Boston University

OpenBU **http://open.bu.edu**

Theses & Dissertations Boston University Theses & Dissertations

2017

The differential geometric structure in supervised learning of classifiers

https://hdl.handle.net/2144/22449

Boston University

BOSTON UNIVERSITY

GRADUATE SCHOOL OF ARTS AND SCIENCES

Dissertation

THE DIFFERENTIAL GEOMETRIC STRUCTURE IN SUPERVISED LEARNING OF CLASSIFIERS

by

QINXUN BAI

B.E., Tsinghua University, China, 2006
M.S., Chinese Academy of Sciences, China, 2010

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2017

© Copyright by
QINXUN BAI
2017

Approved by

First Reader	Stan Sclaroff, PhD Professor of Computer Science
Second Reader	Steven Rosenberg, PhD Professor of Mathematics and Statistics
Third Reader	Suvrit Sra, PhD Principal Research Scientist Laboratory for Information & Decision Systems Massachusetts Institute of Technology

Acknowledgments

My acknowledgments must begin with my major advisor Prof. Stan Sclaroff and thesis co-advisor Prof. Steven Rosenberg. I can write another thesis on the guidance and support I have received from both of them, but I could never say enough how much I appreciate for what they have done for me. Stan's influence on my PhD study and future career is tremendous and invaluable. It is about almost every aspect a graduate student should grow mature, including but not limited to approaching problems, validating new ideas, presenting research work orally and verbally, managing multiple projects, collaborating with colleagues, and being a constructive member of the community. Another amazing thing about Stan is that he carefully maintains a perfect balance between concrete guidance and encouragement of self-motivation, which inspires me to be the best of myself. Steve has deeply shaped my way of thinking during the passing four years, bringing me to a wonderland full of intellectual pleasure and geometric insights that I could never imagine on my own. In fact, the maths I have learned from him surpasses the total amount I have ever obtained elsewhere. Beyond research, Steve has also been an amazing friend in my life.

Prof. Henry Lam has also provided precious guidance and influenced my way of thinking during our two years' collaboration on another project. I have greatly benefited from his statistical perspective of approaching problems and technical excellence in characterizing things rigorously.

I also owe a lot to other members of my thesis committee. Prof. Margrit Betke has been helping me and supporting me at every milestone of my PhD study. Dr. Suvrit Sra has been giving very constructive advices since the thesis proposal. Prof. Lorenzo Orecchia has provided precious help on some algorithm aspects of this thesis.

There are a number of other colleagues in this community who have helped me

with their insights and expertise. Dr. Sebastian Nowozin and Dr. Daniel Tarlow have closely advised me and shown me the “hypothesis driven” way of approaching research problems during my internship at Microsoft Research Cambridge. Prof. Shivani Agarwal has explained thoroughly some concepts and principles in statistical learning that have puzzled me for a long time, and provided very helpful feedback on a submission related to this thesis. Dr. Katia Oleinik has patiently tutored me for one month, during which I acquired all my GPU programming skills. Camille Monnier has been very patient and supportive during our academia-industry collaboration.

I would also like to thank many fellow labmates for all the inspiring discussions and great memories. Special thanks to Dr. Zheng Wu for closely collaborating with me and getting us through some challenging experiments with his insights and application skills. With Kun, Shugao, Jianming, and Fatih, I have benefited so much from our frequent discussions of course study and research problems, sharing of insights, and preparation for the written exam together. Diane, Danna, Sarah, and Mikhail have always been so encouraging and helpful whenever I have to prepare presentations for conferences and interviews. I thank Prof. Erik Learned-Miller, Zhenyu, Wenxin, Mehrnoosh, Ajjen and Andrew for those interesting discussions in research and life. I am also grateful to some senior alumni of the IVC group. I have learned a lot from Tai-Peng, Ashwin, Vitaly and John during my first two years of PhD study. Liliana has always been so warm and supportive, those Italian courses she gave Kun and me are definitely some of the best memories of my PhD life.

Last but not the least, I must give my deepest appreciation to my wonderful parents Ding and Qiaomei for their consistent support and understanding, being their son is the most fortunate thing I can think of in this world.

THE DIFFERENTIAL GEOMETRIC STRUCTURE IN SUPERVISED LEARNING OF CLASSIFIERS

QINXUN BAI

Boston University, Graduate School of Arts and Sciences, 2017

Major Professor: Stan Sclaroff, Professor of Computer Science

Co-advisor: Steven Rosenberg, Professor of Mathematics

ABSTRACT

In this thesis, we study the overfitting problem in supervised learning of classifiers from a geometric perspective. As with many inverse problems, learning a classification function from a given set of example-label pairs is an ill-posed problem, i.e., there exist infinitely many classification functions that can correctly predict the class labels for all training examples. Among them, according to Occam's razor, simpler functions are favored since they are less overfitted to training examples and are therefore expected to perform better on unseen examples. The standard technique to enforce Occam's razor is to introduce a regularization scheme, which penalizes some type of complexity of the learned classification function. Some widely used regularization techniques are functional norm-based (Tikhonov) techniques, ensemble-based techniques, early stopping techniques, etc. However, there is important geometric information in the learned classification function that is closely related to overfitting, and has been overlooked by previous methods. In this thesis, we study the complexity of a classification function from a new geometric perspective. In particular, we investigate the differential geometric structure in the submanifold corresponding to the estimator of the class probability $P(y|\boldsymbol{x})$, based on the observation that overfitting produces rapid local oscillations and hence large mean curvature of this submanifold. We also show that our geometric perspective of supervised learning is naturally related to an elastic model in physics, where our complexity measure is a high dimensional extension of the surface energy in physics. This study leads to a new geometric regularization approach for supervised learning of classifiers. In our approach, the learning process can be viewed as a submanifold fitting problem that is solved by a mean curvature flow method. In particular, our approach finds the submanifold by iteratively fitting the training examples in a curvature or volume decreasing manner. Our technique is unified for both binary and multiclass classification, and can be applied to regularize any classification function that satisfies two requirements: firstly, an estimator of the class probability can be obtained; secondly, first and second derivatives of the class probability estimator can be calculated. For applications, where we apply our regularization technique to standard loss functions for classification, our RBF-based implementation compares favorably to widely used regularization methods for both binary and multiclass classification. We also design a specific algorithm to incorporate our regularization technique into the standard forward-backward training of deep neural networks. For theoretical analysis, we establish Bayes consistency for a specific loss function under some mild initialization assumptions. We also discuss the extension of our approach to situations where the input space is a submanifold, rather than a Euclidean space.

Contents

1 Introduction 1

1.1 Problem Definition 2

1.2 Main Idea 3

1.3 Contributions 6

1.4 Roadmap of Thesis 6

1.5 Related Paper 9

2 Related Work 10

2.1 Supervised Learning of Classifiers 10

2.2 Regularization Techniques 11

2.2.1 Geometric Regularization 12

2.2.2 Sobolev Regularization 12

2.3 Manifold Learning 13

2.4 Variational Problems and Gradient Flow 14

2.5 Theory of Minimal Surfaces 15

2.6 Training in Neural Networks Robust to Adversarial Examples 15

3 Geometric Perspective of Supervised Learning 18

3.1 Insight from Statistical Learning and Optimization 18

3.1.1 Bayesian Perspective and Occam's Razor 18

3.1.2 Optimization and Statistical Learning 20

3.2 A Physical Model for Supervised Learning 23
3.3 Geometric Foundation 27
  3.3.1 Geometry of $\mathcal{M}$ 28
  3.3.2 Gradient Flow on $\mathcal{M}$ 29
3.4 Summary 31

4 Differential Geometric Regularization 32

4.1 Formal Setup 32
4.2 Variational Formulation 35
  4.2.1 The Empirical Term 36
  4.2.2 The Regularization Term 37
  4.2.3 The Simplex Constraint 40
  4.2.4 Algorithm Summary 42

5 Theoretical Analysis 44

5.1 Bayes Consistency 45
  5.1.1 Step 1 49
  5.1.2 Step 2 50
  5.1.3 Step 3 53
5.2 Riemannian Curvature Based Geometric Penalty 55
5.3 Existence of Gradient Flow 57
  5.3.1 Existence and Uniqueness of The Geometric Flow 58
  5.3.2 Existence for the Empirical Term 61
5.4 Extension to Nonlinear Input Spaces 63
5.5 Summary 63

6 Applications **65**

6.1 RBF Representation 66

6.1.1 Formulations and Algorithm 66

6.1.2 Qualitative Experiments on Synthetic Data 70

6.1.3 Quantitative Experiments on Benchmarks 73

6.2 Deep Neural Networks 79

6.2.1 Formulations and Algorithm 79

6.2.2 Computational Details 83

6.2.3 Qualitative Experiments on Synthetic Data 88

6.2.4 Quantitattive Experiments on Benchmarks 94

6.3 Summary 99

7 Conclusions and Future Work **101**

7.1 Main Contributions 101

7.2 Strength and Limitations 103

7.3 Interesting Directions for Future Research 104

Bibliography **111**

Curriculum Vitae of Qinxun Bai **118**

List of Tables

4.1 Notation 33

5.1 Notation for Bayes consistency 46

6.1 Performance of RBF-based implementation on UCI Datasets 74

6.2 Classification performance on CIFAR-10 96

6.3 Classification performance on perturbed training set 98

6.4 Classification performance on perturbed testing set 98

List of Figures

1.1 An image classification example 2

1.2 An adversarial example for DNN 4

1.3 Cartoon example of "small local oscillations" 5

3.1 Example of 2D classification 23

3.2 Initial surfaces of 2D classification example 24

3.3 Resulting surfaces of 2D classification example 25

3.4 Detailed plots for the 2D classification example 26

3.5 Real-world examples related to our physical model 27

3.6 Intuitive plots of the gradient vector 31

4.1 Example of three-class learning 34

6.1 RBF-based implementation on 2D classification example 71

6.2 Reversed gradient for 2D classification example 72

6.3 Exemplar input images from FMD and MNIST 77

6.4 Testing accuracy on FMD and MNIST 78

6.5 A feedforward neural network 80

6.6 Effect of one geometric gradient step directly applied on the output of the network function 90

6.7 Effect of geometric gradient update through back-propagation 91

6.8 Simple network with ReLU activation (ignoring singularities) on 2D classification example 92
6.9 Simple network with ReLU activation (substitution method) on 2D classification example 93

Chapter 1

Introduction

Supervised learning is one of the fundamental tasks in machine learning. It takes as input a training set of labeled data and outputs a function that is capable of predicting the desired value associated with the data. The two main topics in supervised learning are classification and regression. In this thesis, we study the classification problem in supervised learning, where a classifier is learned from a given set of labeled training data. With the ever increasing availability of labeled data and computational resources, classification techniques are playing an important role and achieving great success in many applications. The representation of the problem and the classification function, the optimization technique in learning the classifier, and the regularization technique to prevent overfitting during training are among the most important factors that affect the performance of a classification algorithm. This thesis focuses on regularization techniques. In the following sections of this chapter, we first define the problem we study, and motivate our main idea from challenges observed in practice. Then we briefly introduce the main contributions of this thesis. Finally we give a roadmap of this thesis and list the related publication.

1.1 Problem Definition

The major problem studied in this thesis is supervised learning of classifiers. In machine learning, the problem of classification is to determine to which of a given set of categories an observation belongs. The set of categories is commonly represented by a discrete set of labels, with each label corresponding to one category. The observation, which serves as the input of a classification task, is usually in the form of some sensed data. For example, the input could be an image, a video, a recording, a paragraph of text, etc. Figure 1.1 shows an example of image classification, where an input image is given on the left, and the task is to decide which is the most proper label assigned to it given a set of labels on the right. The function or mapping that performs this task is called a classifier.

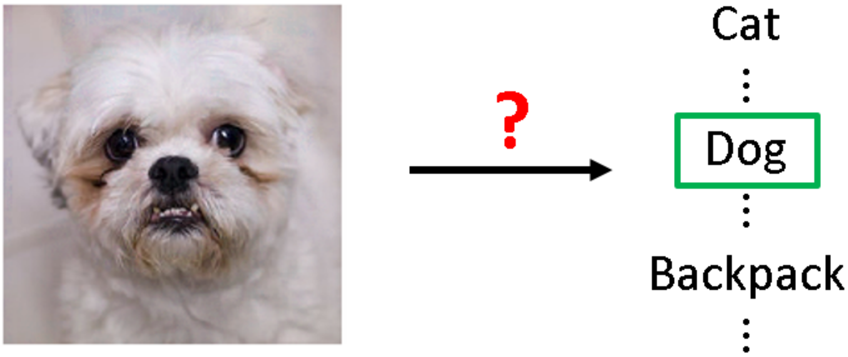


Figure 1.1: An image classification example. Image courtesy of the ImageNet database [22].

Supervised learning of classifiers is the problem of constructing a classifier for a specific type of classification task, based on a training set of input-label pairs of the same type. The performance of an algorithm in solving this problem is evaluated by the classification accuracy of the constructed classifier on unseen data of the same type. Just as with many inverse problems, supervised learning of classifiers is also an ill-posed problem, since there exist infinitely many functions (classifiers) that are able to correctly predict the labels of all examples from the given (finite) training set. As a result, if the classification accuracy on training examples is the only criterion in constructing the classifier, our algorithm is very likely to come up with one that performs perfectly on the training set, but poorly on unseen examples. This is known as the problem of overfitting to training data. The standard technique to prevent overfitting is by introducing an additional information/criterion in learning the classifier, which is known as regularization.

