\|^2) \tag{4.21}$$

In our final step, we bound the MSE between f_{span}^* and f^* and give the main theorem for the feature method. We have:

Lemma 4.11. (Bound on $\mathrm{E}\left[\|f_{span}^* - f^*\|^2\right]$)

$$\mathrm{E}\left[\|f_{span}^* - f^*\|^2\right] \leq \begin{cases} \frac{2}{\lambda_1}(L(\bar{0}) - L(f^*)) & \textit{if} \quad l(\cdot,\cdot) \textit{ is strongly convex} \\ \frac{1}{\sigma}(L(\bar{0}) - L(f^*)) & \textit{if} \quad l(\cdot,\cdot) \textit{ is strictly convex} \\ \frac{1}{\sigma_{D_1}}(L(\bar{0}) - L(f^*)) & \textit{if} \quad l(\cdot,\cdot) \textit{ is convex.} \end{cases} \tag{4.22}$$

Proof. Using the definition of strongly convex B.3, let $f(\cdot)$ be the expected loss $L(f) = E_{(x,y)\sim\mathcal{P}}[l(f(x), y)]$, $\lambda = \lambda_1$, $y = f_{span}^*$, $x = f^*$ and use the fact that origin point $\bar{0}$ and $f_{span}^* \in span\{f_i\}$ and $\nabla L(f^*) = \bar{0}$ we have:

$$\begin{aligned} & L(f_{span}^*) \geq L(f^*) + \left\langle \nabla L(f^*), f_{span}^* - f^* \right\rangle + \frac{\lambda_1}{2}\|f_{span}^* - f^*\|^2 \\ \Rightarrow\ & L(\bar{0}) \geq L(f_{span}^*) \geq L(f^*) + \frac{\lambda_1}{2}\|f_{span}^* - f^*\|^2 \\ \Rightarrow\ & \mathrm{E}\left[\|f_{span}^* - f^*\|^2\right] \leq \frac{2}{\lambda_1}(L(\bar{0}) - L(f^*)). \end{aligned}$$

Although we never know the value of $L(\bar{0})$ and $L(f^*)$, but they are fixed and finite.

For the case of strictly convex, using the definition of strictly convex B.1, we have for $\forall x \neq y \in domL(\cdot)$ and a positive function $\sigma(x,y)$:

$$\begin{aligned} & L(y) > L(x) + \langle \nabla L(x), y - x \rangle \\ \Rightarrow\ & L(y) = L(x) + \langle \nabla L(x), y - x \rangle + \sigma(x,y)\|y - x\|^2. \end{aligned}$$

Now let $x = f^*$ and $\sigma = \min\limits_{\substack{y \neq f^* \\ y \neq \infty}} \sigma(f^*, y) > 0$. We can see that σ is uniquely determined by $L(\cdot)$ (hence fixed for fixed $l(\cdot,\cdot)$ and $\mathcal{P}$) and irrelevant to y. So $L(y) \geq L(f^*) + \sigma\|y - f^*\|^2$

holds for any finite $y \neq f^* \in domL(\cdot)$ and using the same method in s.c. case, we get the result. Note that $y \neq \infty$ means no component of y is ∞ or $-\infty$ since we usually assume that data points are bounded and hence the output classifiers, so is f^*_{span}. The actual bound should be better since we can set y to be some point as long as $\sigma(f^*, y) \leq \sigma(f^*, f^*_{span})$.

For the case of convex, we have for $\forall x, y \in domL(\cdot)$, $L(y) \geq L(x) + \langle \nabla L(x), y - x \rangle$. we decompose the domain of $L(\cdot)$ into two disjoint sets D_1 and D_2 s.t. for $\forall y \in D_1$, $L(y) > L(f^*)$ and for $\forall y \in D_2$, $L(y) = L(f^*)$. Note that D_1 and D_2 are fixed for fixed $L(\cdot)$. For the first case ($f^*_{span} \in D_1$), using the same idea in strictly convex and set $\sigma_{D_1} = \min_{\substack{y \in D_1, y \neq f^* \\ y \neq \infty}} \sigma(f^*, y)$, we get the result. For the second case ($f^*_{span} \in D_2$), D_2 is the optimal set of $\min L(\cdot)$ and we simply set $f^* = f^*_{span}$. □

Although Lemma 4.11 provide a nontrivial bound on $\|f^*_{span} - f^*\|$, we believe that this is not the best bound. One reason is that the value of σ (σ_{D_1}) might be very small. Intuitively when N is small ($N \leq d$) and increase, the dimension of $span\{f_i\}$ increase w.h.p and hence f^*_{span} become close to f^*. Also notice that larger N means more data, which will improve performance. This is the reason that when testing the effect of N on the system in Chapter 5, the error rate decrease when N enlarge. So we have to consider the effect of increasing N and dimension of $span\{f_i\}$ together to see how it affect the final result.

Theorem 4.12. (Feature method)

Under the conditions of Lemma 4.8, 4.10,4.11 and inequality 4.21, we have:

$$\begin{aligned}
&\mathrm{E}\left[\|f_{\omega_{min}} - f^*\|^2\right] \\
&\leq 2\Bigg(\frac{8N}{(\lambda_1 + \Lambda\lambda_2)^2}\left(\frac{8C_4C_1^2}{m} + 2\Lambda^2 C_2 + 4C_1^2\right) + \frac{8N}{\lambda_1}(L(\bar{0}) - L(f^*)) + 4N\|f^*\|^2\Bigg) \\
&\times \frac{8}{(\lambda_1 + \Lambda\lambda_2)^2}\left(\frac{8C_4C_1^2}{m_0} + 4C_1^2 + 2\Lambda^2 C_3\right) + \frac{2}{\lambda_1}(L(\bar{0}) - L(f^*))\Big). \qquad (4.23)
\end{aligned}$$

Proof.

$$\begin{aligned}
\mathrm{E}\left[\|f_{\omega_{min}} - f^*\|^2\right] &\leq 2(\mathrm{E}\left[\|f_{\omega_{min}} - f^*_{span}\|^2\right] + \mathrm{E}\left[\|f^*_{span} - f^*\|^2\right]) \\
&\leq 2(2\mathrm{E}\left[|||A^T_{trans}|||^2_{op}\right]\mathrm{E}\left[\|\omega_{min} - \omega^*\|^2\right] + \mathrm{E}\left[\|f^*_{span} - f^*\|^2\right]), \qquad (4.24)
\end{aligned}$$

then we plug the results in Lemma 4.8, 4.10 and 4.11 into the inequality 4.24, we will get the result. □

Rewrite Theorem 4.12 in another way, we have:

$$\mathrm{E}\left[\|f_{\omega_{min}} - f^*\|^2\right] = \mathcal{O}(\frac{N}{mm_0}) + \mathcal{O}(\frac{N}{m}) + \mathcal{O}(\frac{N}{m_0}) + \mathcal{O}(N) + \frac{4}{\lambda_1}(L(\bar{0}) - L(f^*)). \tag{4.25}$$

It can be seen from inequality (4.25) that the convergence rate is reverse proportional to m and m_0. Thus larger size of data set definitely make things better. Since the bound is a linear increasing function of N, proper size of N tend to have good result. We will explore these three parameters experimentally in Chapter 5 to demonstrate our theoretical results. Besides m, m_0 and N, we can see that large regularization parameter Λ always bring us benefit since it enlarge the domain. Large strongly convex parameter λ make it more easier to find the minimum of function. It is interesting to see that small gradient of $R(\cdot)$ at f^* make things better. This is due to reason that if we take gradient of objective function of ERM and set it 0 at f^*, we will see that smaller $\nabla R(f^*)$ meas better result when fix data and domain.

Note that the result does not converge to 0 due to the constant terms. But it says that with limited data, the bound indeed shrink w.r.t some parameters. And we hope that the performance will become better.

Since we have the result to describe the behavior of feature method in non-private case, there is an interest to further explore the effect of privacy on whole system. We replace the normal regularized ERM procedure by the objective perturbation algorithm 2 at all local sites. Now there are two sources of randomness in our system: data and noise. From now on, we use $\mathrm{E}_{data}[\cdot]$ instead of $\mathrm{E}[\cdot]$ to emphasize that the expectation is w.r.t data. We introduce a lemma which replaces the role of Lemma 4.9 at the local sites. This corresponds to the **"Public-Private" case** in Sarwate et al. [8].

First denote $A_{transpp}$ (for Public-Private case) by the stack of private classifiers compute from local sites (matrix M_f in the feature method 5). We replace all A_{trans} by $A_{transpp}$ in the following:

$$A_{transpp} = \begin{bmatrix} \cdots & f_{priv1} & \cdots \\ \cdots & f_{priv2} & \cdots \\ & \vdots & \\ \cdots & f_{privN} & \cdots \end{bmatrix},$$

then we have the following lemma:

Lemma 4.13. (Bound on $\mathrm{E}\left[|||A^T_{transpp}|||^2_{op}\right]$ using objective perturbation at the local sites) *Let d, ϵ, δ be the dimension of data, privacy parameter and some constants in*

$(0,1)$ *respectively. Using the notations in Lemma 4.9 and under the conditions of inequality* (4.12)*, Assumption 4.3.5, inequality* (4.19) *and w.p. as least* $(1-\delta)^N$*:*

$$\mathrm{E}_{data}\left[|||A^T_{transpp}|||^2_{op}\right] \leq \frac{16Nd^2\log^2(d/\delta)}{(\lambda_1+\Lambda\lambda_2)^2m^2\epsilon^2} + \frac{16N}{(\lambda_1+\Lambda\lambda_2)^2}\left(\frac{8C_4C_1^2}{m} + 2\Lambda^2C_2 + 4C_1^2\right) + \frac{8N}{\lambda_1}(L(\bar{0}) - L(f^*)) + 4N\|f^*\|^2.$$

Proof. We bound the error $\mathrm{E}_{data}\left[\|f_{priv_i} - f^*_{span}\|^2\right]$ by two parts:

$$\mathrm{E}_{data}\left[\|f_{priv_i} - f^*_{span}\|^2\right] \leq 2(\mathrm{E}_{data}\left[\|f_{priv_i} - f_i\|^2\right] + \mathrm{E}_{data}\left[\|f_i - f^*_{span}\|^2\right]). \tag{4.26}$$

The intuition of inequality (4.26) is clear: the first part is caused by noise added at local site i and the second part is normal regularized ERM. Now we bound the first part.

Let b be the noise vector drawn from distribution (3.1) and

$$f_i = \underset{f}{\operatorname{argmin}}\, J_R(f, D_{local,i}),$$
$$f_{priv_i} = \underset{f}{\operatorname{argmin}}\left(J_R(f, D_{local,i}) + \frac{1}{m}b^Tf\right).$$

Applying lemma 7 in Chaudhuri et al. [78], we have

$$\|f_i - f_{priv_i}\| \leq \frac{1}{\lambda_1+\Lambda\lambda_2}\|\nabla(\frac{1}{m}b^Tf)\|,$$
$$\Rightarrow \mathrm{E}_{data}\left[\|f_i - f_{priv_i}\|^2\right] \leq \frac{\|b\|^2}{m^2(\lambda_1+\Lambda\lambda_2)^2}. \tag{4.27}$$

Note that the right part of inequality (4.27) contains no randomness from data.

The second part is from Lemma 4.9.

Since we known that $\|b\|$ is drawn from the distribution of $\Gamma(d, \frac{2}{\epsilon})$. Then using Lemma 17 in Chaudhuri et al. [78], w.p. at least $1-\delta$, $\|b\|^2 \leq \frac{4d^2\log^2(d/\delta)}{\epsilon^2}$. Now combining these two parts we get: then w.p. at least $1-\delta$:

$$\mathrm{E}_{data}\left[\|f_{priv_i} - f^*_{span}\|^2\right] \leq \frac{8d^2\log^2(d/\delta)}{(\lambda_1+\Lambda\lambda_2)^2m^2\epsilon^2} + \frac{8}{(\lambda_1+\Lambda\lambda_2)^2}\left(\frac{8C_4C_1^2}{m} + 2\Lambda^2C_2 + 4C_1^2\right). \tag{4.28}$$

Finally, we substitute inequality 4.28 into inequality 4.19 (replace f_i by f_{priv_i}) and then use Assumption 4.3.5 and inequality (4.12) we can get the result. □

Now we have the following theorem for feature method with privacy at all local sites.

Theorem 4.14. (Feature method in public-private case)

Under the conditions of inequality 4.24, Lemmas 4.8, 4.11, 4.13, w.p. at least $(1-\delta)^N$, *we have:*

$$\begin{aligned}
\mathrm{E}_{data}\left[\|f_{\omega_{min}} - f^*\|^2\right] \leq 2((&\frac{16Nd^2\log^2(d/\delta)}{(\lambda_1+\Lambda\lambda_2)^2m^2\epsilon^2} + \frac{16N}{(\lambda_1+\Lambda\lambda_2)^2}\left(\frac{8C_4C_1^2}{m} + 2\Lambda^2C_2 + 4C_1^2\right)\\
&+ \frac{8N}{\lambda_1}(L(\bar{0}) - L(f^*)) + 4N\|f^*\|^2)\\
&\times \frac{8}{(\lambda_1+\Lambda\lambda_2)^2}\left(\frac{8C_4C_1^2}{m_0} + 2\Lambda^2C_3 + 4C_1^2\right) + \frac{2}{\lambda_1}(L(\bar{0}) - L(f^*)))\\
&= \mathcal{O}(\frac{N}{m^2\epsilon^2}) + \mathcal{O}(\frac{N}{m_0m^2\epsilon^2}) + \mathcal{O}(\frac{N}{mm_0}) + \mathcal{O}(\frac{N}{m}) + \mathcal{O}(\frac{N}{m_0}) + \mathcal{O}(N)\\
&+ \frac{4}{\lambda_1}(L(\bar{0}) - L(f^*)). \qquad (4.29)
\end{aligned}$$

Proof. We use the decomposition (4.24) and plug the results in Lemma 4.8, 4.11 and 4.13, the only difference here is that we replace A_{trans} by $A_{transpp}$. □

It can be seen that larger ϵ makes performance better, which matches our intuition and the results in single machine in Chapter 3. We will also test this by experiments in Chapter 5.

