Politecnico di Torino

DIPARTIMENTO DI SCIENZE MATEMATICHE
M.Sc. in Engineering Mathematics

Final dissertation

Robust Risk Minimization for Noisy Labels

Supervisor: Santiago Mazuelas
Co-supervisor: Giuseppe Calafiore

Candidate: Filippo Buoncompagni

ACADEMIC YEAR 2019/2020

Summary

More often than not the learner has access to a corrupted dataset and still he has to find a way to get a good predictive model out of it. This means, in case of classification problems, to be able to design a *learning algorithm* whose aim is to output from corrupted data a *classification rule* with small *generalization error*, or *risk*, on new clean samples. The learning algorithm used throughout this work is what is called *Robust Risk Minimization* (RRM) which consists in minimizing the worst case risk incurred over an *uncertainty set* of probability distributions. The most popular learning algorithm *Empirical Risk Minimization* (ERM) can be seen as a particular case of RRM when the uncertainty set contains only the empirical distribution. In this work we will perform RRM by considering the $0-1$ loss, the set of non deterministic classification rules as *Hypothesis Space*, and uncertainty sets defined by linear constraints on the expected value of a function, called the *feature map*. We will show that, with such approach, we can derive a classification rule by solving a Linear Programming problem. By suitably modifying the feature map, we will make the classifier robust against one particular type of corruption which consists in having class conditional noise in the labels. We will see that our approach extends the current literature since it is able to deal with non reconstructible noise, while existing techniques cannot. Lastly, we will present some *generalization bounds* for the risk of our classifier learned from datasets with noisy labels.

All the work is supported by numerical results showing that the performances of our classifier are competitive with the performances of the current state of the art techniques for dealing with noisy labels.

Acknowledgements

The research presented in this master thesis has been done at "BCAM, Basque Center for Applied Mathematics" in Bilbao, Spain. I have been there for an internship of four months: June, July, August and September 2019, under the supervision of Dr. Santiago Mazuelas and his group.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The recent scientific field known as *machine learning* consists in developing computational methods to make accurate predictions from past available information which usually consists in collected data. Nowadays, the interest of the corporate world towards machine learning has increased enormously. In fact machine learning techniques are a valuable support tool in making important decisions that guide the growth process of a company. Companies posses large amount of data and they want to find a way to extract useful information in order to improve efficiency and reduce costs. In parallel, also among academics the interest toward a relatively new field like machine learning has increased in recent years, and almost every day we see the birth of new and more efficient techniques. This has led to a deep and solid collaboration between university and industry as we have never seen before.

Each machine learning technique was born to solve a particular class of learning problem. Some major class of learning problems are:

- *Classification*: we have a set of items and we want to assign a categorical value to each of them. For categorical value we may think as a label, which can be a color or a number, that identifies a category. An example of classification problem is the one that faces a bank when it has to decide whether to give or not a loan to a new client. The bank usually has an history of all the past clients to whom it had lent money. This history may contain for every past client some of his features like age, occupation, salary, education, etc, along with a label that indicates if the clients has repaid the loan or not. The bank based on the features of the new client and on past labeled data will output a decision yes/no at the request of credit from the client.
- *Regression*: given a set of items we want to predict a real value for each of them. An example of a regression problem is the one of predicting the stock values.
- *Clustering*: we have a set of items and we want to group them into homogeneous regions according to some measure of similarity.

While the first two major class of problems, classification and regression, are called *supervised learning* problems because the learner receives a set of labeled examples as training data and makes predictions for all unseen points, the clustering problem is referred to *unsupervised learning* because the learner exclusively receives unlabeled training data [Hastie et al. (2005)]. Sometimes it might be the case that unlabeled data is easily accessible but labels are expensive to obtain. In this last case the learner usually has very few labeled examples and many unlabeled examples. The hope is that the distribution of unlabeled data accessible to the learner can help him achieve a better performance than in a supervised setting where he has only few labeled data. We refer to the last scenario as *semi-supervised learning*[Zhu & Goldberg (2009)].

In this work we will deal mainly with supervised learning, with a brief section regarding semi-supervised learning, and in particular with classification problems.

The problem we want to address with this work is how to deal with situations in which the data at our disposal is affected by some noise, in particular how to guarantee an efficient learning that is able to still make good predictions. Dealing with corrupted dataset is more a rule than an exception. In this work we will concentrate on the very specific situation in which the noise in the dataset affects only the labels. This problem in the machine learning community is known as *learning with noisy labels*. Creating supervised learning algorithms that are able to learn efficiently from data with noisy labels is a problem of great importance. [Angluin & Laird (1988)] proved that the random classification noise, in which every label is flipped independently with a probability $\rho \in [0, 0.5)$, is PAC-learnable. [Kearns (1998)] proposed the statistical query model to learn with random classification noise. [Natarajan et al. (2013)] studied the problem of binary classification in the presence of asymmetric random classification noise, by using Empirical risk minimization with a suitably modified loss function. [Van Rooyen & Williamson (2017)] presented a generic method for learning from a fixed, known and reconstructible corruption in the labels that generalizes the method proposed in [Natarajan et al. (2013)]. We will show that our method is able to deal with non reconstructible corruptions since it does not require to invert the transition matrix. [Liu & Tao (2015)] instead, proved that any surrogate loss function can be used for classification with noisy labels by using importance reweighting.

The machine learning technique used throughout all the work is what is called *Linear Probabilistic Classifier*(LPC) which is still under development, whose details can be found in [Mazuelas et al. (2019)] and a brief summary in Appendix B.

In all the following simulations I've used CVX, a package for specifying and solving convex programs [Grant & Boyd (2014), Grant & Boyd (2008)].

In this first introductory chapter we present briefly what is the goal of supervised

learning and the most used approach to deal with, which is *Empirical Risk Minimization*. Then we will introduce how can we model corruptions of data.

1.1 Framework and Problem Description

Suppose we have access to a set of data points $\{x_1, \dots, x_n\}$ living in a space called the *feature space* $\mathcal{X}$, each of which is a realization of a random variable x, following a fixed but unknown distribution $p(x)$. Every data point also carries its own label according to a fixed but unknown conditional distribution $p(y|x)$. From now on, we will call the set of data points we have access to, together with their labels, $(x_1, y_1), \dots, (x_n, y_n)$, the *training data*. The training data is a set of i.i.d. samples drawn from the joint distribution $p(x, y) = p(y|x)p(x)$, that we will call p.

Let $h : \mathcal{X} \to \mathcal{Y}$ be a function from the feature space $\mathcal{X}$ to the label space $\mathcal{Y}$. The function h is called a classification rule. Moreover, let $l : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ be a loss function which measures the loss $l(h, (x, y))$ incurred when we classify a given sample (x, y) with label $h(x)$. The natural loss function used in classification is the $0 - 1$ loss, defined as $l(h, (x, y)) = \mathbb{1}[h(x) \neq y]$. The goal of supervised learning is to find a classification rule h that minimizes the expected loss, called the *risk* or *generalization error*, over unseen data distributed according to p [Vapnik (1999)]. That is we want to solve

$$\min_{h:\mathcal{X}\to\mathcal{Y}} \mathbb{E}_p l(h, (x, y)). \tag{1.1}$$

Problem (1.1) can be solved in theory by writing

$$\min_{h:\mathcal{X}\to\mathcal{Y}} \sum_{x\in\mathcal{X}, y\in\mathcal{Y}} p(x, y)(1 - \mathbb{1}[h(x) = y]) = \tag{1.2}$$

$$\min_{h:\mathcal{X}\to\mathcal{Y}} 1 - \sum_{x\in\mathcal{X}} p(x, h(x)) = \tag{1.3}$$

$$1 - \max_{h:\mathcal{X}\to\mathcal{Y}} \sum_{x\in\mathcal{X}} p(x, h(x)). \tag{1.4}$$

Maximizing the sum in (1.4) is the same as maximizing each term of the sum, and so the best classification rule h is such that, $\forall x \in \mathcal{X}$

$$h(x) \in \operatorname*{argmax}_{y\in\mathcal{Y}} p(x, y), \tag{1.5}$$

and is called the Bayes rule, while

$$R^* = 1 - \sum_{x\in\mathcal{X}} \max_{y\in\mathcal{Y}} p(x, y), \tag{1.6}$$

is called the Bayes Risk and is the smallest possible risk we can achieve. Since however the true distribution p is not known, solving the optimization problem

above in practice is not possible. In order to simplify the notation let's indicate the risk of a given classifier h as $R(h) \doteq \mathbb{E}_p l(h,(x,y))$ and its empirical risk as

$$\hat{R}(h) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[h(x_i) \neq y_i]. \tag{1.7}$$

Usually instead of considering all the possible functions $h : \mathcal{X} \to \mathcal{Y}$, we fix a hypothesis set $\mathcal{H}$ and do the minimization over $\mathcal{H}$. Notice that, for a fixed $h \in \mathcal{H}$, the empirical risk $\hat{R}(h)$ is a random variable since it depends on the data $(x_1,y_1),\ldots,(x_n,y_n)$ which are i.i.d. samples from p. Taking the expectation of the empirical error over all the possibles training data we get the generalization error $R(h)$, that is

$$\mathbb{E}[\hat{R}(h)] = R(h). \tag{1.8}$$

It is possible to bound the difference between $R(h)$ and $\hat{R}(h)$ in many ways with the so called Generalization Bounds (see Appendix A). The idea of that is to find some upper bounds for the generalization error $R(h)$ of a given classifier $h \in \mathcal{H}$.

Empirical Risk Minimization (ERM) is by far the most used approach for approximating (1.1). The idea is to use the training data and it consists in minimizing the sample average of the loss incurred at the observed samples

$$\min_{h\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n} l(h,(x_i,y_i)), \tag{1.9}$$

which can be written in terms of the empirical distribution p_e as

$$\min_{h\in\mathcal{H}} \mathbb{E}_{p_e} l(h,(x,y)). \tag{1.10}$$

From a practical point of view, problem (1.9) is tractable only if the objective function is convex, which is not the case for the $0-1$ loss. What is usually done is to replace the $0-1$ loss with a convex loss function such as the logistic loss, the Hinge loss or the quadratic loss.

Usually in supervised learning it is assumed that training samples follow the same distribution as test samples. However this situation is rarely seen in reality. In many real situations the datasets we have access to might have been previously corrupted deliberately or unintentionally. Suppose, for example, that an adversary corrupted the labels of our training dataset, and we have access to $(x_1,\tilde{y}_1)\ldots(x_n,\tilde{y}_n)$ which are i.i.d. samples from a distribution $\tilde{p} = p(\tilde{y}|x)p(x)$, while $(x_1,y_1)\ldots(x_n,y_n)$ are i.i.d. test samples from the clean distribution $p = p(y|x)p(x)$. We model the above situation by saying that originally the training data was following p, but then a corruption occurred and now it follows $\tilde{p}$. More formally a corruption in the label can be modeled as a linear map $T : \Delta(\mathcal{X}\times\mathcal{Y}) \to \Delta(\mathcal{X}\times\tilde{\mathcal{Y}})$, where $\Delta(\mathcal{X}\times\mathcal{Y})$ is the simplex of probability distributions in $\mathbb{R}^{|\mathcal{X}||\mathcal{Y}|}$, that transforms a probability

distribution p into a probability distribution $\tilde{p}$ [Mazuelas & Pérez (2019)]. Our goal now is to learn a model from those corrupted samples that performs well on new clean samples, limiting as much as possible the damage caused by the adversary.

As a starting point let's consider the case of binary classification where $y \in \{-1, +1\}$. Let's define ρ_+ the probability of flipping the true label $y = +1$ to $\tilde{y} = -1$, and ρ_- the probability of flipping the true label $y = -1$ to $\tilde{y} = +1$. In formulas

$$\rho_+ = \mathbb{P}(\tilde{y} = -1 | y = +1) \tag{1.11}$$
$$\rho_- = \mathbb{P}(\tilde{y} = +1 | y = -1). \tag{1.12}$$

Usually ρ_+ and ρ_- are called the noise rates, depend on the class label, and a very important point is that we assume to know those rates. In practice we can see the corruption process of a data as going through every training instance-label pair and flipping a biased coin according to the conditional probability ρ_+ and ρ_-. In theory in the case of binary classification the linear map T can be expressed as a stochastic matrix $\mathbb{R}^{|\mathcal{X}||\tilde{\mathcal{Y}}| \times |\mathcal{X}||\mathcal{Y}|}$ whose columns add up to one: $T = \mathbb{I} \otimes T_y$, where

$$T_y = \begin{bmatrix} 1 - \rho_+ & \rho_- \\ \rho_+ & 1 - \rho_- \end{bmatrix}, \tag{1.13}$$

and with the compact notation $T = \mathbb{I} \otimes T_y$, we mean

$$T = \begin{bmatrix} T_y & 0 & 0 & \dots & 0 \\ 0 & T_y & 0 & \dots & 0 \\ 0 & 0 & T_y & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & T_y \end{bmatrix}. \tag{1.14}$$

Let's see what is the effect of the linear map T on a generic probability distribution p. Notice that p is a $|\mathcal{X}||\mathcal{Y}|$-dimensional vector, where $|\mathcal{X}|$ denotes all the possible instances of the features

$$p = \begin{bmatrix} p(x_1, +1) \\ p(x_1, -1) \\ \vdots \\ p(x_{|\mathcal{X}|}, +1) \\ p(x_{|\mathcal{X}|}, -1) \end{bmatrix} \in \mathbb{R}^{|\mathcal{X}||\mathcal{Y}|}, \tag{1.15}$$

and T is a $|\mathcal{X}||\tilde{\mathcal{Y}}| \times |\mathcal{X}||\mathcal{Y}|$ dimensional matrix such that

$$\begin{bmatrix} T_y & 0 & 0 & \dots & 0 \\ 0 & T_y & 0 & \dots & 0 \\ 0 & 0 & T_y & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & T_y \end{bmatrix} \begin{bmatrix} p(x_1, y = +1) \\ p(x_1, y = -1) \\ \vdots \\ p(x_{|\mathcal{X}|}, y = +1) \\ p(x_{|\mathcal{X}|}, y = -1) \end{bmatrix} = \begin{bmatrix} \tilde{p}(x_1, \tilde{y} = +1) \\ \tilde{p}(x_1, \tilde{y} = -1) \\ \vdots \\ \tilde{p}(x_{|\mathcal{X}|}, \tilde{y} = +1) \\ \tilde{p}(x_{|\mathcal{X}|}, \tilde{y} = -1) \end{bmatrix}, \tag{1.16}$$

or in a compact way

$$Tp = \tilde{p}. \tag{1.17}$$

In particular we have $\forall i = 1, \ldots, |\mathcal{X}|$:

$$\tilde{p}(x_i, \tilde{y} = +1) = (1 - \rho_+)p(x_i, y = +1) + (\rho_-)p(x_i, y = -1), \tag{1.18}$$
$$\tilde{p}(x_i, \tilde{y} = -1) = (\rho_+)p(x_i, y = +1) + (1 - \rho_-)p(x_i, y = -1). \tag{1.19}$$

Up to now we have consider the situation in which the original dataset gets corrupted homogeneously. However more complicated corruptions can happen. For instance we could have different corruptions for different subsets of the original dataset. That is

$$\begin{gathered}(x_{11}, \tilde{y}_{11}) \ldots (x_{1n_1}, \tilde{y}_{1n_1}) \text{ i.i.d. from } \tilde{p}_1 \\ (x_{21}, \tilde{y}_{21}) \ldots (x_{2n_2}, \tilde{y}_{2n_2}) \text{ i.i.d. from } \tilde{p}_2 \\ \vdots \\ (x_{m1}, \tilde{y}_{m1}) \ldots (x_{mn_m}, \tilde{y}_{mn_m}) \text{ i.i.d. from } \tilde{p}_m,\end{gathered}$$

which is called *Heterogeneous* noisy labels scenario [Mazuelas & Pérez (2019)] and will be discussed later.

Chapter 2

Noisy Labels using Linear Probabilistic Classifiers

In this chapter we will introduce the concept of Linear Probabilistic Classifier [Mazuelas et al. (2019)] which is a classifier derived by using Robust Risk Minimization, a different approach for approximating problem (1.1). We will also see the correction that has to be applied on a Linear Probabilistic Classifier in order to make it robust against noisy labels, along some simulations on synthetic datasets whose results confirm the validity of our approach. Moreover we will present a generalization bound for the risk of a LPC learned using the correction. Finally we will briefly discuss a semi-supervised learning setting and multi classification.

2.1 Linear Probabilistic Classifier

Rather than using Empirical Risk Minimization, a different way for approximating problem (1.1) is to implement what is called *Robust Risk Minimization*. Since we don't know p in (1.1), the idea is to derive from the training data $(x_1, y_1), \ldots, (x_n, y_n)$ an uncertainty set $\mathcal{U}$ made of probability distributions, that contains the true distribution p with high probability. Then we find a classification rule h that minimizes the worst case risk incurred over the uncertainty set $\mathcal{U}$ of probability distributions [Farnia & Tse (2016), Lanckriet et al. (2002)]. We can express this idea with the following optimization problem

$$\min_{h \in \mathcal{H}} \max_{p \in \mathcal{U}} \mathbb{E}_p l(h, (x, y)). \tag{2.1}$$

The *Linear Probabilistic Classifier* (LPC) presented in [Mazuelas et al. (2019)] is a classifier built by solving (2.1), where the loss considered is the $0-1$ loss, the classification rule h is unconstrained, and the uncertainty set $\mathcal{U}$ is defined in terms of inequalities that constrain the expected value of a function Φ called the feature map.