1.2 Main Idea

Our study of the overfitting problem and regularization techniques in supervised learning of classifiers is motivated by some widely observed phenomena and challenges in practice. In many real world classification problems, if the feature space is meaningful, then all examples that are locally within a small enough neighborhood of a training example should have class probability $P(y|\boldsymbol{x})$ similar to the training example. For instance, a small enough perturbation of RGB values at some pixels of a human face image should not change dramatically the likelihood of correct identification of this image during face recognition. We refer to this phenomenon, i.e., a small perturbation in the input results in small changes in its class probability, as the "small local oscillations" of the class probability. However, such a widely observed phenomenon is not explicitly incorporated by previous regularization techniques. For instance, as reported by [31], linear models and their combinations can be easily fooled by barely perceptible perturbations of a correctly predicted image, even though a L^2 regularizer is adopted. Such an example is shown in Figure 1.2, where the left image is correctly recognized by the learned classifier as dog. However, after adding some small random noise, the resulting perturbed image, which looks almost identical to human eyes, is predicted incorrectly by the classifier.

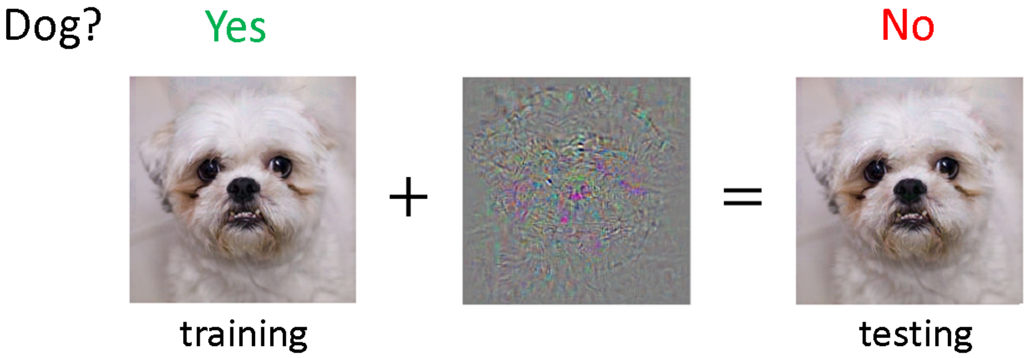
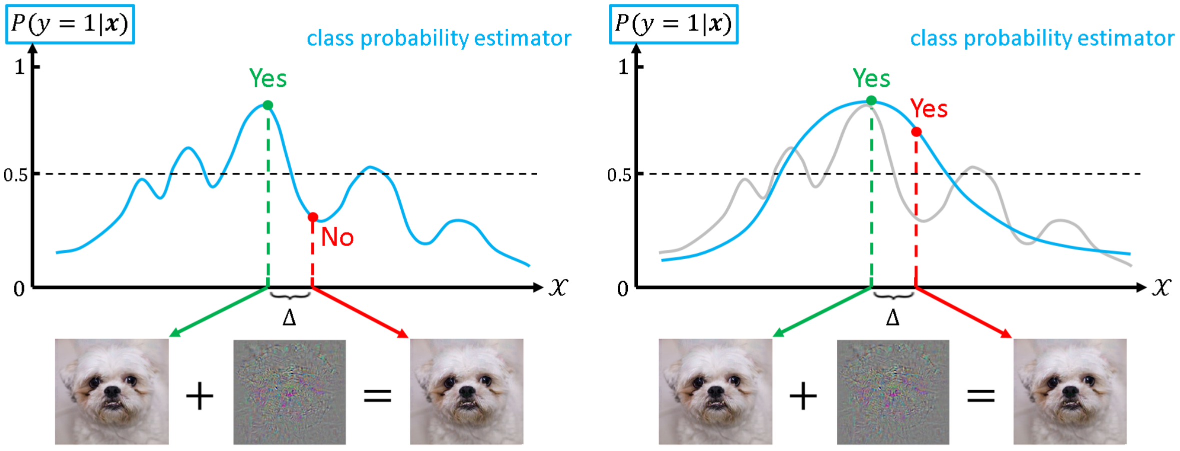


Figure 1.2: An example of fooling deep neural networks by slightly perturbed images, reported in [79]. The left image is correctly recognized by a learned classifier. The right image is constructed by adding some small random noise to the left image, where the added random noise is plotted (with 10 times the magnitude) in the middle. The right image, however, is predicted incorrectly by the same classifier.

Let us use a cartoon example to reveal the hints behind these observations. As shown in Figure 1.3, in both plots, the x-axis denotes the high dimensional space $\mathcal{X}$ of input images, i.e., every point on x-axis is an input image, and the original and perturbed dog images in Figure 1.2 correspond to two points close to each other in $\mathcal{X}$. The y-axis denotes the class probability of being dog, i.e., $P(y = 1|\boldsymbol{x})$. Then any class probability estimator corresponds to some curve between $y = 0$ and $y = 1$ in these plots. Two exemplar class probability estimators are shown as cyan curves in the left and right plots respectively. The left curve exhibits rapid oscillations in the neighborhood of the original dog image, and therefore predicts a negative result for the slightly perturbed image. On the other hand, since the right curve exhibits small oscillations in the neighborhood of the original image, a slight perturbation will not change the prediction of being positive.

The main idea of this thesis is to generalize the insight from Figure 1.3 to high-


$P(y = 1|\boldsymbol{x})$
class probability estimator
1
0.5
0
Yes
No
$\mathcal{X}$
Δ
+
=
$P(y = 1|\boldsymbol{x})$
class probability estimator
1
0.5
0
Yes
Yes
$\mathcal{X}$
Δ
+
=

(a) rapid local oscillations (b) small local oscillations

Figure 1.3: A cartoon example depicting the "small local oscillations" phenomenon of the class probability.

dimensional input space and general multiclass classification settings. Let $\boldsymbol{f} : \mathcal{X} \to \Delta^{L-1}$ be a class probability estimator, where $\mathcal{X}$ is the input space and Δ^{L-1} is the probabilistic simplex for L-class problem. Then there exists a submanifold corresponding to $\boldsymbol{f}$, in $\mathcal{X} \times \Delta^{L-1}$, namely the functional graph (in the geometric sense) of $\boldsymbol{f}$: $\{(\boldsymbol{x}, \boldsymbol{f}(\boldsymbol{x}))|\boldsymbol{x} \in \mathcal{X}\}$. By carefully studying the geometry of this submanifold, we argue that "small local oscillations" of the class probability can be characterized by the Riemannian geometry of this submanifold, and in particular, can be measured by the local volume or the more sensitive local curvature of this submanifold. We then propose a regularization approach based on this specific measure of the amount of local oscillations. In our approach, the supervised learning process can be viewed as a submanifold fitting procedure following a geometric flow. As we will see in later chapters, this regularization approach naturally handles binary and multiclass classification in a unified way, while many previous geometric methods focus on the geometry of the decision boundary, which are originally designed for binary classification and rely on "one versus one", "one versus all" or more efficiently a binary coding strategy [83] to generalize to the multiclass case.

1.3 Contributions

The main contributions of this thesis include the following:

- We conduct the first study of the differential geometric structure of class probability estimators, from a perspective combining statistical learning, differential geometry, and practical algorithm design.
- We provide a new geometric perspective on overfitting in supervised learning of classifiers.
- We provide a new complexity measure of classification functions, which is a natural extension of the surface energy in physics.
- Our study leads to a new regularization framework that is unified for both binary and multiclass classification.
- We implement a highly optimized library[1] of this regularization approach for applications in a variety of classification models, including feedforward deep neural networks.

1.4 Roadmap of Thesis

We organize the rest of the thesis as follows:

Chapter 2: Related Work

---

[1] `http://cs-people.bu.edu/qinxun/geo/geo.html`

This chapter reviews related literature on the following subjects: supervised learning of classifiers, regularization techniques, especially geometric regularization and Sobolev regularization techniques, manifold learning, variational problems and gradient flow, and the theory of minimal surfaces. Recent work in deep learning related to the application of our approach is also reviewed.

Chapter 3: Geometric Perspective of Supervised Learning

This chapter describes a new geometric perspective of supervised learning. To incorporate the phenomenon of "small local oscillations", we first draw insights from Bayesian arguments for Occam's Razor and optimization arguments for statistical learning. Then we motivate our geometric perspective by describing an elastic model in physics that naturally fits the process of supervised learning of a classifier. In the last section of this chapter, we establish the mathematical foundation of our geometric perspective, in particular, we study the Riemannian geometry and the gradient flow on the functional space in which our learning algorithm works.

Chapter 4: Differential Geometric Regularization

This chapter presents the methodology of the proposed differential geometric regularization method for supervised learning of classifiers. We first give a formal setup of the problem and related terminologies, then present in detail the formulation of our approach. In particular, we present the general variational formula for the regularized learning process, detailed formulas for the empirical term and the regularization term, and detailed formulas for solving the variational formula with gradient flow. Special concerns to the simplex constraint and a summary of the learning algorithm are also given.

Chapter 5: Theoretical Analysis

This chapter discusses some theoretical aspects and extensions of the proposed approach. Firstly, with a particular empirical loss function, we establish Bayes consistency for our geometric regularization scheme, under some mild initialization assumptions. Secondly, we discuss the theory and formulation of an alternative geometric regularizer based on the Riemannian curvature. Thirdly, we discuss the existence of the gradient flow under different topologies and point out that the existence of the gradient flow, in a strict mathematical sense, is non-trivial and not automatic. Lastly, we discuss the extension of our geometric perspective and regularization approach to situations where the input space is a submanifold with local charts, rather than a Euclidean space.

Chapter 6: Applications

This chapter discusses applications of the proposed approach in two representative classification models, i.e., the linear combination of radial basis functions (RBFs) and feedforward neural networks. For each model, we introduce specific formulations for applying our regularization approach to that model, and design specific algorithms incorporating our regularization approach into the training process of that model. Implementation details and practical concerns are also discussed for both models. In experiments with the RBF-based model, we demonstrate the effectiveness of our approach in benchmarks for binary and multiclass classification tasks. In experiments with deep neural networks, we analyze in detail the experimental results and suggest recipes to improve the implementation for follow-up research.

Chapter 7: Conclusions and Future Work

This chapter summarizes the thesis, and discusses key contributions and observations of our study. Some follow-up directions and open problems related to regularization in supervised learning are also discussed.

1.5 Related Paper

Part of the material for this thesis is based on the following paper:

> **Qinxun Bai**, Steven Rosenberg, Zheng Wu, and Stan Sclaroff. Differential Geometric Regularization for Supervised Learning of Classifiers. In *Proceedings of International Conference on Machine Learning (ICML)*, 2016.

Chapter 2

Related Work

In this chapter, we review the related literature on the following subjects: supervised learning of classifiers, regularization techniques, especially geometric regularization and Sobolev regularization techniques, manifold learning, variational problems and gradient flow, and the theory of minimal surfaces. We also review recent work related to our applications in deep learning, in particular, those related to adversarial examples of deep neural networks.

2.1 Supervised Learning of Classifiers

Many supervised learning algorithms for solving the classification problem have been proposed in machine learning literature, and there exist different ways of categorizing them. From statistical learning standpoint, there are two main types of classifiers: the empirical risk minimization (ERM) type, such as those introduced in [82], and the plug-in type, where the class label prediction is based on a class probability estimator [1], such as the nearest neighbor classifier [20]. In following paragraphs, we review some representative classification techniques following the categorization of [63].

One main type of classifiers is logic (rule) based, such as decision tree classifiers [61]. A decision tree classifier consists of a hierarchical decision making process following a tree structure, and the learning process depends on heuristics for splitting the feature space at every node while constructing the tree. Combining with ensemble techniques, some decision tree based classifiers have shown very strong performance in practice, such as AdaBoost [28] and Random forests [12].

Another main type of classifiers is based on the notion of the perceptron [65], which is inherently a linear and binary classifier. Variants along this line include some of the most powerful classifiers. A regularization and kernel extension of the perceptron leads to the well-known support vector machines [70]. A deep structure extension of the perceptron leads to multilayered perceptrons and deep neural networks [30]. State-of-the-art algorithms based on deep neural networks already achieve human level accuracy on some challenging classification tasks [67].

2.2 Regularization Techniques

In supervised learning for classification, the idea of regularization seeks a balance between a perfect description of the training data and the potential for generalization to unseen data. One popular type of regularization technique is defined in the form of penalizing some functional norms, such as L^1-norm, L^2-norm, etc. One of the most successful classification methods, the support vector machine (SVM) [82, 70] and its variants [6, 76], use a RKHS norm as a regularizer. Another type of regularization technique makes use of the ensemble principle to reduce the variance of the learning model, such as the very effective dropout strategy for training neural networks[75], which approximates some sort of "geometric averaging" over a large ensemble of possible sub-networks. There are also heuristic based regularization techniques, such as early stopping [60] in training neural networks. In the following subsections, we review two types of functional norm based regularization techniques that are most closely related to our approach.

2.2.1 Geometric Regularization

Geometric regularization techniques have also been studied in machine learning. Belkin *et al.* [11] employed geometric regularization in the form of the L^2 norm of the gradient magnitude supported on a manifold. This approach exploits the geometry of the marginal distribution $P(\boldsymbol{x})$ for semi-supervised learning, rather than the geometry of the class probability $P(y|\boldsymbol{x})$. Other related geometric regularization methods are motivated by the success of level set methods in image segmentation [15, 83] and Euler's Elastica in image processing [46, 45]. In particular, the Level Learning Set [15] combines a counting function of training samples and a geometric penalty on the surface area of the decision boundary. The Geometric Level Set [83] generalizes this idea to standard empirical risk minimization schemes with margin-based loss and carefully treats the variational problem with a Radial Basis Function (RBF) approximation. Along this line, the Euler's Elastica Model [46, 45] proposes a regularization technique that penalizes both the gradient oscillations and the curvature of the decision boundary. However, all three methods focus on the geometry of the decision boundary supported in the domain of the feature space, and the "small local oscillation" of the class probability is not explicitly addressed.

2.2.2 Sobolev Regularization

There exist other regularization methods that are related to some aspects of our work. Most notably, Sobolev regularization involves functional norms of a certain number of derivatives of the prediction function. For instance, the manifold regularization [11] uses a first derivative-based functional norm,

$$\int_{x \in \mathcal{M}} \|\nabla_{\mathcal{M}} f\|^2 dP(x), \tag{2.1}$$

where f is a smooth function on a manifold $\mathcal{M}$. A discrete version of (2.1) corresponds to the graph Laplacian regularization [89]. Lin *et al.* [45] discuss in detail the difference between a Sobolev norm and a curvature-based norm for the purpose of exploiting the geometry of the decision boundary.