Finally, corresponding to the **fully-private case** mentioned in Sarwate et al. [8], we use objective perturbation at both the local sites and the aggregation site.

It is easy to get the following Lemma to describe the learning procedure at the aggregation site using objective perturbation:

Lemma 4.15. *Let* $\omega_{priv} = \mathrm{argmin}_\omega\left(J_R(\omega, D_0) + \frac{1}{m_0}b^T\omega\right)$, *where* $b \in \mathcal{R}^N$. *Using the notations in Lemma 4.8, then w.p. at least* $1-\delta$*:*

$$\begin{aligned}
\mathrm{E}_{data}\left[\|\omega_{priv} - \omega^*\|^2\right] \leq &\frac{8N^2\log^2(N/\delta)}{(\lambda_1+\Lambda\lambda_2)^2m_0^2\epsilon^2}\\
&+ \frac{8}{(\lambda_1+\Lambda\lambda_2)^2}\left(\frac{8C_4C_1^2}{m_0} + 4C_1^2 + 2\Lambda^2C_3\right).
\end{aligned}$$

Proof. The proof here is the same as inequality 4.28 except that we consider everything in $\mathcal{R}^N$ and $\|b\|$ is drawn from the distribution of $\Gamma(N, \frac{2}{\epsilon})$. □

With the help of previous results, we have the following theorem for fully-private case feature method:

Theorem 4.16. (Feature method in fully-private case)

Under the conditions of Lemmas 4.11, 4.13, 4.15 and decomposition (4.24), w.p. at least $(1-\delta)^{N+1}$*, we have:*

$$\begin{aligned}
\mathrm{E}_{data}\left[\|f_{\omega_{min}} - f^*\|^2\right] \leq 2((&\frac{16Nd^2\log^2(d/\delta)}{(\lambda_1+\Lambda\lambda_2)^2m^2\epsilon^2} + \frac{16N}{(\lambda_1+\Lambda\lambda_2)^2}\left(\frac{8C_4C_1^2}{m} + 4C_1^2 + 2\Lambda^2C_2\right)\\
&+ \frac{8N}{\lambda_1}(L(\bar{0}) - L(f^*)) + 4N\|f^*\|^2)\\
&\times (\frac{16N^2\log^2(N/\delta)}{(\lambda_1+\Lambda\lambda_2)^2m_0^2\epsilon^2} + \frac{16}{(\lambda_1+\Lambda\lambda_2)^2}\left(\frac{8C_4C_1^2}{m_0} + 4C_1^2 + 2\Lambda^2C_3\right))\\
&+ \frac{2}{\lambda_1}(L(\bar{0}) - L(f^*)))\\
&= \mathcal{O}(\frac{N^3\log^2(N/\delta)}{m^2m_0^2\epsilon^2\epsilon_0^2}) + \mathcal{O}(\frac{N^3\log^2(N/\delta)}{mm_0^2\epsilon_0^2}) + \mathcal{O}(\frac{N^3\log^2(N/\delta)}{m_0^2\epsilon_0^2})\\
&+ \mathcal{O}(\frac{N}{m^2\epsilon^2}) + \mathcal{O}(\frac{N}{m_0m^2\epsilon^2}) + \mathcal{O}(\frac{N}{mm_0}) + \mathcal{O}(\frac{N}{m}) + \mathcal{O}(\frac{N}{m_0}) + \mathcal{O}(N)\\
&+ \frac{4}{\lambda_1}(L(\bar{0}) - L(f^*)).
\end{aligned} \tag{4.30}$$

where we use ϵ_0 as the privacy parameter at the aggregation site.

Proof. Replacing A_{trans} by $A_{transpp}$ and ω_{min} by ω_{priv}, we substitute Lemmas 4.11, 4.13 and 4.15, into decomposition (4.24) and get the result. □

It is worth noting that we assume that the variables like parameters ϵ, m and Λ and functions $R(\cdot)$ are the same at all sites. The main results in this chapter can be easily generalized when these variables are different from site to site due to their own choice. Note that $l(\cdot,\cdot)$ should be the same at all sites since we want to approach the same f^*. Also, if we replace the loss function and regularization term in Theorem 4.12, 4.14 and 4.16 with specific choice such as logistic regression loss and huber loss, we will get more useful bounds. For convenience, these variables we test in Chapter 5 are the same at all sites unless otherwise specified.

Also, it is necessary to explain why we use the MSE of output classifier and f^* as the measurement of the performance of whole system's output classifier. Chaudhuri et al. [78] use $L(f)$ to measure the performance of output classifier f, where L is the expected loss (see Page x). This is reasonable since data point do not come from testing set but from its distribution and any output classifier is evaluated under this real-world case. The disadvantage of this is that we can never use $L(f)$ as y-axis in plotting since we don't know data distribution $\mathcal{P}$, which make the theoretical results not that intuitive. Kifer et al. [80] use $J_R(f,D)$ as the measurement. $J_R(f,D)$ can be used as y-axis in plotting but it costs a lot to calculate it. Using MSE can be both intuitive and easy-to-calculate. According to our definition of misclassification error rate 5.1,

$\frac{1}{k}\sum_k I_0(fx_jy_j < 0) \approx -\frac{1}{k}\sum_k fx_jy_j \approx -\mathrm{E}_{data}[fxy]$ can be used as a measurement of generalized error rate. In other words, $\mathrm{E}_{data}[f^*xy - fxy]$ is also a good choice, where (x,y) is a data-label pair draw from $\mathcal{P}$. Apparently we have $\mathrm{E}_{data}[f^*xy - fxy] \leq \mathrm{E}_{data}[\|f - f^*\|\|xy\|] \leq a\mathrm{E}_{data}[\|f - f^*\|]$ for constant a if data is bounded. If we plot fxy as y-axis then we have $\mathrm{E}_{data}[fxy] \leq a\sqrt{\mathrm{E}_{data}[\|f - f^*\|^2]} + b$. We can see that the value of $\frac{1}{k}\sum_k I_0(fx_jy_j < 0)$ is "consistent" with the value of $\mathrm{E}_{data}\left[\|f - f^*\|^2\right]$. So the plotting reflects the error rate directly and fxy is very easy to calculate.

Now we have three Theorems 4.12 4.14 4.16 to describe the behavior of the feature method under different privacy conditions. In Chapter 5 we will conduct series of experiments to test the parameters in them.

Chapter 5

Comparison of two methods

In this chapter, we conduct some experiments to validate of our theoretical analysis in chapter 4 and the advantages of the distributed learning model w.r.t the local and global models. We illustrate different trade-offs to provide some intuitions about when distributed learning is feasible.

We provide a series of experiments under several privacy conditions to illustrate the average method and the feature method. These experiments correspond to the three theorems in chapter 4. We will analyze them and draw some important conclusions about the performance of feature method compare with the average method.

In the last section we will conduct experiments to show that by using differentially private classifiers learned from multiple local sites, an aggregation site can learn a classifier that significantly outperforms classifiers learned from a single local site. This illustrates the benefit of differential privacy: it enables sharing data derivatives, and incentivizes sharing access to data that can improve overall accuracy.

In this chapter we mainly use six parameters. We test the effect of parameter ϵ (privacy parameter), m (number of data points in one local site), m_0 (number of data points in the aggregation site) and N (number of local sites) while fixing other parameters. Also h denotes Huber constant and Λ denotes regularization parameter.

We use misclassification error rate to measure performance in our experiments. For any linear classifier f and a data point (x_j, y_j), we denote misclassification error (empirical error) by e_j and define it as:

$$\mathrm{e}_j = \begin{cases} 1 & \text{if} \quad y_j f^T x_j < 0 \\ 0 & \text{if} \quad y_j f^T x_j \geq 0. \end{cases}$$

Then we define the error rate as:

$$\text{error rate} = \frac{1}{k}\sum_{k} \mathrm{e}_j. \tag{5.1}$$

if we have totally k data points in testing set.

From now on, we denote $\mathrm{e}(\epsilon, N, m, m_0, h, \Lambda)$ to be the error rate and the parameters that are associated with it. Also, we use subscripts $\mathrm{e}_{\mathsf{Avg,None}}$, $\mathrm{e}_{\mathsf{Avg,Loc}}$, $\mathrm{e}_{\mathsf{Avg,Agg}}$, $\mathrm{e}_{\mathsf{Avg,Full}}$ as error rate of the average method without privacy, with privacy only at the local sites, with privacy only at the aggregation site and with privacy at both local and aggregation sites. With same meaning, $\mathrm{e}_{\mathsf{Feat,None}}$, $\mathrm{e}_{\mathsf{Feat,Loc}}$, $\mathrm{e}_{\mathsf{Feat,Agg}}$, $\mathrm{e}_{\mathsf{Feat,Full}}$ are the corresponding subscripts for the feature method. In the following description, we will use the combination of these notations to denote the meaning.

5.1 Trade-offs in distributed system

There are several interesting trade-offs in distributed learning. Through these trade-offs we can see how the performance of distributed learning changes w.r.t some important factors.

5.1.1 Privacy vs. accuracy in one local site

How does the error e **vary with as the privacy level** ϵ **changes in a single local site?** Fix the number of sites N, number of training points m per local site, number of points m_0 at the aggregation site, and parameters h and Λ from the ERM algorithm. We set the first local site in our system as an "auxiliary site." The auxiliary site has a privacy parameter ϵ' and the remaining $N-1$ sites have privacy parameter ϵ. The system is shown in Figure 5.1. We consider two scenarios: one where the $N-1$ sites do non-private training ($\epsilon = \infty$) and the other one where they do differentially private training ($\epsilon = 2$). This two experiments represent practical situations because in reality each local institute that provides data can decide its own level of privacy independently. Also, the first case can be seen as a base experiment in which no privacy is provided at local sites other than the test site. Our goal is to understand how the auxiliary site's different privacy requirement ϵ' affects the performance of the average and feature method. We compare the average method with feature method under non-private training by plotting $\mathrm{e}_{\mathsf{Avg,None}}$ and $\mathrm{e}_{\mathsf{Feat,None}}$ in Figure 5.2 as a function of ϵ'. Similarly, we show $\mathrm{e}_{\mathsf{Avg,Loc}}$ and $\mathrm{e}_{\mathsf{Feat,Loc}}$ in Figure 5.3. The results shown in these two figures indicate that even changing the privacy level ϵ_0 at a single site (auxiliary site

here) can have a significant effect on the final performance. The reasons are two-fold: firstly the noise we add on local sites (other than the test site) is low. Secondly, both the average and the feature methods combine the classifiers from local sites so that even a small change will significantly affect the result. This indicates that when the overall performance is poor, we should discard some classifiers or reduce their weight artificially. From these two figures we can see that the feature method works better than the average method. This is because the feature method weighs each classifier from the local sites using extra data by non-private ERM, rather than setting the importance of each local classifier equally. However, it can be seen that this benefit only holds under certain conditions, rather than being universally true. Another interesting phenomena that can be seen when comparing these two figures is that the feature method is stabler when no noise is added at local site other than the test site. This is because we add extra noise on the second case, which make the error rate higher. Also, when all the classifiers from local sites other than the test set are good (non-private), a change in performance of the test site affects the overall result little. So the first case is stabler.

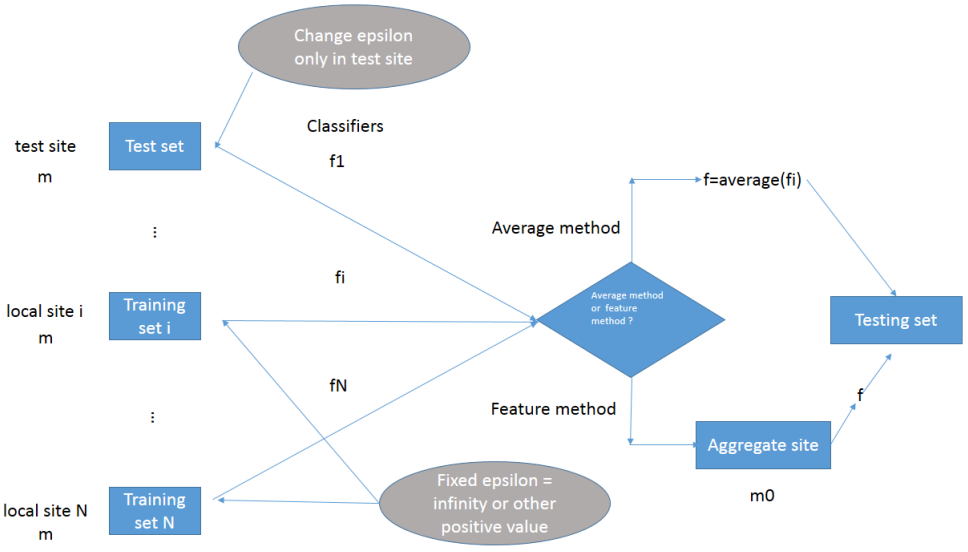


FIGURE 5.1: Privacy vs. accuracy in one local site

5.1.2 Number of data points in one local site vs. accuracy

How does the error e **vary with as the number of data points** m **changes in one local site?** Fix the number of sites N, privacy parameter ϵ per local site, number of points m_0 at the aggregation site, and parameters h and Λ from the ERM algorithm. We set the number of data points in auxiliary site as m' and the remaining $N-1$ sites have m data points in each site. The system is shown in Figure 5.1. We also consider the cases where either all N local sites do non-private training or differentially private training.