In the LPC framework we define the uncertainty set $\mathcal{U}$ as a subset in $\Delta(\mathcal{X}\times\mathcal{Y}) \subset \mathbb{R}^{|\mathcal{X}||\mathcal{Y}|}$ in the following way [1]

$$\mathcal{U}_{\Phi}^{a,b} = \{p \in \Delta(\mathcal{X}\times\mathcal{Y}) : a \preceq \mathbb{E}_p\{\Phi(x,y)\} \preceq b\}\,, \tag{2.2}$$

where $\Phi : \mathcal{X}\times\mathcal{Y} \to \mathbb{R}^m$ is a vector function, called the feature map, and $a, b \in \mathbb{R}^m$ are given vectors such that $a \preceq b$. In words, the uncertainty set $\mathcal{U}_{\Phi}^{a,b}$ contains probability distributions such that the mean of Φ with respect to those distributions is between a and b. Suppose now that $(x_1, y_1), \ldots, (x_n, y_n)$ are i.i.d. samples from p, we would like to solve problem (2.1) over an uncertainty set $\mathcal{U}_{\Phi}^{a,b}$ that contains p with high probability, but at the same time ensuring that all the probability distributions in $\mathcal{U}_{\Phi}^{a,b}$ are close to p. This can be achieved through the vectors a and b as the following. First of all, let's define

$$\tau \doteq \mathbb{E}_p\{\Phi(x,y)\} = \mathbf{\Phi} p, \tag{2.3}$$

where the equality is just the definition of the expectation of Φ with respect to p. From (2.3) we see that τ is a convex linear combination of the columns of the matrix $\mathbf{\Phi} \in \mathbb{R}^{m\times|\mathcal{X}||\mathcal{Y}|}$, i.e., $\tau \in \mathrm{Conv}(\Phi) \subset \mathbb{R}^m$. Then let's define τ_n as an estimate for τ

$$\tau_n = \frac{1}{n}\sum_{i=1}^{n} \mathbf{\Phi}(x_i, y_i), \tag{2.4}$$

and by the Law of Large Numbers, we know that as $n \to \infty$,

$$\tau_n \to \tau.$$

We can build the component wise confidence interval for τ by recalling that, as $n \to \infty$, Central Limit Theorem states

$$\frac{(\tau_n)_j - (\tau)_j}{\sqrt{\sigma_j^2/n}} \to^d \mathcal{N}(0,1),\ j = 1 \ldots m,$$

where σ_j^2 is the variance of Φ_j. So, the $1 - \alpha/m$ confidence interval for τ_j will be:

$$(\tau_n)_j \pm z_{1-\alpha/2m}\sqrt{\sigma_j^2/n}.$$

If we put

$$(a_n)_j = (\tau_n)_j - z_{1-\alpha/2m}\sqrt{\sigma_j^2/n},\ j = 1 \ldots m \tag{2.5}$$

$$(b_n)_j = (\tau_n)_j + z_{1-\alpha/2m}\sqrt{\sigma_j^2/n},\ j = 1 \ldots m, \tag{2.6}$$

and we use in (2.2) $a = a_n$ and $b = b_n$, we obtain an uncertainty set that contains p with probability $(1 - \alpha)$.[2]

[1] When we write p in text it is referred to the true distribution, while in the definition of the uncertainty set $\mathcal{U}_{\Phi}^{a,b}$, p are all the probability distributions belonging to that uncertainty set.

[2] Since we have $\mathbb{P}\{A_j\} = 1 - \alpha/m,\ \forall j = 1, \ldots, m$, by using the Union Bound we get $\mathbb{P}\{\cap_j A_j\} \geq \sum_j^m \mathbb{P}\{A_j\} + 1 - m = 1 - \alpha$, where A_j is the event $|\tau_j - (\tau_n)_j| \leq z_{1-\alpha/2m}\sqrt{\sigma_j^2/n}$.

2.1.1 From the noisy domain to the clean domain

We have seen that uncertainty sets are defined by linear inequalities

$$\mathcal{U}_{\Phi}^{a,b} = \{p \in \Delta(\mathcal{X} \times \mathcal{Y}) : a \preceq \mathbb{E}_p\{\Phi(x,y)\} \preceq b\}\,, \tag{2.7}$$

and we have also shown how to build an uncertainty set that contains the true distribution p with tunable confidence from the training data. We could in principle repeat exact the same reasoning as if we were in a noisy domain, that is if we had training samples with noisy labels $(x_1, \tilde{y}_1), \dots (x_n, \tilde{y}_n)$ following $\tilde{p}$, instead of having training samples $(x_1, y_1) \dots (x_n, y_n)$ following p. Repeating the same reasoning would mean to build uncertainty sets in the noisy domain, solve the optimization problem (2.1) and, once we have the LPC, testing on a unseen dataset which is in the clean domain. Doing that, however does not take into account any correction for the noise. In fact, the learning algorithm would learn from a noisy dataset and consequently performing poorly on a clean unseen one.

The idea is to build uncertainty sets that contain the clean probability distribution p, from training data that follow a different distribution $\tilde{p}$. Let $\tilde{\Phi} : \mathcal{X} \times \tilde{\mathcal{Y}} \to \mathbb{R}^m$ be the fixed feature map in the noisy domain. Then we can write, in the same way as (B.2), the uncertainty set in the noisy domain

$$\mathcal{U}_{\tilde{\Phi}}^{a,b} = \left\{\tilde{p} \in \Delta(\mathcal{X} \times \tilde{\mathcal{Y}}) : a \preceq \mathbb{E}_{\tilde{p}}\{\tilde{\Phi}(x,\tilde{y})\} \preceq b\right\}, \tag{2.8}$$

where the true expectation in the noisy domain is with respect to the true noisy distribution $\tilde{p}$ which is the output of a corruption transition T on p, the true probability distribution in the clean domain. That is

$$\mathbb{E}_{\tilde{p}}\{\tilde{\Phi}(x,\tilde{y})\} = \tilde{\mathbf{\Phi}}\tilde{p}, \text{ and } Tp = \tilde{p}. \tag{2.9}$$

If we build an uncertainty set that contains the true distribution in the noisy domain (let's call it $\tilde{p}$) with confidence level $1-\alpha$, we obtain easily an uncertainty set that contains the true distribution in the clean domain p with the same confidence level. This is due to the fact that

$$\tilde{\mathbf{\Phi}}\tilde{p} = \tilde{\mathbf{\Phi}}Tp = \mathbf{\Phi}p, \tag{2.10}$$

that is, the expectations in the two different domain are the same provided that

$$\mathbf{\Phi} = \tilde{\mathbf{\Phi}}T. \tag{2.11}$$

A confidence interval for the expectation $\tilde{\mathbf{\Phi}}\tilde{p}$ (which means finding a and b and defining an uncertainty set that contains $\tilde{p}$ with probability $1-\alpha$) will also contain the expectation $\mathbf{\Phi}p$, as long as $\mathbf{\Phi} = \tilde{\mathbf{\Phi}}T$. We can interpret (2.11) as the *correction* for the noise we were talking above, and it can also be seen as a way to move uncertainty sets between different domains. At the end the correction is just a modification of the fixed $\tilde{\Phi}$ in such a way that the true expectations in both domains are preserved.

2.1.2 Minimax problem

Once we have understood the correction that has to be applied in order to learn from noisy labels, we proceed by solving the optimization problem (2.1) to derive a robust LPC against noise. All the details regarding how to solve problem (2.1), where the loss considered is the $0-1$ loss, the classification rule h is unconstrained, and the uncertainty set $\mathcal{U}$ is defined as (2.8) where instead of using $\tilde{\boldsymbol{\Phi}}$ we use $\boldsymbol{\Phi} = \tilde{\boldsymbol{\Phi}}T$, are in Appendix B. At the end the important thing for learning the LPC is solving the following convex problem

$$\begin{aligned} \underset{\gamma,\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{maximize}} \quad & \boldsymbol{\alpha}^{\boldsymbol{T}}\boldsymbol{a} - \boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{b} + \gamma \\ \text{subject to} \quad & \|(T^T\tilde{\boldsymbol{\Phi}}^{\boldsymbol{T}}(\boldsymbol{\alpha}-\boldsymbol{\beta}) + \mathbf{1}\gamma)^+\|_{1,\infty} \leq 1, \\ & \boldsymbol{\alpha} \succeq \mathbf{0}, \\ & \boldsymbol{\beta} \succeq \mathbf{0}. \end{aligned} \tag{2.12}$$

Problem (2.12) can be reformulate as a linear optimization problem (LP) by rewriting the constraint

$$\|(T^T\tilde{\boldsymbol{\Phi}}^{\boldsymbol{T}}(\boldsymbol{\alpha}-\boldsymbol{\beta}) + \mathbf{1}\gamma)^+\|_{1,\infty} \leq 1. \tag{2.13}$$

First of all (2.13) is equal to

$$\|(T^T\tilde{\boldsymbol{\Phi}}_{\boldsymbol{x}}^{\boldsymbol{T}}(\boldsymbol{\alpha}-\boldsymbol{\beta}) + \mathbf{1}\gamma)^+\|_1 \leq 1, \forall x \in \mathcal{X}, \tag{2.14}$$

then in order to understand how (2.14) can be develop further, let's study $\|\boldsymbol{v}^+\|_1$ where $\boldsymbol{v}$ is a n-vector. If all the elements of $\boldsymbol{v}$ are positive then $\|\boldsymbol{v}^+\|_1 = \sum_{i=1}^n v_i$. If instead some elements of $\boldsymbol{v}$ are negative, the sum will be only done on a subset of n where the elements of $\boldsymbol{v}$ are positive. Then we can write

$$\|\boldsymbol{v}^+\|_1 \geq \sum_{i\in\mathcal{S}} v_i, \ \forall \mathcal{S} \in \{1,\dots,n\}, \tag{2.15}$$

and so, saying $\|\boldsymbol{v}^+\|_1 \leq 1$ is equivalent to say

$$\max_{\mathcal{S}} \sum_{i\in\mathcal{S}} v_i \leq 1 \iff \sum_{i\in\mathcal{S}} v_i \leq 1, \ \forall \mathcal{S} \in \{1,\dots,n\}. \tag{2.16}$$

It is straightforward now to see that (2.14) is equivalent to

$$\sum_{y\in\mathcal{S}} (T_y^T\tilde{\boldsymbol{\Phi}}_{\boldsymbol{x}}^{\boldsymbol{T}})_y(\boldsymbol{\alpha}-\boldsymbol{\beta}) + |\mathcal{S}|\gamma \leq 1, \ \forall \mathcal{S} \in \mathcal{Y}, \ x \in \mathcal{X}. \tag{2.17}$$

Once we have solved (2.12), meaning we have found $\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \gamma^*$, we can construct the Linear Probabilistic Classifier as

$$h^{a,b}(x,y) = ((T_y^T\tilde{\boldsymbol{\Phi}}_{\boldsymbol{x}}^{\boldsymbol{T}})_y(\boldsymbol{\alpha}^*-\boldsymbol{\beta}^*) + \gamma^*)^+ + \frac{1 - \|(T_y^T\tilde{\boldsymbol{\Phi}}_{\boldsymbol{x}}^{\boldsymbol{T}}(\boldsymbol{\alpha}^*-\boldsymbol{\beta}^*) + \mathbf{1}\gamma^*)^+\|_1}{|\mathcal{Y}|}, \tag{2.18}$$

which gives the probability of classifying a given x with label y. In particular for a given x we have an entire probability distribution $\boldsymbol{h}_{\boldsymbol{x}}^{\boldsymbol{T}}$. For example in case of binary classification where $y \in \{1, 2\}$, we have for a given x

$$h^{a,b}(x,1) = (((1-\rho_1)\tilde{\Phi}(x,1)^T + (\rho_1)\tilde{\Phi}(x,2)^T)(\boldsymbol{\alpha}^* - \boldsymbol{\beta}^*) + \gamma^*)^+ + \frac{1 - \|(T_y^T \tilde{\boldsymbol{\Phi}}_{\boldsymbol{x}}^{\boldsymbol{T}}(\boldsymbol{\alpha}^* - \boldsymbol{\beta}^*) + \mathbf{1}\gamma^*)^+\|_1}{2}, \quad (2.19)$$

$$h^{a,b}(x,2) = (((\rho_2)\tilde{\Phi}(x,1)^T + (1-\rho_2)\tilde{\Phi}(x,2)^T)(\boldsymbol{\alpha}^* - \boldsymbol{\beta}^*) + \gamma^*)^+ + \frac{1 - \|(T_y^T \tilde{\boldsymbol{\Phi}}_{\boldsymbol{x}}^{\boldsymbol{T}}(\boldsymbol{\alpha}^* - \boldsymbol{\beta}^*) + \mathbf{1}\gamma^*)^+\|_1}{2}. \quad (2.20)$$

To summarize The learning stage consists in solving the convex problem (2.12), whereas the prediction stage consists, given an input x, in generating a y according to $\boldsymbol{h}_{\boldsymbol{x}}^{\boldsymbol{T}}$ given by (2.18).

By applying the correction, i.e., using $\boldsymbol{\Phi} = \tilde{\boldsymbol{\Phi}}T$ instead of $\tilde{\boldsymbol{\Phi}}$, we should get better results in testing on a clean dataset, than in the case where the learning algorithm directly learns from a noisy domain. This is indeed the case as shown in the following simulation. Let's consider the synthetic dataset in Figure 2.1 generated by a mixture of four Gaussians, with weights $0.4, 0.6, 0.3, 0.7$ and covariances $0.5^2\boldsymbol{I}$ and $0.25^2\boldsymbol{I}$. The means of the Gaussians are $(1, 1.5), (3, 1), (1, 3), (3, 3.5)$. We split then the dataset in two parts one for training and one for testing. Regarding the training part, we apply an increasing symmetric noise to the labels (see how the training data set changes due to the corruption in Figure 2.2)

$$T_y = \begin{bmatrix} 1-\rho & \rho \\ \rho & 1-\rho \end{bmatrix}, \ \rho = 0,\ 0.1,\ 0.2,\ 0.3,\ 0.4. \quad (2.21)$$

and for each noise level we learn a LPC which then will be tested on the clean test part. The learning stage it has been done both with the correction for the noise and without, then the performances are compared.

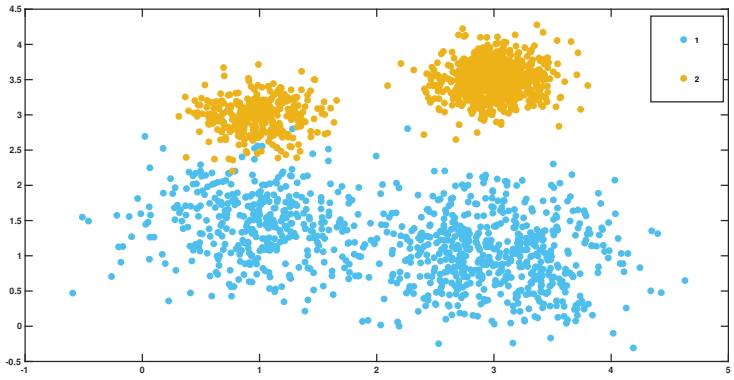


Figure 2.1: Dataset generated by a mixture of gaussians.

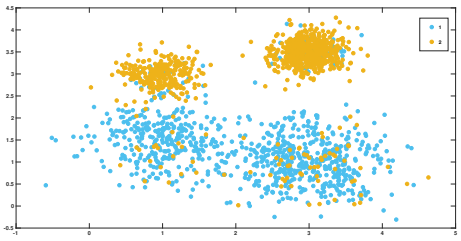
The accuracy as the noise level varies are shown in Figure 2.3 and Table 2.1. We see that by applying the correction for the noise we get better accuracy even for high levels of noises.

Noise	Correction yes	Correction no
0	0.930	0.930
0.1	0.920	0.885
0.2	0.925	0.805
0.3	0.905	0.765
0.4	0.865	0.760

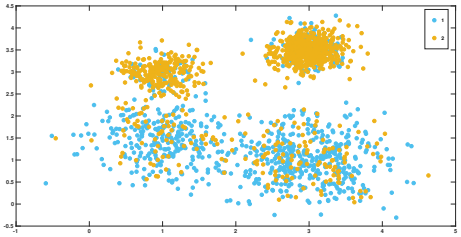
Table 2.1: Accuracy

In order to make things more statistically significant we can consider more training data drawn from the same distribution.

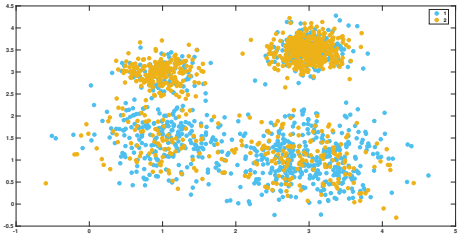
For each instantiation of a training sample of size $n = 10000$ the accuracy is computed for every level of noise pair $(0.1, 0.1), (0.2, 0.2), (0.3, 0.3), (0.4, 0.4)$ on a clean test sample of size $n_t = 1000$. In Figure 2.4 it can be observed that by considering $k = 50$ different samples the mean of all the accuracy of the LPC is almost the same for high levels of noise.



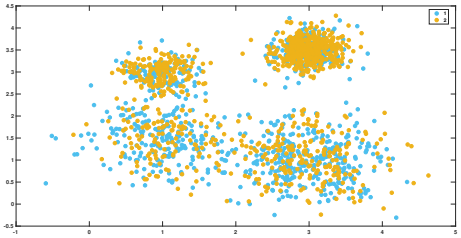
(a) Noise level:0.1



(b) Noise level:0.2



(c) Noise level:0.3



(d) Noise level:0.4

Figure 2.2: Corruption of the training set.