For our purpose, while imposing, say, a high Sobolev norm[1] will also lead to a flattening of the submanifold $\mathrm{gr}(\boldsymbol{f})$, these norms are not specifically tailored to measuring the flatness of $\mathrm{gr}(\boldsymbol{f})$. In other words, a high Sobolev norm bound will imply the volume bound we desire, but not *vice versa*. As a result, imposing high Sobolev norm constraints (regardless of computational difficulties) overshrinks the hypothesis space from a learning theory point of view. In contrast, our regularization term, as we will see in §4.2.2, involves only the combination of first derivatives of $\boldsymbol{f}$ that specifically address the geometry behind the "small local oscillation" prior observed in practice.

2.3 Manifold Learning

Our training procedure for finding the optimal graph of a function is, in a general sense, also related to the manifold learning problem [80, 66, 10, 25, 88, 47]. The main difference is that we are studying the geometry of a specific manifold, i.e., the functional graph of a class probability estimator, which is well-defined given the estimator and unrelated to the distribution of the data. This is in contrast with

---

[1] "High Sobolev norm" is the conventional term for a Sobolev norm with a high number of derivatives.

the traditional manifold learning problem and assumptions therein. For instance, a common assumption made by most manifold learning methods is that the data lie in a low dimensional smooth manifold embedded in the high dimensional space. Actually, even checking the validity of this manifold hypothesis is both mathematically and algorithmically extremely involved [27]. Nevertheless, regarding the optimization process of finding the optimal submanifold, the most closely related work is [25], which seeks a flat submanifold of Euclidean space that contains a dataset. Again, there are key differences. Since the goal of [25] is dimensionality reduction, their manifold has high codimension, while our functional graph has codimension $L-1$, which may be as low as 1, where L is the number of classes. More importantly, we do not assume that the graph of our target function is a flat (or volume minimizing) submanifold, and we instead flow towards a function whose graph is as flat (or volume minimizing) as possible.

2.4 Variational Problems and Gradient Flow

Gradient flow procedures are widely used in variational problems, such as level set methods [59, 71], the Mumford-Shah functional [54], etc. In the classification literature, Varshney and Willsky *al.* [83] were the first to use gradient flow methods to solve level set based energy functions, then followed by [46, 45] to solve Euler's Elastica models. In our case, we are exploiting the geometry of submanifolds of the space $\mathcal{X} \times \Delta^{L-1}$, rather than standard vector spaces. In this regard, our work is related to a large body of literature on gradient flow/Morse theory in finite dimensions [52] and infinite dimensions [2], and on mean curvature flow, see [17, 51] and the references therein.

2.5 Theory of Minimal Surfaces

As will be presented in more detail in §3.2, our regularization approach moves the submanifold following the mean curvature flow, i.e., in the maximally volume decreasing direction. Since minimal surfaces are critical points of the volume functional used as our regularization term, our work is also related to the theory of minimal surfaces. For more details about this topic, please refer to Chapter One of [42]

2.6 Training in Neural Networks Robust to Adversarial Examples

Since the observations [79, 56] that neural networks are vulnerable to certain perturbations of the input data which are hardly noticeable to the human eye, there has been much effort to alleviate vulnerability of neural networks to adversarial perturbations.

Goodfellow *et al.* [31] argue that this phenomenon arises from high-dimensional linearity and proposes adding adversarial examples into the training set, where adversarial examples are generated by perturbation of training examples in the direction that most damage the classification loss. [57] further shows that training with adversarial samples alone also slightly improves the performance on standard testing sets. [33] penalizes the Frobenius norm of the Jacobian through a series of approximations and suffers from some performance drop on standard testing sets.

Following [31], Lyu *et al.* [48] proposed a min-max formulation to train models that are robust to adversarial examples. This min-max formulation is then transformed into a Jacobian regularization technique by first order approximation and yields performance improvement when used together with dropout.

While these early works all have difficulties in computing the derivatives of the Jacobian, Ororbia *et al.* [58] proposed a unified framework to compute derivatives of the Jacobian penalty with respect to the network's parameters and reported experimental evidence on adversarial testing sets. However, there is still a step using finite difference approximation in computing the derivatives, which introduces some error into the process of gradient update.

Inspired by [31], some recent works [53, 38] also apply the min-max formula where an inner constrained optimization problem is solved to find the most adversarial direction, with respect to the sensitivity of the class probability (softmax output of the network), in the sense of KL divergence [53] and disagreement [38] respectively. Miyato *et al.* [53] show improvement in standard testing sets over the adversarial training in [31]. Huang *et al.* [38] show improvement in adversarial testing sets over [31] and dropout.

All of these methods, except for [38], only test on simple networks no deeper than three layers. No adversarial training results have been reported on state-of-the-art deep architectures, such as Resnet. Moreover, most of these methods focus on some specific type of "perturbation scheme" and apply more or less approximations in gradient update with respect to the extra penalty on adversarial examples. In contrast, we propose a general regularization scheme for training neural networks which are robust to small perturbations, without using adversarial examples and without being specifically tailored to any particular type of perturbation mechanism. Our approach provides an exact formulation for gradient update with this regularization scheme, which can be incorporated into the standard back-propagation. We also test its effectiveness using state-of-the-art network models on both standard testing sets and adversarial testing sets.

Applying our regularization techniques to deep neural networks is also motivated by the recent theoretical study [26], which suggests that the robustness of classifiers regarding small perturbations in input space is closely related to the curvature of the classifier's decision boundary. This work makes it even more interesting to investigate the possibility of generalizing our geometric regularization technique to deep neural networks. Since the classifier's decision boundary is in fact a level set of the class probability estimator studied in this thesis, the geometry (mean curvature) exploited by our approach encodes much more information than the curvature of the decision boundary.

Chapter 3

Geometric Perspective of Supervised Learning

To address the observation of “small local oscillations” described in §1.2, in this chapter, we propose a geometric perspective of supervised learning.

Roadmap for this chapter

In §3.1, we first draw insights from Bayesian arguments for Occam’s Razor and optimization arguments for statistical learning. Then in §3.2, we motivate our geometric perspective by describing an elastic model in physics that naturally fits the process of supervised learning of a classifier, and we propose a volume-based “energy” as the complexity measure of classification functions. In §3.3, we establish the mathematical foundation of our geometric perspective. In particular, we study the Riemannian geometry and the gradient flow on the functional space in which our learning algorithm operates.

3.1 Insight from Statistical Learning and Optimization

3.1.1 Bayesian Perspective and Occam’s Razor

To solve the ill-posed problem of supervised learning of classifiers, regularization techniques are adopted to prevent the learning function from being overfitted to the training data. The effectiveness of regularization plays a central role in the generalization ability of the trained classification function. There are different perspectives for mathematically justifying the usefulness of regularization. We describe two perspectives that motivate our study of differential geometric regularization.

Bayesian perspective. One justification of regularization from the Bayesian perspective is that it imposes certain prior distributions on the search space of classification functions. This prior is encoded in the expression of the regularization term, which should reflect some characteristics of the problem on hand. In other words, the regularization term should penalize classification functions that are less likely to incorporate the underlying data distribution $P(\boldsymbol{x}, y)$. In classification problems, as explained in §1.2, one commonly observed prior is that the underlying class probability $P(y|\boldsymbol{x})$ has "small local oscillations" around confidently classified examples.

Occam's razor perspective. Another justification for regularization is that it imposes Occam's razor on the search space of classification functions. Occam's razor states that "the simplest model that fits the data is always preferred." It can be explained in probability theory without assuming any prior biased towards simpler models. We briefly review this argument in the following; for more details, please refer to Chapter 28 of [50].

Assume we are choosing between two models/functions F_1 and F_2 based on observation/data D. We consider the posterior of both models, i.e., $P(F_1|D)$ and $P(F_2|D)$. According to Bayes' rule, we have

$$\frac{P(F_1|D)}{P(F_2|D)} = \frac{P(F_1)}{P(F_2)} \frac{P(D|F_1)}{P(D|F_2)}. \tag{3.1}$$

Assuming a uniform prior over the function space, i.e., $P(F_1) = P(F_2)$, then our choice between F_1 and F_2 depends on the second ratio on the righthand side of

Eqn. (3.1), which compares the integration of of both models' data likelihood on the given observation set D. If F_2 is more complex than F_1, then it is capable of fitting a greater variety of data, i.e., the data likelihood of F_2 is supported on and thus spreads over a larger measurable subset of the data space than that of F_1. As a result, if both data likelihoods are supported on the observation set D, the integration of the more "concentrated" data likelihood, i.e., $P(D|F_1)$, is expected to be greater than the integration of the widerly spread data likelihood, i.e., $P(D|F_2)$. This explains why the less complex model F_1 is more preferable.

The next question is what type of simplicity should we pursue following Occam's razor. Again, we still focus on the data likelihood alone. There is always a trade-off between the capability of fitting the variability of the data and concentrating on just explaining the most likely data. The former favors a more complex model, while the latter favors a simpler one. Ideally, we want a model whose data likelihood is well aligned with the data distribution of the problem. In other words, by imposing Occam's razor, it is preferable to penalize the complexity of models in a way that penalizes deviation from the underlying data distribution of the classification problem. Of course, this is wishful thinking in practice.

3.1.2 Optimization and Statistical Learning

Recall the standard regularized loss minimization scheme for supervised learning,

$$\min_{\boldsymbol{f} \in \mathcal{M}} \mathcal{P}(\boldsymbol{f}) = \min_{\boldsymbol{f} \in \mathcal{M}} \{L(\boldsymbol{f}) + \lambda G(\boldsymbol{f})\}, \tag{3.2}$$

where $\boldsymbol{f} : \mathcal{X} \to \mathcal{Y}$ is a (classification) function from the input space $\mathcal{X}$ to the output label space $\mathcal{Y}$, $\mathcal{M}$ is the functional space our learning algorithm is searching in, L is the empirical data term, G is the regularization term, and λ is the trade-off parameter between them.

From an optimization perspective, including the regularization term into the minimization objective is equivalent to optimizing the original loss within a shrunken functional space, i.e.,

$$\min_{\boldsymbol{f}\in\mathcal{M}}\{L(\boldsymbol{f})+\lambda G(\boldsymbol{f})\} \iff \min_{\boldsymbol{f}\in\mathcal{M}_\lambda} L(\boldsymbol{f}), \tag{3.3}$$

where $\mathcal{M}_\lambda = \{\boldsymbol{f} \in \mathcal{M}, G(\boldsymbol{f}) \leq s(\lambda)\}$ is the shrunken functional space, and $s(\lambda)$ is some monotonically decreasing function of λ.

The key question to ask is how to properly shrink the functional space. Before answering this question, we need to understand the role of $\mathcal{M}$ based on statistical learning theory. Following the above setup, we denote the generalization error (risk) of a classification function $\boldsymbol{f}$ by

$$R_P(\boldsymbol{f}) = \mathbb{E}_P[\mathbb{1}_{\boldsymbol{f}(\boldsymbol{x})\neq y}], \tag{3.4}$$

where P denotes the underlying data distribution $P(\boldsymbol{x}, y)$. The lowest possible generalization error achievable by any function $h : \mathcal{X} \to \mathcal{Y}$ is defined as the Bayes error:

$$R_P^* = \inf_{\boldsymbol{h}:\mathcal{X}\to\mathcal{Y}} R_P(\boldsymbol{h}). \tag{3.5}$$

For a function $\boldsymbol{f} \in \mathcal{M}$, it is useful to study the difference between $R_P(\boldsymbol{f})$ and R_P^*, known as the excess error of $\boldsymbol{f}$, which can be decomposed as the following two terms,

$$R_P(\boldsymbol{f}) - R_P^* = (R_P(\boldsymbol{f}) - R_P(\mathcal{M})) + (R_P(\mathcal{M}) - R_P^*), \tag{3.6}$$

where $R_P(\mathcal{M}) = \inf\limits_{\boldsymbol{h}\in\mathcal{M}} R_P(\boldsymbol{h})$ refers to the optimal error achievable in $\mathcal{M}$.

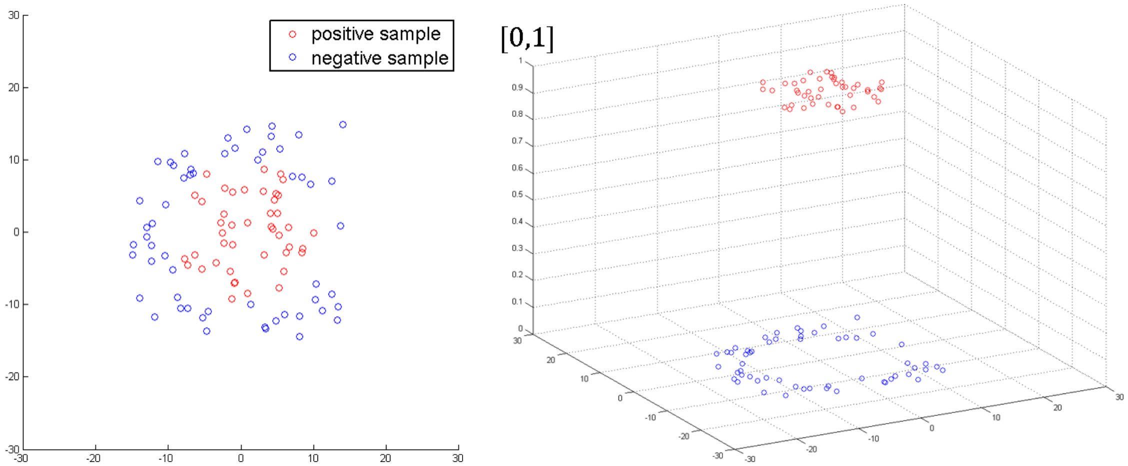
The first term on the right hand side of Eqn. (3.6) is known as the estimation error, which measures how close $\boldsymbol{f}$ is to the optimal choice in $\mathcal{M}$. The second term is known as the approximation error, which measures how close one can get to the Bayes error by searching a function in $\mathcal{M}$. In other words, the estimation error measures the optimality of the optimization algorithm working in $\mathcal{M}$, while the approximation error measures the capacity of the functional space $\mathcal{M}$. In general, the larger and the more complex $\mathcal{M}$ is, the lower the possible approximation error. However, there is no free lunch. The larger and the more complex $\mathcal{M}$ is, the harder it is for a learning algorithm to find the optimal function within it, given a fixed amount of training data, and the more likely it is that the trained function is overfitted to the training data. In other words, for the estimation error a smaller $\mathcal{M}$ is preferred, while for the approximation error a larger $\mathcal{M}$ is preferred.