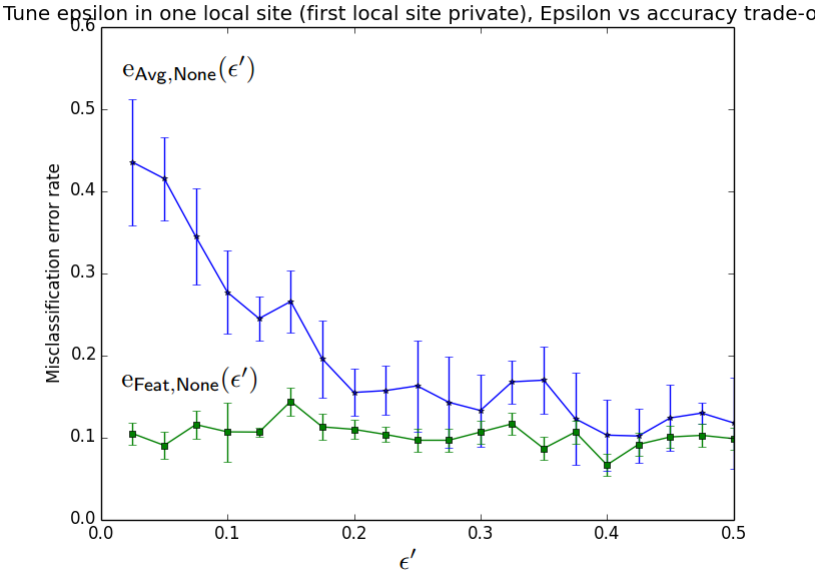


FIGURE 5.2: Private in auxiliary site, $e(\epsilon', \epsilon = \infty, N = 10, m = 789, m_0 = 789, h = 0.5, \Lambda = 0.01)$ vs ϵ' using '37' in MNIST dataset, ϵ' is from 0.025 to 0.5 with step 0.025, run the whole system 10 times for fixed ϵ' and plot with error bar.

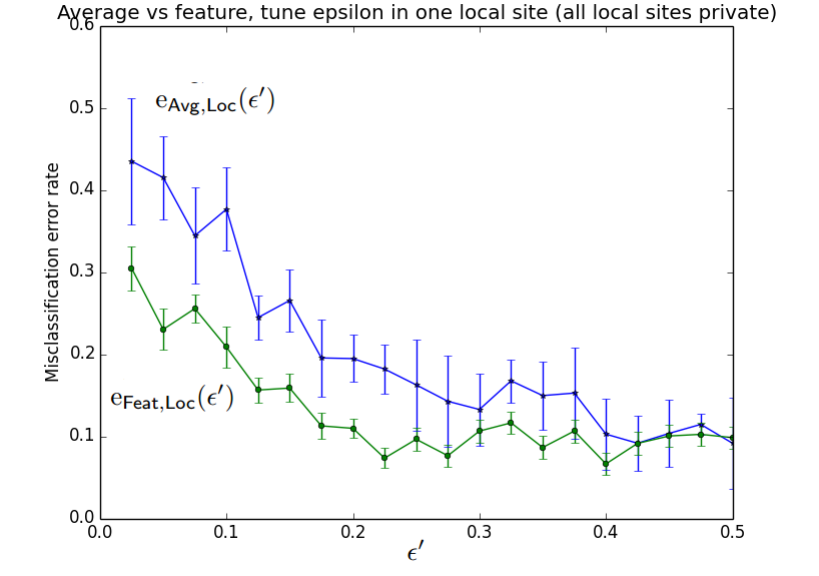


FIGURE 5.3: Private in all sites, $e(\epsilon', \epsilon = 2, N = 10, m = 789, m_0 = 789, h = 0.5, \Lambda = 0.01)$ vs ϵ' using '37' in MNIST dataset, ϵ' is from 0.025 to 0.5 with step 0.025, run the whole system 10 times for fixed ϵ' and plot with error bar.

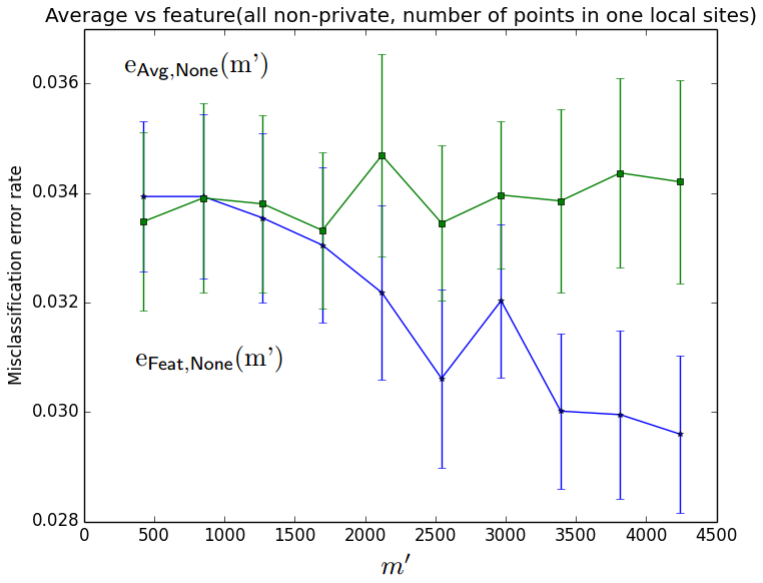


FIGURE 5.4: All non-private, $e(\epsilon = \infty, N = 10, m = 413, m_0 = 413, h = 0.5, \Lambda = 0.01)$ vs m' using '37' in MNIST dataset, m' is from 413 to 4546 with step 413, run the whole system 10 times for fixed m' and plot with error bar.

This two experiments represent the situations in reality where each local institute that provides data independently. Our goal is to understand how the auxiliary site's different amount of data points m' affects the performance of the average and feature method. We compare the average and feature method under non-private training by plotting $e_{Avg,None}$ and $e_{Feat,None}$ in Figure 5.4 as a function of m'. Similarly, we show $e_{Avg,Loc}$ and $e_{Feat,Loc}$ in Figure 5.5. From the results in Figure 5.4 and Figure 5.5, we conclude that when we increase the data from one local site (or any local site), we enjoy a better performance. We can see that the performance in Figure 5.4 is much better than that in Figure 5.5 due to the effect of noise on local sites. Also it is interesting to see that in both experiments the feature method is stabler than average method. This is because the feature method uses data in aggregation site to adjust the weight of classifier of each local site when the classifier changes. This makes the system "react" to the changes of parameters. But the average method can not achieve this goal. Note that the performance of the feature method becomes better than the average method in Figure 5.5. This phenomenen can also be explained by the mechanism of the feature method. This experiment stimulates the situation in reality that different sites have different numbers of data points. But we cannot get as much data as we need and balancing the number of points in each site and the number of sites, is important, too. As a base experiment, these two results (together with last two experiments) show that the distributed system behaves the same as a single site (comparing with experiments in Chapter 3), as long as we test one parameter at one time. The phenomenon is also demonstrated in many other studies [54, 76, 106, 107] with different convergence rates.

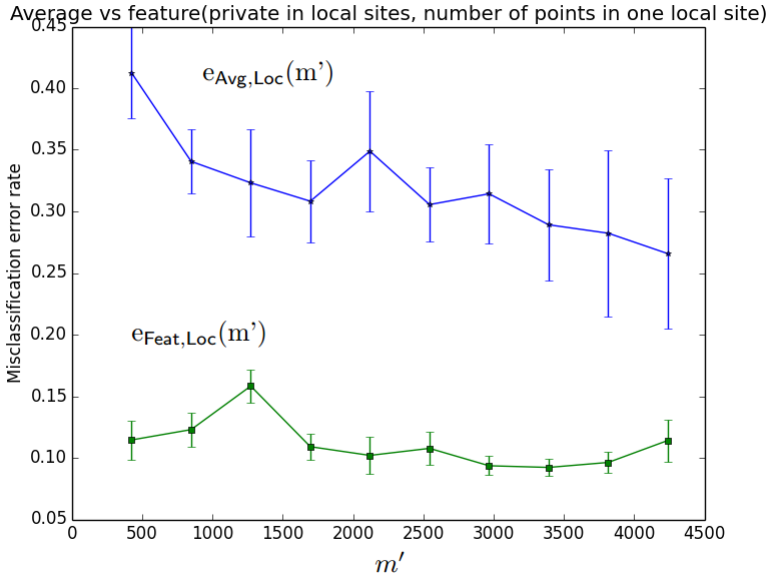


FIGURE 5.5: Private in all sites, e($\epsilon = 2, N = 10, m = 413, m_0 = 413, h = 0.5, \Lambda = 0.01$) vs m' using '37' in MNIST dataset, m' is from 413 to 4546 with step 413, run the whole system 10 times for fixed m' and plot with error bar.

5.2 Feature method vs average method: experiments

In this section, we will compare the average method with the feature method by experiments on real data. In each experiment, we first provide the situation the experiment describe. Then we set the goal and information of experiment. We give analysis of each figure and show how they demonstrate the previous theoretical results (Theorem 4.12 4.14 and 4.16 in Chapter 4) and our intuitions. Then we compare the two methods. Also, we compare the results with results in previous subsections and do some analysis on interesting phenomena and differences among data sets. Finally, we give conclusions of this experiment.

5.2.1 Experiment I: non-private case

How average method and feature method perform when there is no privacy concern? Fix the number of sites N, number of training points m per local site, number of points m_0 at the aggregation site, and parameters h and Λ from the ERM algorithm. Our goal is to show the performance of the average method and the feature method under non-private case. We compare them by plotting histogram in Figure 5.6 to Figure 5.9. The regularization constant here is chosen by a 5-fold cross validation. It can be seen from these two histograms that when all the parameters are fixed, feature method performs better than the average method, especially when using Covertype

data set. Also, they have similar distribution. It is not surprising that this happens because given the same local classifiers, feature method has the ability to choose the weight of each local classifier adaptively according to the aggregation site and output a combination of them. This is generally better than treat each local classifier the same. But this also requires a good aggregation site. So it is possible that the results of two methods are close to each other. Therefore, we conclude that under the condition of non-differentially private distributed system[1], when parameters are all the same, we prefer feature method to perform distributed learning.

How does the error e **vary with as the number of data points** m **changes at each local site?**The goal of this experiment is to understand how the number of training points m of each data provider affects the performance of the average and feature method. We plot $\mathrm{e}_{\mathsf{Avg,None}}$ and $\mathrm{e}_{\mathsf{Feat,None}}$ in Figure 5.10 and 5.11 as a function of m. Theorem 1 in Zhang et al. [88] and Theorem 4.12 indicate that increasing[2] data at local site(s) will decrease the MSE between the output classifier and f^* in both methods. We therefore conclude that MSE is decreasing when we add data on local sites.

How does the error rate e **vary with as the number of data points** m_0 **changes at the aggregation site?** Our goal is to understand how the number of data points in the aggregation site m_0 affects the performance of the average and feature method. We compare the average and feature method under non-private training by plotting $\mathrm{e}_{\mathsf{Avg,None}}$ and $\mathrm{e}_{\mathsf{Feat,None}}$ in Figure 5.12 and Figure 5.13 as a function of m_0. The results demonstrate the analysis in Theorem 4.12 that when number of data points increase at the aggregation site, MSE gets smaller (hence error rate goes down). The results show that the feature method performs much better than the average method when we have a large aggregation data set (This phenomena is especially clear when using Covertype data set). This phenomena shows the advantage of feature method: it uses extra data to improve the result. So we have the conclusion that the MSE decreases when we add data on the aggregation site and we should consider using the feature method when there is a large aggregation data set.

How does the error e **vary with as the number of local sites** N **changes?**Our goal is to understand how the number of data providers N affects the performance of the average and feature method. We compare the average and feature method under non-private training by plotting $\mathrm{e}_{\mathsf{Avg,None}}$ and $\mathrm{e}_{\mathsf{Feat,None}}$ in Figure 5.14 and Figure 5.15 as a function of N. Theorem 1 in Zhang et al. [88] indicates that the average method performs better when number of local sites increase, which is shown in these two figures. Theorem 4.12 shows that the feature method gets worse when number of local sites

[1]This condition holds for all experiments in this section

[2]When tuning one parameter, we always assume that other parameters are fixed unless otherwise specified, similarly hereinafter.

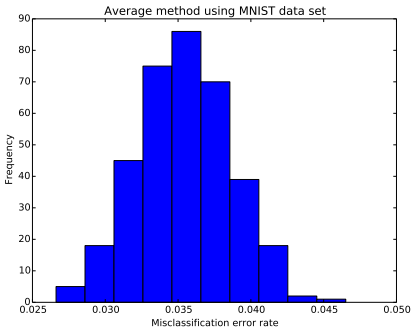


FIGURE 5.6: $e(\epsilon = \infty, N = 10, m = 789, h = 0.5, \Lambda = 0.01)$, run the whole system 360 times and plot the histogram.