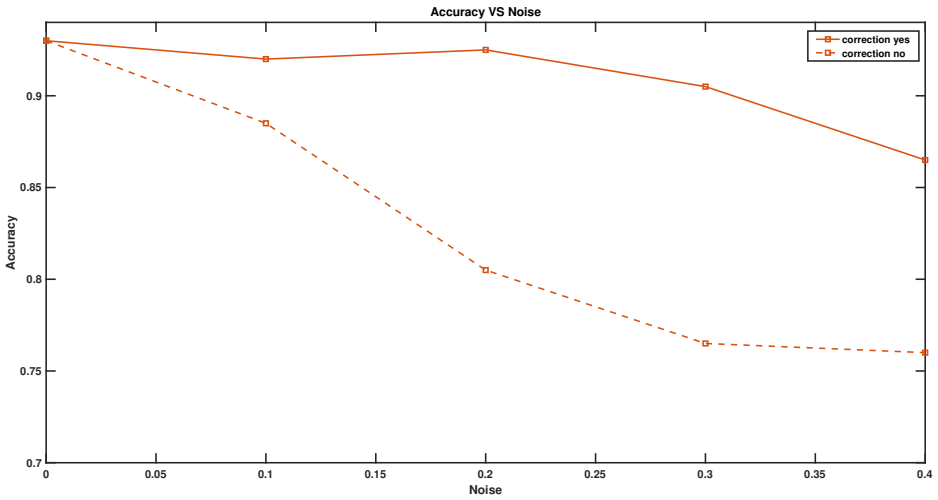


Figure 2.3: Accuracy versus noise

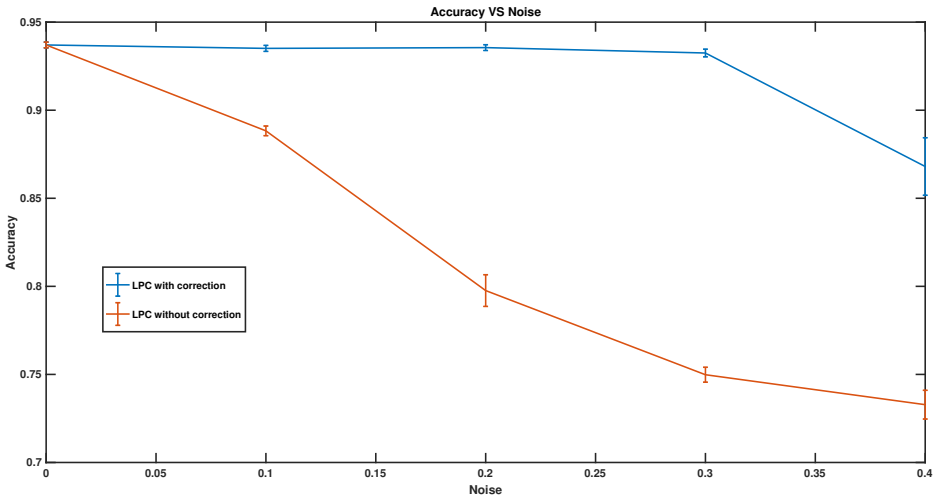


Figure 2.4: Mean accuracy versus noise with 95 per cent confidence intervals for the mean

2.1.3 Point estimation uncertainty set

Up to now we have seen how to derive a LPC by solving problem (2.1) using uncertainty sets of type (2.2), where a and b define a confidence interval for the expectation of the function Φ. In the case of $a = b$ we have what we call *Point estimation uncertainty set* which can be built by estimating a with $\tilde{\tau}_n = \frac{1}{n}\sum_{i=1}^{n} \tilde{\mathbf{\Phi}}(x_i, \tilde{y}_i)$

$$\mathcal{U}_{\tilde{\Phi}}^{\tilde{\tau}_n} = \left\{ \tilde{p} \in \Delta(\mathcal{X} \times \tilde{\mathcal{Y}}) : \tilde{\mathbf{\Phi}}\tilde{p} = \tilde{\tau}_n \right\}, \tag{2.22}$$

where $\tilde{\tau}_n$ can also be written as $\tilde{\mathbf{\Phi}}\tilde{p}_e$,[3] and $\tilde{\mathbf{\Phi}}\tilde{p} = \mathbb{E}_{\tilde{p}}\{\tilde{\Phi}(x, \tilde{y})\}$. The problem with this construction is that the true noisy distribution $\tilde{p}$ will not in general be in $\mathcal{U}_{\tilde{\Phi}}^{\tilde{\tau}_n}$, since it is not true that the expectation of $\tilde{\Phi}$ with respect to the true noisy distribution $\tilde{p}$ is equal to the sample average of $\tilde{\Phi}$ computed on the samples. This implies that when we move $\mathcal{U}_{\tilde{\Phi}}^{a}$ to $\mathcal{U}_{\Phi}^{a}$ the true distribution p will not be inside the uncertainty set since

$$\tilde{\mathbf{\Phi}}\tilde{p} \neq \tilde{\tau}_n \implies \tilde{\mathbf{\Phi}}Tp \neq \tilde{\tau}_n \implies \mathbf{\Phi}p \neq \tilde{\tau}_n. \tag{2.23}$$

In practice this is not a big problem, and sometimes it is even better to solve problem (2.1) on an uncertainty set that does not contain the true distribution p but has a worst case distribution not too far from p, rather than solving the problem on an uncertainty set that contains p but has a worst case distribution very far from p. Since in this last case the minimizer h of the worst case risk can be very different from the minimizer of the risk. Of course the ideal case would be to have an uncertainty set that contains p and a worst case distribution very close to p.

However, something more subtle may happen. Suppose that there are no $p \in \Delta(\mathcal{X} \times \mathcal{Y})$ such that $\tilde{p} = Tp \in \mathcal{U}_{\tilde{\Phi}}^{a}$, i.e., the intersection between the image of T and $\mathcal{U}_{\Phi}^{a}$ is empty, then when we try to move the uncertainty set from the noisy domain to the clean domain we end up with an empty uncertainty set $\mathcal{U}_{\Phi}^{a}$. If we then try to solve the optimization problem[4]

$$\begin{aligned} \underset{\gamma, \boldsymbol{\lambda}}{\text{maximize}} \quad & \boldsymbol{\lambda}^{\boldsymbol{T}}\boldsymbol{a} + \gamma \\ \text{subject to} \quad & \|(T^T\tilde{\mathbf{\Phi}}^{\boldsymbol{T}}\boldsymbol{\lambda} + \mathbf{1}\gamma)^+\|_{1,\infty} \leq 1, \end{aligned} \tag{2.24}$$

we get as optimum value $+\infty$ which means that the minimax problem (2.1) is infeasible (see Appendix B). In order to avoid that, we need to find a way to make the intersection between the image of T and $\mathcal{U}_{\tilde{\Phi}}^{a}$ not empty. One possibility is to replace $\mathcal{U}_{\tilde{\Phi}}^{a}$ $(a = \tilde{\tau}_n)$ with $\mathcal{U}_{\tilde{\Phi}}^{\pi}$, where π is the projection of $\tilde{\tau}_n$ onto the convex hull of the modified $\mathbf{\Phi} = \tilde{\mathbf{\Phi}}T$ [Luenberger (1997)].

$$\text{Conv}(\mathbf{\Phi}) = \left\{ \sum_{i=1}^{n} \mathbf{\Phi}(x_i, y_i)p(x_i, y_i) : \sum_{i=1}^{n} p(x_i, y_i) = 1,\ p(x_i, y_i) \geq 0\ \forall i \right\}. \tag{2.25}$$

The problem we want to solve is then the following convex QP which admits a

[3]Note that the empirical noisy distribution $\tilde{p}_e$ it is in $\mathcal{U}_{\tilde{\Phi}}^{\tilde{\tau}_n}$ by construction.

[4]It is the same problem of (2.12) but since we have $a = b$ we have modified it according to what explained in Appendix B.

closed form solution

$$\begin{aligned} \underset{p,\pi}{\text{minimize}} \quad & \|\boldsymbol{\pi} - \tilde{\tau}_n\|_2^2 \\ \text{subject to} \quad & \mathbf{1}^{\boldsymbol{T}} p = 1, \\ & \boldsymbol{\pi} = \tilde{\boldsymbol{\Phi}} T p, \\ & p \succeq 0. \end{aligned} \tag{2.26}$$

Once we found $\boldsymbol{\pi}_{opt}$ and built $\mathcal{U}_{\tilde{\Phi}}^{\pi_{opt}}$ at least the distribution $\tilde{p}$ such that $\tilde{p} = Tp_{opt}$ will be in the image of T and in $\mathcal{U}_{\tilde{\Phi}}^{\pi_{opt}}$.

In order to find $\boldsymbol{\pi_{opt}}$, let's write the Lagrangian

$$\mathcal{L}(p, \lambda, \nu) = \left\| \tilde{\boldsymbol{\Phi}} \boldsymbol{T} p - \tilde{\tau}_n \right\|_2^2 - \lambda^T p + \nu(1 - \mathbf{1}^{\boldsymbol{T}} p), \tag{2.27}$$

where $\lambda \in \mathbb{R}^n$ and $\nu \in \mathbb{R}$, and impose the KKT conditions

$$p^*, \lambda^*, \nu^* \text{primal and dual optima} \iff \nabla_p \mathcal{L}(p^*, \lambda^*, \nu^*) = 0 \tag{2.28}$$

$$\lambda^* \geq 0 \tag{2.29}$$

$$p^* \geq 0 \tag{2.30}$$

$$\mathbb{1}^T p^* = 1 \tag{2.31}$$

$$\lambda_i^* p^*(x_i, y_i) = 0, \forall i. \tag{2.32}$$

From (2.28) we get:

$$p^* = \frac{1}{2}(\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1}(\lambda^* + \nu^* \mathbb{1} + 2\boldsymbol{\Phi}^T \tilde{\tau}_n), \tag{2.33}$$

$$\boldsymbol{\pi_{opt}} = \sum_{i=1}^{n} \Phi(x_i, y_i) p^*(x_i, y_i) = \boldsymbol{\Phi} p^* = \boldsymbol{\Phi}(\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \frac{\lambda^* + \nu^* \mathbb{1} + 2\boldsymbol{\Phi}^T \tilde{\tau}_n}{2}. \tag{2.34}$$

By calling $d = (\lambda^* + \nu^* \mathbb{1} + 2\boldsymbol{\Phi}^T \tilde{\tau}_n)/2$, we obtain the following linear system

$$\boldsymbol{\Phi}^T \boldsymbol{\pi_{opt}} = d. \tag{2.35}$$

In the following simulation we compare the performances of a Linear Probabilistic Classifier learned using point estimation uncertainty sets, first considering the case without projection and then the case with the projection. The dataset is the same as the one in Figure 2.1. We split the dataset in two parts one for training and one for testing. Regarding the training part, we apply an increasing noise to the labels, and for each noise level we learn a LPC which then will be tested on the

clean test part. In the first simulation we consider as feature map Φ the following

$$\Phi(x, y) = \begin{bmatrix} 1\mathbb{1}[y = 1] \\ 1\mathbb{1}[y = 2] \\ x_1\mathbb{1}[y = 1] \\ x_1\mathbb{1}[y = 2] \\ x_2\mathbb{1}[y = 1] \\ x_2\mathbb{1}[y = 2] \end{bmatrix}. \tag{2.36}$$

The results are shown in Table 2.2 and Figure 2.5.

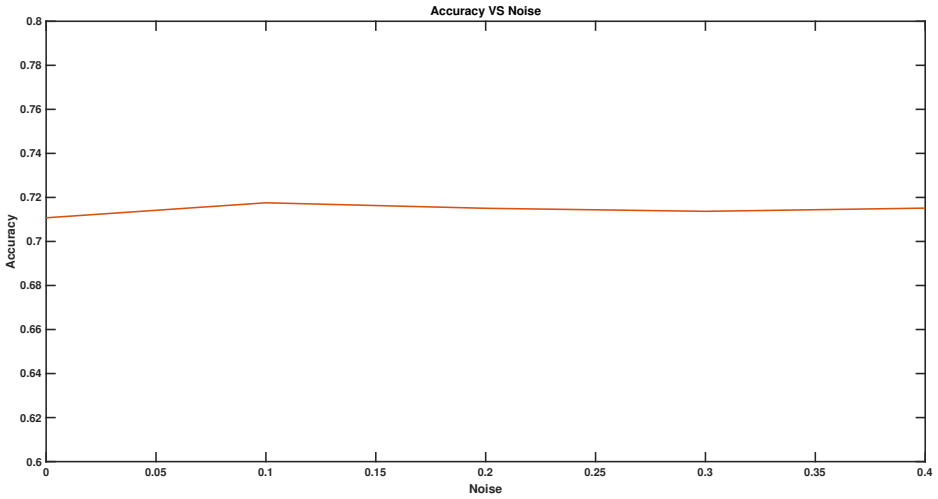


Figure 2.5: Accuracy versus noise. Averaged 50 times.

Noise	Accuracy
0	0.7107
0.1	0.7176
0.2	0.7151
0.3	0.7137
0.4	0.7151

Table 2.2: Accuracy

In this first case with the considered Φ we don't have any problem of feasibility of the primal optimization problem and there is no need in projecting. However the performances with this simple Φ, even though the LPC seems to be robust to noise, are not very impressive. With a more complicated Φ than the one considered before, which takes in account also higher order polynomial terms such the second and third order, we get the results in Table 2.3.

We notice that when the primal problem is infeasible the accuracy of the learned LPC is around 0.5, that is the LPC is a random classifier. If we do the projection then the performances increase substantially (Figure 2.6).

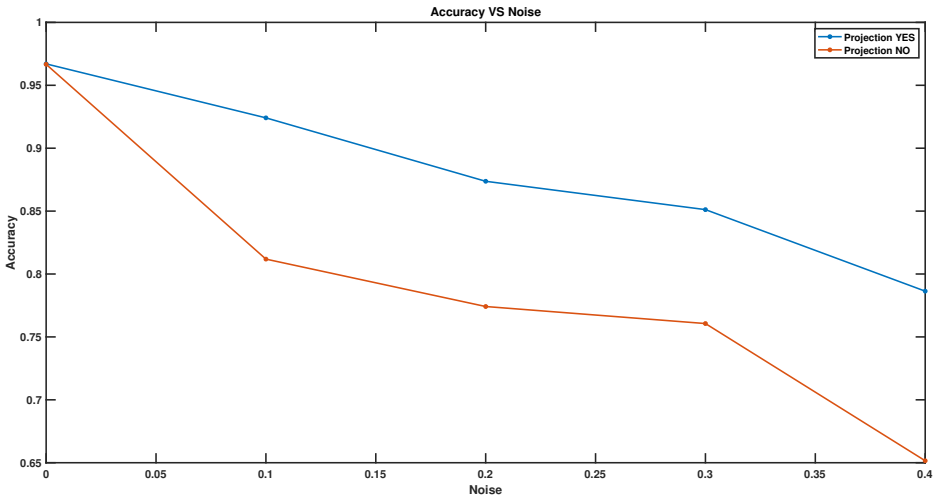


Figure 2.6: Accuracy versus noise. Averaged 50 times.

Noise	Projection no	Projection yes
0	0.9667	0.9670
0.1	0.8118	0.9241
0.2	0.7742	0.8737
0.3	0.7606	0.8512
0.4	0.6516	0.7864

Table 2.3: Accuracy with and without projection

2.2 Generalization Bounds

When dealing with noise we would like to derive some *generalization bounds*, i.e., some learning guarantees of our algorithm in terms of the noise levels ρ_+ and ρ_-. What we want to quantify is how much a fixed noise level prevent us from a satisfactory learning.

2.2.1 Hoeffding's inequality

Before we go into generalization bounds let's introduce another way to build uncertainty sets $\mathcal{U}_\Phi^{a,b}$ which uses Hoeffding's inequality [Wainwright (2019)]. Let $x_1, \dots, x_n$ a sequence of independent bounded random variables with $x_i \in [l_i, u_i]$, called $\overline{x} = \frac{1}{n}\sum_{i=1}^n x_i$ the sample mean, Hoeffding's inequality says

$$\mathbb{P}\left\{|\overline{x} - \mathbb{E}[\overline{x}]| \leq t\right\} \geq 1 - 2\exp\left\{\left(\frac{-2n^2t^2}{\sum_{i=1}^n (l_i - u_i)^2}\right)\right\}, \tag{2.37}$$

that is, for every n the probability that the sample mean stays near its expectation less than a certain amount t is bounded below. The important point is that Hoeffding's inequality holds for every n, and we don't need to assume that n goes to infinity as when we use the Central Limit Theorem. As before let's call

$$\tau \doteq \mathbb{E}_p\{\Phi(x,y)\} = \mathbf{\Phi} p, \tag{2.38}$$

the true expectation, and

$$\tau_n = \frac{1}{n}\sum_{i=1}^{n} \mathbf{\Phi}(x_i, y_i). \tag{2.39}$$

its estimator. If we apply Hoeffding's inequality for every component $j = 1, \ldots, m$ we get

$$\mathbb{P}\left\{\left|(\tau_n)_j - (\tau)_j\right| \leq t_j\right\} \geq 1 - 2\exp\left\{\left(\frac{-2n^2 t_j^2}{\sum_{i=1}^{n}(l_j - u_j)^2}\right)\right\}, \tag{2.40}$$

where

$$l_j = \max_{x_i, y_i} (\mathbf{\Phi}(x_i, y_i))_j \tag{2.41}$$

$$u_j = \min_{x_i, y_i} (\mathbf{\Phi}(x_i, y_i))_j. \tag{2.42}$$

Let's call the event inside the probability in (2.40) A_j and $c_j = l_j - u_j$. Thus we get

$$\mathbb{P}\{A_j\} \geq 1 - 2\exp\left\{\left(\frac{-2nt_j^2}{c_j^2}\right)\right\}. \tag{2.43}$$

We would like to set this probability to be greater or equal to $1 - \alpha/m$, with α usually small

$$1 - \alpha/m = 1 - 2\exp\left\{\left(\frac{-2nt_j^2}{c_j^2}\right)\right\} \iff \tag{2.44}$$

$$\log\frac{2}{\alpha} + \log m = \frac{2nt_j^2}{c_j^2} \longrightarrow t_j = c_j\sqrt{\frac{\log\frac{2}{\alpha} + \log m}{2n}} \leq 2\|\mathbf{\Phi}_j\|_\infty \sqrt{\frac{\log\frac{2}{\alpha} + \log m}{2n}}. \tag{2.45}$$

Using the union bound we have

$$\mathbb{P}\{\cap_{j=1}^{m} A_j\} \geq \sum_{j=1}^{m} \mathbb{P}\{A_j\} + 1 - m \geq 1 - \alpha \tag{2.46}$$

where the second inequality follows from the fact that we have put $\mathbb{P}\{A_j\} \geq 1 - \alpha/m$.