Returning to the question of how to properly shrink the functional space $\mathcal{M}$ by introducing some regularization $G(\boldsymbol{f})$, there should be a subtle trade-off concerning its effect on both the estimation error and the approximation error. Combining the analysis from Section (§3.1.1), ideally, the regularization term $G(\boldsymbol{f})$ should precisely encode our observed/believed prior on the underlying data distribution, without overshrinking $\mathcal{M}$. This motivates our study of the geometric structure underlying the class probability.

Smoothness vs. Mean Curvature. A natural question to ask is whether previous regularization methods have properly encoded the “small local oscillation” phenomenon of the class probability widely observed in many classification problems. As described in §2.2, we feel that the answer is no. Most importantly, most functional-norm based regularization methods focus on enforcing smoothness of different kinds, e.g., a high Sobolev or Hölder norm. Enforcing a strong enough smoothness will finally lead to flattening all the local oscillations; however, it may overshrink the functional space, because a Sobolev-type regularization will suppress all partial derivatives, an extremely strong criterion. This is by way of analogy like carving with an axe, when what we really need is a sculptor's knife. On the other hand, studying the differential geometry on the submanifold corresponding to the class probability leads us to propose a mean curvature based regularization, which is a specific measure of the amount of local oscillation. This measurement generalizes to high dimensional spaces and handles binary and multiclass cases in a uniform way.

3.2 A Physical Model for Supervised Learning

In this section, we motivate our study from an elastic model in physics and the theory of minimal surfaces.



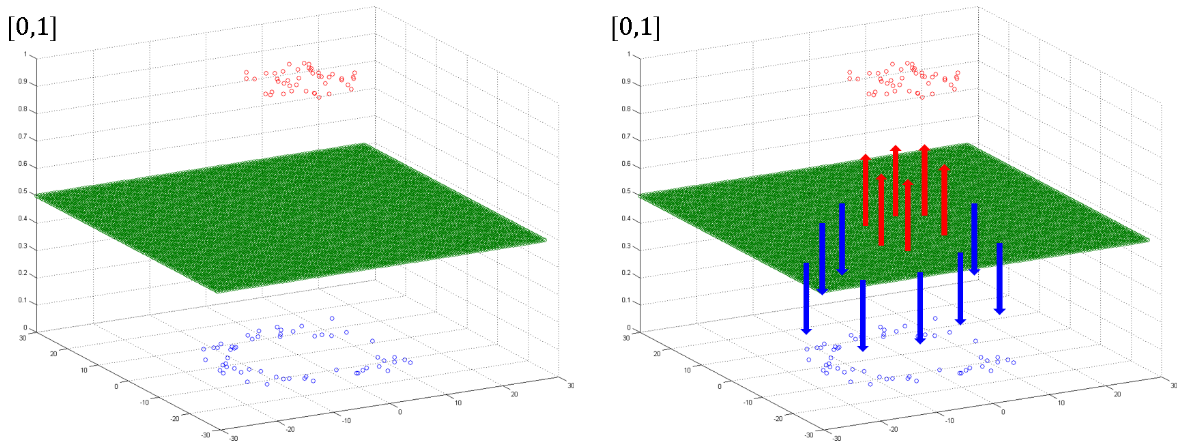
(*a*) (*b*)

Figure 3.1: Example of 2D classification.

We demonstrate our learning process using a binary classification example of 2D points, as shown in Figure 3.1. The left figure plots all training points of the two classes, with input space $\mathcal{X} \subset \mathbb{R}^2$. The right figure shows the product space

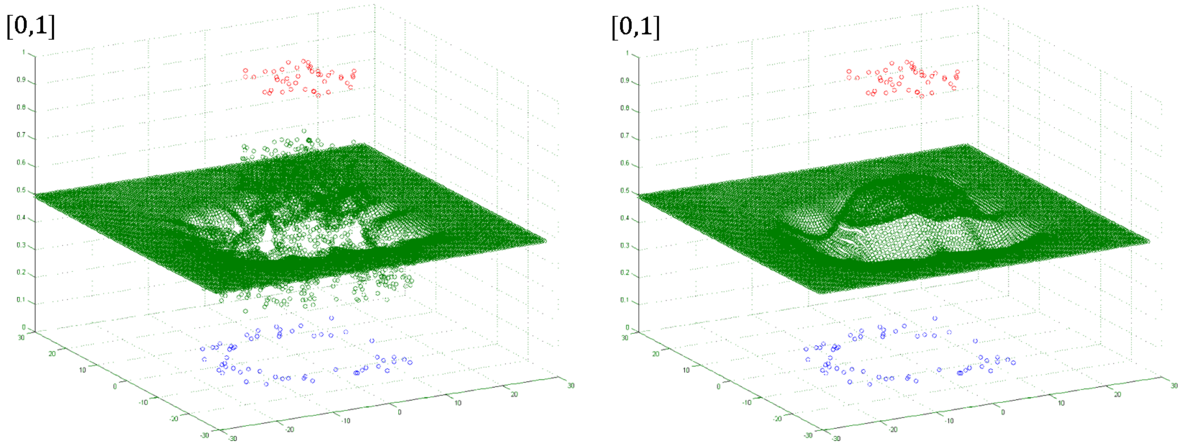
$\mathcal{X} \times [0,1]$, where $[0,1]$ is actually the output space of an estimator of the class probability $P(y=1|\boldsymbol{x})$. It is natural that all the positive training points lie on the plane $P(y=1|\boldsymbol{x})=1$, and all the negative training points lie on the plane $P(y=1|\boldsymbol{x})=0$. Without training, an initial guess of the class probability should assign equal probability of being positive and negative at every location of $\mathcal{X}$, i.e., $P(y=1|\boldsymbol{x})=\frac{1}{2}$ for all $\boldsymbol{x} \in \mathcal{X}$, which forms a perfectly flat plane at $\frac{1}{2}$, as shown in the left image of Figure 3.2.

When training starts, no matter what classification loss is used, the initial plane will deform towards both positive and negative training points, in order to explain their labels correctly, as shown in the right image of Figure 3.2. In physics, this is as if the initial surface is attracted by gravitational force due to point masses centered at the training data. If this is the only force that the surface is subject to, the surface will end up like the one shown in the left image of Figure 3.3. However, in elastic models in physics, the surface itself will simultaneously remain as tight as possible, due to the presence of surface tension. Then the overall effect of both forces will deform the surface in the way shown in the right image of Figure 3.3.



(a) (b)

Figure 3.2: Initial surfaces of 2D classification example.



(*a*) (*b*)

Figure 3.3: Resulting surfaces of 2D classification example.

Comparing the two images in Figure 3.3, the left surface deforms under a constrained gravitational force and exhibits rapid local oscillation. This is consistent with the learning process where there is only an empirical data term, and as a result, the learned function is overfitted to the training data. On the other hand, the right surface deforms subject to both a gravitational force and its surface tension. The resulting surface is much flatter than the left surface and exhibits smaller local oscillation. This is consistent with the learning process where a proper regularization term is combined with the classification loss and thus prevents the learned function from being overfitted to the training data. More figures depicting the detailed learning process as well as the corresponding decision boundaries are included in Figure 3.4.

Our study of the geometric structure of the learning process is inspired by this physical model. The regularization technique we propose can be regarded as a high-dimensional extension of the surface tension effect in this elastic model. In elastic physics, the effect of surface tension follows the rule of minimizing the surface energy, which is proportional to the surface area. The regularization term we propose is based on the volume measurement of the submanifold corresponding to the class

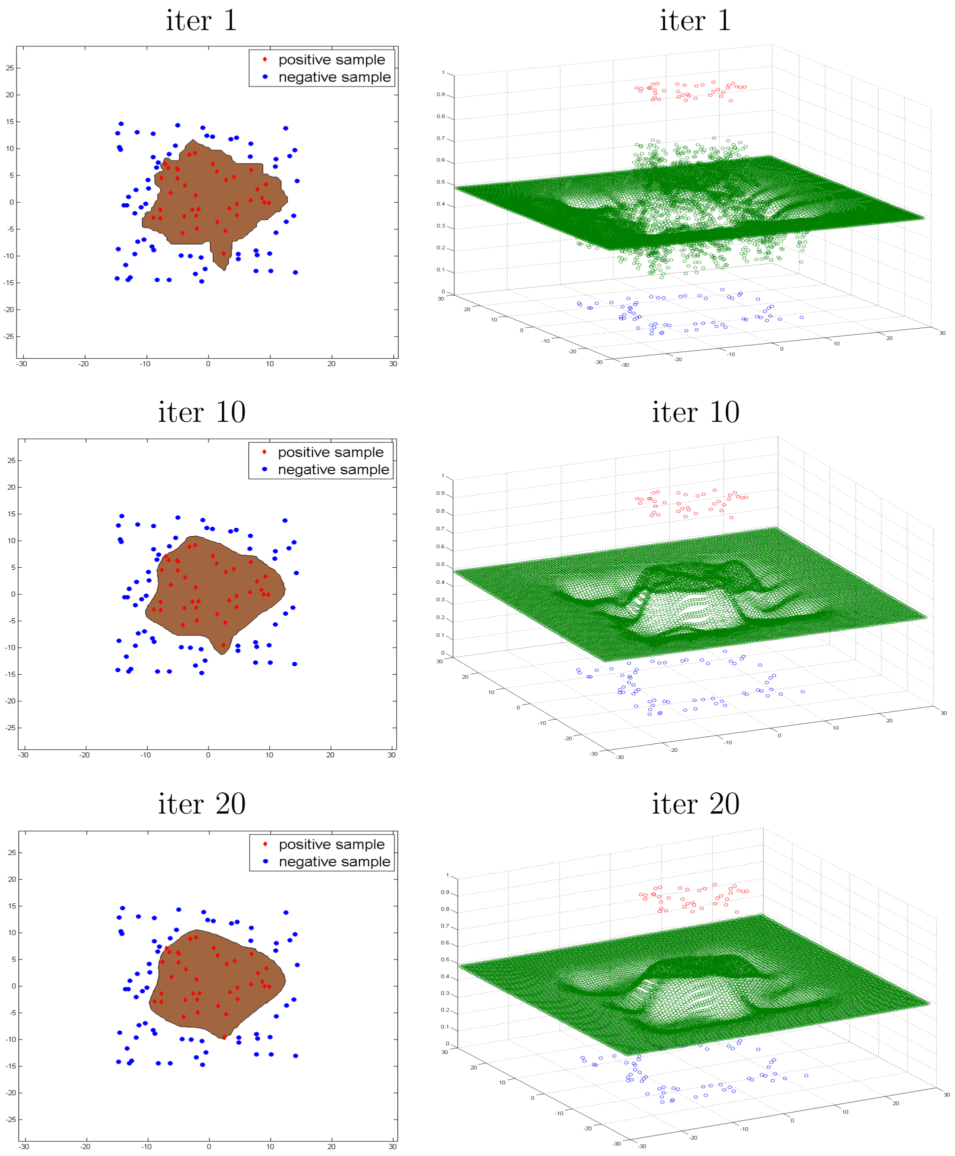
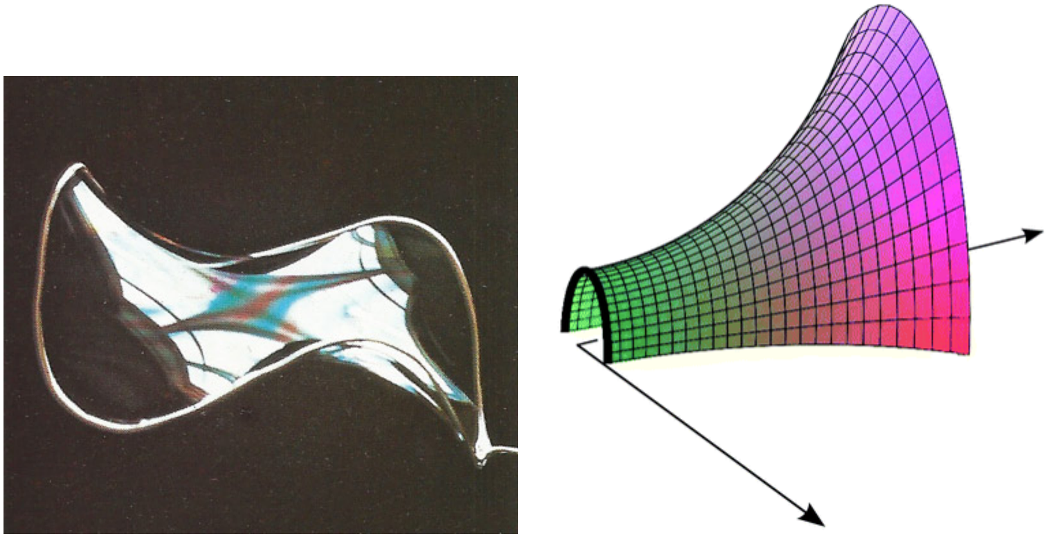


Figure 3.4: Example of binary learning, where input space $\mathcal{X}$ is $2d$. Training points are sampled uniformly within the region $[-15, 15] \times [-15, 15]$, and labeled by the function $y = \mathrm{sign}(10 - \|\boldsymbol{x}\|_2)$. We plot the surface obtained by our method in the right column and the corresponding decision boundary in the left column. Note that the vertical axis of the right image is the 1-simplex $\Delta^1 \subset \mathbb{R}^2$.

probability estimator, which is exactly a natural extension of surface area to the high-dimensional case. Following the insight drawn from previous section, to impose Occam's razor, we propose the volume-based "energy" of the functional graph as the complexity measure of functions, which is a high-dimensional extension of the surface energy in elastic physics.