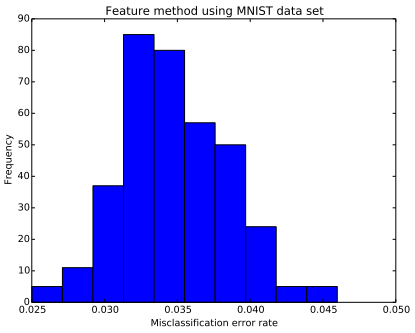


FIGURE 5.7: $e(\epsilon = \infty, N = 10, m = 789, m_0 = 789, h = 0.5, \Lambda = 0.01)$, run the whole system 360 times and plot the histogram.

increase, which directly contradicts our intuition that whenever the total number of training data points increases, we get a better result. But it can be seen from these two figures that the error rate of the feature method decreases before increasing, which can not be explained by Theorem 4.12. An explanation of this phenomenon is that when the number of local sites is small, increasing the number of local sites is equivalent to adding the dimension of $span\{f_i\}$ (see Definition 4.2), which means that the $\|f^*_{span} - f^*\|$ decrease w.h.p. Since we have the decomposition $\mathrm{E}[\|f_{output} - f^*_{span}\|^2] \leq 2(\mathrm{E}[\|f_{output} - f^*_{span}\|^2] + \mathrm{E}[\|f^*_{span} - f^*\|^2])$, we may guess that overall performance will get better. When the number of local sites increases, $f^* \in span\{f_i\}$ almost certainly. (hence $f^* = f^*_{span}$) and adding more classifiers (local sites) will be useless and thus becomes a burden to system. At this time, the model fits Theorem 4.12 and error rate goes up. Also, notice that the fluctuations in the figures indicate that even if the dimension of $span\{f_i\}$ is high, $f^* \in span\{f_i\}$ does not necessarily hold because of the randomness that comes from data redistribution. This tells us that we must choose suitable N so that in real world cases the error rate is small enough and in other experiments N won't become a factor that separate the average method and the feature method. We conclude here that when we increase the number of local sites, the MSE decreases when we use the average method but displace a local minimum when we use the feature method.

5.2.2 Experiment II: private at local sites

Our next series of experiments focuses on the affect of privacy effect at local sites. Here we use the objective perturbation (Algorithm 2) at all local sites.

How does the privacy parameter ϵ at local sites affect the performance?

Our goal is to understand how ϵ affects the performance of the average and feature method by tuning ϵ at each local site simultaneously. We compare the average method

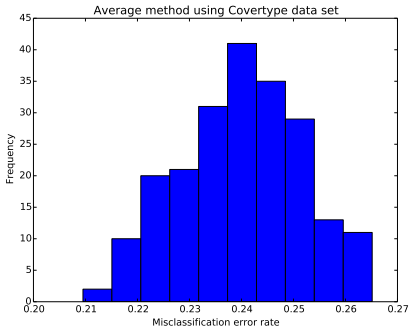


FIGURE 5.8: e($\epsilon = \infty, N = 10, m = 2411, h = 0.5, \Lambda = 0.00001$), run the whole system 200 times and plot the histogram.

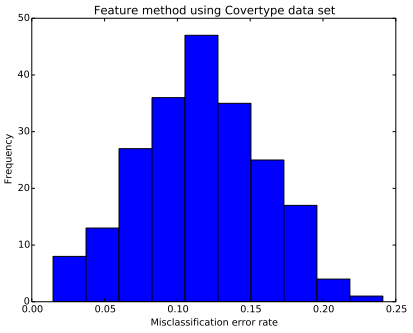


FIGURE 5.9: e($\epsilon = \infty, N = 10, m = 2411, m_0 = 2411, h = 0.5, \Lambda = 0.00001$), run the whole system 200 times and plot the histogram.

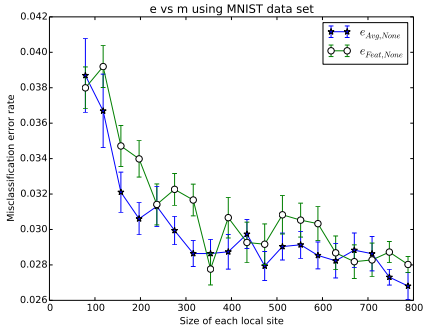


FIGURE 5.10: e($m, \epsilon = \infty, N = 10, m_0 = 789, h = 0.5, \Lambda = 0.01$) vs m, m is from 39 to 789 with step 39, run the whole system 10 times for fixed m and plot with error bar.

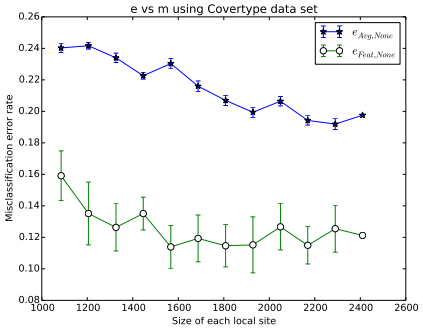


FIGURE 5.11: e($m, \epsilon = \infty, N = 10, m_0 = 2411, h = 0.5, \Lambda = 10^{-5}$) vs m, m is from 1085 to 2410 with step 120, run the whole system 10 times for fixed m and plot with error bar.

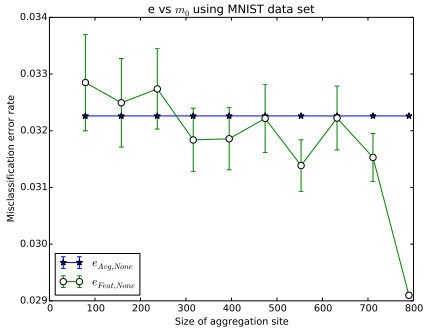


FIGURE 5.12: e($m_0, \epsilon = \infty, N = 10, m = 789, h = 0.5, \Lambda = 0.01$) vs m_0, m_0 is from 79 to 789 with step 79, run the whole system 10 times for fixed m_0 and plot with error bar.

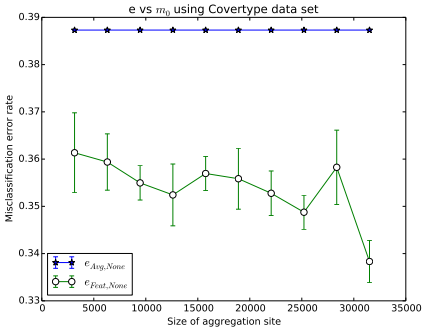


FIGURE 5.13: e($m_0, \epsilon = \infty, N = 10, m = 31509, h = 0.5, \Lambda = 10^{-6}$) vs m_0, m_0 is from 3151 to 31509 with step 3151, run the whole system 10 times for fixed m_0 and plot with error bar.

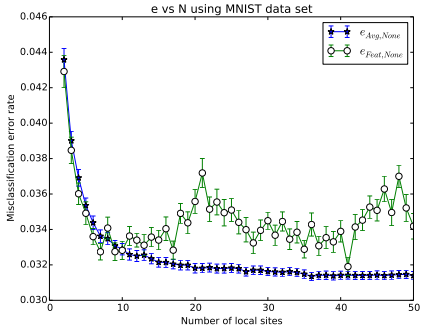


FIGURE 5.14: $e(N, \epsilon = \infty, m = 170, m_0 = 170, h = 0.5, \Lambda = 0.01)$ vs N, N is from 2 to 50 with step 1, run the whole system 100 times for fixed N and plot with error bar.

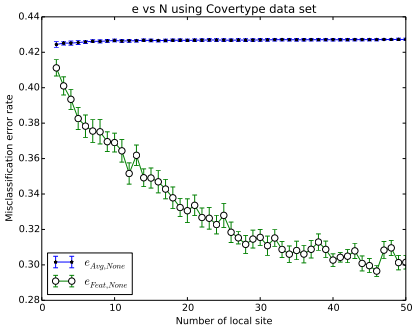


FIGURE 5.15: $e(N, \epsilon = \infty, m = 6796, m_0 = 6796, h = 0.5, \Lambda = 10^{-6})$ vs N, N is from 2 to 50 with step 1, run the whole system 50 times for fixed N and plot with error bar.

with feature method by plotting $e_{\mathsf{Avg,Loc}}$ and $e_{\mathsf{Feat,Loc}}$ in Figure 5.16 and5.17. These two figures fit our intuition and results in Chapter 3, where we use single data set and found that the performance became better and variance became smaller when we increased ϵ. Also, it shows an important character of the feature method under differential privacy: the feature method is less affected by noise than the average method. In these two experiments the randomness comes from noise instead of data redistribution[3]. Now we give an explanation of this phenomenon. From Theorem 4.14 we can see that the term $\mathcal{O}(\frac{N}{m^2\epsilon^2}) + \mathcal{O}(\frac{N}{m_0 m^2 \epsilon^2})$ that contains ϵ, which related with privacy, are relative small comparing with other terms due to the m and m_0 terms in dominator. So changing ϵ at local sites has little effect on the performance. But the average method uses each classifier from local sites more directly than the feature method, without the help of the aggregation site. The overall result is directly related with performance of each classifier. From Theorem 3.4 and Figure 3.1, increasing ϵ will raise performance. Thus, the average method gets better when we decrease the privacy risk at each local site. So our conclusion for this experiment is that for both method, increasing ϵ makes the final performance better and the feature method performs better than the average method and less affected by noise than the average method.

The next three experiments are the same as last three experiments in last section but now we limit privacy concern at local sites.

Our goal of this experiment is to understand how the number of data points in each local site m affects the performance of the average and feature method. We compare the average method with feature method by plotting $e_{\mathsf{Avg,Loc}}$ and $e_{\mathsf{Feat,Loc}}$ in Figure 5.18 and 5.19 as a function of m. The results demonstrate the theoretical prediction in Theorem 4.14 that increasing the number of data points at local sites will decrease the

[3]Note that in theoretical analysis in Chapter 4, the randomness comes from both noise and data

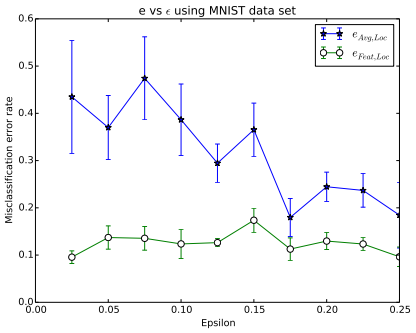


FIGURE 5.16: $e(\epsilon, N = 10, m = 789, m_0 = 789, h = 0.5, \Lambda = 0.01)$ vs ϵ, ϵ is from 0.025 to 0.25 with step 0.025, run the whole system 10 times for fixed ϵ and plot with error bar.

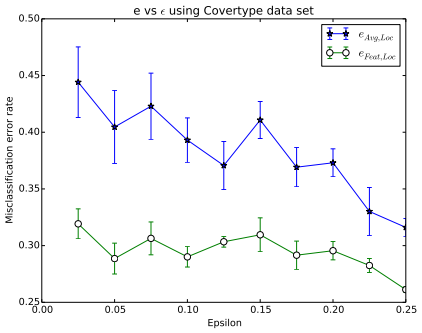


FIGURE 5.17: $e(\epsilon, N = 10, m = 42762, m_0 = 42762, h = 0.5, \Lambda = 0.0000001)$ vs ϵ, ϵ is from 0.025 to 0.25 with step 0.025, run the whole system 10 times for fixed ϵ and plot with error bar.

error rate. From these two figures we can see that feature method is more stable than the average method. Notice that they perform worse than the previous two results in Figure 5.10 and Figure 5.11 due to noise added at local sites (noise terms).

We want to understand how the number of data points at the aggregation site m_0 affects the performance of the average and feature method. We compare the average method with feature method under public-private training by plotting $\mathsf{e}_{\mathsf{Avg,Loc}}$ and $\mathsf{e}_{\mathsf{Feat,Loc}}$ in Figure 5.20 and 5.21 as a function of m_0. Our results and conclusions here are the same as the experiments corresponding to Figure 5.12 and Figure 5.13 except for the effect of the noise, which make performance worse. Notice that adding noise or not and changing data points at the aggregation (or local) site(s) work independently of each other, so the figures keep the general trend as previous ones. So the conclusion here is that under public-private case, system's performance get better when size of data at the aggregation site enlarge.

We want to understand how the number of local sites N affects the performance of the average and feature method. We compare the average method with feature method under public-private training by plotting $\mathsf{e}_{\mathsf{Avg,Loc}}$ and $\mathsf{e}_{\mathsf{Feat,Loc}}$ in Figure 5.22 and 5.3 as a function of N. These two figures show that the error of the average method decreases little and it performs worse than the previous results in Figure 5.14 and Figure 5.15. This is due to the effect of noise. Also, the average method performs worse than the feature method. This is because the average method is more sensitive to the noise than the feature method. The feature method performs better when N increases.

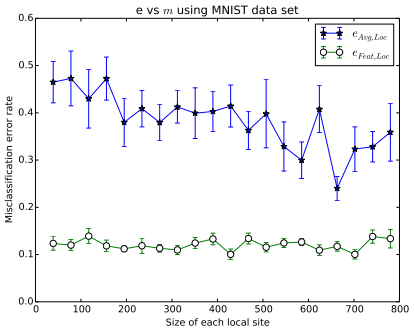


FIGURE 5.18: $e(m, \epsilon = 0.1, N = 10, m_0 = 789, h = 0.5, \Lambda = 0.01)$ vs m, m is from 39 to 789 with step 39, run the whole system 10 times for fixed m and plot with error bar.

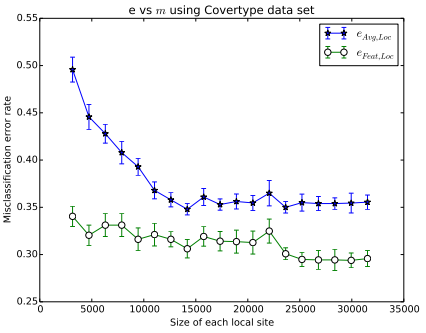


FIGURE 5.19: $e(m, \epsilon = 0.1, N = 10, m_0 = 31509, h = 0.5, \Lambda = 0.0000001)$ vs m, m is from 3151 to 31509 with step 1576, run the whole system 10 times for fixed m and plot with error bar.

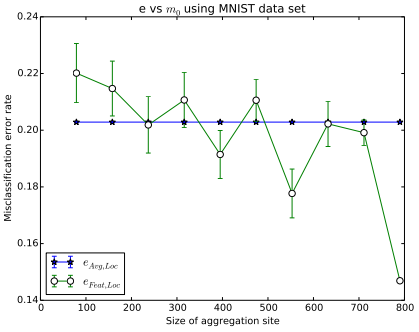


FIGURE 5.20: $e(m_0, \epsilon = 0.1, N = 10, m = 789, h = 0.5, \Lambda = 0.01)$ vs m_0, m_0 is from 79 to 789 with step 79, run the whole system 10 times for fixed m_0 and plot with error bar.