At the end, by putting $a = \tau_n - t$ and $b = \tau_n + t$ where t given by (2.45) we build an uncertainty set $\mathcal{U}_\Phi^{a,b}$ that contains p with probability $1 - \alpha$. To conclude, with probability $1 - \alpha$ we also have

$$\|\tau_n - \tau\|_2 \leq \|c\|_2 \sqrt{\frac{\log\frac{2}{\alpha} + \log m}{2n}}. \tag{2.47}$$

2.2.2 Generalization Bounds in the clean domain

Before deriving the generalization bounds in the noisy domain, it is useful to look at the case without noise. Let's start by recalling that the minimax problem

$$\min_{h\in\Delta(\mathcal{X},\mathcal{Y})}\max_{p\in\mathcal{U}_\Phi^{a,b}}\mathbb{E}_p l(h,(x,y)), \tag{2.48}$$

can be interpreted in learning theory not only as an optimization problem but also as a learning algorithm. The output of the learning algorithm will be a LPC h_S, the minimizer of the worst case risk, that hopefully will have a small risk or generalization error $R(h_S)$, defined as the expected loss of h_S over the true distribution p, which is also the distribution of new samples

$$R(h_S) = \mathbb{E}_p l(h_S,(x,y)). \tag{2.49}$$

The risk tells us how good is the specific classifier generated by the learning algorithm (B.4), but since we don't have access to p we cannot evaluate directly $R(h_S)$. For this reason in general we are interested in finding an upper bound for $R(h_S)$ with high probability. A first very simple upper bound for $R(h_S)$ is the worst case risk $R_\Phi^{a,b}$ of h_S given directly by solving (B.4). In fact, if the true distribution p is in $\mathcal{U}_\Phi^{a,b}$ with confidence $1-\alpha$, we have with probability $1-\alpha$ that

$$R(h_S) \leq R_\Phi^{a,b}. \tag{2.50}$$

Before looking at another bound, we introduce the smallest possible worst case risk R_Φ^τ for a fixed feature map Φ as

$$R_\Phi^\tau = \min_{h\in\Delta(\mathcal{X},\mathcal{Y})}\max_{p\in\mathcal{U}_\Phi^{\tau\infty}}\mathbb{E}_p l(h,(x,y)), \tag{2.51}$$

where τ is the expectation of the feature map Φ with respect to the true distribution p. In this way p is in $\mathcal{U}_\Phi^\tau$ by construction. In order to see why R_Φ^τ is the smallest possible worst case risk, note that given an arbitrarily $\mathcal{U}_\Phi^{a,b}$ that contains p with probability $1-\alpha$ it will then contains $\mathcal{U}_\Phi^\tau$ with the same probability level. This implies

$$R_\Phi^\tau \leq R_\Phi^{a,b}. \tag{2.52}$$

Fixing a, b and Φ, we can actually bound the difference between the worst case risk $R_\Phi^{a,b}$ and the smallest possible one of Φ, R_Φ^τ. We know that[5]

$$R_\Phi^{a,b} = 1 - a^T\alpha^* + b^T\beta^* - \gamma^*, \tag{2.53}$$

[5] see Appendix B.

where α^*,β^* and γ^* is solution of

$$\begin{array}{ll} \underset{\gamma,\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{maximize}} & \boldsymbol{\alpha}^{\boldsymbol{T}}\boldsymbol{a} - \boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{b} + \gamma \\ \text{subject to} & \|(\boldsymbol{\Phi}^{\boldsymbol{T}}(\boldsymbol{\alpha}-\boldsymbol{\beta}) + \mathbf{1}\gamma)^+\|_{1,\infty} \leq 1, \\ & \boldsymbol{\alpha} \succeq \mathbf{0}, \\ & \boldsymbol{\beta} \succeq \mathbf{0}, \end{array} \tag{2.54}$$

and

$$R_\Phi^\tau = 1 - \tau^T\lambda^* - \gamma^*, \tag{2.55}$$

where $\boldsymbol{\lambda^*}$ and γ^* is solution of

$$\begin{array}{ll} \underset{\gamma,\boldsymbol{\lambda}}{\text{maximize}} & \boldsymbol{\lambda}^{\boldsymbol{T}}\boldsymbol{\tau} + \gamma \\ \text{subject to} & \|(\boldsymbol{\Phi}^{\boldsymbol{T}}\boldsymbol{\lambda} + \mathbf{1}\gamma)^+\|_{1,\infty} \leq 1. \end{array} \tag{2.56}$$

It is easy to see that a solution of problem (2.56) is a feasible point for problem (2.54) by writing $\lambda^* = (\lambda^*)^+ - (-\lambda^*)^+$. Then we can write

$$R_\Phi^{a,b} \leq 1 - a^T(\lambda^*)^+ + b^T(-\lambda^*)^+ - \gamma^*, \tag{2.57}$$

because $R_\Phi^{a,b}$ is the optimal value of problem (2.54). Adding and subtracting $\tau^T\lambda^*$ in the right end side, we get

$$R_\Phi^{a,b} \leq R_\Phi^\tau + (\lambda^*)^+(\tau - a) + (-\lambda^*)^+(b - \tau). \tag{2.58}$$

Depending now, on how we choose a and b we get different bounds. For example if we follow the construction using Hoeffding's inequality we have

$$a = \tau_n - c\sqrt{\frac{\log m + \log(2/\alpha)}{2n}},\ b = \tau_n + c\sqrt{\frac{\log m + \log(2/\alpha)}{2n}}. \tag{2.59}$$

By substituting in (2.58) we have

$$R_\Phi^{a,b} \leq R_\Phi^\tau + (\tau - \tau_n)^T\lambda^* + \sqrt{\frac{\log m + \log(2/\alpha)}{2n}}c^T((\lambda^*)^+ + (-\lambda^*)^+), \tag{2.60}$$

and using result (2.47) and Cauchy-Schwarz inequality we get a bound for the difference between the worst case risk and the smallest possible one.

$$R_\Phi^{a,b} \leq R_\Phi^\tau + 2\|\lambda^*\|_2\|c\|_2\sqrt{\frac{\log m + \log(2/\alpha)}{2n}}. \tag{2.61}$$

From this it follows that we can build a bound on $R(h_S)$ as

$$R(h_S) \leq (R_\Phi^{a,b} - R_\Phi^\tau) + R_\Phi^\tau \leq 2\|\lambda^*\|_2\|c\|_2\sqrt{\frac{\log m + \log(2/\alpha)}{2n}} + R_\Phi^\tau. \tag{2.62}$$

2.2.3 Generalization bounds with noise

In presence of noisy labels it is possible to construct an upper bound for the risk $R(h_S)$ where the risk is computed with respect to the clean distribution p and h_S is built with a solution of (2.12) as in (2.18).

First of all let's fix a feature map $\tilde{\Phi} : \mathcal{X} \times \tilde{\mathcal{Y}} \to \mathbb{R}^m$, with $|\tilde{\mathcal{Y}}|= 2$, such that

$$\tilde{\Phi}_i(x, 1) \cdot \tilde{\Phi}_i(x, 2) = 0, \ \forall x \in \mathcal{X}, \ \forall i = 1, \dots, m. \tag{2.63}$$

That is we are considering feature maps that take points $(x, y) \in \mathcal{X} \times \tilde{\mathcal{Y}}$ into orthogonal[6] vectors in $\mathbb{R}^m$ if the labels are different. Given n points $(x_1, \tilde{y}_1), \dots, (x_n, \tilde{y}_n)$ consider the row i of the matrix $\tilde{\boldsymbol{\Phi}} \in \mathbb{R}^{m\times 2n}$

$$\tilde{\boldsymbol{\Phi}}_i = \Big[\tilde{\Phi}_i(x_1, 1), \tilde{\Phi}_i(x_1, 2), \dots, \tilde{\Phi}_i(x_n, 1), \tilde{\Phi}_i(x_n, 2)\Big] \in \mathbb{R}^{2n}, \tag{2.64}$$

and imagine to build two subvectors $\tilde{\boldsymbol{\Phi}}_i^{\,1}$ and $\tilde{\boldsymbol{\Phi}}_i^{\,2} \in \mathbb{R}^n$, i.e, grouping $\tilde{\boldsymbol{\Phi}}_i$ according to the label. Then it is true that

$$\|\tilde{\boldsymbol{\Phi}}_i\|_\infty = \max(\|\tilde{\boldsymbol{\Phi}}_i^{\,1}\|_\infty, \|\tilde{\boldsymbol{\Phi}}_i^{\,2}\|_\infty). \tag{2.65}$$

Notice that once we fix a $\tilde{\Phi}$ and a $\boldsymbol{T}$ we can write $\boldsymbol{\Phi} = \tilde{\boldsymbol{\Phi}}\boldsymbol{T}$ where $\boldsymbol{\Phi}_i = \tilde{\boldsymbol{\Phi}}_i\boldsymbol{T}$.[7] Having in mind that, it is possible to write the infinite norm of the i-th row of $\boldsymbol{\Phi}$ in terms of the noise rates ρ_+ and ρ_- as

$$\|\boldsymbol{\Phi}_i\|_\infty = \|\tilde{\boldsymbol{\Phi}}_i\boldsymbol{T}\|_\infty = \max\Big(\|(1-\rho_+)\tilde{\boldsymbol{\Phi}}_i^{\,1} + (\rho_+)\tilde{\boldsymbol{\Phi}}_i^{\,2}\|_\infty, \|(\rho_-)\tilde{\boldsymbol{\Phi}}_i^{\,1} + (1-\rho_-)\tilde{\boldsymbol{\Phi}}_i^{\,2}\|_\infty\Big) \tag{2.66}$$

$$= \max\Big[\max\Big((1-\rho_+)\|\tilde{\boldsymbol{\Phi}}_i^{\,1}\|_\infty, (\rho_+)\|\tilde{\boldsymbol{\Phi}}_i^{\,2}\|_\infty\Big), \max\Big((\rho_-)\|\tilde{\boldsymbol{\Phi}}_i^{\,1}\|_\infty, (1-\rho_-)\|\tilde{\boldsymbol{\Phi}}_i^{\,2}\|_\infty\Big)\Big] \tag{2.67}$$

$$= \max\Big[(1-\rho_+)\|\tilde{\boldsymbol{\Phi}}_i^{\,1}\|_\infty, (\rho_+)\|\tilde{\boldsymbol{\Phi}}_i^{\,2}\|_\infty, (\rho_-)\|\tilde{\boldsymbol{\Phi}}_i^{\,1}\|_\infty, (1-\rho_-)\|\tilde{\boldsymbol{\Phi}}_i^{\,2}\|_\infty\Big]. \tag{2.68}$$

Then if $(1-\rho_+)\|\tilde{\boldsymbol{\Phi}}_i^{\,1}\|_\infty > (1-\rho_-)\|\tilde{\boldsymbol{\Phi}}_i^{\,2}\|_\infty$ we can have

- $\|\tilde{\boldsymbol{\Phi}}_i^{\,1}\|_\infty > \|\tilde{\boldsymbol{\Phi}}_i^{\,2}\|_\infty \longrightarrow \|\boldsymbol{\Phi}_i\|_\infty = (1-\rho_+)\|\tilde{\boldsymbol{\Phi}}_i\|_\infty$
- $\|\tilde{\boldsymbol{\Phi}}_i^{\,1}\|_\infty < \|\tilde{\boldsymbol{\Phi}}_i^{\,2}\|_\infty \longrightarrow \|\boldsymbol{\Phi}_i\|_\infty = (1-\rho_+)\|\tilde{\boldsymbol{\Phi}}_i^{\,1}\|_\infty < (1-\rho_+)\|\tilde{\boldsymbol{\Phi}}_i\|_\infty$.

So if $(1-\rho_+)\|\tilde{\boldsymbol{\Phi}}_i^{\,1}\|_\infty > (1-\rho_-)\|\tilde{\boldsymbol{\Phi}}_i^{\,2}\|_\infty$ we get

$$\|\boldsymbol{\Phi}_i\|_\infty \leq (1-\rho_+)\|\tilde{\boldsymbol{\Phi}}_i\|_\infty, \tag{2.69}$$

[6] Notice that they are more than orthogonal since the product of each component is zero.

[7] Fixing a matrix $\boldsymbol{T}$ means fixing the noise levels. For different $\boldsymbol{T}$ we get different $\boldsymbol{\Phi}$.

otherwise

$$\|\boldsymbol{\Phi}_i\|_\infty \leq (1-\rho_-)\|\tilde{\boldsymbol{\Phi}}_i\|_\infty. \tag{2.70}$$

At the end we have the following relationship between $\|\boldsymbol{\Phi}_i\|_\infty$ and $\|\tilde{\boldsymbol{\Phi}}_i\|_\infty$

$$\|\boldsymbol{\Phi}_i\|_\infty \leq \max\Big[(1-\rho_+),(1-\rho_-)\Big]\|\tilde{\boldsymbol{\Phi}}_i\|_\infty, \tag{2.71}$$

or

$$\|\tilde{\boldsymbol{\Phi}}_i\|_\infty \leq \max\Big[\frac{1-\rho_+}{1-\rho_+-\rho_-},\frac{1-\rho_-}{1-\rho_+-\rho_-}\Big]\|\boldsymbol{\Phi}_i\|_\infty.^{8} \tag{2.72}$$

Recalling now the bound of (2.62) in the clean domain, we have in the presence of noise the analogous bound

$$R(h_S) \leq 2\|\boldsymbol{\lambda}^*\|_2\|\tilde{c}\|_2\sqrt{\frac{\log m+\log(2/\alpha)}{2n}}+R^\tau_\Phi, \tag{2.73}$$

where $\boldsymbol{\Phi}=\tilde{\boldsymbol{\Phi}}\boldsymbol{T}$ and $\boldsymbol{\lambda}^*$ solution of problem (2.24). Now notice that due to (2.72) we can write

$$\tilde{c}_j \leq 2\|\tilde{\boldsymbol{\Phi}}_j\|_\infty \leq 2\max\Big[\frac{1-\rho_+}{1-\rho_+-\rho_-},\frac{1-\rho_-}{1-\rho_+-\rho_-}\Big]\|\boldsymbol{\Phi}_i\|_\infty, \tag{2.74}$$

from which follows

$$\|\tilde{c}\|_2 \leq 2\sqrt{m}\max\Big[\frac{1-\rho_+}{1-\rho_+-\rho_-},\frac{1-\rho_-}{1-\rho_+-\rho_-}\Big]\max_i\|\boldsymbol{\Phi}_i\|_\infty. \tag{2.75}$$

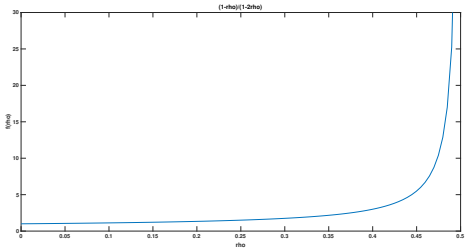
At the end we can upper bound $R(h_S)$ with a bound that depends on the noise rates that quantify how much the noise prevent us from learning.

$$R(h_S) \leq$$
$$4\|\lambda^*\|_2\sqrt{m}\max\Big[\frac{1-\rho_+}{1-\rho_+-\rho_-},\frac{1-\rho_-}{1-\rho_+-\rho_-}\Big]\max_i\|\boldsymbol{\Phi}_i\|_\infty\sqrt{\frac{\log m+\log(2/\alpha)}{2n}}+R^\tau_\Phi. \tag{2.76}$$

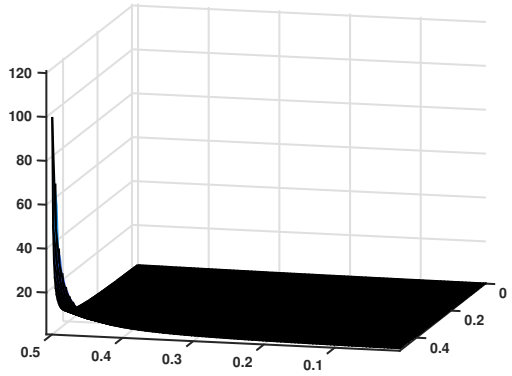
From bound (2.76) interesting conclusions can be drawn. First of all, notice that the influence of noise enters only in the first term of right end side of the inequality because of the expression: $\max\Big[\frac{1-\rho_+}{1-\rho_+-\rho_-},\frac{1-\rho_-}{1-\rho_+-\rho_-}\Big]$ (Figures 2.7(a), 2.7(b) and 2.7(c)). As ρ_+ and ρ_- approach 0.5 the bound increases very fast giving no guarantees for

[8] This second expression is obtained, instead of fixing $\tilde{\Phi}$ in the noisy domain, by fixing Φ in clean domain. In this way in order to get the $\tilde{\Phi}$ we have to invert the matrix $\boldsymbol{T}$.