The process of minimizing this volume-based energy is also closely related to the theory of minimal surfaces. Some real-world example of minimal surfaces are shown in Figure 3.5.



(*a*) soap film (*b*) a fraction of catenoid

Figure 3.5: Real-world examples related to our physical model. The soap film is balanced by the gravity and surface tension under boundary condition. The catenoid is a minimal surface where the mean curvature is zero at every point on the surface.

3.3 Geometric Foundation

The learning process and physical model described in previous section is intuitive, so we need to rigorously formulate it in the language of differential geometry. In particular, recalling that the regularized loss minimization formula in Eqn. (3.2), for supervised learning of classifiers, the functional space our algorithm works in is

$$\mathcal{M} = \mathrm{Maps}(\mathcal{X}, \Delta^{L-1}), \tag{3.7}$$

the set of smooth functions from $\mathcal{X}$ to Δ^{L-1}, where Δ^{L-1} is the standard $(L-1)$-simplex in $\mathbb{R}^L$ for L-class problems. Since $\mathcal{M}$ is an infinite dimensional manifold with corners, in order to apply the standard regularized loss minimization on it, in the following sections we first study the geometry of $\mathcal{M}$, and then discuss how to do optimization on it.

3.3.1 Geometry of $\mathcal{M}$

Since our learning process is actually applied to the infinite dimensional manifold $\mathcal{M}$, we have to understand both the topology and the Riemannian geometry of $\mathcal{M}$. We think of the input space $\mathcal{X} \subset \mathbb{R}^N$ as large enough so that the training examples are actually sampled well inside $\mathcal{X}$. This allows us to treat $\mathcal{X}$ as a closed manifold in our setup, so that boundary effects can be ignored. Given that $\mathcal{M}$ is not a vector space, we first study the topology and the Riemannian geometry of its ambient space $\mathcal{M}' = \mathrm{Maps}(\mathcal{X}, \mathbb{R}^L)$, the set of smooth maps from $\mathcal{X}$ to $\mathbb{R}^L$. For the topology, we put the Fréchet topology on $\mathcal{M}'$, and take the induced topology on $\mathcal{M}$. The Fréchet topology is given as follows: a basis of the topology at a function $\boldsymbol{f} \in \mathcal{M}$ consists of all functions $\boldsymbol{f}'$ with $\sup_{\alpha_k} \|\partial_{\alpha_k}(\boldsymbol{f}' - \boldsymbol{f})\|_\infty < \delta_k$ for all $k \in \mathbb{Z}_{\geq 0}$, for choices of $\delta_k > 0$, where ∂_{α_k} ranges over all partial derivatives of order k. Intuitively speaking, two functions in $\mathcal{M}$ are close if the functions and all their partial derivatives are pointwise close. For the tangent space of $\mathcal{M}$, since $\mathcal{M}$ is a closed Fréchet submanifold with corners inside the vector space $\mathcal{M}'$, we can identify each $T_{\boldsymbol{f}}\mathcal{M}$ [1] with a closed cone inside $\mathcal{M}'$. For the Riemannian metric on $\mathcal{M}$, we restrict the L^2 metric on $\mathcal{M}'$ to $T_{\boldsymbol{f}}\mathcal{M}$:

$$\langle \phi_1, \phi_2 \rangle := \int_{\mathcal{X}} \phi_1(\boldsymbol{x})\phi_2(\boldsymbol{x})\mathrm{dvol}_{\boldsymbol{x}}, \tag{3.8}$$

with $\phi_i \in \mathcal{M}'$ and $\mathrm{dvol}_{\boldsymbol{x}}$ being the volume form of the induced Riemannian metric on the functional graph of $\boldsymbol{f}$ (the submanifold corresponding to $\boldsymbol{f}$). Strictly speaking, the volume form is pulled back to $\mathcal{X}$ by $\boldsymbol{f}$, usually denoted by $\boldsymbol{f}^*\mathrm{dvol}_{\mathbb{R}^{N+L}}$. We could alternatively use the Lebesgue measure on $\mathcal{X}$ in this integral, but dvol generalizes to the case where $\mathcal{X}$ is itself a manifold, we will discuss this in more detail in §5.4.

For the Riemannian metric on the functional graph of $\boldsymbol{f}$, since $\boldsymbol{f} : \mathcal{X} \to \Delta^{L-1} \subset \mathbb{R}^L$, we use the standard dot product on $\mathbb{R}^{N+L}$ to define the induced Riemannian metric on the graph of $\boldsymbol{f}$.

3.3.2 Gradient Flow on $\mathcal{M}$

Following Eqn. (3.2), we are minimizing a penalty function $\mathcal{P} : \mathcal{M} \to \mathbb{R}$ on the infinite dimensional manifold $\mathcal{M}$. Based on the geometric setup described in the previous section, this optimization problem leads to solving a variational formula. We leave the detailed formula and other practical concerns to the next chapter, while focusing on the basic principles in this section.

The standard technique for solving variational formulas is the Euler-Lagrange PDE. However, due to our geometric regularization term, finding the minimal solutions of the Euler-Lagrange equations for $\mathcal{P}$ is difficult. Instead, we solve for argmin $\mathcal{P}$ using gradient flow in functional space $\mathcal{M}$. Note that $\mathcal{P}$ should be Fréchet smooth or at least differentiable, so that the gradient method can be applied.

---

[1] $T_{\boldsymbol{f}}\mathcal{M}$ is the tangent space to $\mathcal{M}$ at $\boldsymbol{f}$.

For explanation purposes only, we replace $\mathcal{M}$ with a finite dimensional Riemannian manifold M. Without loss of generality, we also assume that $\mathcal{P}$ is smooth, so that it has a differential $d\mathcal{P}_{\boldsymbol{f}} : T_{\boldsymbol{f}}M \to \mathbb{R}$ for each $\boldsymbol{f} \in M$, where $T_{\boldsymbol{f}}M$ is the tangent space to M at $\boldsymbol{f}$. Since $d\mathcal{P}_{\boldsymbol{f}}$ is a linear functional on $T_{\boldsymbol{f}}M$, there is a unique tangent vector, denoted $\nabla\mathcal{P}_{\boldsymbol{f}}$, such that $d\mathcal{P}_{\boldsymbol{f}}(\boldsymbol{v}) = \langle \boldsymbol{v}, \nabla\mathcal{P}_{\boldsymbol{f}} \rangle$ for all $\boldsymbol{v} \in T_{\boldsymbol{f}}M$. $\nabla\mathcal{P}_{\boldsymbol{f}}$ points in the direction of maximal increase of $\mathcal{P}$ at $\boldsymbol{f}$. Define the gradient flow equation as follows,

$$d\boldsymbol{f}_t/dt = -\nabla\mathcal{P}_{\boldsymbol{f}_t}, \text{ for } \boldsymbol{f}_t \in M \text{ and } t \geq 0. \tag{3.9}$$

Its solution is a flow line of steepest descent of $\mathcal{P}$ starting at an initial $\boldsymbol{f}_0$. For a dense open set of initial points, flow lines approach a local minimum of $\mathcal{P}$ as $t \to \infty$. For $t \gg 0$, the length of the gradient vector along the flow line goes to zero, and we could stop the flow when this length is below a specified cutoff. In finite dimensions, by standard ordinary differential equation theory, the existence of flow lines is guaranteed if M is complete.

However, in the infinite dimensional case of spaces of functions, the existence of flow lines is not automatic, and we will discuss this in more detail in §5.3. Assuming the existence of flow lines, we always choose the initial function $\boldsymbol{f}_0$ to be the “neutral” choice $\boldsymbol{f}_0(\boldsymbol{x}) \equiv (\frac{1}{L}, \ldots, \frac{1}{L})$, i.e., assigning equal conditional probability to all classes, which we assume is generic.

Figure 3.6 gives an intuitive idea about the gradient vector $\nabla\mathcal{P}_{\boldsymbol{f}_t}$. The gradient vector at a point $\boldsymbol{f}_t$ of the functional space $\mathcal{M}$ is plotted as the blue arrow in the left image. Since a point on $\mathcal{M}$ corresponds to a function whose functional graph is a surface in the right image, the gradient vector $\nabla\mathcal{P}_{\boldsymbol{f}_t}$ at $\boldsymbol{f}_t$ then corresponds to a vector field on the graph of $\boldsymbol{f}_t$, i.e., blue arrows in the right image. With $\boldsymbol{f}_t$ moving along the negative flow line on $\mathcal{M}$, the corresponding functional graph (submanifold) deforms in a volume minimizing manner.

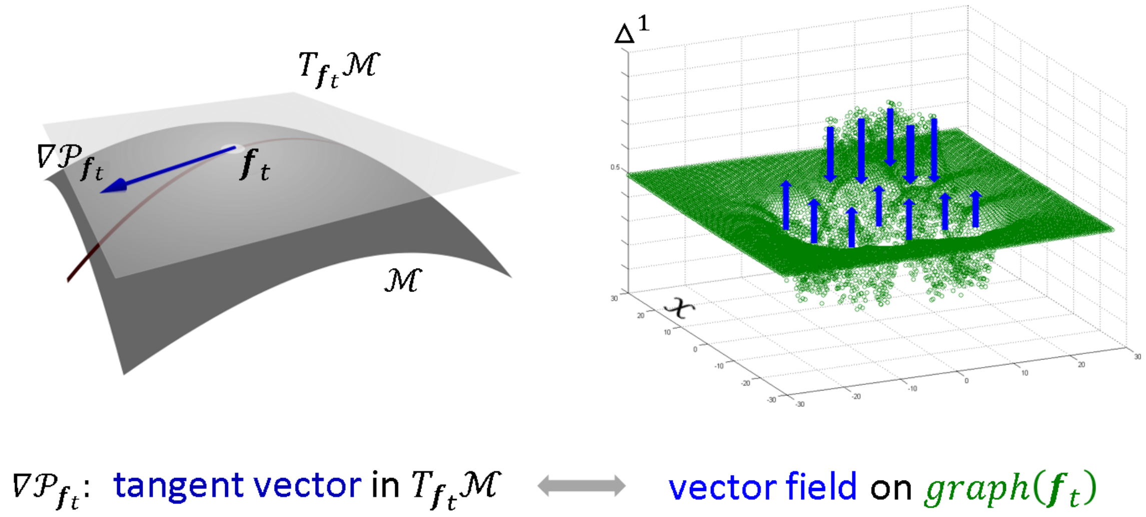

$T_{f_t}\mathcal{M}$
$\nabla\mathcal{P}_{f_t}$
f_t
$\mathcal{M}$
Δ^1
$\mathcal{X}$
0.5
30
20
10
0
-10
-20
-30
-30
-20
-10
0
10
20
30
$\nabla\mathcal{P}_{f_t}$: tangent vector in $T_{f_t}\mathcal{M}$
vector field on $graph(f_t)$

Figure 3.6: Intuitive plots of the gradient vector $\nabla\mathcal{P}_{f_t}$.

3.4 Summary

In this chapter, we propose a new geometric perspective of supervised learning. To incorporate the phenomenon of "small local oscillations", we first draw insights from Bayesian arguments for Occam's Razor and optimization arguments for statistical learning. Then we motivate our geometric perspective from an elastic model in physics that naturally fits the process of supervised learning of classifiers. We also establish the mathematical foundation for our geometric perspective. In particular, we study the Riemannian geometry and the gradient flow on the functional space in which our learning algorithm works.

Chapter 4

Differential Geometric Regularization

In this chapter, we introduce the detailed formulation that explores the geometric perspective of supervised learning presented in the previous chapter. This leads to a new differential geometric regularization method for supervised learning of classifiers.

Roadmap for this chapter

We first give a formal setup of the problem and related terminology in §4.1. We then present in detail the formulations of our approach in §4.2. In particular, we present the detailed formulas for the empirical term in §4.2.1, and the detailed formulas for the regularization term in §4.2.2. Special concerns to the simplex constraint and a summary of the learning algorithm are given in §4.2.3 and §4.2.4 respectively. Related mathematical notations are summarized in Table 4.1.