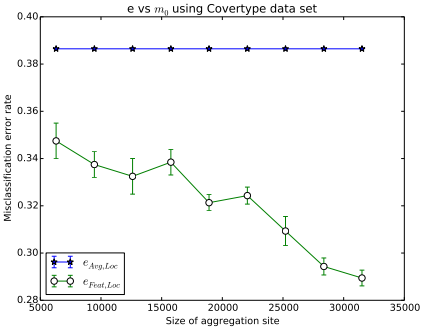


FIGURE 5.21: $e(m_0, \epsilon = 0.1, N = 10, m = 31509, h = 0.5, \Lambda = 0.0000001)$ vs m_0, m_0 is from 3151 to 31509 with step 3151, run the whole system 10 times for fixed m_0 and plot with error bar.

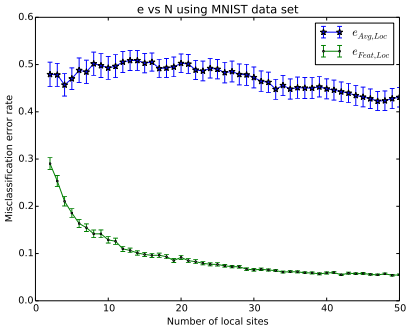


FIGURE 5.22: $e(N, \epsilon = 0.1, m = 170, m_0 = 170, h = 0.5, \Lambda = 0.01)$ vs N, N is from 2 to 50, run the whole system 50 times for fixed N and plot with error bar.

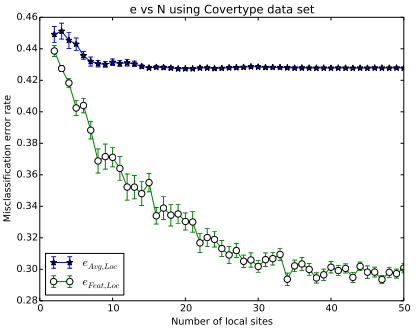


FIGURE 5.23: $e(N, \epsilon = 0.1, m = 9223, m_0 = 9223, h = 0.5, \Lambda = 0.0000001)$ vs N, N is from 2 to 50, run the whole system 50 times for fixed N and plot with error bar.

5.2.3 Experiment III: all-private case

Now we redo the four experiments in section 5.2.2 but this time we add a fully-private feature method and compare with the public-private case.

Our goal here is to understand how the local sites and the aggregation site privacy requirement ϵ affects the performance of the average and feature method. We compare the average method with feature method under fully-private training by plotting $\mathrm{e}_{\mathsf{Avg,Loc}}$, $\mathrm{e}_{\mathsf{Feat,Loc}}$ and $\mathrm{e}_{\mathsf{Feat,Full}}$ in Figure 5.24 and Figure 5.25 as a function of ϵ. Not surprisingly, all the results get better when ϵ increases, which is similar to previous results (Figure 5.16, 5.17). We can see that the fully-private feature method (green line) performs worse than the public-private feature method (red line). This demonstrates the difference of Theorem 4.14 and Theorem 4.16. Also, the fully-private feature method is better than the average method. This shows that the effect of extra aggregation site is much larger than the noise at it. So we may conclude here that feature method with non-private aggregation site is the best choice in real world distributed learning system.

In this experiment we want to understand how the number of data points at each local site m affects the performance of the average and feature method. We compare the average method with feature method under fully-private training by plotting $\mathrm{e}_{\mathsf{Avg,Loc}}$, $\mathrm{e}_{\mathsf{Feat,Loc}}$ and $\mathrm{e}_{\mathsf{Feat,Full}}$ in Figure 5.26 and Figure 5.27 as a function of m. It can be see that the average method performs better as the number of data points increases and the feature method seems unaffected by this. This phenomenon can be explained with the help of Theorem 4.14 and Theorem 4.16. We may guess, although we can not prove it directly since we know nothing about f^*, that the terms contain m change little due to their small value. Also notice that the feature method with privacy at the aggregation site performs worse than the feature method without privacy: this has already been explained in the last paragraph. So our conclusion here is also the same as the last paragraph.

Our goal is of this experiment to understand how the number of points m_0 affects the performance of the average and feature method. We compare the average method with feature method under fully-private training by plotting $\mathrm{e}_{\mathsf{Avg,Loc}}$, $\mathrm{e}_{\mathsf{Feat,Loc}}$ and $\mathrm{e}_{\mathsf{Feat,Full}}$ in Figure 5.28 and Figure 5.29 as a function of m_0. These two figures demonstrate the theoretical results in Theorem 4.14 and Theorem 4.16 that adding data points at the aggregation site makes the performance better. This is especially significant for the fully-private feature method. The fully-private feature method performs the worst in this experiment due to the reason that it both have noise added at local sites and at the aggregation site. Both the average method and the public-private feature method fluctuate significantly due to the noise.

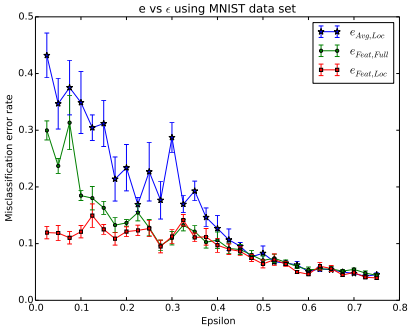


FIGURE 5.24: $e(\epsilon, N = 10, m = 789, m_0 = 789, h = 0.5, \Lambda = 0.01)$ vs ϵ, ϵ is from 0.025 to 0.75 with step 0.025, run the whole system 10 times for fixed ϵ and plot with error bar.

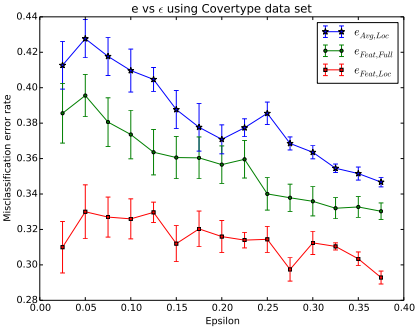


FIGURE 5.25: $e(\epsilon, N = 10, m = 31509, m_0 = 31509, h = 0.5, \Lambda = 0.0000001)$ vs ϵ, ϵ is from 0.025 to 0.5 with step 0.025, run the whole system 10 times for fixed ϵ and plot with error bar.

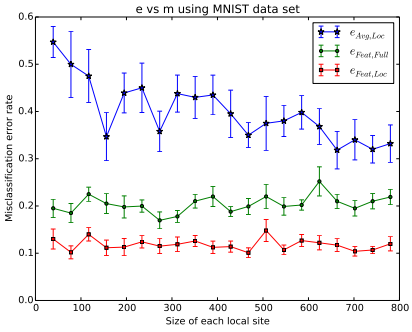


FIGURE 5.26: $e(m, \epsilon = 0.1, N = 10, m_0 = 789, h = 0.5, \Lambda = 0.01)$ vs m, m is from 39 to 789 with step 39, run the whole system 10 times for fixed m and plot with error bar.

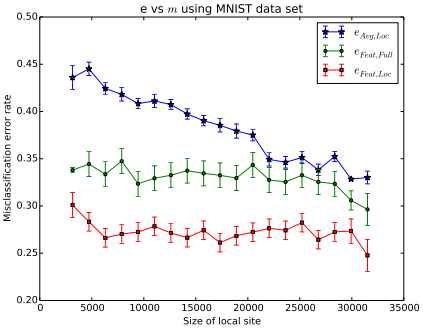


FIGURE 5.27: $e(m, \epsilon = 0.1, N = 10, m_0 = 31509, h = 0.5, \Lambda = 0.0000001)$ vs m, m is from 3151 to 31509 with step 1576, run the whole system 10 times for fixed m and plot with error bar.

Finally we want to understand how the number of local site N affects the performance of the average and feature method. We compare the average method with feature method under fully-private training by plotting $e_{Avg,Loc}$, $e_{Feat,Loc}$ and $e_{Feat,Full}$ in Figure 5.30 N. From this figure we can see that the error rate of two methods become better when N increase, which match our theoretical results in Chapter 4.

5.3 Three models

In this section we focus our attention on the comparisons among local model, global model and distributed model that we mentioned in section 4.2.

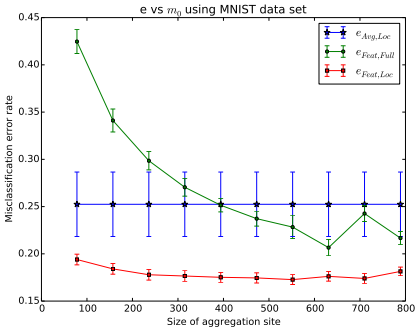


FIGURE 5.28: $e(m_0, \epsilon = 0.1, N = 10, m = 789, h = 0.5, \Lambda = 0.01)$ vs m_0, m_0 is from 79 to 789 with step 79, run the whole system 50 times for fixed m_0 and plot with error bar.

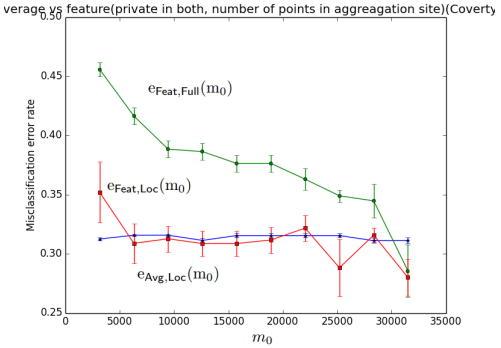


FIGURE 5.29: Private in both, $e(m_0, \epsilon = 0.1, N = 10, m = 31509, h = 0.5, \Lambda = 0.0000001)$ vs m_0 using cover type 1 and 2 in Covertype data set, m_0 is from 3151 to 31509 with step 3151, run the whole system 10 times for fixed m_0 and plot with error bar.

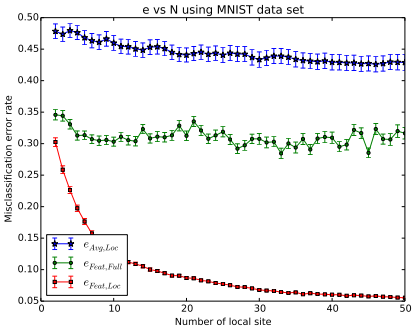


FIGURE 5.30: $e(N, \epsilon = 0.1, m_0 = 170, m = 170, h = 0.5, \Lambda = 0.01)$ vs N, N is from 2 to 50, run the whole system 50 times for fixed N and plot with error bar.

How the three models perform when changing ϵ ? In this series of experiments, we are ready to demonstrate our claim that distributed model indeed provides us with more benefit than local model and global model. Our experiments are to compare these three models in two cases: all non-private case and fully-private case (see Chapter 4). Here we use "l1" to "l10" to represent 10 local sites (local model) and "G" for global model, "F" for the feature method and "A" for the average method (distributed model).

The result of all non-private case is given in Figures 5.41. The results of fully-private case are shown from Figure 5.31 to Figure 5.40.

Despite of the decreasing error rate of all models due to increasing in ϵ, there are some other interesting phenomena that we can find from these figures. We can see that global model and distributed performs better than local model in general. This is due to the lack of data in local model. Next, as ϵ increasing, the local model changes the fastest, its error rate changes 36% and the global model changes the slowest, about 2%. This is due

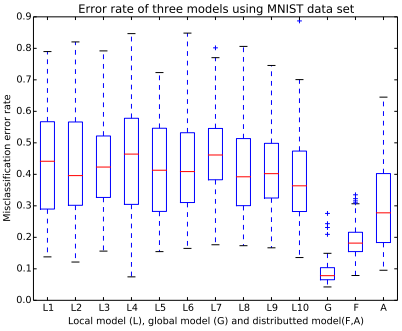


FIGURE 5.31: Comparison among models (private), $\epsilon = 0.1$

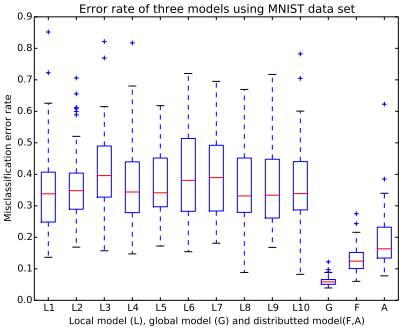


FIGURE 5.32: Comparison among models (private), $\epsilon = 0.2$

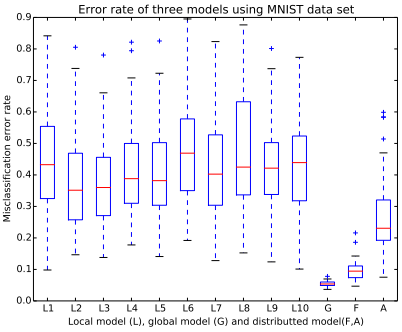


FIGURE 5.33: Comparison among models (private), $\epsilon = 0.3$

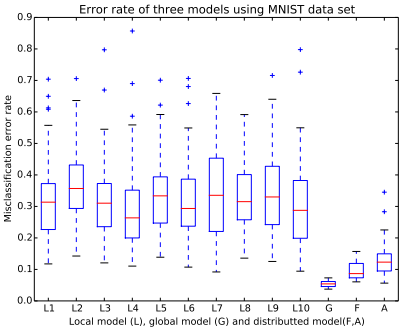


FIGURE 5.34: Comparison among models (private), $\epsilon = 0.4$

to the reason that the norm of noise vector draw from gamma distribution and its shape parameter $\theta \propto \frac{1}{n_{data}\epsilon}$. Hence for a smaller mount of data n_{data}, same change in ϵ makes larger change in shape parameter and hence makes the shape of gamma distribution change faster[4]. It can be seen that the distributed model has higher error rate than global model at start. As ϵ increases, the distributed model (both the average method and the feature method) performs closer to global model and finally better than it. This tells us that when system needs a low level of privacy (small ϵ), distributed model is definitely our best choice.