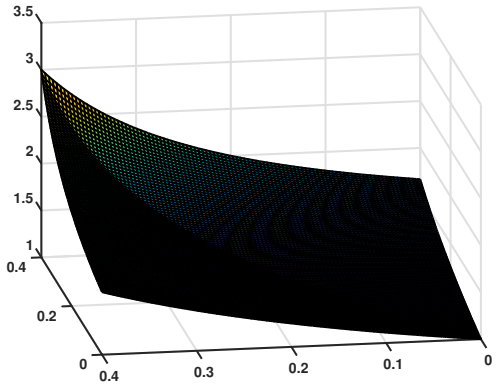
learning. In this case the bound doesn't decrease with the number of training samples n. On the other hand for noise levels' values far from 0.5, if n is very high the first term of right end side of the inequality is almost zero and we would have $R(h_S) \leq R^{\tau}_{\Phi}$, making the noise negligible. Lastly from (2.76) and Figure 2.7(c) we can also say that learning with symmetric noise levels ($\rho_+ = \rho_- = k/2$) is easier than learning from situations of class conditional noise levels in which the all noise affects just one class ($\rho_+ = k$ and $\rho_- = 0$). In other words, from a learning point of view it is better to equally split a total amount of noise into the classes than having a clean class and the other receiving all the amount of noise.



(a) Plot of the function $\frac{1-\rho}{1-2\rho}$. This is the case of symmetric noise.



(b) As $\rho_+ = \rho_-$ go to 0.5 the function increases very fast making the bound less tight.



(c) If we consider noise levels up to 0.4 we get a better insight. By fixing the sum of the noise levels, that is $\rho_+ + \rho_- = k$ we get a bound which is tighter for symmetric noise levels ($\rho_+ = k/2, \rho_- = k/2$) than for noise levels ($\rho_+ = 0, \rho_- = k$).

Figure 2.7

2.3 Heterogenous Noisy Labels

Up to now we have considered cases in which we have applied a random noise to all the original dataset. However more complicated situations can happen. For example we could have in the training set groups of samples following different distributions

$$\begin{aligned}(x_{11}, \tilde{y}_{11}) \dots (x_{1n_1}, \tilde{y}_{1n_1}) \text{ i.i.d. from } \tilde{p}_1 \\ (x_{21}, \tilde{y}_{21}) \dots (x_{2n_2}, \tilde{y}_{2n_2}) \text{ i.i.d. from } \tilde{p}_2 \\ \vdots \\ (x_{m1}, \tilde{y}_{m1}) \dots (x_{mn_m}, \tilde{y}_{mn_m}) \text{ i.i.d. from } \tilde{p}_m,\end{aligned}$$

2.3.1 Semi-supervised Learning with LPC

Consider the case where the noise in the labels is applied only to a fraction of the training dataset

$$\begin{aligned}(x_{11}, y_{11}) \dots (x_{1n_1}, y_{1n_1}) \text{ i.i.d. from } p \\ (x_{21}, y_{21}) \dots (x_{2n_2}, y_{2n_2}) \text{ i.i.d. from } \tilde{p},\end{aligned}$$

where p is the true distribution and $\tilde{p} = Tp$, with $T = \mathbb{I} \otimes T_y$. In the case of

$$T_y = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}, \tag{2.77}$$

we are in a situation where for a group of samples the label is uninformative since the labelling is random. We can think at an unbiased coin that labels those samples. This situation is equivalent to *semi-supervised* learning where some training samples are unlabeled [Zhu & Goldberg (2009)]. Let q be the fraction of clean data in the training set, we want to see how the performances of the LPC degrades as we decrease the value of q. That is, we are interested in seeing if we can still have a satisfactory learning even when most of the training dataset has been labeled randomly. For instance, consider the dataset in Figure 2.8 of size $n = 10000$ where 90% of instances (grey dots) have with an equal probability of being in class 1 or in class 2. If we learn from that dataset we would have an accuracy of 0.923, as shown in Figure 2.9 and Table 2.4.

q	Accuracy
0.1	0.923
0.2	0.931
0.3	0.939
0.5	0.946
0.7	0.948
0.9	0.950

Table 2.4: Accuracy degrades by varying q

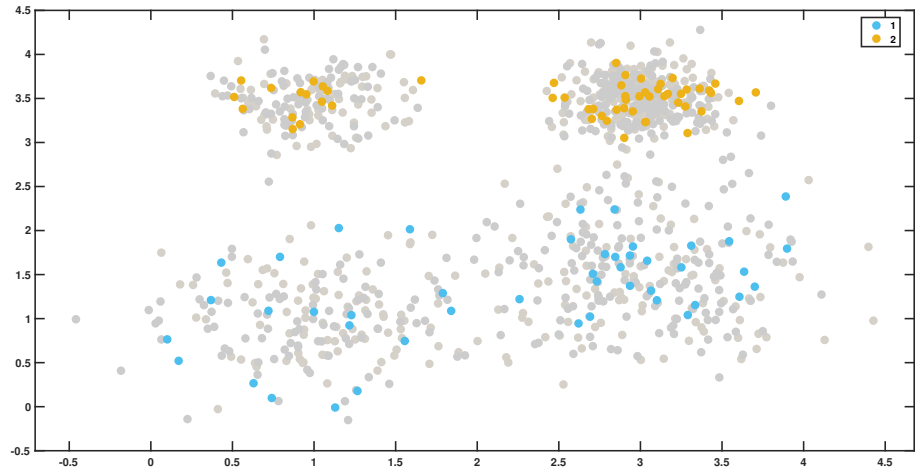


Figure 2.8: dataset where 90% of instances (grey points) have random label.

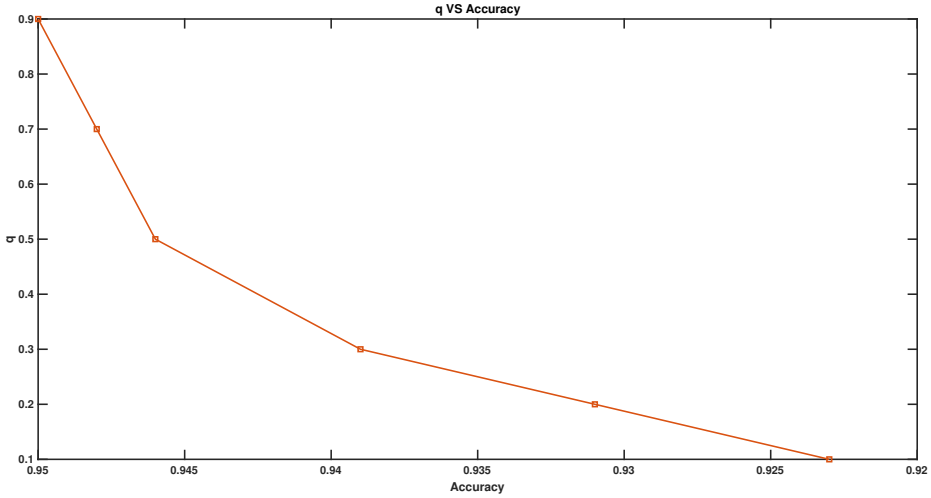


Figure 2.9: How the accuracy diminishes as we decreased the fraction of clean data. However even when ninety percent of the dataset is randomly labeled we have a satisfactory learning

The goal of semi-supervised learning is to train a classifier h from both the labeled and unlabeled data, such that it is better than the supervised classifier trained on the labeled data alone. If we eliminate the noisy labels and consider only the correct labeled data we see in Figure 2.10 and Table 2.5 that the performance are worst than in the case when we consider the all corrupted dataset suggesting that, even if some labels are uninformative, the associated instances provide useful information regarding the distributions.

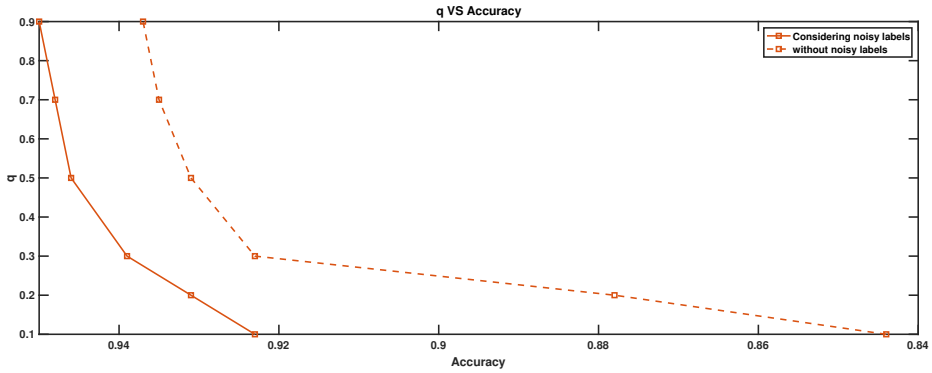


Figure 2.10: It is better to learn from a corrupted dataset with some fraction of clean data rather than considering only the learning from a small fraction of clean data.

q	Accuracy considering noisy labels	Accuracy eliminating noisy labels
0.1	0.923	0.844
0.2	0.931	0.878
0.3	0.939	0.923
0.5	0.946	0.931
0.7	0.948	0.935
0.9	0.950	0.937

Table 2.5: Comparison of the performances between considering all the corrupted dataset and considering only the fraction of clean labels.

A very simple algorithm for semi-supervised learning is called *Self-learning*, because the learning process uses its own predictions to teach itself. In particular, it consists in training a model on labeled data and then use it for predicting the unlabeled data. By doing this, we end up with a full labeled training dataset from which we can learn. In Figure 2.11 is shown the comparison between Self-learning and our approach that learns directly from noisy labels. Notice that when $q = 0.1$, meaning that 90% of the dataset is unlabeled, Self-learning method performs poorly because it is only learning from 10% of all the observations and predicting the 90% part of unlabeled data. At the end we get a dataset with 90% of it being self-learned. It is interesting that, for low value of q, seems better to have pure random noisy labels and know how to correct for the noise, rather than trying to label the unlabeled part with a classifier learned from only few labels. For high value of q instead, we get what we expect: the predictions of self-learning are much more reliable since the labeled part is much bigger.

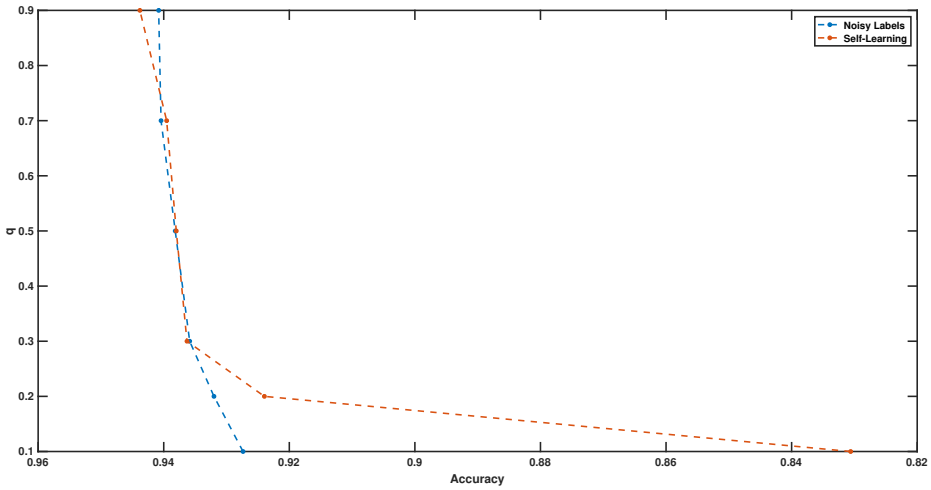


Figure 2.11: The two lines are the average of $k = 30$ values.

2.3.2 Different noise levels for different groups

Above we have applied to two sub groups of the dataset corruptions given by

$$T_{y1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, T_{y2} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}. \tag{2.78}$$

In particular we have applied T_{y1} to a fraction q of the original dataset. Now we show what would happen if we use different T_{yi} than the ones in (2.78). Let's call ρ_1 and ρ_2 the noise levels for the two different groups[9]. Figure 2.12 shows, for different pairs ρ_1, ρ_2, how the accuracy changes as varying q, where q in this case is the fraction of data with less noise applied. In particular we have considered the following noise pairs $(0, 0.5), (0.1, 0.4), (0.2, 0.3), (0.1, 0.5), (0.2, 0.5), (0.3, 0.5), (0, 0.4), (0, 0.3), (0, 0.2)$. In Figures 2.13(a) and 2.13(b) it is shown more clearly what happens for values of q near 0.9 and 0.1.

[9]We are in the case where in each group we have symmetric noise. That is $\rho_+ = \rho_- = \rho_i$ within each group i. So for example in (2.78) we have $\rho_1 = 0$ and $\rho_2 = 0.5$.

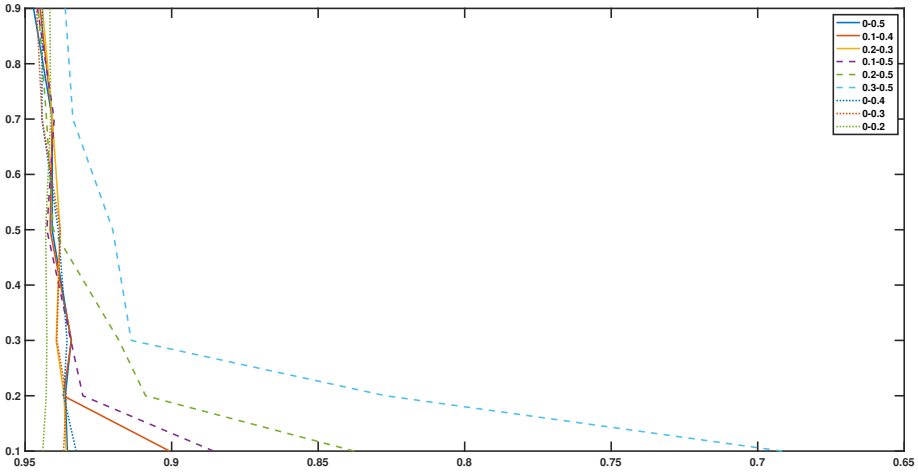
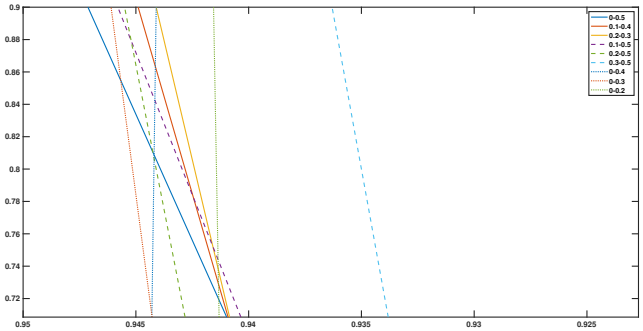
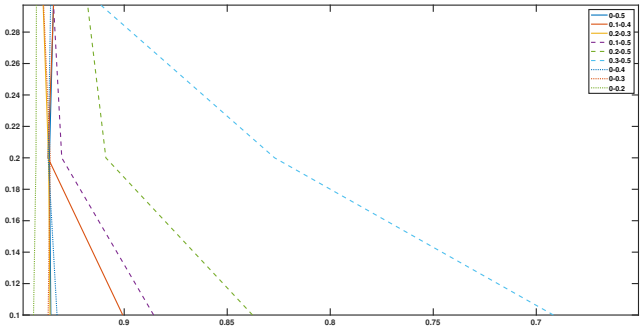


Figure 2.12: For different corruptions of two sub groups, we see how the accuracy changes by varying the proportion of the two sub groups. Averaged 10 times.



(a) For values of q near 0.9.



(b) For values of q near 0.1.

Figure 2.13

2.4 Multiclass Classification

Now let's extend the case of binary classification to the case of multiclass classification. Consider the synthetic dataset in Figure 2.14, and suppose that points belonging to the purple class can independently go to the blue class with probability 1/3, go to the red class with probability 1/3 or stay in the purple class with probability 1/3. At the same time points belonging to the blue or red class will remain in their class with probability 2/3 and will go to the purple class with probability 1/3. That is to the dataset in Figure 2.14 we inject a random noise corresponding to

$$T_y = \begin{bmatrix} 2/3 & 0 & 1/3 \\ 0 & 2/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}, \tag{2.79}$$

which produce the dataset in Figure 2.15.

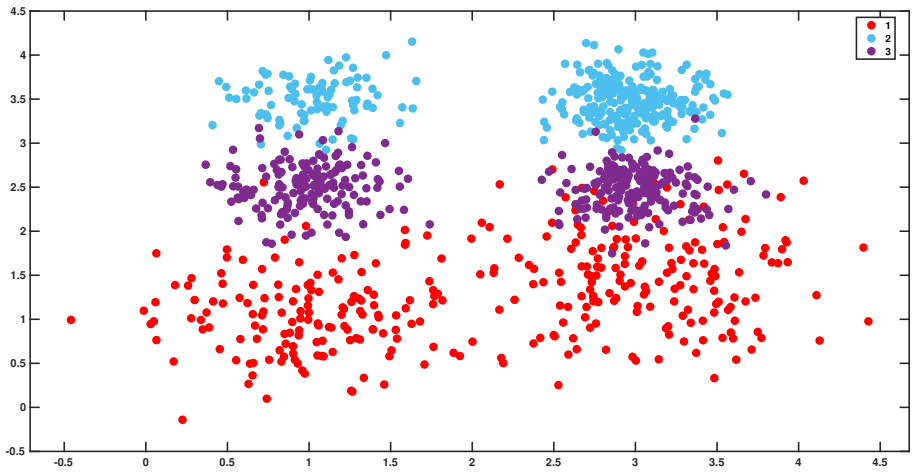


Figure 2.14: Dataset for multiclass classification

If we train a LPC on the clean dataset we get an accuracy of 0.89, whereas learning from the noisy dataset with the correction for the noise leads to an accuracy of 0.74 which is remarkably high considering the level of noise we have injected. If we had not applied the correction we would have an accuracy of 0.61.