4.1 Formal Setup

Following the probabilistic setting of classification, given a sample (feature) space $\mathcal{X} \subset \mathbb{R}^N$, a label space $\mathcal{Y} = \{1, \ldots, L\}$, and a finite training set of labeled samples $\mathcal{T}_m = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$, where each training sample is generated i.i.d. from a distribution P over $\mathcal{X} \times \mathcal{Y}$, our goal is to find a $h_{\mathcal{T}_m} : \mathcal{X} \to \mathcal{Y}$ such that for any new sample $\boldsymbol{x} \in \mathcal{X}$, $h_{\mathcal{T}_m}$ predicts its label $\hat{y} = h_{\mathcal{T}_m}(\boldsymbol{x})$. The optimal generalization risk (Bayes risk) is achieved by the classifier $h^*(\boldsymbol{x}) = \operatorname{argmax}\{\eta^\ell(\boldsymbol{x}), \ell \in \mathcal{Y}\}$, where $\boldsymbol{\eta} = (\eta^1, \ldots, \eta^L)$

Table 4.1: Notation

$\delta^i_j = \mathbb{1}_{\{i=j\}} = \begin{cases} 1, & i=j \\ 0, & i\neq j \end{cases}$
$h_{\boldsymbol{f}}(\boldsymbol{x}) = \underset{\ell\in\mathcal{Y}}{\operatorname{argmax}} f^{\ell}(\boldsymbol{x})$: plug-in classifier of $\boldsymbol{f} : \mathcal{X} \to \Delta^{L-1}$
Δ^{L-1} : the standard$(L-1)$-simplex in $\mathbb{R}^L$;
$\boldsymbol{\eta}(\boldsymbol{x}) = (\eta^1(\boldsymbol{x}), \dots, \eta^L(\boldsymbol{x}))$: class probability: $\eta^{\ell}(\boldsymbol{x}) = P(y=\ell\|\boldsymbol{x})$
$\mathcal{M} : \{\boldsymbol{f} : \mathcal{X} \to \Delta^{L-1} : \boldsymbol{f} \in C^{\infty}\}$
$\mathcal{M}' : \{\boldsymbol{f} : \mathcal{X} \to \mathbb{R}^L : \boldsymbol{f} \in C^{\infty}\}$
$T_f\mathcal{M}$: the tangent space to $\mathcal{M}$ at some $f \in \mathcal{M}$
The graph of $f \in \mathcal{M}$ (or $\mathcal{M}'$) : $\mathrm{gr}(\boldsymbol{f}) = \{(\boldsymbol{x}, f(\boldsymbol{x})) : \boldsymbol{x} \in \mathcal{X}\}$
$g_{ij} = \frac{\partial f}{\partial x^i}\frac{\partial f}{\partial x^j}$: The Riemannian metric on $\mathrm{gr}(\boldsymbol{f})$ induced from the standard dot product on $\mathbb{R}^{N+L}$
$(g^{ij}) = g^{-1}$, with $g = (g_{ij})_{i,j=1,\dots,N}$
$\mathrm{dvol} = \sqrt{\det(g)}dx^1 \dots dx^N$, the volume element on $\mathrm{gr}(\boldsymbol{f})$
$\{e_i\}_{i=1}^N$: a smoothly varying orthonormal basis of the tangent spaces $T_{(\boldsymbol{x},\boldsymbol{f}(\boldsymbol{x}))}\mathrm{gr}(\boldsymbol{f})$ of $\mathrm{gr}(\boldsymbol{f})$
$\mathrm{Tr\ II} \in \mathbb{R}^{N+L}$: the trace of the second fundamental form of $\mathrm{gr}(\boldsymbol{f})$ $\mathrm{Tr\ II} = \left(\sum_{i=1}^N D_{e_i} e_i\right)^{\perp}$: with $\perp$ the orthogonal projection to the subspace perpendicular to the tangent space of $\mathrm{gr}(\boldsymbol{f})$, and $D_y w$ is the directional derivative of w in y direction
$\mathrm{Tr\ II}^L$: the projection of Tr II onto the last L coordinates of $\mathbb{R}^{N+L}$
$\mathcal{P} : \mathcal{M} \to \mathbb{R}$: a penalty function on a possibly infinite dimensional manifold $\mathcal{M}$
$\nabla\mathcal{P}$: the gradient vector field of $\mathcal{P}$

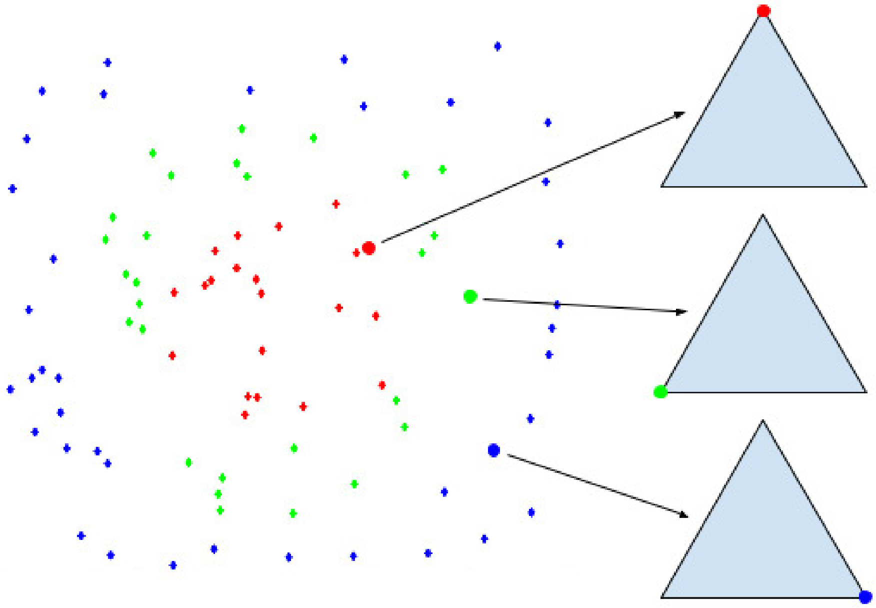


Figure 4.1: Example of three-class learning, i.e., $L = 3$, where the input space $\mathcal{X}$ is $2d$. Training samples of the three classes are marked with red, green and blue dots respectively. The class label for each training sample corresponds to a vertex of the simplex Δ^{L-1}. As a result, each mapped training point $(\boldsymbol{x}_i, \boldsymbol{z}_i)$ lies on one face (corresponding to its label y_i) of the space $\mathcal{X} \times \Delta^2$.

with $\eta^\ell : \mathcal{X} \to [0, 1]$ being the ℓ^{th} class probability, i.e. $\eta^\ell(\boldsymbol{x}) = P(y = \ell | \boldsymbol{x})$.

Our regularization approach exploits the geometry of the class probability estimator, and can be regarded as a "hybrid" plug-in/ERM scheme [1]. A regularized loss minimization problem is set up to find an estimator $\boldsymbol{f} : \mathcal{X} \to \Delta^{L-1}$, where Δ^{L-1} is the standard $(L-1)$-simplex in $\mathbb{R}^L$, and $\boldsymbol{f} = (f^1, \ldots, f^L)$ is an estimator of $\boldsymbol{\eta}$ with $f^\ell : \mathcal{X} \to [0, 1]$. The estimator $\boldsymbol{f}$ is then "plugged-in" to get the classifier $h_{\boldsymbol{f}}(\boldsymbol{x}) = \operatorname{argmax}\{f^\ell(\boldsymbol{x}), \ell \in \mathcal{Y}\}$.

Figure 4.1 shows an example of the setup of our approach, for a synthetic three-class classification problem. The submanifold corresponding to estimator $\boldsymbol{f}$ is the graph (in the geometric sense) of $\boldsymbol{f}$: $\operatorname{gr}(\boldsymbol{f}) = \{(\boldsymbol{x}, f^1(\boldsymbol{x}), \ldots, f^L(\boldsymbol{x})) : \boldsymbol{x} \in \mathcal{X}\} \subset \mathcal{X} \times \Delta^{L-1}$. We denote a point in the space $\mathcal{X} \times \Delta^{L-1}$ by $(\boldsymbol{x}, \boldsymbol{z}) = (x^1, \ldots, x^N, z^1, \ldots, z^L)$, where $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{z} \in \Delta^{L-1}$. In this product space, a training pair $(\boldsymbol{x}_i, y_i = \ell)$ naturally maps to the point $(\boldsymbol{x}_i, \boldsymbol{z}_i) = (\boldsymbol{x}_i, 0, \ldots, 1, \ldots, 0)$, with the one-hot vector $\boldsymbol{z}_i$ (with the 1 in its ℓ-th slot) at the vertex of Δ^{L-1} corresponding to $P(y = \ell|\boldsymbol{x}) = 1$.

We point out two properties of this geometric setup. Firstly, it inherently handles multiclass classification, with binary classification as a special case. Secondly, while the dimension of the ambient space, i.e. $\mathbb{R}^{N+L}$, depends on both the feature dimension N and number of classes L, the intrinsic dimension of the submanifold $\mathrm{gr}(\boldsymbol{f})$ only depends on N.

4.2 Variational Formulation

We want $\mathrm{gr}(\boldsymbol{f})$ to approach the mapped training points while remaining as flat as possible, so we impose a penalty on $\boldsymbol{f}$ consisting of an empirical loss term $\mathcal{P}_{\mathcal{T}_m}$ and a geometric regularization term $\mathcal{P}_G$. For $\mathcal{P}_{\mathcal{T}_m}$, we can choose either the widely-used cross-entropy loss function for multiclass classification or the simpler Euclidean distance function between the simplex coordinates of the graph point and the mapped training point. For $\mathcal{P}_G$, following the physical model introduced in the previous chapter, we measure the graph's volume, which is proportional to the graph's energy in low dimensions, $\mathcal{P}_G(\boldsymbol{f}) = \int_{\mathrm{gr}(\boldsymbol{f})} \mathrm{dvol}$, where dvol is the induced volume from the Lebesgue measure on the ambient space $\mathbb{R}^{N+L}$.

More precisely, we find the function that minimizes the following penalty $\mathcal{P}$:

$$\mathcal{P} = \mathcal{P}_{\mathcal{T}_m} + \lambda \mathcal{P}_G : \mathcal{M} = \mathrm{Maps}(\mathcal{X}, \Delta^{L-1}) \to \mathbb{R} \tag{4.1}$$

on the set $\mathcal{M}$ of smooth functions from $\mathcal{X}$ to Δ^{L-1}, where λ is the tradeoff parameter between empirical loss and regularization. It is important to note that any relative scaling of the domain $\mathcal{X}$ will not affect the estimate of the class probability $\boldsymbol{\eta}$, as scaling will distort $\mathrm{gr}(\boldsymbol{f})$ but will not change the critical function estimating $\boldsymbol{\eta}$.

In our setup, we want to find the (or a) best estimator $\boldsymbol{f} : \mathcal{X} \to \Delta^{L-1}$ of $\boldsymbol{\eta}$ on a compact set $\mathcal{X} \subset \mathbb{R}^N$ given a set of training data $\mathcal{T}_m = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$. We think of $\mathcal{X}$ as large enough so that the training data actually is sampled well inside $\mathcal{X}$. This allows us to treat $\mathcal{X}$ as a closed manifold in our gradient calculations, so that boundary effects can be ignored. A similar *natural boundary condition* is also adopted by previous work [83, 46, 45].

4.2.1 The Empirical Term

We consider two widely-used loss functions for the empirical penalty term $\mathcal{P}_{\mathcal{T}_m}$.

Quadratic loss. Since $\mathcal{P}_{\mathcal{T}_m}$ measures the deviation of $\mathrm{gr}(\boldsymbol{f})$ from the mapped training points, it is natural to choose the quadratic function of the Euclidean distance in the simplex Δ^{L-1},

$$\mathcal{P}_{\mathcal{T}_m}(\boldsymbol{f}) = \sum_{i=1}^{m} \|\boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{z}_i\|^2, \tag{4.2}$$

where $\boldsymbol{z}_i$ is the one-hot vector corresponding to the ground truth label of $\boldsymbol{x}_i$. The gradient vector with respect to $\boldsymbol{f}$ evaluated at $\boldsymbol{x}_i$ is

$$\nabla \mathcal{P}_{\mathcal{T}_m, \boldsymbol{f}}(\boldsymbol{x}_i) = 2(\boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{z}_i). \tag{4.3}$$

Cross-entropy loss. The cross-entropy loss function is also widely used for probabilistic output in classification,

$$\mathcal{P}_{\mathcal{T}_m}(\boldsymbol{f}) = -\sum_{i=1}^{m} \sum_{\ell=1}^{L} z_i^\ell \log f^\ell(\boldsymbol{x}_i). \tag{4.4}$$

We will discuss the detailed formula for the gradient vector of this loss function in

§6.1 when we represent $\boldsymbol{f}$ in a parametric form.

4.2.2 The Regularization Term

Following our geometric perspective, we wish to penalize graphs for high "energy", and we use the following function, which measures the volume of $\mathrm{gr}(\boldsymbol{f})$:

$$\mathcal{P}_G(\boldsymbol{f}) = \int_{\mathrm{gr}(\boldsymbol{f})} \mathrm{dvol} = \int_{\mathrm{gr}(\boldsymbol{f})} \sqrt{\det(g)} dx^1 \dots dx^N, \tag{4.5}$$

where $g = (g_{ij})$ with $g_{ij} = \delta^i_j + f^a_i f^a_j$, is the Riemmanian metric on $\mathrm{gr}(\boldsymbol{f})$ induced from the standard dot product on $\mathbb{R}^{N+L}$, and $\boldsymbol{f} = (f^1, \dots, f^L)$. We use the summation convention on repeated indices. Note that this regularization term is clearly very different from the standard Sobolev norm of any order.

It is standard that $\nabla \mathcal{P}_G = -\mathrm{Tr}\ \mathrm{II} \in \mathbb{R}^{N+L}$ on the space $\mathcal{M}'$ [1] of all embeddings of $\mathcal{X}$ in $\mathbb{R}^{N+L}$, where Tr II is the trace of second fundamental form of $\mathrm{gr}(\boldsymbol{f})$. If we restrict to the submanifold of graphs of $\boldsymbol{f} \in \mathcal{M}'$, it is easy to calculate that the gradient of geometric penalty (4.5) is

$$\nabla \mathcal{P}_{G,\boldsymbol{f}} = V_{G,\boldsymbol{f}} = -\mathrm{Tr}\ \mathrm{II}^L, \tag{4.6}$$

where $\mathrm{Tr}\ \mathrm{II}^L$ denotes the last L components of Tr II.

The formulation given above is general in that it encompasses both the binary and the multiclass cases. For both cases, evaluation of $\mathrm{Tr}\ \mathrm{II}^L$ at any point $\boldsymbol{x}$ can be performed explicitly by the following theorem.

---

[1] recall that $\mathcal{M}' = \mathrm{Maps}(\mathcal{X}, \mathbb{R}^L)$ is the ambient space of $\mathcal{M} = \mathrm{Maps}(\mathcal{X}, \Delta^{L-1})$.