[4]For a random variable X draw from gamma distribution $f(x;\alpha,\beta)$, $\mathrm{E}(X) = \alpha\beta$,$\mathrm{V}(X) = \alpha\beta^2$

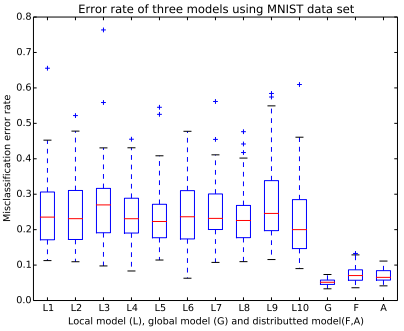


FIGURE 5.35: Comparison among models (private), $\epsilon = 0.5$

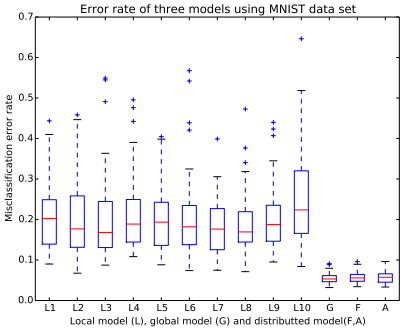


FIGURE 5.36: Comparison among models (private), $\epsilon = 0.6$

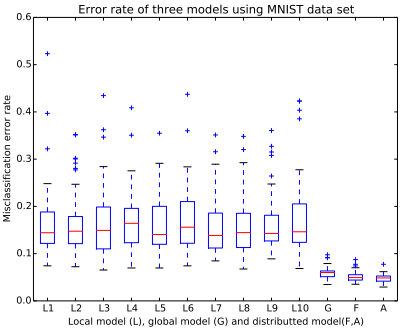


FIGURE 5.37: Comparison among models (private), $\epsilon = 0.7$

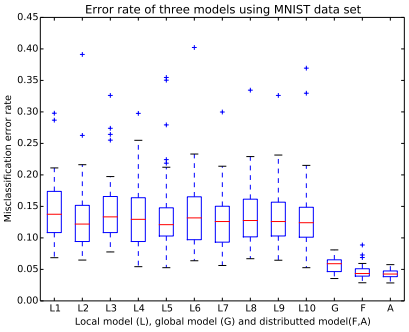


FIGURE 5.38: Comparison among models (private), $\epsilon = 0.8$

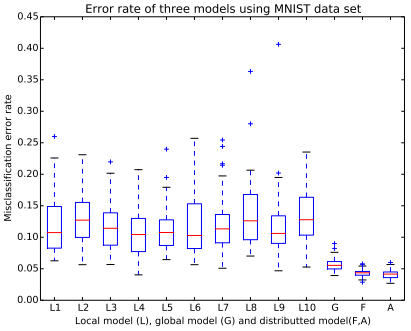


FIGURE 5.39: Comparison among models (private), $\epsilon = 0.9$

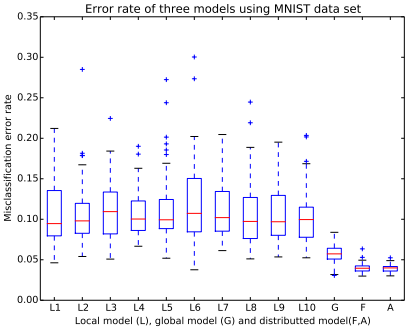


FIGURE 5.40: Comparison among models (private), $\epsilon = 1.0$

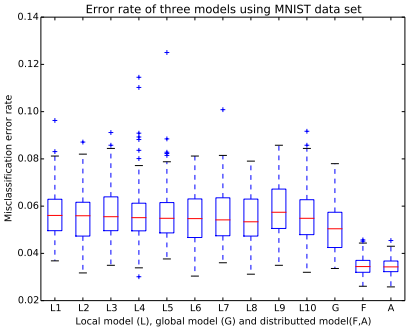


FIGURE 5.41: Comparison among models (non-private)

Chapter 6

Conclusion and Future work

6.1 Conclusion

Previously in Chapter 4 and 5 we discussed the average method and the feature method. We summarize our research results in Table 6.1 to give a brief review.

TABLE 6.1: Feature method vs average

PC \ P	ϵ	m	m_0	N
All non-private	$A > F$	$A(\downarrow) \approx F(\downarrow)$	$A > F(\downarrow)$	$A(\downarrow) \approx F(\downarrow)$ then $A(\downarrow) < F(\text{“}s\text{”})$)
Public-private	$A(\downarrow) > F(\text{“}s\text{”})$	$A(\downarrow) > F(\text{“}s\text{”})$	$A > F(\downarrow)$	$A(\downarrow) > F(\downarrow)$
Fully-private	$A(\downarrow) > F(\text{“}s\text{”})$	$A(\downarrow) > F(\text{“}s\text{”})$	$A > F(\downarrow)$	$A > F$

Where PC is for privacy conditions and P is for parameters, “A” is for the average method and “F” is for the feature method. Error rate has been compared here and we also use arrows and “s” (stable) to indicate the general trends of curves.

Under all non-private case, the feature method has small advantages over the average method. Experiments suggest that when we add more data at aggregation site, the feature method outperforms the average method. We have to be careful when using the feature method when N is large since the performance gets worse. In public-private case, the feature method is more stable and can take advantage of extra data points at the aggregation site to reduce the effect of noise. In fully private case, we find that the curves are the same as in the public-private case. The most relevant scenario for applications is where the privacy constraint is at the local sites. In this case the feature method is superior: it achieves a favorable trade-off between accuracy and privacy risk. So we are very confident to say here that the feature method is better than the traditional average method.

6.2 Future work

In the last section of this thesis, we provide some guidelines for future work based on our investigations. Previously we discussed differentially private ERM and feature aggregation methods for binary classification problems. From the theoretical and experimental results in Chapter 3 to 5 we can see that they perform pretty well and easy to implement. But we also noticed that there are some disadvantages of these two algorithms. In this section we provide possible solutions to these problems.

6.2.1 Improved differential private ERM algorithm using random matrix

Chaudhuri et al. [78] design two delicate differential private ERM algorithms and give original ideas on how to bound probability quotient. Kifer et al. [80] improve this result by introducing extra regularization term, assumptions and different noise distribution. The algorithms in Chapter 3 give us the hint that one way to look at differential private ERM algorithm is to use a noise matrix, which is shown below:

$$J(f, D) = \frac{1}{n}\sum_{i=1}^{n} l(f^T x_i, y_i) + \Lambda f^T M f, \tag{6.1}$$

where M is a $d \times d$ real, p.d. and symmetric random matrix.

We call this method the "matrix perturbation". Here we construct a noise matrix so that it both have function of privacy and regularization. It is easy to see that the diagonal terms of matrix M have the effect of regularization. The difference comparing with algorithms in Chaudhuri et al. [78] and Kifer et al. [80] is that the regularization parameter is random (due to noise) and different in different coordinate. We should pay attention to the noise distribution that the matrix draw from since intuitively, a random matrix introduce more noise than a random vector. For reference of random matrix, Terence Tao's book "Topics in random matrix theory" [108] is worth reading. Also, we can gain some insight into it from Pathak et al. [35], where we have to pay attention to how to draw the noise distribution and the norm of matrix.

A possible way of considering matrix perturbation is simply draw the diagonal of M i.i.d according to some distribution on $\mathbb{R}^{+1}$ and set all off-diagonal components to 0, where we need to make sure that all diagonal components are positive so that matrix is p.d.. In this case we have a regularization term with random weight: $\sum_i \mathrm{diag}_i(M) f_i^2$, which have the effect of both regularization and privacy preservation. So we use one term to meet two requirements, which may make things better. One thing need our attention is

[1] distribution like exponential distribution whose domain is non-negative

that we must find a bijection between the output vector (classifier) f and noise vector (the diagonal of noise matrix here) so that we can transform $\frac{P(f|D)}{P(f|D')}$ and bound it.

6.2.2 Improved feature method

The feature method is a different approach to distributed learning that uses data at the aggregation site to let the data "speak" by learning weights for the local classifiers. We believe there are several interesting future directions.

The first improvement of feature method is to make the best classifier f^*_{span} equal to the global optimal classifier f^*, in other words, $f^* \in span\{f_i\}$. One way to do this is to borrow data from N local sites where $N \gg d$, and pick d best independent local classifiers before conducting the learning procedure in the aggregation site. Although this method guarantee $f^*_{span} = f^*$ and leave the 'useful' classifiers, we need further rigorous proof to show that this 'pick out' procedure indeed bring us more benefit.

The second improvement may be adding more levels of learning. More specifically, we can use the feature method in the following way:

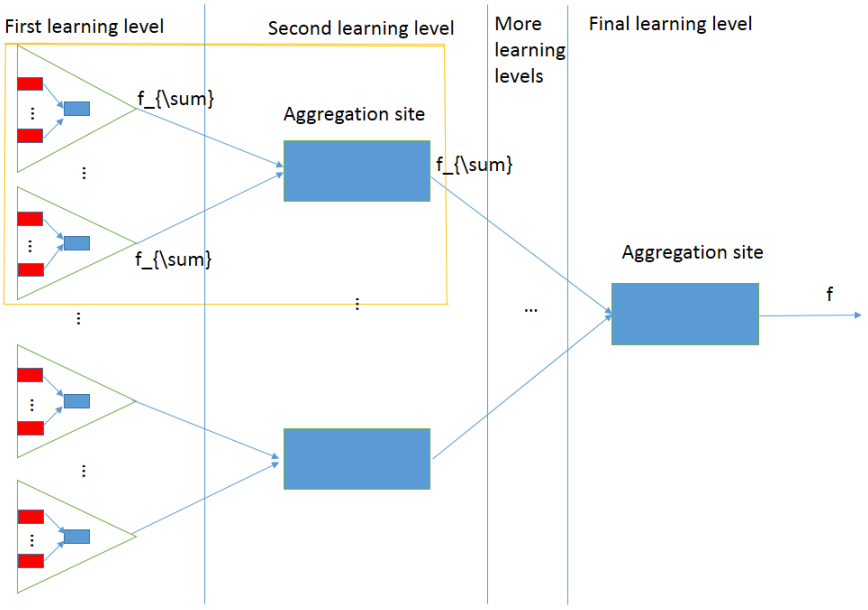


FIGURE 6.1: Multilevel feature method, red rectangular for the local site and blue one for the aggregation site

As we can see from Chapter 4, more levels of learning means faster rate since the number of data points in each learning level acts in a multiplicative way. Take the content in yellow square in Figure 6.1 as an example. In each triangle, we use the original feature method 5 and output the combined classifier $f_{\sum}$[2]. We gathered these outputs, combine with aggregation site in next learning level (the second learning level in Figure 6.1) and

[2]Note that here we output $f_{\sum} \in \mathcal{R}^d$ rather than the weighted parameter $\omega_{min} \in \mathcal{R}^N$

use the feature method to output combined classifier $f_{\sum}$. Then we do the same thing for all triangles and in all levels. In what situation should we use this multilevel method? The answer is that when the public-accessible data in aggregation site is gathered in real-time and is very big. Another situation is when the communication links between levels are bad and the data can't stand too much loss in transmission. Using a multilevel learning framework, all the computations are finished locally (strong local computation ability) at aggregation site so that we can only transmit one classifier (each triangle) instead of whole data set in the aggregation site. This framework may be applied in research in meteorology, oceanography and geology.

The third improvement is to explore the number of local sites N. As we see in Theorem 4.12, large N makes things worse, which is counter to the results in the average method [88]. This can also be seen from experiments in Chapter 5. Intuitively, the feature method prefers fewer good local classifiers over a large number of bad ones. We haven't figured out ways to move N from numerator to denominator in the bound (or at least remove it from numerator). But what we can do is to control the number of sites we borrow data from and do sufficient investigation and research on each data set they provide.

The fourth improvement is to find a better way to deal with the covariance in Section 4.21. This covariance part can be reduced since we didn't give a further analysis of two learning levels and bound it just by basic property of random variable.

The feature method can also be improved with the help of other machine learning methods. Ensemble learning methods such as bootstrap aggregating and boosting improve the overall result by weighting. A comprehensive introduction to boosting is given by Schapire and Freund's book [102]. The feature method may also be analyzed in a stochastic gradient descent way, which can be find in Shamir et al. [86].

There are many other ways to improve the feature method, including some technical tricks. We can communicate between levels of learning to help reducing the constants. We can choose suitable regularization term and loss function to get tighter bound. Notice that in Lemma 4.11 we use the fact that $\|\cdot\| \leq C \rightarrow \mathrm{E}\left[\|\cdot\|\right] \leq C$, where the bound can be improve by some methods. We can add more assumptions about the property of functions. Also, it is useful to find the exact relation between distribution of $\mathcal{P}$ and distribution of f^*_{span}. The only randomness of f^*_{span} comes from $span\{f_i\}$ (see Definition 4.3), whose randomness comes from f_i, which are independently distributed. The most challenging part here is to find the distribution of argmin$(\cdot)$. You may get some inspirations from Habibi and Reza [109]. Then we are able to know the distribution of f_i and hence $\mathrm{E}\left[f^*_{span}\right]$. Further more, the estimation of the data distribution $\mathcal{P}$ is a popular topic in statistic and machine learning.

Finally, if we want to look at the distributed learning procedure in a higher way, a deeper knowledge and understanding of optimization theory, probability theory, modern

analysis and high-dimensional geometry is necessary. Although linear classifier learned from one local site represent the character of data in that site, we can get more features and inner structures of a data set by analyze it geometrically. Also, PCA can be applied to reveal the information that a data set contains. Papers related to distributed PCA can be found in Liang et al. [110] and Qu et al. [111].