The above example is interesting since the matrix T_y in (2.79) has determinant equal to zero, i.e., is not *invertible*. When this happens, in literature we say that T is not *reconstructible*. While existing techniques in [Van Rooyen & Williamson (2017) and Natarajan et al. (2013)] deal only with reconstructible noise our approach is able to treat both.

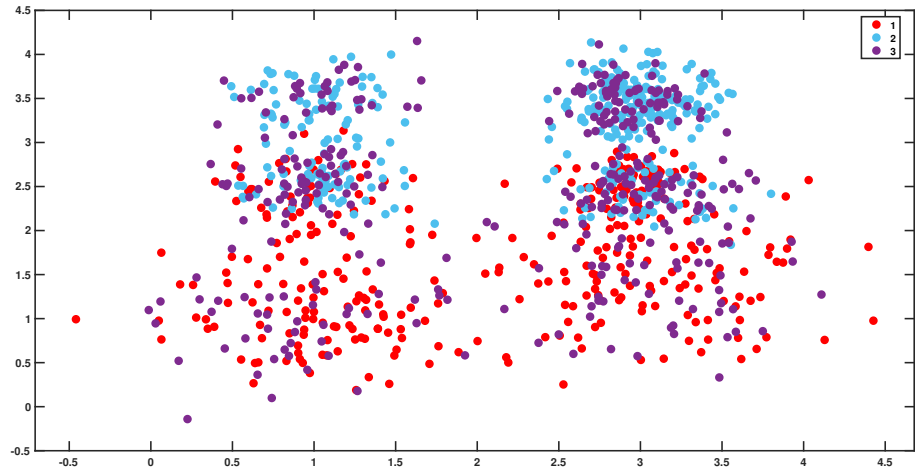


Figure 2.15: Dataset corrupted for multiclass classification

2.4.1 Iris dataset

Let's see on a real dataset corrupted with some noise how LPC behaves. In particular we consider the well known Iris dataset made of 4 features and 150 instances each of which has a label $y \in \{1, 2, 3\}$. The dataset is shown in Figure 2.16 where we have plotted only the first two features *Sepal length* and *Sepal width*.

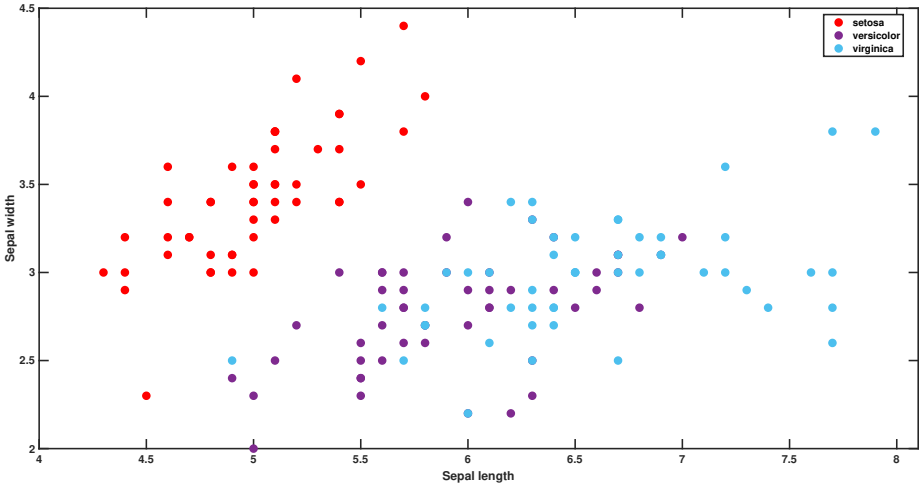


Figure 2.16: Iris Dataset

We have performed a 10-fold cross validation for each of the following cases: we first have considered the case without noise, then the case with noise given by (2.79)

taking in account the correction for it, and lastly the case with noise without any correction. The results are respectively 0.70, 0.60 and 0.46 and confirm that by applying the correction we increase the accuracy of our classifier.

Chapter 3

Performances on Real Datasets

In this final chapter we compare on real dataset our proposed approach against noisy labels using Linear Probabilistic Classifiers, with the approach of [Natarajan et al. (2013)] that uses the method of Unbiased Estimators, and we will show that our approach is competitive.

3.1 Method of Unbiased Estimators

The method of unbiased estimator presented in [Natarajan et al. (2013)] for binary classification is another approach to deal with noisy labels. Suppose we are in the same setting we have considered so far: the learning algorithm sees samples with labels that have been independently changed with probability that is class dependent. So the learner has access to samples drawn from $\tilde{p}$ which is a noisy version of the true distribution p. Given a loss function $l(h, y)$,[10] instead of doing empirical risk minimization with $l(h, y)$, the method consists of minimizing the empirical risk of a modified version of the loss $l(h, y)$, which we indicate as $\tilde{l}(h, y)$. [Natarajan et al. (2013)] construct $\tilde{l}(h, y)$ using the noise rates ρ_{+1} and ρ_{-1} in such a way that $\tilde{l}(h, y)$ is an unbiased estimator of $l(h, y)$. That is

$$\mathbb{E}_{\tilde{y}}[\tilde{l}(h, \tilde{y})] = l(h, y), \tag{3.1}$$

explicitly

$$l(h, +1) = (1 - \rho_{+1})\tilde{l}(h, +1) + (\rho_{+1})\tilde{l}(h, -1), \tag{3.2}$$

$$l(h, -1) = (\rho_{-1})\tilde{l}(h, +1) + (1 - \rho_{-1})\tilde{l}(h, -1), \tag{3.3}$$

$$\boldsymbol{l} = T\tilde{\boldsymbol{l}}, \qquad T = \begin{bmatrix} 1 - \rho_{+1} & \rho_{+1} \\ \rho_{-1} & 1 - \rho_{-1} \end{bmatrix} \tag{3.4}$$

[10] We omit the dependence on x in order to simplify the notation.

and solving for $\tilde{l}$ we get

$$\tilde{\boldsymbol{l}} = R\boldsymbol{l}, \qquad R = T^{-1} = \frac{1}{\det(T)} \begin{bmatrix} 1-\rho_{-1} & -\rho_{+1} \\ -\rho_{-1} & 1-\rho_{+1} \end{bmatrix} \tag{3.5}$$

$$\tilde{l}(h, y) \doteq \frac{(1-\rho_{-y})l(h, y) - \rho_y l(h, -y)}{1-\rho_{+1}-\rho_{-1}}. \tag{3.6}$$

[Natarajan et al. (2013)] then proposes to minimize the empirical risk of the modified loss function in (3.6). From (3.5) is also clear why this method only works for reconstructible noise, i.e., for invertible T. In the following we will compare our performances on some real dataset with the ones of the method presented in [Natarajan et al. (2013)] in which the loss considered is the linear Logistic loss

$$l(h, y) = \log(1 + \exp(-yh_w)), \tag{3.7}$$

with $h_w \in \mathcal{H} = \{h_w : h_w = w^T x, x \in \mathcal{X}\}$.

3.2 Performances on Real Datasets

We now conduct experiments on real data to illustrate the performance of our proposed approach and compare it with the one of [Natarajan et al. (2013)].

The real datasets considered are UCI benchmarks dataset provided by Gunnar Ratsch: http://theoval.cmp.uea.ac.uk/matlab. Each dataset is randomly split 10 times in train and test and then the labels of the training set flipped according to the given noise rates ρ_{+1} and ρ_{-1} . The mean accuracies on the 10 test sets are computed.

3.2.1 Breast cancer dataset

We start by considering the first dataset which is the Breast cancer dataset made of 9 features and 263 rows. The train and set sizes are 186 and 77 respectively. We first start by considering symmetric noise (which is class independent) with four pairs of noise levels $(0.1, 0.1), (0.2, 0.2), (0.3, 0.3), (0.4, 0.4)$. Results are shown in Figure 3.1 and Table 3.1. We see that for this dataset the LPC approach is the one which is more robust to noise.

Let's then have a look at the case of a class conditional random noise. For instance let's consider as noise levels the pairs $(\rho_+, \rho_-) = (0.3, 0.1), (0.1, 0.3), (0.4, 0.2), (0.2, 0.4)$. Results are shown in Table 3.2.

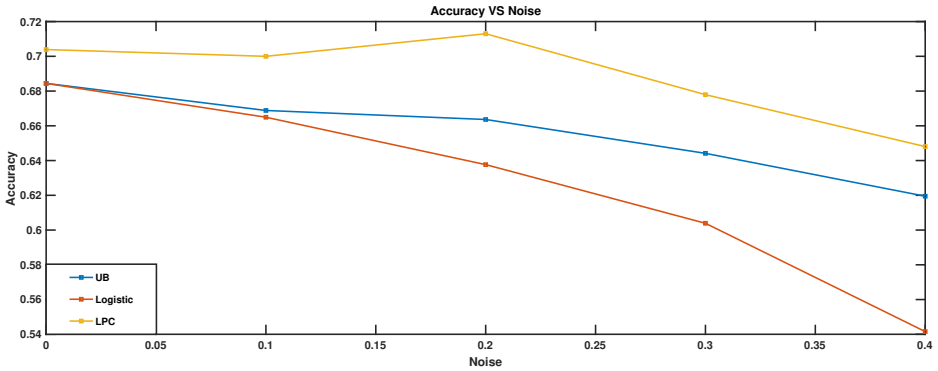


Figure 3.1: Comparison between LPC and the method of Unbiased Estimator(UB). The red curve is the profile of accuracy of the ERM classifier minimizing the logistic loss.

ρ	LPC	UB	Logistic
0	0.7039±0.0366	0.6840±0.0397	0.6840±0.0397
0.1	0.7000±0.0499	0.6690±0.0688	0.6650±0.0695
0.2	0.7130±0.0343	0.6650±0.0648	0.6400±0.0722
0.3	0.6779±0.0708	0.6450±0.0648	0.6050±0.1104
0.4	0.6481±0.1101	0.6200±0.1006	0.5450±0.1101

Table 3.1: Comparison between LPC and the method of Unbiased Estimator(UB). The standard deviation is also computed.

(ρ_+, ρ_-)	LPC	UB	Logistic
(0.3,0.1)	0.7026±0.0315	0.6710±0.0721	0.6888±0.0659
(0.1,0.3)	0.7026±0.0359	0.6590±0.0782	0.5722±0.1001
(0.4,0.2)	0.6896±0.0549	0.6429±0.0835	0.6552±0.0800
(0.2,0.4)	0.6818±0.0619	0.6449±0.0829	0.5268±0.1064

Table 3.2: Comparison between LPC and the method of Unbiased Estimator(UB). The standard deviation is also computed.

3.2.2 Heart dataset

We then consider the Heart dataset made of 13 features and 270 rows. The train and set sizes are 170 and 100 respectively. We first start by considering symmetric noise with four pairs of noise levels $(0.1, 0.1), (0.2, 0.2), (0.3, 0.3), (0.4, 0.4)$. Results are shown in Figure 3.3 and Table 3.3. We see that for this dataset the LPC approach is worst than UB but still is more robust to noise with respect the Logistic.

Let's then have a look at the case of a class conditional random noise. For instance let's consider as noise levels the pairs $(\rho_+, \rho_-) = (0.3, 0.1), (0.1, 0.3), (0.4, 0.2), (0.2, 0.4)$. Results are shown in Table 3.4.

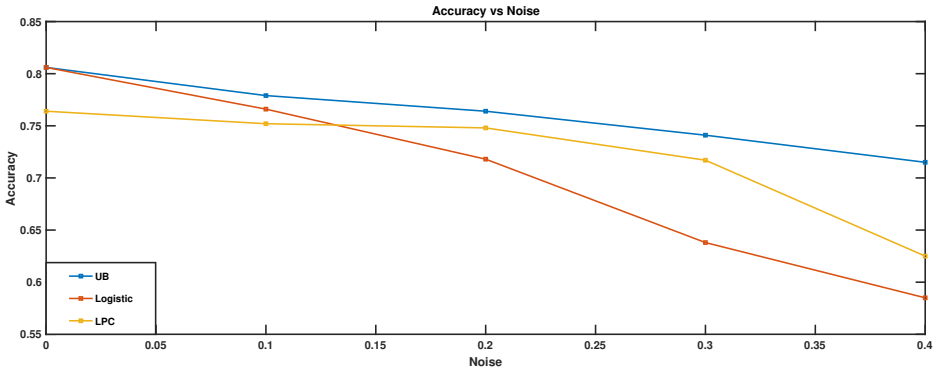


Figure 3.2: Comparison between LPC and the method of Unbiased Estimator(UB). The red curve is the profile of accuracy of the ERM classifier minimizing the logistic loss.

ρ	LPC	UB	Logistic
0	0.7640±0.0445	0.8060±0.0337	0.8060±0.0337
0.1	0.7520±0.0426	0.7790±0.0443	0.7660±0.0517
0.2	0.7480±0.0379	0.7640±0.0626	0.7180±0.0893
0.3	0.7170±0.0773	0.7410±0.0689	0.6380±0.0692
0.4	0.6250±0.0562	0.7150±0.0896	0.5850±0.0971

Table 3.3: Comparison between LPC and the method of Unbiased Estimator(UB). The standard deviation is also computed.

(ρ_+, ρ_-)	LPC	UB	Logistic
(0.3,0.1)	0.7390±0.0484	0.7700±0.0759	0.7248±0.0849
(0.1,0.3)	0.7620±0.0402	0.7770±0.0640	0.6959±0.0907
(0.4,0.2)	0.6910±0.0576	0.7417±0.0892	0.6584±0.1095
(0.2,0.4)	0.7010±0.0482	0.7424±0.0850	0.6315±0.1096

Table 3.4: Comparison between LPC and the method of Unbiased Estimator(UB). The standard deviation is also computed.

3.2.3 German dataset

Lastly let's consider the German dataset made of 20 features and 1000 rows. The train and set sizes are 700 and 300 respectively. We first start by considering symmetric noise with four pairs of noise levels $(0.1, 0.1), (0.2, 0.2), (0.3, 0.3), (0.4, 0.4)$. Results are shown in Figure 3.3 and Table 3.5. We see that for this dataset the LPC approach is better than UB for high level of noise.

Let's then have a look at the case of a class conditional random noise. For instance let's consider as noise levels the pairs $(\rho_+, \rho_-) = (0.3, 0.1), (0.1, 0.3), (0.4, 0.2), (0.2, 0.4)$. Results are shown in Table 3.6.

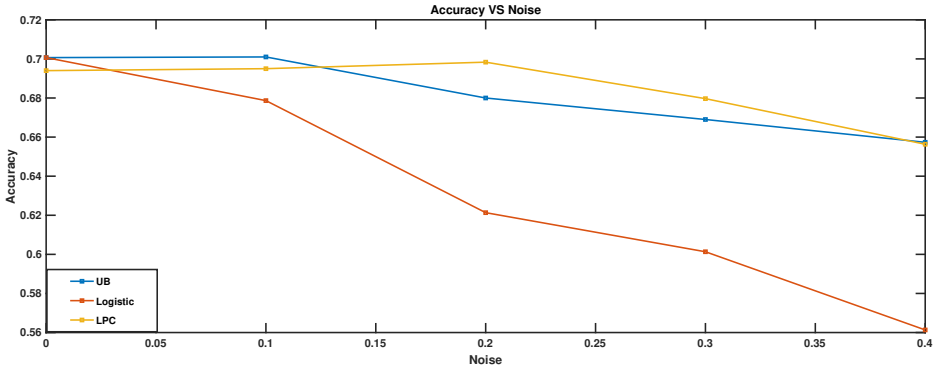


Figure 3.3: Comparison between LPC and the method of Unbiased Estimator(UB). The red curve is the profile of accuracy of the ERM classifier minimizing the logistic loss.

ρ	LPC	UB	Logistic
0	0.6940±0.0275	0.7007±0.0289	0.7007±0.0289
0.1	0.6950±0.0287	0.7010±0.0349	0.6787±0.0370
0.2	0.6983±0.0190	0.6800±0.0366	0.6213±0.0508
0.3	0.6797±0.0285	0.6690±0.0570	0.6013±0.0697
0.4	0.6563±0.0392	0.6573±0.0459	0.5613±0.0450

Table 3.5: Comparison between LPC and the method of Unbiased Estimator(UB). The standard deviation is also computed.

(ρ_+, ρ_-)	LPC	UB	Logistic
(0.3,0.1)	0.6987±0.0241	0.6880±0.0377	0.6877±0.0368
(0.1,0.3)	0.6923±0.0254	0.6937±0.0419	0.6047±0.0645
(0.4,0.2)	0.6787±0.0264	0.6867±0.0415	0.6673±0.0525
(0.2,0.4)	0.6910±0.0196	0.6790±0.0301	0.5170±0.0443

Table 3.6: Comparison between LPC and the method of Unbiased Estimator(UB). The standard deviation is also computed.

Conclusions

With this work we have presented a new approach for learning with noisy labels which is competitive with the current state of the art techniques. In particular we have shown how to build a robust classifier against noise by modifying the feature map, and a derivation of a generalization bound that quantifies how much is hard learning with a fixed pair of noise rates. We have also considered the heterogeneous noise case, in which different parts of the dataset get corrupted with different noise rates, and showed that semi-supervised setting is a particular case of this scenario when we have totally random labels for some portion of the dataset. Lastly, the presented approach updates the literature of learning with noisy labels, since it is able to deal with non reconstructible noise while current techniques cannot. Future work could consist in trying to *kernelize* the learning algorithm in order eliminate the dependence on the feature map.