Theorem 1. *For* $\boldsymbol{f} : \mathbb{R}^N \to \Delta^{L-1}$*,* $\mathrm{Tr\ II}^L$ *for* $\mathrm{gr}(\boldsymbol{f})$ *is given by*

$$\mathrm{Tr\ II}^L \;=\; (g^{-1})^{ij}\bigg(f^1_{ji} - (g^{-1})^{rs} f^a_{rs} f^a_i f^1_j, \ldots, f^L_{ji} - (g^{-1})^{rs} f^a_{rs} f^a_i f^L_j\bigg), \tag{4.7}$$

where f^a_i, f^a_{ij} *denote partial derivatives of* f^a*.*

Proof. For $\boldsymbol{f} : \mathbb{R}^N \to \Delta^{L-1} \subset \mathbb{R}^L$,

$$\{r_j = r_j(\boldsymbol{x}) = (0, \ldots, \overset{j}{1}, \ldots, 0, f^1_j, \ldots, f^L_j) : j = 1, \ldots N\} \tag{4.8}$$

is a basis of the tangent space $T_{\boldsymbol{x}}\mathrm{gr}(\boldsymbol{f})$ to $\mathrm{gr}(\boldsymbol{f})$. Here $f^i_j = \partial_{x^j} f^i$. Let $\{e_i\}$ be an orthonormal frame of $T_{\boldsymbol{x}}\mathrm{gr}(\boldsymbol{f})$. We have

$$e_i = B^j_i r_j, \tag{4.9}$$

for some invertible matrix B^j_i.

Define the metric matrix g for the basis $\{r_j\}$ by

$$g = (g_{kj}) \text{ with } g_{kj} = r_k \cdot r_j = \delta^k_j + f^i_k f^i_j. \tag{4.10}$$

Then

$$\begin{aligned} \delta^i_j = e_i \cdot e_j = B^k_i B^t_j r_k \cdot r_t = B^k_i B^t_j g_{kt} \\ \Rightarrow I = (BB^T)g \Rightarrow BB^T = g^{-1}, \end{aligned} \tag{4.11}$$

thus BB^T is computable in terms of derivatives of $\boldsymbol{f}$.

Let $D_u w$ be the $\mathbb{R}^{N+L}$ directional derivative of w in the direction u, then

$$\mathrm{Tr\ II} \;=\; P^\nu D_{e_i} e_i = P^\nu D_{B^j_i r_j} B^k_i r_k = B^j_i P^\nu D_{r_j} B^k_i r_k$$

$$\begin{aligned} &= B_i^j P^\nu [(D_{r_j} B_i^k) r_k] + B_i^j B_i^k P^\nu D_{r_j} r_k \\ &= B_i^j B_i^k P^\nu D_{r_j} r_k \\ &= (g^{-1})^{jk} P^\nu D_{r_j} r_k, \end{aligned} \tag{4.12}$$

the equality in the second line holds since $P^\nu r_k = 0$.

From Eqn. 4.8, we have

$$\begin{aligned} r_k &= (0, \ldots, 1, \ldots, f_k^1(x^1, \ldots, x^N), \ldots, f_k^L(x^1, \ldots, x^N)) \\ &= \partial_k^{\mathbb{R}^{N+L}} + \sum_{i=1}^{L} f_k^i \partial_{N+i}^{\mathbb{R}^{N+L}}, \end{aligned} \tag{4.13}$$

so in particular, $\partial_\ell^{\mathbb{R}^{N+L}} r_k = 0$ if $\ell > N$. Thus

$$D_{r_j} r_k = (0, \ldots, \overset{N}{0}, f_{kj}^1, \ldots, f_{kj}^L). \tag{4.14}$$

So far, we have

$$\text{Tr II} = (g^{-1})^{jk} P^\nu (0, \ldots, \overset{N}{0}, f_{kj}^1, \ldots, f_{kj}^L). \tag{4.15}$$

Since g is given in terms of derivatives of $\boldsymbol{f}$, we need to write $P^\nu = I - P^T$ in terms of derivatives of $\boldsymbol{f}$. Here P^T is the projection to the tangent space of $\text{gr}(\boldsymbol{f})$. For any $u \in \mathbb{R}^{N+L}$, we have

$$\begin{aligned} P^T u &= (P^T u \cdot e_i) e_i = (u \cdot B_i^j r_j) B_i^k r_k \\ &= B_i^j B_i^k (u \cdot r_j) r_k \\ &= (g^{-1})^{jk} (u \cdot r_j) r_k. \end{aligned} \tag{4.16}$$

Thus

$$\begin{aligned}
\mathrm{Tr}\ \mathrm{II} &= (g^{-1})^{jk}P^{\nu}(0,\dots,\overset{N}{0},f^1_{kj},\dots,f^L_{kj})\\
&= (g^{-1})^{jk}(0,\dots,\overset{N}{0},f^1_{kj},\dots,f^L_{kj}) - P^T\big[(g^{-1})^{jk}(0,\dots,\overset{N}{0},f^1_{kj},\dots,f^L_{kj})\big]\\
&= (g^{-1})^{jk}(0,\dots,\overset{N}{0},f^1_{kj},\dots,f^L_{kj})\\
&\quad -(g^{-1})^{jk}\big[(g^{-1})^{rs}(0,\dots,\overset{N}{0},f^1_{rs},\dots,f^L_{rs})\cdot r_j\big]r_k\\
&= (g^{-1})^{jk}(0,\dots,\overset{N}{0},f^1_{kj},\dots,f^L_{kj}) - (g^{-1})^{jk}(g^{-1})^{rs}\left(f^i_{rs}f^i_j\right)r_k\\
&= (g^{-1})^{ij}\Bigg(0,\dots,\overset{j}{-(g^{-1})^{rs}f^a_{rs}f^a_i},\dots,0,\\
&\quad f^1_{ji}-(g^{-1})^{rs}f^a_{rs}f^a_i f^1_j,\dots,f^L_{ji}-(g^{-1})^{rs}f^a_{rs}f^a_i f^L_j\Bigg), \qquad (4.17)
\end{aligned}$$

after a relabeling of indices. Therefore, the last L component of Tr II are given by

$$\mathrm{Tr}\ \mathrm{II}^L = (g^{-1})^{ij}\Bigg(f^1_{ji}-(g^{-1})^{rs}f^a_{rs}f^a_i f^1_j,\dots,f^L_{ji}-(g^{-1})^{rs}f^a_{rs}f^a_i f^L_j\Bigg). \qquad (4.18)$$

□

4.2.3 The Simplex Constraint

An allowable class probability estimator $\boldsymbol{f} : X \to \Delta^{L-1}$ always takes values in $\Delta^{L-1} \subset \mathbb{R}^L$. Recall that in our notation, $\boldsymbol{f} \in \mathcal{M}'$. However, the gradient vector field $\nabla P_{G,f} \in T_{\boldsymbol{f}}\mathcal{M}'$ for our geometric penalty $\mathcal{P}_G$, i.e., the set of vector fields along $\boldsymbol{f}$ with values in $\mathbb{R}^L$, may not lie in $T_{\boldsymbol{f}}\mathcal{M}$, and in particular may not take values in $T\Delta^{L-1}$. There are two ways to enforce this constraint for the geometric gradient vector field. Firstly, since our initial function $\boldsymbol{f}_0$ takes values at the center of Δ^{L-1}, we can orthogonally project the geometric gradient vector $V_{G,\boldsymbol{f}}$ to $V'_{G,\boldsymbol{f}}$ in the tangent space $Z = \{(y^1,\dots,y^L) \in \mathbb{R}^L : \sum_{\ell=1}^{L} y^\ell = 0\}$ of the simplex, and then scale $\tau V'_{G,\boldsymbol{f}}$ (τ

is the stepsize) to ensure that the range of the new $\boldsymbol{f}$ lies in Δ^{L-1}. We then iterate.

To compute the projection of $V_{G,\boldsymbol{f}}$ to $T_{\boldsymbol{f}}\mathcal{M}$, let $\mathcal{P} : \mathbb{R}^L \to Z$ be the orthogonal projection. Then we claim that for a vector field X defined along $\mathrm{gr}(\boldsymbol{f})$, the projection $\mathcal{P}X$ of X into $T_{\boldsymbol{f}}\mathcal{M}$ is given by $X_x \mapsto PX_{\boldsymbol{x}}$ for $\boldsymbol{x} \in \mathcal{X}$. For the projection $\mathcal{P}$ by definition satisfies $\mathcal{P}X = \operatorname{argmin}\{\|X - Y\|_{L^2} : Y \in T_{\boldsymbol{f}}\mathcal{M}\}$. Since $PX_{\boldsymbol{x}} = \operatorname{argmin}\{|X_{\boldsymbol{x}} - Y_{\boldsymbol{x}}| : Y_{\boldsymbol{x}} \in Z\}$, we clearly have

$$\|X - Y\|_{L^2} = \int_{\mathcal{X}} |X_{\boldsymbol{x}} - Y_{\boldsymbol{x}}|^2 d\boldsymbol{x} \tag{4.19}$$

is minimized over $Y \in T_{\boldsymbol{f}}\mathcal{M}$ by $Y_{T_{\boldsymbol{f}}\mathcal{M}} = PX_{\boldsymbol{x}}$.

Take the following basis of Z,

$$\{h_1 = (1,-1,0,\ldots,0), h_2 = (1,0,-1,0,\ldots,0), \ldots, h_{L-1} = (1,0,\ldots,0,-1)\}. \tag{4.20}$$

For $\{e_i\}$ being an orthonormal basis of Z, we have $e_i = C_i^j h_j$ for some coefficients C_i^j. Then

$$\delta_j^i = e_i \cdot e_j = C_i^k C_j^\ell h_k \cdot h_\ell \Rightarrow CC^T = (h_i \cdot h_j)^{-1}. \tag{4.21}$$

Since

$$h_i \cdot h_j = \begin{cases} 2, & i = j, \\ 1, & i \neq j, \end{cases} \tag{4.22}$$

it is easy to compute $H = (h_i \cdot h_j)^{-1}$. Then we get

$$\begin{aligned} \mathcal{P}V_{G,\boldsymbol{f}} &= (V_{G,\boldsymbol{f}} \cdot e_i)e_i \\ &= (V_{G,\boldsymbol{f}} \cdot C_i^j h_j)C_i^k h_k \\ &= (CC^T)^{jk}(V_{G,\boldsymbol{f}} \cdot h_j)h_k \end{aligned}$$

$$= \quad (H^{-1})^{jk}(V_{G,\boldsymbol{f}} \cdot h_j)h_k. \qquad (4.23)$$

Secondly, to enforce this constraint for the geometric gradient vector field, we can also select $L-1$ of the L components of $\boldsymbol{f}(\boldsymbol{x})$, call the new function $\boldsymbol{f}' : \mathcal{X} \to R^{L-1}$, which is equivalent to projecting $\mathrm{gr}(\boldsymbol{f})$ to $\mathcal{X} \times \mathbb{R}^{K-1}$, and compute the $(L-1)$-dimensional gradient vector $V_{G,\boldsymbol{f}'}$ following (4.6) and Theorem 1. The omitted component of the desired L-gradient vector is determined by $-\sum_{\ell=1}^{L-1} V^{\ell}_{G,\boldsymbol{f}'}$, by the definition of tangent space Z. Our implementation follows this second approach, where we choose the $(L-1)$ components of $\boldsymbol{f}$ by omitting the component corresponding to the class with least number of training samples.

4.2.4 Algorithm Summary

Combining both the empirical term and the geometric term, the total gradient vector at point $\boldsymbol{x}$ can be computed by,

$$\nabla \mathcal{P}_{\boldsymbol{f}}(\boldsymbol{x}) = \nabla \mathcal{P}_{\mathcal{T}_m,\boldsymbol{f}}(\boldsymbol{x}) + \lambda \nabla \mathcal{P}_{G,\boldsymbol{f}}(\boldsymbol{x}), \qquad (4.24)$$

where λ is the trade-off parameter. Then a gradient flow step for updating $\boldsymbol{f}$ at point $\boldsymbol{x}$ is given by,

$$\boldsymbol{f}(\boldsymbol{x}) \longleftarrow \boldsymbol{f}(\boldsymbol{x}) - \tau \nabla \mathcal{P}_{\boldsymbol{f}}(\boldsymbol{x}), \qquad (4.25)$$

where τ is some step-size parameter. Note that for explaining the principles, we have been using the general non-parametric form of function $\boldsymbol{f}$ throughout this chapter. In practice, however, we will apply some parametric representation of $\boldsymbol{f}$, and Eqn.(4.25) will be converted to a formula updating some parameters based on the concrete representation of $\boldsymbol{f}$. Examples of representing $\boldsymbol{f}$ in the form of RBF functions and neural networks are discussed in detail in Chapter 6.

Algorithm 1 gives a summary of the classifier learning procedure with our geometric regularization. It is a general algorithm for any classification function $\boldsymbol{f}$ that satisfies two requirements: firstly, an estimator of the class probability can be obtained; secondly, first and second derivatives of the class probability estimator can be calculated. It is also unified for both binary and multiclass classification. The input to the algorithm is the training set $\mathcal{T}_m$, the trade-off parameter λ, and the step-size parameter τ. For initialization, our algorithm initializes the function values of $\boldsymbol{f}$ at every training point with equal probability to all classes. In the subsequent steps, at each iteration, our algorithm first evaluates the gradient vector field $\nabla\mathcal{P}_{\boldsymbol{f}}$ at every training point, then updates the estimator function $\boldsymbol{f}$ based on Eqn. (4.25) and the parametric representation of $\boldsymbol{f}$.

Algorithm 1 Geometric regularized classification

```
**Input:** training data $\mathcal{T}_m = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$, trade-off parameter $\lambda$, step-size $\tau$
**Initialize:** $\boldsymbol{f}(\boldsymbol{x}_i) = (\frac{1}{L}, \ldots, \frac{1}{L})$, $\forall i \in \{1, \cdots, m\}$
for $t = 1$ to $T$ do
    – Evaluate the total gradient vector $\nabla\mathcal{P}_{\boldsymbol{f}}(\boldsymbol{x}_i)$ at every training point according to Eqn. (4.24).
    – Update the $\boldsymbol{f}$ by (4.25).
end for
**Output:** class probability estimator $\boldsymbol{f}$.
```

While our approach is based on a solid Riemannian geometric foundation, the resulting algorithm, as shown in Algorithm 1, turns out to be simple and straightforward. And more importantly, it is easily parallelizable, regardless of the parametric representation of $\boldsymbol{f}$, given that the computation of the gradient vector at every training point is independent of each other.