Appendix A

PCA and data preprocessing

PCA is a kind of dimension reduction strategy. It uses orthogonal transformation to convert a set of points(vectors) which are possibly linearly correlated with each other to a set of points which are linearly uncorrelated. The dimension of transformed data points are less than or equal to original ones. Intuitively, PCA keep the useful information of data and remove the redundancy. Also, PCA can be used for interpretation of data, finding meaningful structure of data and for illustration purposes [6]. Some good introductions of PCA are in Shlens and Jonathon [112], Smith and Lindsay I [113] and Wold et al. [114]

After we load the data set, the first important step is to prepossess it. This step aims at reducing the running time.

Take the MNIST handwrite digit data as an example. We re-range the data set into a 782×11552 matrix. Here 782 is the dimension of original data (also we called measurement type) and 11552 is the number of data points we have. Our goal is to reduce the matrix to $d \times 11552$, here d is the dimension that contains 90% of the information of date's shape.

$$Original\,data\,matrix = \begin{bmatrix} x_{1\times1} & x_{1\times2} & \cdots & x_{1\times11552} \\ x_{2\times1} & x_{2\times2} & \cdots & x_{2\times11552} \\ \vdots & \vdots & & \vdots \\ x_{784\times1} & x_{784\times2} & \cdots & x_{784\times11552} \end{bmatrix}$$

$\rightarrow$

$$Data\,matrix\,after\,PCA = \begin{bmatrix} y_{1\times1} & y_{1\times2} & \cdots & y_{1\times11552} \\ y_{2\times1} & y_{2\times2} & \cdots & y_{2\times11552} \\ \vdots & \vdots & & \vdots \\ y_{d\times1} & y_{d\times2} & \cdots & y_{d\times11552} \end{bmatrix}$$

Basically we imagine the data are represented as a huge cloud in 3D space (even the number of dimensions can be more than 3), what PCA does is to change the point of

view to find the most significant gradients (shape or characters) of this 'data cloud' and extract them.

At first step, we compute the m-dimensional mean vector where m equals 784 here.

$$mean\,vector = \begin{bmatrix} m_{1\times 1} \\ m_{2\times 1} \\ \vdots \\ m_{784\times 1} \end{bmatrix}$$

where $m_{k\times 1} = \frac{1}{11552}\sum_{i=1}^{11552} x_{k\times i}$

This step is to find the center of cloud.

At second step, subtract every column of the original data matrix by using mean vector. At this step we move the coordinate to the center of the data cloud.

$$centered\,data\,matrix = \begin{bmatrix} x_{1\times 1} & x_{1\times 2} & \cdots & x_{1\times 11552} \\ x_{2\times 1} & x_{2\times 2} & \cdots & x_{2\times 11552} \\ \vdots & \vdots & & \vdots \\ x_{784\times 1} & x_{784\times 2} & \cdots & x_{784\times 11552} \end{bmatrix} - mean\,matrix$$

here the mean matrix is a matrix that has the same shape with original data matrix and every column is a mean vector.

At third step, compute the covariance matrix using centered data matrix. The covariance measures the degree of the linear relationship between two arbitrary points. A large positive value indicates positively correlated data. Likewise, a large negative value denotes negatively correlated data. The absolute magnitude of the covariance measures the degree of redundancy. So the covariance matrix represents the point-wise relation between data points. Let cov_m be the covariance matrix, cov_{cm} be the centered data matrix be the we have: $cov_m = \frac{1}{11552}\, cov_{cm}\, cov_{cm}^T$. The $ij^{th}\, i, j = 1, \cdots, 784$element of cov_m is the dot product between the vector of the i^{th} measurement type and the vector of the j^{th} measurement type.

At fourth step, we compute eigenvectors and corresponding eigenvalues. At the last step, covariance matrix captures the covariance between all possible pairs of measurements. The covariance values reflect the noise and redundancy in our measurements. In the diagonal terms, large values correspond to interesting structure. In the off-diagonal terms large magnitudes correspond to high redundancy. The eigenvectors here are the directions that shows the characters (shape) of data. We denote these eigenvector-eigenvalue pairs as $(vec_i - val_i)$

At fifth step, we sort and choos k eigenvectors with the largest eigenvalues. Since we already got the eigenvalues, we discard least 10

At last step, we map the data onto the new subspace.

Data matrix after PCA = transpose of transform matrix * original data matrix =

$$\begin{bmatrix} y_{1\times1} & y_{1\times2} & \cdots & y_{1\times11552} \\ y_{2\times1} & y_{2\times2} & \cdots & y_{2\times11552} \\ \vdots & \vdots & & \vdots \\ y_{d\times1} & y_{d\times2} & \cdots & y_{d\times11552} \end{bmatrix}$$

The mathematical proof of PCA can be found in Chapter 23 in Shalev-Shwartz, Shai and Ben-David, Shai [6].

Appendix B

Definitions

We introduce some useful definitions and examples in this chapter.

Definition B.1. *(Convex and strictly function)* let $\mathcal{F}$ be a nonempty convex set in a real vector space and let $H(f) : \mathcal{F} \to \mathcal{R}$ be a function. $H(f)$ is convex if $\forall f_1, f_2 \in \mathcal{F}$ and $\forall t \in [0,1]$, we have $H(tf_1 + (1-t)f_2) \leq tH(f_1) + (1-t)H(f_2)$. If for all $f_1 \neq f_2$ and $\forall t \in (0,1)$ we have $H(tf_1 + (1-t)f_2) < tH(f_1) + (1-t)H(f_2)$, then we say the function is strictly convex. Another expression are $f(y) \geq (>) f(x) + \langle \nabla f(x), y - x \rangle$ for all $x, y \in$ domain of $f(\cdot)$ $(x \neq y \in$ domain of $f(\cdot))$.

Strictly convex function is useful because it can have unique global minimum. Suppose there are two distinct local minimum points in the domain of $H(f)$. We denote them as f_1 and f_2. Assuming that $H(f_1) \leq H(f_2)$. Now using the definition of the strictly convex function we have for $\forall t \in (0,1)$, $H(tf_1 + (1-t)f_2) < tH(f_1) + (1-t)H(f_2) \leq tH(f_2) + (1-t)H(f_2) = H(f_2)$. If we take t sufficiently small so that $tf_1 + (1-t)f_2$ is near to f_2. So there exists a neighborhood of f_2 such that for any points x in this neighborhood, $H(x) < H(f_2)$, and hence that H does not have a local minimum at f_2, a contradiction. We conclude that a strictly convex function has at most one local minimum. Using similar idea we can conclude that if $H(f)$ does have a local minimum at f_2 then it is also a global minimum at f_2.
Note that convex function can have no minimum. For example function $f(x) = e^{-x}$.

Examples 1. *Logistic regression function* $f(x) = \log(1 + e^x)$ *is often used in ERM. Using second derivative test we have* $f(x)'' = \frac{1}{(1+e^{-x})(1+e^x)} \geq 0$ *and thus it is a convex function.*

Also, other loss functions like Huber loss and SVM loss are convex.

Definition B.2. *(Strongly convex function 1)* A continuous function f is strongly convex over a convex set S w.r.t a norm $\|\cdot\|$ and with parameter λ if for all $x, y \in S$ and $t \in (0,1)$, $f(tx + (1-t)y) \leq tf(x) + (1-t)f(y) - \frac{1}{2}\lambda t(1-t)\|x - y\|_2^2$.

Definition B.3. *(Strongly convex function 2)* Another useful definition of strongly convex function is: A differentiable function f is strongly convex over a convex set S w.r.t a norm $\|\cdot\|$ and with parameter λ if for all $x, y \in S$, $f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\lambda}{2}\|y - x\|^2$. Or in other way, $\langle \nabla f(x) - \nabla f(y)), (x - y) \rangle \geq \lambda \|y - x\|^2$.

Note that strongly convex function $\subset$ strictly convex function $\subset$ convex function.

Examples 2. *Some times we use negative entropy $\sum_i x_i \log x_i$ as regularization term. Lemma 8 in Shalev-Shwartz and Singer [115] mentioned that negative entropy is 1-strongly convex w.r.t L_1 norm.*

Interestingly, in 1-dimensional case, x^2 is strongly convex but x^4 is not.

Definition B.4. *(Lipschitz continuity)* Given an open set $S \subseteq R^d$. A function $f : \mathcal{R}^d \to \mathcal{R}^k$ is Λ-Lipschitz continuous on the open subset S if there exists a constant $\Lambda \in R_0^+$ (called the Lipschitz constant of f on S) such that $\|f(x) - f(y)\| \leq \Lambda \|x - y\|, \forall x, y \in S$.

Examples 3. *Absolute value function $|x|$ is 1-Lipschitz continuity over $\mathcal{R}$. For $\forall x, y$, we have:*

$$\begin{aligned} |x| - |y| &= |x - y + y| - |y| \\ &\leq |x - y| + |y| - |y| \\ &= 1 \times |x - y|. \end{aligned}$$

Definition B.5. *(β-smooth)* A differentiable function f: $R^d \to R$ is β-smooth if its gradient is β-Lipschitz continuity. That is, for all $x, y \in$ domain of f, we have $\|\nabla f(y) - \nabla f(x)\| \leq \beta \|y - x\|$. Another expression is: $f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\beta}{2}\|y - x\|^2$

Examples 4. *The well known regularizer $\frac{1}{2}\|x\|^2$ is both 1-smooth and 1-strongly convex. $f(w) = \log(1 + e^{-y<w,x>})$ is $\frac{\|x\|^2}{4}$-smooth for $x \in R^d$ and $y = \pm 1$*

Definition B.6. *(Laplace distribution and its property)* A random variable has a Laplace distribution if its probability density function is $f(x|\mu, b) = \frac{1}{2b} e^{-\frac{|x-\mu|}{b}}$, Here, μ is the location parameter and $b \geq 0$, which is the scale parameter. In our application we often set $\mu = 0$

Bibliography

[1] J. H. Friedman, "On bias, variance, 0/1—loss, and the curse-of-dimensionality," *Data mining and knowledge discovery*, vol. 1, no. 1, pp. 55–77, 1997.

[2] P. Domingos and M. Pazzani, "On the optimality of the simple bayesian classifier under zero-one loss," *Machine learning*, vol. 29, no. 2-3, pp. 103–130, 1997.

[3] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using em," *Machine learning*, vol. 39, no. 2-3, pp. 103–134, 2000.

[4] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," *Machine learning*, vol. 36, no. 1-2, pp. 105–139, 1999.

[5] C. M. Bishop *et al.*, *Pattern recognition and machine learning*, vol. 1. springer New York, 2006.

[6] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[7] J.-B. Poline, J. L. Breeze, S. Ghosh, K. Gorgolewski, Y. O. Halchenko, M. Hanke, C. Haselgrove, K. G. Helmer, D. B. Keator, D. S. Marcus, *et al.*, "Data sharing in neuroimaging research," *Frontiers in neuroinformatics*, vol. 6, 2012.

[8] A. D. Sarwate, S. M. Plis, J. A. Turner, M. R. Arbabshirani, and V. D. Calhoun, "Sharing privacy-sensitive access to neuroimaging and genetics data: a review and preliminary validation," *Frontiers in neuroinformatics*, vol. 8, 2014.

[9] S. R. Ganta, S. P. Kasiviswanathan, and A. Smith, "Composition attacks and auxiliary information in data privacy," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 265–273, ACM, 2008.

[10] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pp. 111–125, IEEE, 2008.

[11] C. Dwork, "Differential privacy," in *Automata, languages and programming*, pp. 1–12, Springer, 2006.

[12] C. Dwork, "An ad omnia approach to defining and achieving private data analysis," in *Privacy, Security, and Trust in KDD*, pp. 1–13, Springer, 2008.

[13] S. P. Kasiviswanathan and A. Smith, "On the'semantics' of differential privacy: A bayesian formulation," *Journal of Privacy and Confidentiality*, vol. 6, no. 1, p. 1, 2014.

[14] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2013.

[15] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography*, pp. 265–284, Springer, 2006.

[16] K. Chaudhuri, A. D. Sarwate, and K. Sinha, "A near-optimal algorithm for differentially-private principal components," *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 2905–2943, 2013.

[17] P. Jain and A. Thakurta, "Differentially private learning with kernels," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 118–126, 2013.

[18] A. G. Thakurta and A. Smith, "Differentially private feature selection via stability arguments, and the robustness of the lasso," in *Conference on Learning Theory*, pp. 819–850, 2013.

[19] X. Xiao, G. Wang, and J. Gehrke, "Differential privacy via wavelet transforms," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 23, no. 8, pp. 1200–1214, 2011.

[20] A. Friedman and A. Schuster, "Data mining with differential privacy," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 493–502, ACM, 2010.

[21] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor, "Optimizing linear counting queries under differential privacy," in *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 123–134, ACM, 2010.

[22] C. Dwork, G. N. Rothblum, and S. Vadhan, "Boosting and differential privacy," in *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pp. 51–60, IEEE, 2010.

[23] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, pp. 94–103, IEEE, 2007.

[24] O. Williams and F. McSherry, "Probabilistic inference and differential privacy," in *Advances in Neural Information Processing Systems*, pp. 2451–2459, 2010.

[25] M. Hardt and K. Talwar, "On the geometry of differential privacy," in *Proceedings of the forty-second ACM symposium on Theory of computing*, pp. 705–714, ACM, 2010.

[26] P. Jain and A. G. Thakurta, "(near) dimension independent risk bounds for differentially private learning," in *Proceedings of The 31st International Conference on Machine Learning*, pp. 476–484, 2014.

[27] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum, "Differential privacy under continual observation," in *Proceedings of the forty-second ACM symposium on Theory of computing*, pp. 715–724, ACM, 2010.