Appendices

Appendix A

Generalization Bounds

Here we present some generalization bounds in the case when the hypothesis set $\mathcal{H}$ is finite or infinite. For the interested reader more details can be found in [Mohri et al. (2018)].

A.0.1 Generalization bounds for finite $\mathcal{H}$

Let's start by fixing a classification rule $h \in \mathcal{H}$, then, for any $\delta > 0$, the following inequality holds with probability at least $1 - \delta$

$$R(h) \leq \hat{R}(h) + \sqrt{\frac{\log\left(\frac{2}{\delta}\right)}{2n}}. \tag{A.1}$$

However this inequality is only true when h is fixed, and not for a h_S which is a random variable derived from a random sample S. In fact in this case also $R(h_S)$ is a random variable and of course is not true that the expectation of $\hat{R}(h_S)$ is equal to $R(h_S)$. Having notice that we need a bound that is true for all $h \in \mathcal{H}$ simultaneously. We have the following result that says that for any $\delta > 0$, with probability at least $1 - \delta$, the following inequality holds for every $h \in \mathcal{H}$

$$R(h) \leq \hat{R}(h) + \sqrt{\frac{\log|\mathcal{H}| + \log\frac{2}{\delta}}{2n}}. \tag{A.2}$$

Bound in (A.2) present a trade-off between the empirical error and size of $\mathcal{H}$. By enlarging the size of the set $\mathcal{H}$ we typically reduce the empirical error but we increase the second term of (A.2) which is called the complexity term and it is a function of $|\mathcal{H}|$ and n.

It is also possible to decompose the generalization error $R(h)$ for a classification rule $h \in \mathcal{H}$ in the following way

$$R(h) = (R(h) - R(h^*)) + (R(h^*) - R^*) + R^*, \tag{A.3}$$

where R^* is the Bayes Risk and $h^* = \underset{h \in H}{\text{arginf}} R(h)$ the best-in-class classifier. Let's give names to every term. The first difference in the right end side is called the

estimation error and measures the difference between the risk of the given classifier h and the smallest risk obtainable considering the set $\mathcal{H}$ which is achieved at h^*. It is a sort of measure of quality of the given classifier h with respect to the best possible one in $\mathcal{H}$. The second difference of (A.3) is called the approximation error and it is a property of the hypothesis set $\mathcal{H}$ measuring how far is the best possible risk within the class $\mathcal{H}$ from the best possible risk considering all the possibles $h : \mathcal{X} \to \mathcal{Y}$ which is the Bayes Risk.

A.0.2 Generalization bounds for infinite $\mathcal{H}$

Bound (A.2) in the case when $\mathcal{H}$ is infinite is not useful in practice, and we need a different way to build informative bounds for the generalization error $R(h)$. In this setting we can derive different bounds for $R(h)$ depending on which complexity term we may want to use. Essentially there are two classes of bounds when we deal with infinite $\mathcal{H}$: the ones defined in term of the Rademacher complexity and the ones defined by the notion of VC-dimension. let's start by looking at the first class of bounds by introducing the Rademacher complexity.

Rademacher complexity bounds

In order to simplify the notation let's associate to each h a function g that maps $(x, y) \in \mathcal{X} \times \mathcal{Y}$ to $l(h(x), y)$ with l fixed. For example, if $\mathcal{H}$ is the family of functions taking values in $\{-1, +1\}$, then G is the family of loss functions associated to $\mathcal{H}$ for the $0 - 1$ loss. In this way the generalization error will be $R(g) \doteq \mathbb{E}[g]$. Let's define the empirical Rademacher complexity of a class of functions G with respect to a sample S of size n as

$$\hat{\mathcal{R}}_S(G) = \mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{g \in G} \frac{1}{n} \sum_{i=1}^{n} \sigma_i g(x_i, y_i)\right] = \mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{g \in G} \frac{\boldsymbol{\sigma} \cdot \boldsymbol{g_S}}{n}\right], \tag{A.4}$$

where $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)^T$ are called Rademacher variables and are independent uniform random variable taking values in $\{-1, +1\}$, and $\boldsymbol{g}_S$ is the restriction of the function g to the given sample S. Notice that the empirical Rademacher complexity is sample dependent, it depends on the particular sample that has been drawing from the unknown distribution p. If we want a sample independent quantity we can take the expectation over all the possible samples of size n that are drawn according to p, that is

$$\mathcal{R}_n(G) = \mathbb{E}_{S \sim p}[\hat{\mathcal{R}}_S(G)]. \tag{A.5}$$

The quantity $\mathcal{R}_n(G)$ is called the Rademacher complexity of G and it is define as the expectation of the empirical Rademacher complexity over all samples of size n from p. Having introduced these two notions of complexity we can write two generalization bounds based on Rademacher complexity. We have that, for any

$\delta > 0$, with probability at least $1 - \delta$ these two bounds holds for all $g \in G$

$$\mathbb{E}[g(x,y)] \leq \frac{1}{n}\sum_{i=1}^{n} g(x_i, y_i) + 2\mathcal{R}_n(G) + \sqrt{\frac{\log \frac{1}{\delta}}{2n}} \tag{A.6}$$

$$\mathbb{E}[g(x,y)] \leq \frac{1}{n}\sum_{i=1}^{n} g(x_i, y_i) + 2\hat{\mathcal{R}}_S(G) + 3\sqrt{\frac{\log \frac{2}{\delta}}{2n}}. \tag{A.7}$$

In the case of binary classification and using as loss function the $0-1$ loss we can relate the Empirical Rademahcer complexities of G and $\mathcal{H}$ in the following way

$$\hat{\mathcal{R}}_S(G) = \frac{1}{2}\hat{\mathcal{R}}_{S_\mathcal{X}}(\mathcal{H}), \tag{A.8}$$

where $S_\mathcal{X} = (x_1, \ldots, x_n)$. Now we can state the Rademacher complexity bounds in term of the Rademacher complexity of $\mathcal{H}$ rather than in term of the one of G

$$R(h) \leq \hat{R}(h) + \mathcal{R}_n(\mathcal{H}) + \sqrt{\frac{\log \frac{1}{\delta}}{2n}} \tag{A.9}$$

$$R(h) \leq \hat{R}(h) + \hat{\mathcal{R}}_S(\mathcal{H}) + 3\sqrt{\frac{\log \frac{2}{\delta}}{2n}}. \tag{A.10}$$

An interesting observation is that the bound (A.10) is data-dependent since it is present the empirical Rademacher complexity. However computing the empirical Rademacher complexity might be so hard as an Empirical risk minimization problem.

VC-dimension bounds

These type of bounds make use of a different notion of complexity which is easier to compute than the Rademacher complexity. The price to pay will be that these new bounds are less tight than those with the Rademacher complexity. Let's start by defining the Growth function $\Pi_\mathcal{H}$ for a hypothesis set $\mathcal{H}$ as

$$\Pi_\mathcal{H}(n) = \max_{\{x_1,\ldots,x_n\}\in X} \left| \{(h(x_1), \ldots, h(x_n)) : h \in \mathcal{H}\} \right|, \tag{A.11}$$

which is the maximum number of distinct labelling of n points using the functions in $\mathcal{H}$. The notion of the Growth function is purely combinatorial and does not depend on the distribution of the data. It can be shown that we can use the Growth function $\Pi_\mathcal{H}$ to upper bound the Rademacher complexity as the following

$$\mathcal{R}_n(\mathcal{H}) \leq \sqrt{\frac{2\log \Pi_\mathcal{H}(n)}{n}}, \tag{A.12}$$

and so we can build the bound

$$R(h) \leq \hat{R}(h) + \sqrt{\frac{2\log \Pi_\mathcal{H}(n)}{n}} + \sqrt{\frac{\log \frac{1}{\delta}}{2n}}. \tag{A.13}$$

The computation of the Growth function $\Pi_{\mathcal{H}}$ however might not be so easy to do. In fact given n points and quantify in how many possible ways those points can be classified by some functions $h \in \mathcal{H}$ it is not obvious at all. For this reason usually is preferred to use rather than the Growth function a closely related quantity that is instead a scalar easier to deal with: the so called VC-dimension. Let $\mathcal{H}$ be a hypothesis set of functions on $\mathcal{X}$ that takes values in $\{-1, +1\}$ (Binary classification case). Given a set of n points $S = \{x_1, \dots, x_n\}$ we say that S is *shattered* by $\mathcal{H}$ if all the possible 2^n binary labellings of the points can be realized with the functions in $\mathcal{H}$. Then the VC-dimension is defined as the cardinality of largest set of points that can be shattered by $\mathcal{H}$. Formally it is defined in terms of the growth function as

$$\mathrm{VCdim}(\mathcal{H}) = \max\{n : \Pi_{\mathcal{H}}(n) = 2^n\}. \tag{A.14}$$

If an arbitrarily large set of points in $\mathcal{X}$ can be shattered by $\mathcal{H}$ we say that $\mathrm{VCdim}(\mathcal{H}) = \infty$. Using Sauer's lemma we can upper bound the Growth function $\Pi_{\mathcal{H}}(n)$ with an expression that contains the VC-dimension of $\mathcal{H}$. Let $\mathrm{VCdim}(\mathcal{H}) = d$ and for all $n \in \mathbb{N}$ the following inequality holds

$$\Pi_{\mathcal{H}}(n) \leq \sum_{i=0}^{d} \binom{n}{i} \leq \left(\frac{n}{d}\right)^d e^d, \tag{A.15}$$

which gives the following bound based on the VC-dimension, where $\mathcal{H}$ is a hypothesis set with VC-dimension equal to d. Then, for any $\delta > 0$, the following holds with probability $1 - \delta$ for all $h \in \mathcal{H}$

$$R(h) \leq \hat{R}(h) + \sqrt{\frac{2d \log \frac{en}{d}}{n}} + \sqrt{\frac{\log \frac{1}{\delta}}{2n}}. \tag{A.16}$$

A.1 Empirical Risk Minimization

Empirical Risk Minimization (ERM) is by far the most used approach for approximating (1.1). The idea is to use the training data and it consists in minimizing the sample average of the loss incurred at the observed samples:

$$\min_{h \in H} \frac{1}{n} \sum_{i=1}^{n} l(h, (x_i, y_i)), \tag{A.17}$$

which more formally can be written in terms of the empirical distribution p_e as:

$$\min_{h \in H} \mathbb{E}_{p_e} l(h, (x, y)). \tag{A.18}$$

Suppose that h_S is a classifier obtained by solving the empirical minimization problem which by definition means $\hat{R}(h_S) \leq \hat{R}(h),\ \forall h \in \mathcal{H}$. Then we can bound

the estimation error of h_S in terms of the generalization bound (A.2)

$$R(h_S) - R(h^*) = R(h_S) - \hat{R}(h_S) + \hat{R}(h_S) - R(h^*) \tag{A.19}$$

$$\leq R(h_S) - \hat{R}(h_S) + \hat{R}(h^*) - R(h^*) \tag{A.20}$$

$$\leq 2 \sup_{h \in \mathcal{H}} |R(h) - \hat{R}(h)|. \tag{A.21}$$

Now the right end side of (A.21) can be bounded by (A.2). The theoretical problem of Empirical Risk Minimization is that it minimizes $\hat{R}(h)$ disregarding the complexity term in (A.2). From a practical point of view instead, problem (A.17) is tractable only if the objective function is convex, which is not the case for the $0-1$ loss. What is usually done is to replace the $0-1$ loss with a convex loss function such as the logistic loss, the Hinge loss or the quadratic loss.

Appendix B

Linear Probabilistic Classifier

In this appendix more details of LPC are presented. In particular it is justified where (2.12) comes from and why a LPC is constructed as (2.18). More details regarding LPC can be found in [Mazuelas et al. (2019)], while details regarding Lagrange Duality in [Boyd & Vandenberghe (2004)] and [Calafiore & El Ghaoui (2014)].

B.1 Robust Risk Minimization

Robust Risk Minimization is another way to approximate (1.1). Since we don't know p in (1.1), we would like to derive from the training data $(x_1, y_1) \dots (x_n, y_n)$ an uncertainty set $\mathcal{U}$ made of probability distributions, that contains the true distribution p with high probability.

The optimization problem we want to solve is:

$$\min_{h \in \mathcal{H}} \max_{p \in \mathcal{U}} \mathbb{E}_p l(h, (x, y)). \tag{B.1}$$

The idea is to find a classification rule h that minimizes the worst case expected loss incurred over the uncertainty set $\mathcal{U}$ of probability distributions.

The Linear Probabilistic Classifier (LPC) is a classifier built using RRM. LPC is derived by solving problem (B.1) where the loss considered is the $0-1$ loss, the classification rule h is unconstrained, and the uncertainty set $\mathcal{U}$ is defined in terms of inequalities that constrain the expected value of a function called feature map.

B.2 Polyhedral uncertainty sets

In the minimax problem (B.1), the *max* part is done over an uncertainty set $\mathcal{U}$ whose elements are probability distributions. In the LPC framework we define the uncertainty set in the following way:

$$\mathcal{U}_{\Phi}^{a,b} = \{p \in \Delta(\mathcal{X} \times \mathcal{Y}) : a \preceq \mathbb{E}_p\{\Phi(x, y)\} \preceq b\}, \tag{B.2}$$

where $\Phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^m$ is a function, called the feature map, and $a, b \in \mathbb{R}^m$ are given vectors such that $a \preceq b$. In words, the uncertainty set $\mathcal{U}_\Phi^{a,b}$ contains probability distributions such that the mean of Φ with respect to those distributions is between a and b. Note, that uncertainty sets like (B.2) are subset $\Delta(\mathcal{X} \times \mathcal{Y}) \subset \mathbb{R}^{|\mathcal{X}||\mathcal{Y}|}$. Suppose now that $(x_1, y_1), \ldots, (x_n, y_n)$ are i.i.d. samples from p, i.e., $(X, Y) \sim p$; we would like to solve problem (B.1) over an uncertainty set $\mathcal{U}_\Phi^{a,b}$ that contains p with high probability.

B.3 The minimax problem

Let's see now how to solve the minimax problem

$$\min_{h \in \Delta(\mathcal{X},\mathcal{Y})} \max_{p \in \mathcal{U}_\Phi^{a,b}} \mathbb{E}_p l(h, (x, y)), \tag{B.3}$$

where $\mathcal{U}_\Phi^{a,b}$ is built as explained in section B.2. Another difference with respect to (B.1) is that here we don't restrict the classification rule h to belong to some hypothesis space $\mathcal{H}$, and we allow for classification rules that randomly classify each feature vector x. That is, $h : \mathcal{X} \to \Delta(\mathcal{Y})$ and $\Delta(\mathcal{X}, \mathcal{Y})$ is the set of these probabilistic transformations. It is useful to think the function h as a function from $\mathcal{X} \times \mathcal{Y}$ to $\mathbb{R}$, or equivalently as a row vector $\boldsymbol{h}^{\boldsymbol{T}}$ of dimension $1 \times |\mathcal{X}||\mathcal{Y}|$, where for every x the subvector $\boldsymbol{h}_{\boldsymbol{x}}^{\boldsymbol{T}}$, of dimension $1 \times |\mathcal{Y}|$, is a probability distribution, i.e., $\boldsymbol{h}_{\boldsymbol{x}}^{\boldsymbol{T}} \succeq 0$ and $\boldsymbol{h}_{\boldsymbol{x}}^{\boldsymbol{T}} \mathbb{1} = 1$. In this way a classification rule h obtained will classify a feature vector x with label y with probability $h(x, y)$.

B.3.1 Solving the minimax problem with Lagrange Duality

In order to solve problem (B.3) first notice that since $\Delta(\mathcal{X}, \mathcal{Y})$ and $\mathcal{U}_\Phi^{a,b}$ are closed and convex sets [Grünwald et al. (2004)] we have the following equality[11]

$$\min_{h \in \Delta(\mathcal{X},\mathcal{Y})} \max_{p \in \mathcal{U}_\Phi^{a,b}} \mathbb{E}_p l(h, (x, y)) = \max_{p \in \mathcal{U}_\Phi^{a,b}} \min_{h \in \Delta(\mathcal{X},\mathcal{Y})} \mathbb{E}_p l(h, (x, y)). \tag{B.4}$$

We will use the notation $l(h, p)$ for $\mathbb{E}_p l(h, (x, y))$. If we now concentrate on the maximin problem, the inner *min* part is very easy to solve. First, the loss function l is the $0 - 1$ loss and the loss incurred at a (x, y) pair is just the probability of classifying it wrongly

$$l(h, (x, y)) = 1 - h(x, y). \tag{B.5}$$

[11] The equality should be meant as an equality of the values of the problems at the optimum. It is not an equality between problems.