[28] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: Differential privacy for location-based systems," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 901–914, ACM, 2013.

[29] G. Jagannathan, K. Pillaipakkamnatt, and R. N. Wright, "A practical differentially private random decision tree classifier," in *Data Mining Workshops, 2009. ICDMW'09. IEEE International Conference on*, pp. 114–121, IEEE, 2009.

[30] M. Hardt, K. Ligett, and F. McSherry, "A simple and practical algorithm for differentially private data release," in *Advances in Neural Information Processing Systems*, pp. 2339–2347, 2012.

[31] Q. Yu and R. Rao, "An improved approach of data integration based on differential privacy," in *Progress in Informatics and Computing (PIC), 2014 International Conference on*, pp. 395–399, IEEE, 2014.

[32] R. Bassily, A. Smith, and A. Thakurta, "Private empirical risk minimization, revisited," *arXiv preprint arXiv:1405.7085*, 2014.

[33] J. Ullman, "Private multiplicative weights beyond linear queries," *arXiv preprint arXiv:1407.1571*, 2014.

[34] F. K. Dankar and K. El Emam, "The application of differential privacy to health data," in *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, pp. 158–166, ACM, 2012.

[35] M. A. Pathak and B. Raj, "Large margin gaussian mixture models with differential privacy," *Dependable and Secure Computing, IEEE Transactions on*, vol. 9, no. 4, pp. 463–469, 2012.

[36] J. Lee and C. Clifton, "How much is enough? choosing ε for differential privacy," in *Information Security*, pp. 325–340, Springer, 2011.

[37] L. Wasserman and S. Zhou, "A statistical framework for differential privacy," *Journal of the American Statistical Association*, vol. 105, no. 489, pp. 375–389, 2010.

[38] A. Wald, "Statistical decision functions.," 1950.

[39] M. Mahdavi, L. Zhang, and R. Jin, "Binary excess risk for smooth convex surrogates," *arXiv preprint arXiv:1402.1792*, 2014.

[40] L. Rosasco, E. Vito, A. Caponnetto, M. Piana, and A. Verri, "Are loss functions all the same?," *Neural Computation*, vol. 16, no. 5, pp. 1063–1076, 2004.

[41] Y. Lin, "A note on margin-based loss functions in classification," *Statistics & probability letters*, vol. 68, no. 1, pp. 73–82, 2004.

[42] V. Vapnik, "Principles of risk minimization for learning theory," in *Advances in neural information processing systems*, pp. 831–838, 1992.

[43] V. Vapnik, *The nature of statistical learning theory*. springer, 2000.

[44] R. Frostig, R. Ge, S. M. Kakade, and A. Sidford, "Competing with the empirical risk minimizer in a single pass," *arXiv preprint arXiv:1412.6606*, 2014.

[45] P. L. Bartlett and S. Mendelson, "Empirical minimization," *Probability Theory and Related Fields*, vol. 135, no. 3, pp. 311–334, 2006.

[46] F. Pérez-Cruz, A. Navia-Vázquez, A. R. Figueiras-Vidal, and A. Artes-Rodriguez, "Empirical risk minimization for support vector classifiers," *Neural Networks, IEEE Transactions on*, vol. 14, no. 2, pp. 296–303, 2003.

[47] K. Talwar, A. Thakurta, and L. Zhang, "Private empirical risk minimization beyond the worst case: The effect of the constraint set geometry," *arXiv preprint arXiv:1411.5417*, 2014.

[48] G. Lecué, "Suboptimality of penalized empirical risk minimization in classification," in *Learning theory*, pp. 142–156, Springer, 2007.

[49] V. Koltchinskii *et al.*, "Sparsity in penalized empirical risk minimization," *Ann. Inst. Henri Poincaré Probab. Stat*, vol. 45, no. 1, pp. 7–57, 2009.

[50] L. Devroye, T. Linder, and G. Lugosi, "Nonparametric estimation and classification using radial basis function nets and empirical risk minimization," *Neural Networks, IEEE Transactions on*, vol. 7, no. 2, pp. 475–487, 1996.

[51] S. Mukherjee, P. Niyogi, T. Poggio, and R. Rifkin, "Learning theory: stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization," *Advances in Computational Mathematics*, vol. 25, no. 1-3, pp. 161–193, 2006.

[52] P. J. Huber *et al.*, "Robust estimation of a location parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964.

[53] O. Bousquet and L. Bottou, "The tradeoffs of large scale learning," in *Advances in neural information processing systems*, pp. 161–168, 2008.

[54] S. Shalev-Shwartz and N. Srebro, "Svm optimization: inverse dependence on training set size," in *Proceedings of the 25th international conference on Machine learning*, pp. 928–935, ACM, 2008.

[55] Y. Lee, "Stat 881 advanced statistical learning." http://www.stat.osu.edu/~yklee/881/index.html.

[56] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[57] H. Frezza, "Support vector machines tutorial," 2013.

[58] P.-H. Chen, C.-J. Lin, and B. Schölkopf, "A tutorial on ν-support vector machines," *Applied Stochastic Models in Business and Industry*, vol. 21, no. 2, pp. 111–136, 2005.

[59] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[60] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *The Journal of Machine Learning Research*, vol. 2, pp. 45–66, 2002.

[61] G. M. Fung and O. L. Mangasarian, "Multicategory proximal support vector machine classifiers," *Machine Learning*, vol. 59, no. 1-2, pp. 77–97, 2005.

[62] S.-i. Amari and S. Wu, "Improving support vector machine classifiers by modifying kernel functions," *Neural Networks*, vol. 12, no. 6, pp. 783–789, 1999.

[63] Y.-J. Lee and O. L. Mangasarian, "Ssvm: A smooth support vector machine for classification," *Computational optimization and Applications*, vol. 20, no. 1, pp. 5–22, 2001.

[64] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: Primal estimated sub-gradient solver for svm," *Mathematical programming*, vol. 127, no. 1, pp. 3–30, 2011.

[65] T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler, "Support vector machine classification and validation of cancer tissue samples using microarray expression data," *Bioinformatics*, vol. 16, no. 10, pp. 906–914, 2000.

[66] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine learning*, vol. 46, no. 1-3, pp. 389–422, 2002.

[67] E. Byvatov and G. Schneider, "Support vector machine applications in bioinformatics.," *Applied bioinformatics*, vol. 2, no. 2, pp. 67–77, 2002.

[68] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2009.

[69] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, and R. Tibshirani, *The elements of statistical learning*, vol. 2. Springer, 2009.

[70] H. T. Jongen, K. Meer, and E. Triesch, *Optimization theory*. Springer Science & Business Media, 2004.

[71] A. Antoniou and W.-S. Lu, *Practical optimization: algorithms and engineering applications*. Springer Science & Business Media, 2007.

[72] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The computer journal*, vol. 7, no. 4, pp. 308–313, 1965.

[73] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the nelder–mead simplex method in low dimensions," *SIAM Journal on optimization*, vol. 9, no. 1, pp. 112–147, 1998.

[74] M. S. Alvim, M. E. Andrés, K. Chatzikokolakis, P. Degano, and C. Palamidessi, "Differential privacy: on the trade-off between utility and information leakage," in *Formal Aspects of Security and Trust*, pp. 39–54, Springer, 2012.

[75] A. D. Sarwate and K. Chaudhuri, "Signal processing and machine learning with differential privacy: Algorithms and challenges for continuous data," *Signal Processing Magazine, IEEE*, vol. 30, no. 5, pp. 86–94, 2013.

[76] S. M. Kakade, K. Sridharan, and A. Tewari, "On the complexity of linear prediction: Risk bounds, margin bounds, and regularization," in *Advances in neural information processing systems*, pp. 793–800, 2009.

[77] K. Chaudhuri and C. Monteleoni, "Privacy-preserving logistic regression," in *Advances in Neural Information Processing Systems*, pp. 289–296, 2009.

[78] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *The Journal of Machine Learning Research*, vol. 12, pp. 1069–1109, 2011.

[79] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby, "Selective sampling using the query by committee algorithm," *Machine learning*, vol. 28, no. 2-3, pp. 133–168, 1997.

[80] D. Kifer, A. Smith, and A. Thakurta, "Private convex empirical risk minimization and high-dimensional regression," *Journal of Machine Learning Research*, vol. 1, p. 41.

[81] R. Bekkerman, M. Bilenko, and J. Langford, *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press, 2011.

[82] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[83] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pp. 1–10, IEEE, 2010.

[84] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: cluster computing with working sets," in *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, pp. 10–10, 2010.

[85] V. K. Potluru, J. Diaz-Montes, A. D. Sarwate, S. M. Plis, V. D. Calhoun, B. A. Pearlmutter, and M. Parashar, "Cometcloudcare (c 3): Distributed machine learning platform-as-a-service with privacy preservation,"

[86] O. Shamir, N. Srebro, and T. Zhang, "Communication efficient distributed optimization using an approximate newton-type method," *arXiv preprint arXiv:1312.7853*, 2013.

[87] Y. Zhang and L. Xiao, "Communication-efficient distributed optimization of self-concordant empirical loss," *arXiv preprint arXiv:1501.00263*, 2015.

[88] Y. Zhang, M. J. Wainwright, and J. C. Duchi, "Communication-efficient algorithms for statistical optimization," in *Advances in Neural Information Processing Systems*, pp. 1502–1510, 2012.

[89] R. Mcdonald, M. Mohri, N. Silberman, D. Walker, and G. S. Mann, "Efficient large-scale distributed training of conditional maximum entropy models," in *Advances in Neural Information Processing Systems*, pp. 1231–1239, 2009.

[90] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, "Parallelized stochastic gradient descent," in *Advances in Neural Information Processing Systems*, pp. 2595–2603, 2010.

[91] S. Han, U. Topcu, and G. J. Pappas, "Differentially private distributed constrained optimization," *arXiv preprint arXiv:1411.4105*, 2014.

[92] J. C. Duchi, M. I. Jordan, M. J. Wainwright, and Y. Zhang, "Information-theoretic lower bounds for distributed statistical estimation with communication constraints," *arXiv preprint arXiv:1405.0782*, 2014.

[93] Y. Zhang, J. Duchi, M. I. Jordan, and M. J. Wainwright, "Information-theoretic lower bounds for distributed statistical estimation with communication constraints," in *Advances in Neural Information Processing Systems*, pp. 2328–2336, 2013.

[94] Z. Huang, S. Mitra, and N. Vaidya, "Differentially private distributed optimization," *arXiv preprint arXiv:1401.2596*, 2014.

[95] Z. Ji, X. Jiang, S. Wang, L. Xiong, and L. Ohno-Machado, "Differentially private distributed logistic regression using private and public data," *BMC medical genomics*, vol. 7, no. Suppl 1, p. S14, 2014.

[96] "Distributed stochastic optimization and learning,"

[97] Y. Zhang, J. C. Duchi, and M. J. Wainwright, "Divide and conquer kernel ridge regression: A distributed algorithm with minimax optimal rates," *arXiv preprint arXiv:1305.5029*, 2013.

[98] J. Rosenblatt and B. Nadler, "On the optimality of averaging in distributed statistical learning," *arXiv preprint arXiv:1407.2724*, 2014.

[99] K. I. Tsianos, A. D. Sarwate, and M. G. Rabbat, "Tradeoffs for task parallelization in distributed optimization," in *Machine Learning for Signal Processing (MLSP), 2014 IEEE International Workshop on*, pp. 1–6, IEEE, 2014.

[100] M. Seeger, J. Langford, and N. Megiddo, "An improved predictive accuracy bound for averaging classifiers," in *Proceedings of the 18th International Conference on Machine Learning*, no. EPFL-CONF-161321, pp. 290–297, 2001.

[101] R. Meir and G. Rätsch, "An introduction to boosting and leveraging," in *Advanced lectures on machine learning*, pp. 118–183, Springer, 2003.

[102] R. E. Schapire and Y. Freund, *Boosting: Foundations and algorithms*. MIT press, 2012.

[103] K. Nissim, S. Raskhodnikova, and A. Smith, "Smooth sensitivity and sampling in private data analysis," in *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pp. 75–84, ACM, 2007.

[104] L. Rokach, "Ensemble-based classifiers," *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 1–39, 2010.

[105] A. De Acosta, "Inequalities for b-valued random vectors with applications to the strong law of large numbers," *The Annals of Probability*, pp. 157–161, 1981.

[106] K. Sridharan, S. Shalev-Shwartz, and N. Srebro, "Fast rates for regularized objectives," in *Advances in Neural Information Processing Systems*, pp. 1545–1552, 2009.

[107] S. Shalev-Shwartz, O. Shamir, K. Sridharan, and N. Srebro, "Stochastic convex optimization," 2009.

[108] T. Tao, *Topics in random matrix theory*, vol. 132. American Mathematical Soc., 2012.

[109] R. Habibi, "Exact distribution of argmax (argmin)," *Economic Quality Control*, vol. 26, no. 2, pp. 155–162, 2011.

[110] Y. Liang, M.-F. Balcan, and V. Kanchanapally, "Distributed pca and k-means clustering," in *The Big Learning Workshop at NIPS*, 2013.

[111] Y. Qu, G. Ostrouchov, N. Samatova, and A. Geist, "Principal component analysis for dimension reduction in massive distributed data sets," in *Proceedings of IEEE International Conference on Data Mining (ICDM)*, 2002.

[112] J. Shlens, "A tutorial on principal component analysis," *arXiv preprint arXiv:1404.1100*, 2014.

[113] L. I. Smith, "A tutorial on principal components analysis," *Cornell University, USA*, vol. 51, p. 52, 2002.

[114] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1, pp. 37–52, 1987.

[115] S. Shalev-Shwartz and Y. Singer, "Logarithmic regret algorithms for strongly convex repeated games," *The Hebrew University*, 2007.