Then we can write explicitly the *min* part as

$$\min_{h\in\Delta(\mathcal{X},\mathcal{Y})} \sum_{x\in\mathcal{X},y\in\mathcal{Y}} p(x,y)(1-h(x,y)) \tag{B.6}$$

$$= \min_{h\in\Delta(\mathcal{X},\mathcal{Y})} 1-\boldsymbol{p}^{\boldsymbol{T}}\boldsymbol{h} \tag{B.7}$$

$$= 1-\max_{h\in\Delta(\mathcal{X},\mathcal{Y})} \boldsymbol{p}^{\boldsymbol{T}}\boldsymbol{h} \tag{B.8}$$

$$= 1-\|\boldsymbol{p}\|_{\infty,1}. \tag{B.9}$$

At the end the maxmin problem in (B.4) reduces[12] to

$$1-\min_{p\in\mathcal{U}_{\Phi}^{a,b}} \|\boldsymbol{p}\|_{\infty,1}. \tag{B.10}$$

Let's explicitly write the constraint $p\in\mathcal{U}_{\Phi}^{a,b}$, and concentrate on the min part

$$\begin{aligned} \underset{\boldsymbol{p}}{\text{minimize}} \quad & \|\boldsymbol{p}\|_{\infty,1}+I^{+}(\boldsymbol{p}) \\ \text{subject to} \quad & \boldsymbol{p}^{\boldsymbol{T}}\mathbf{1}=1, \\ & \boldsymbol{a}\preceq\boldsymbol{\Phi}\boldsymbol{p}\preceq\boldsymbol{b}. \end{aligned} \tag{B.11}$$

We use the the function

$$I^{+}(\boldsymbol{p}) = \begin{cases} 0, & \text{if } \boldsymbol{p}\succeq\mathbf{0} \\ \infty, & \text{otherwise} \end{cases} \tag{B.12}$$

in order to express the constraint $\boldsymbol{p}\succeq\mathbf{0}$. In this way we avoid to introduce too many Lagrangian multipliers when writing the dual problem of (B.10). In fact, notice that $\boldsymbol{p}$ is a very high dimensional vector ($|\mathcal{X}||\mathcal{Y}|\times 1$).

Let's write the Lagrangian function associated to the problem (B.11)

$$\mathcal{L}(\boldsymbol{p},\boldsymbol{\alpha},\boldsymbol{\beta},\gamma) = \|\boldsymbol{p}\|_{\infty,1}+I^{+}(\boldsymbol{p})+\boldsymbol{\alpha}^{\boldsymbol{T}}(\boldsymbol{a}-\boldsymbol{\Phi}\boldsymbol{p})+\boldsymbol{\beta}^{\boldsymbol{T}}(\boldsymbol{\Phi}\boldsymbol{p}-\boldsymbol{b})+\gamma(1-\boldsymbol{p}^{\boldsymbol{T}}\mathbf{1}), \tag{B.13}$$

and the associated dual function $g(\boldsymbol{\alpha},\boldsymbol{\beta},\gamma)=\inf_{\boldsymbol{p}}\mathcal{L}(\boldsymbol{p},\boldsymbol{\alpha},\boldsymbol{\beta},\gamma)$. The dual function $g(\boldsymbol{\alpha},\boldsymbol{\beta},\gamma)$ can be explicitly written as

$$g(\boldsymbol{\alpha},\boldsymbol{\beta},\gamma) = \boldsymbol{\alpha}^{\boldsymbol{T}}\boldsymbol{a}-\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{b}+\gamma+\inf_{\boldsymbol{p}}\Big(\|\boldsymbol{p}\|_{\infty,1}+I^{+}(\boldsymbol{p})+\boldsymbol{p}^{\boldsymbol{T}}(\boldsymbol{\Phi}^{\boldsymbol{T}}(\boldsymbol{\beta}-\boldsymbol{\alpha})-\mathbf{1}\gamma)\Big). \tag{B.14}$$

Notice that

$$\inf_{\boldsymbol{p}}\Big(\|\boldsymbol{p}\|_{\infty,1}+I^{+}(\boldsymbol{p})+\boldsymbol{p}^{\boldsymbol{T}}(\boldsymbol{\Phi}^{\boldsymbol{T}}(\boldsymbol{\beta}-\boldsymbol{\alpha})-\mathbf{1}\gamma)\Big) \tag{B.15}$$

$$= -\sup_{\boldsymbol{p}}\Big(-(\|\boldsymbol{p}\|_{\infty,1}+I^{+}(\boldsymbol{p}))+\boldsymbol{p}^{\boldsymbol{T}}(\boldsymbol{\Phi}^{\boldsymbol{T}}(\boldsymbol{\alpha}-\boldsymbol{\beta})+\mathbf{1}\gamma)\Big) \tag{B.16}$$

$$= -f^{*}(\boldsymbol{\Phi}^{\boldsymbol{T}}(\boldsymbol{\alpha}-\boldsymbol{\beta})+\mathbf{1}\gamma), \tag{B.17}$$

[12] In this case it is corrected to say that the maximin problem is the same problem B.10

where f^* is the conjugate function of $f(\boldsymbol{p}) = \|\boldsymbol{p}\|_{\infty,1}+I^+(\boldsymbol{p})$. Now we can write the Dual optimization problem as

$$\begin{aligned} \underset{\gamma,\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{maximize}} \quad & \boldsymbol{\alpha}^{\boldsymbol{T}}\boldsymbol{a} - \boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{b} + \gamma - f^*(\boldsymbol{\Phi}^{\boldsymbol{T}}(\boldsymbol{\alpha}-\boldsymbol{\beta}) + \mathbf{1}\gamma) \\ \text{subject to} \quad & \boldsymbol{\alpha} \succeq \mathbf{0}, \\ & \boldsymbol{\beta} \succeq \mathbf{0}. \end{aligned} \tag{B.18}$$

Problem (B.18) is the dual problem of (B.11), we can rewrite the dual in its ultimate form by notice that

$$f^*(\boldsymbol{\Phi}^{\boldsymbol{T}}(\boldsymbol{\alpha}-\boldsymbol{\beta}) + \mathbf{1}\gamma) = \begin{cases} 0, & \text{if } \|(\boldsymbol{\Phi}^{\boldsymbol{T}}(\boldsymbol{\alpha}-\boldsymbol{\beta}) + \mathbf{1}\gamma)^+\|_{1,\infty} \leq 1 \\ \infty, & \text{otherwise} \end{cases}, \tag{B.19}$$

and so the final form of the dual problem is

$$\begin{aligned} \underset{\gamma,\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{maximize}} \quad & \boldsymbol{\alpha}^{\boldsymbol{T}}\boldsymbol{a} - \boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{b} + \gamma \\ \text{subject to} \quad & \|(\boldsymbol{\Phi}^{\boldsymbol{T}}(\boldsymbol{\alpha}-\boldsymbol{\beta}) + \mathbf{1}\gamma)^+\|_{1,\infty} \leq 1, \\ & \boldsymbol{\alpha} \succeq \mathbf{0}, \\ & \boldsymbol{\beta} \succeq \mathbf{0}. \end{aligned} \tag{B.20}$$

Slater's condition holds since in problem (B.11) the constraints are affine. This means that there is zero duality gap between (B.11) and (B.20), that is strong duality holds. Let p^* a primal optimal point and $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \gamma^*)$ be a dual optimal point, strong duality then implies that

$$\|\boldsymbol{p}^*\|_{\infty,1}+I^+(\boldsymbol{p}^*) = \boldsymbol{a}^{\boldsymbol{T}}\,\boldsymbol{\alpha}^* - \boldsymbol{b}^{\boldsymbol{T}}\,\boldsymbol{\beta}^* + \gamma^* = \inf_{\boldsymbol{p}} \mathcal{L}(\boldsymbol{p}, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \gamma^*), \tag{B.21}$$

where the first equality is due to strong duality and the second one is just the definition of the dual function. We also notice that $\boldsymbol{p}^*$ is a minimizer[13] of $\mathcal{L}(\boldsymbol{p}, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \gamma^*)$ by the fact that

$$\inf_{\boldsymbol{p}} \mathcal{L}(\boldsymbol{p}, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \gamma^*) \leq \mathcal{L}(\boldsymbol{p}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \gamma^*) \leq \|\boldsymbol{p}^*\|_{\infty,1}+I^+(\boldsymbol{p}^*), \tag{B.22}$$

where the first inequality follows since the infimum of the Lagrangian over p is less than or equal to its value at $p = p^*$, and the second inequality follows from the fact that $\boldsymbol{\alpha}^* \succeq \mathbf{0}$, $\boldsymbol{\beta}^* \succeq \mathbf{0}$, $\boldsymbol{a} - \boldsymbol{\Phi}\boldsymbol{p}^* \preceq 0$ and $\boldsymbol{\Phi}\,\boldsymbol{p}^* - \boldsymbol{b} \preceq 0$. We see then, by comparing (B.21) and (B.22) that all the inequalities hold in reality with equality. The important point is that a solution of (B.11) is also a solution of $\inf_{\boldsymbol{p}} \mathcal{L}(\boldsymbol{p}, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \gamma^*)$.

Now let's define the following uncertainty set

$$\tilde{\mathcal{U}} = \{p : \mathcal{X} \times \mathcal{Y} \to \mathbb{R} : \boldsymbol{p} \succeq 0, \|\boldsymbol{p}\|_{1,\infty} \leq 1\}, \tag{B.23}$$

[13] $\boldsymbol{p}^*$ is simply a minimizer, $\mathcal{L}(\boldsymbol{p}, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \gamma^*)$can also have others.

which allows to write

$$\inf_{\boldsymbol{p}} \|\boldsymbol{p}\|_{\infty,1} + I^{+}(\boldsymbol{p}) + \boldsymbol{\alpha}^{*\boldsymbol{T}}(\boldsymbol{a} - \boldsymbol{\Phi p}) + \boldsymbol{\beta}^{*\boldsymbol{T}}(\boldsymbol{\Phi p} - \boldsymbol{b}) + \gamma^{*}(1 - \boldsymbol{p}^{\boldsymbol{T}}\mathbf{1}) \tag{B.24}$$

as[14]

$$\inf_{\boldsymbol{p}\in\tilde{\mathcal{U}}} \|\boldsymbol{p}\|_{\infty,1} + \boldsymbol{\alpha}^{*\boldsymbol{T}}(\boldsymbol{a} - \boldsymbol{\Phi p}) + \boldsymbol{\beta}^{*\boldsymbol{T}}(\boldsymbol{\Phi p} - \boldsymbol{b}) + \gamma^{*}(1 - \boldsymbol{p}^{\boldsymbol{T}}\mathbf{1}), \tag{B.25}$$

where (B.24) is just

$$\inf_{\boldsymbol{p}} \mathcal{L}(\boldsymbol{p}, \boldsymbol{\alpha}^{*}, \boldsymbol{\beta}^{*}, \gamma^{*}).$$

Then from (B.21), (B.22) and (B.25) we get[15]

$$1 - \min_{\boldsymbol{p}\in\mathcal{U}_{\Phi}^{a,b}} \|\boldsymbol{p}\|_{\infty,1} = 1 + \max_{\boldsymbol{p}\in\tilde{\mathcal{U}}} -\|\boldsymbol{p}\|_{\infty,1} - \boldsymbol{\alpha}^{*\boldsymbol{T}}(\boldsymbol{a} - \boldsymbol{\Phi p}) - \boldsymbol{\beta}^{*\boldsymbol{T}}(\boldsymbol{\Phi p} - \boldsymbol{b}) - \gamma^{*}(1 - \boldsymbol{p}^{\boldsymbol{T}}\mathbf{1}) \tag{B.26}$$

$$= \max_{\boldsymbol{p}\in\tilde{\mathcal{U}}} \min_{h\in\Delta(\mathcal{X},\mathcal{Y})} l(h, \boldsymbol{p}) - \boldsymbol{\alpha}^{*\boldsymbol{T}}(\boldsymbol{a} - \boldsymbol{\Phi p}) - \boldsymbol{\beta}^{*\boldsymbol{T}}(\boldsymbol{\Phi p} - \boldsymbol{b}) - \gamma^{*}(1 - \boldsymbol{p}^{\boldsymbol{T}}\mathbf{1}) \tag{B.27}$$

$$= \max_{\boldsymbol{p}\in\tilde{\mathcal{U}}} \min_{h\in\Delta(\mathcal{X},\mathcal{Y})} \tilde{l}(h, \boldsymbol{p}), \tag{B.28}$$

$$= \min_{h\in\Delta(\mathcal{X},\mathcal{Y})} \max_{\boldsymbol{p}\in\tilde{\mathcal{U}}} \tilde{l}(h, \boldsymbol{p}). \tag{B.29}$$

Recalling (B.4) we have

$$\min_{h\in\Delta(\mathcal{X},\mathcal{Y})} \max_{p\in\mathcal{U}_{\Phi}^{a,b}} l(h, p) = \min_{h\in\Delta(\mathcal{X},\mathcal{Y})} \max_{\boldsymbol{p}\in\tilde{\mathcal{U}}} \tilde{l}(h, \boldsymbol{p}). \tag{B.30}$$

It can be shown that the right end side admits a solution of this type

$$\boldsymbol{h}^{*} \succeq \boldsymbol{\Phi}^{\boldsymbol{T}}(\boldsymbol{\alpha}^{*} - \boldsymbol{\beta}^{*}) + \mathbf{1}\boldsymbol{\gamma}^{*}, \tag{B.31}$$

and we want to show that this solution is also a solution of the left end side of (B.30). We have that

$$\min_{h\in\Delta(\mathcal{X},\mathcal{Y})} \max_{\boldsymbol{p}\in\tilde{\mathcal{U}}} \tilde{l}(h, \boldsymbol{p}) = \max_{\boldsymbol{p}\in\tilde{\mathcal{U}}} \tilde{l}(h^{*}, \boldsymbol{p}) \geq \max_{p\in\mathcal{U}_{\Phi}^{a,b}} l(h^{*}, p) \geq \min_{h\in\Delta(\mathcal{X},\mathcal{Y})} \max_{p\in\mathcal{U}_{\Phi}^{a,b}} l(h, p), \tag{B.32}$$

but due to (B.30) inequalities in (B.32) are equalities, and h^{*} is a solution of

$$\min_{h\in\Delta(\mathcal{X},\mathcal{Y})} \max_{p\in\mathcal{U}_{\Phi}^{a,b}} l(h, p).$$

[14](B.24) and (B.25) are the same problems. So a solution of (B.11) is also a solution for (B.25).

[15]where $\tilde{l}(h, \boldsymbol{p}) = l(h, \boldsymbol{p}) - \boldsymbol{\alpha}^{*\boldsymbol{T}}(\boldsymbol{a} - \boldsymbol{\Phi p}) - \boldsymbol{\beta}^{*\boldsymbol{T}}(\boldsymbol{\Phi p} - \boldsymbol{b}) - \gamma^{*}(1 - \boldsymbol{p}^{\boldsymbol{T}}\mathbf{1})$.

It is important to appreciate that a classification rule satisfying (B.31) can be obtained by solving the dual problem (B.20) and enforcing that $\boldsymbol{h}_{\boldsymbol{x}}^{\boldsymbol{T}}\mathbf{1} = 1$. That is, $\forall(x, y) \in \mathcal{X} \times \mathcal{Y}$

$$h^{a,b}(x, y) = (\boldsymbol{\Phi}(x, y)^T(\boldsymbol{\alpha}^* - \boldsymbol{\beta}^*) + \gamma^*)^+ + \frac{1 - \|(\boldsymbol{\Phi}_{\boldsymbol{x}}^{\boldsymbol{T}}(\boldsymbol{\alpha}^* - \boldsymbol{\beta}^*) + \mathbf{1}\gamma^*)^+\|_1}{|\mathcal{Y}|}. \quad \text{(B.33)}$$

We refer to (B.33) as the Linear Probabilistic Classifier. The learning stage consists in solving the convex problem (B.20), whereas the prediction stage consists, given an input x, in generating a y according to h_x given by (B.33).

Bibliography

Angluin, D., & Laird, P. (1988). Learning from noisy examples. *Machine Learning*, *2*(4), 343–370.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Calafiore, G. C., & El Ghaoui, L. (2014). *Optimization models*. Cambridge university press.

Farnia, F., & Tse, D. (2016). A minimax approach to supervised learning. In *Advances in neural information processing systems* (pp. 4240–4248).

Grant, M., & Boyd, S. (2008). Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, & H. Kimura (Eds.), *Recent advances in learning and control* (pp. 95–110). Springer-Verlag Limited. (`http://stanford.edu/~boyd/graph_dcp.html`)

Grant, M., & Boyd, S. (2014, March). *CVX: Matlab software for disciplined convex programming, version 2.1.* `http://cvxr.com/cvx`.

Grünwald, P. D., Dawid, A. P., et al. (2004). Game theory, maximum entropy, minimum discrepancy and robust bayesian decision theory. *the Annals of Statistics*, *32*(4), 1367–1433.

Hastie, T., Tibshirani, R., Friedman, J., & Franklin, J. (2005). The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, *27*(2), 83–85.

Kearns, M. (1998). Efficient noise-tolerant learning from statistical queries. *Journal of the ACM (JACM)*, *45*(6), 983–1006.

Lanckriet, G. R., Ghaoui, L. E., Bhattacharyya, C., & Jordan, M. I. (2002). A robust minimax approach to classification. *Journal of Machine Learning Research*, *3*(Dec), 555–582.

Liu, T., & Tao, D. (2015). Classification with noisy labels by importance reweighting. *IEEE Transactions on pattern analysis and machine intelligence*, *38*(3), 447–461.

Luenberger, D. G. (1997). *Optimization by vector space methods*. John Wiley & Sons.

Mazuelas, S., & Pérez, A. (2019). General supervision via probabilistic transformations. *arXiv preprint arXiv:1901.08552*.

Mazuelas, S., Zanoni, A., & Perez, A. (2019). Supervised classification via minimax probabilistic transformations. *arXiv preprint arXiv:1902.00693*.

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). *Foundations of machine learning*. MIT press.

Natarajan, N., Dhillon, I. S., Ravikumar, P. K., & Tewari, A. (2013). Learning with noisy labels. In *Advances in neural information processing systems* (pp. 1196–1204).

Van Rooyen, B., & Williamson, R. C. (2017). A theory of learning with corrupted labels. *Journal of Machine Learning Research*, *18*, 228–1.

Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE transactions on neural networks*, *10*(5), 988–999.

Wainwright, M. J. (2019). *High-dimensional statistics: A non-asymptotic viewpoint* (Vol. 48). Cambridge University Press.

Zhu, X., & Goldberg, A. B. (2009). Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, *3*(1), 1–130.