Chapter 5

Theoretical Analysis

In this chapter, we discuss some theoretical aspects of the proposed approach. In particular, the consistency analysis relates our approach with statistical learning theory and shows that under some mild initialization assumptions, our algorithm converges to the statistically optimal classifier (the Bayes classifier) with the number of training examples goes to infinity. This is a basic requirement for a good learning algorithm in a statistical sense. Another fundamental problem for applying our algorithm is the existence of the solutions of the gradient flow equation (3.9). Since our iterative algorithm in essence follows the negative gradient flow line of our penalty function defined on the functional space, without the existence guarantee of the flow line for at least a short time, in theory, we cannot argue that our learning algorithm follows exactly our geometric intuition described in §3.2 with a physical model. To provide theoretical insights on further exploring the potential of the proposed framework in a wider range of applications, we also discuss extensions of our approach to incorporate a Riemannian curvature based geometric penalty and nonlinear input spaces.

Roadmap for this chapter

In §5.1, we establish Bayes consistency of our geometric regularization scheme for a particular empirical loss function, under some mild initialization assumptions. In

§5.2, we describe the theory and formulation of an alternative geometric regularizer based on the Riemannian curvature. In §5.3, we discuss the existence of the gradient flow under different topologies, and show that the existence and uniqueness of solutions for parametric representation of $\boldsymbol{f}$ is guaranteed, which provides a foundation for algorithms applied in the next chapter. In §5.4, we discuss the extension of our geometric perspective and regularization approach to situations where the input space is a submanifold with local charts, rather than a Euclidean space.

5.1 Bayes Consistency

For a training set $\mathcal{T}_m$ of size m, we let $\boldsymbol{f}_{\mathcal{T}_m} = (f^1_{\mathcal{T}_m}, \ldots, f^L_{\mathcal{T}_m})$ be the class probability estimator given by our approach. Recall that the generalization risk of the corresponding plug-in classifier $h_{\boldsymbol{f}_{\mathcal{T}_m}}$ is $R_P(\boldsymbol{f}_{\mathcal{T}_m}) = \mathbb{E}_P[\mathbb{1}_{h_{\boldsymbol{f}_{\mathcal{T}_m}}(\boldsymbol{x}) \neq y}]$. The Bayes risk is defined by $R^*_P = \inf_{h:\mathcal{X}\to\mathcal{Y}} R_P(h) = \mathbb{E}_P[\mathbb{1}_{h_{\boldsymbol{\eta}}(\boldsymbol{x}) \neq y}]$. Our algorithm is Bayes consistent if $\lim_{m\to\infty} R_P(\boldsymbol{f}_{\mathcal{T}_m}) = R^*_P$ holds in probability for all distributions P on $\mathcal{X} \times \mathcal{Y}$.

While showing Bayes consistency for general empirical terms is hard for our regularized scheme, we give an example with an empirical loss that enables Bayes consistency proof under some mild initialization assumptions. Related notation is summarized in Table 5.1.

For ease of reading, we change the notation for empirical penalty $\mathcal{P}_{\mathcal{T}_m}$ in this section to $\mathcal{P}_D$, i.e., $\mathcal{P} = \mathcal{P}_D + \lambda\mathcal{P}_G$. $\mathcal{P}_D$ measures the deviation of $\mathrm{gr}(\boldsymbol{f})$ from the mapped training points. A natural geometric distance penalty term is an L^2 distance in $\mathbb{R}^L$ from $\boldsymbol{f}(\boldsymbol{x})$ to the averaged $\boldsymbol{z}$ component of the k-nearest training points for a fixed choice of k:

$$\mathcal{P}_D(\boldsymbol{f}) = R_{D,\mathcal{T}_m,k}(\boldsymbol{f}) = \int_{\mathcal{X}} d^2\left(\boldsymbol{f}(\boldsymbol{x}), \frac{1}{k}\sum_{i=1}^{k} \tilde{\boldsymbol{z}}_i\right) d\boldsymbol{x}, \tag{5.1}$$

Table 5.1: Notation for Bayes consistency

$h_{\boldsymbol{f}}(\boldsymbol{x}) = \operatorname*{argmax}_{\ell\in\mathcal{Y}} f^{\ell}(\boldsymbol{x})$: plug-in classifier of $\boldsymbol{f} : \mathcal{X} \to \Delta^{L-1}$
$\mathbb{1}_{h_{\boldsymbol{f}}(\boldsymbol{x})\neq y} = \begin{cases} 1, & h_{\boldsymbol{f}}(\boldsymbol{x}) \neq y \\ 0, & h_{\boldsymbol{f}}(\boldsymbol{x}) = y \end{cases}$
$R_P(\boldsymbol{f}) = \mathbb{E}_P[\mathbb{1}_{h_{\boldsymbol{f}}(\boldsymbol{x})\neq y}]$: generalization risk for the estimator $\boldsymbol{f}$
$R_P^* = R_P(\boldsymbol{\eta})$: Bayes risk
$R_{D,P}(\boldsymbol{f}) = \int_{\mathcal{X}} d^2(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{\eta}(\boldsymbol{x}))d\boldsymbol{x}$: D-risk
$R_{D,\mathcal{T}_m}(\boldsymbol{f}) = R_{D,\mathcal{T}_m,k}(\boldsymbol{f}) = \int_{\mathcal{X}} d^2 \left(\boldsymbol{f}(\boldsymbol{x}), \frac{1}{k}\sum_{i=1}^{k} \tilde{\boldsymbol{z}}_i\right) d\boldsymbol{x}$: empirical D-risk, where $\tilde{\boldsymbol{z}}_i$ is the vector of the last L components of $(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{z}}_i)$, with $\tilde{\boldsymbol{x}}_i$ the i^{th} nearest neighbor of $\boldsymbol{x}$ in $\mathcal{T}_m$
$R_{D,P,\lambda}(\boldsymbol{f}) = R_{D,P}(\boldsymbol{f}) + \lambda\mathcal{P}_G(\boldsymbol{f})$: regularized D-risk for estimator $\boldsymbol{f}$
$R_{D,\mathcal{T}_m,\lambda}(\boldsymbol{f}) = R_{D,\mathcal{T}_m}(\boldsymbol{f}) + \lambda\mathcal{P}_G(\boldsymbol{f})$: regularized empirical D-risk for estimator $\boldsymbol{f}$
$\boldsymbol{f}_{D,P,\lambda}$: function attaining the global minimum for $R_{D,P,\lambda}$
$R^*_{D,P,\lambda} = R_{D,P,\lambda}(\boldsymbol{f}_{D,P,\lambda})$: minimum value for $R_{D,P,\lambda}$
$\boldsymbol{f}_{D,\mathcal{T}_m,\lambda} = \boldsymbol{f}_{D,\mathcal{T}_m,k,\lambda}$: function attaining the global minimum for $R_{D,\mathcal{T}_m,\lambda}(\boldsymbol{f})$, note that we assume $\boldsymbol{f}_{D,P,\lambda}$ and $\boldsymbol{f}_{D,\mathcal{T}_m,\lambda}$ exist

where d is the Euclidean distance in $\mathbb{R}^L$, $\tilde{\boldsymbol{z}}_i$ is the vector of the last L components of $(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{z}}_i) = (\tilde{\boldsymbol{x}}_i^1, \ldots, \tilde{\boldsymbol{x}}_i^N, \tilde{\boldsymbol{z}}_i^1, \ldots, \tilde{\boldsymbol{z}}_i^L)$, with $\tilde{\boldsymbol{x}}_i$ the i^{th} nearest neighbor of $\boldsymbol{x}$ in $\mathcal{T}_m$, and $d\boldsymbol{x}$ is the Lebesgue measure. The gradient vector field is

$$\nabla(R_{D,\mathcal{T}_m,k})_{\boldsymbol{f}}(\boldsymbol{x}, \boldsymbol{f}(\boldsymbol{x})) = \frac{2}{k}\sum_{i=1}^{k}(\boldsymbol{f}(\boldsymbol{x}) - \tilde{\boldsymbol{z}}_i). \tag{5.2}$$

However, $\nabla(R_{D,\mathcal{T}_m,k})_{\boldsymbol{f}}$ is discontinuous on the set $\mathcal{D}$ of points $\boldsymbol{x}$ such that $\boldsymbol{x}$ has equidistant training points among its k nearest neighbors. $\mathcal{D}$ is the union of $(N-1)$-dimensional hyperplanes in $\mathcal{X}$, so $\mathcal{D}$ has measure zero. Such points will necessarily exist unless the last L components of the mapped training points are all 1 or all 0. To rectify this, we can smooth out $\nabla(R_{D,\mathcal{T}_m,k})_{\boldsymbol{f}}$ to a vector field

$$V_{D,\boldsymbol{f},\phi} = \frac{2\phi(\boldsymbol{x})}{k}\sum_{i=1}^{k}(\boldsymbol{f}(\boldsymbol{x}) - \tilde{\boldsymbol{z}}_i). \tag{5.3}$$

Here $\phi(\boldsymbol{x})$ is a smooth damping function close to the singular function $\delta_{\mathcal{D}}$, which has $\delta_{\mathcal{D}}(\boldsymbol{x}) = 0$ for $\boldsymbol{x} \in \mathcal{D}$ and $\delta_{\mathcal{D}}(\boldsymbol{x}) = 1$ for $\boldsymbol{x} \notin \mathcal{D}$. Outside any open neighborhood of $\mathcal{D}$, $\nabla R_{D,\mathcal{T}_m,k} = V_{D,\boldsymbol{f},\phi}$ for ϕ close enough to $\delta_{\mathcal{D}}$.

Recall the geometric penalty term, i.e., $\mathcal{P}_G(\boldsymbol{f}) = \int_{\mathrm{gr}(\boldsymbol{f})} \mathrm{dvol}$, with the geometric gradient vector field being $V_{G,\boldsymbol{f}} = -\mathrm{Tr}\ \mathrm{II}^L$.

Then the gradient vector field $V_{tot,\lambda,m,\boldsymbol{f},\phi}$ of this example penalty $\mathcal{P}$ is,

$$\begin{aligned} V_{tot,\lambda,m,\boldsymbol{f},\phi} &= \nabla \mathcal{P}_f = V_{D,\boldsymbol{f},\phi} + \lambda V_{G,\boldsymbol{f}} \\ &= \frac{2\phi(\boldsymbol{x})}{k} \sum_{i=1}^{k} (\boldsymbol{f}(\boldsymbol{x}) - \tilde{\boldsymbol{z}}_i) - \lambda \mathrm{Tr}\ \mathrm{II}^L. \end{aligned} \tag{5.4}$$

Usually, gradient flow methods are applied to a convex functional, so that a flow line approaches the unique global minimum. If the domain of the functional is an infinite dimensional manifold of (e.g. smooth) functions, we always assume that flow lines exist and that the actual minimum exists in this manifold.

Because our functionals are not convex, we can only hope to prove Bayes consistency for the set of initial estimators in the stable manifold of a global minimum point (or sink) of the vector field [34]. Recall that a stable fixed point $\boldsymbol{f}_0$ has a maximal open neighborhood, the stable manifold $\mathcal{S}_{\boldsymbol{f}_0}$, on which flow lines tend towards $\boldsymbol{f}_0$. For the manifold $\mathcal{M}$, the stable manifold for a stable critical point of the vector field $V_{tot,\lambda,m,\boldsymbol{f},\phi}$ is infinite dimensional.

The proof of Bayes consistency for multiclass (including binary) classification follows these steps:

Step 1: $\lim\limits_{\lambda \to 0} R^*_{D,P,\lambda} = 0$.

Step 2: $\lim\limits_{n \to \infty} R_{D,P}(\boldsymbol{f}_n) = 0 \Rightarrow \lim\limits_{n \to \infty} R_P(\boldsymbol{f}_n) = R^*_P$.

Step 3: For all $\boldsymbol{f} \in \mathcal{M} = \mathrm{Maps}(\mathcal{X}, \Delta^{L-1})$, $|R_{D,\mathcal{T}_m}(\boldsymbol{f}) - R_{D,P}(\boldsymbol{f})| \xrightarrow{m\to\infty} 0$ in probability.

Proofs of these steps are provided below. In the notation of Table 5.1, $R^*_{D,P,\lambda}$ is the minimum of the regularized D risk $R_{D,P,\lambda}(\boldsymbol{f})$ for $\boldsymbol{f}$:

$$R_{D,P,\lambda}(\boldsymbol{f}) = R_{D,P}(\boldsymbol{f}) + \lambda \mathcal{P}_G(\boldsymbol{f}), \tag{5.5}$$

with $R_{D,P}(\boldsymbol{f}) = \int_{\mathcal{X}} d^2(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{\eta}(\boldsymbol{x}))d\boldsymbol{x}$ being the D-risk. Also,

$$R_{D,\mathcal{T}_m,\lambda}(\boldsymbol{f}) = R_{D,\mathcal{T}_m}(\boldsymbol{f}) + \lambda \mathcal{P}_G(\boldsymbol{f}), \tag{5.6}$$

with $R_{D,\mathcal{T}_m}(\boldsymbol{f}) = \int_{\mathcal{X}} d^2 \left(\boldsymbol{f}(\boldsymbol{x}), \frac{1}{k}\sum_{i=1}^k \tilde{\boldsymbol{z}}_i\right) d\boldsymbol{x}$ being the empirical D-risk.

Theorem 2 (Bayes Consistency)**.** *Let m be the size of the training data set. Let $\boldsymbol{f}_{1,\lambda,m} \in \mathcal{S}_{\boldsymbol{f}_{D,\mathcal{T}_m,\lambda}}$, the stable manifold for the global minimum $\boldsymbol{f}_{D,\mathcal{T}_m,\lambda}$ of $R_{D,\mathcal{T}_m,\lambda}$, and let $\boldsymbol{f}_{n,\lambda,m,\phi}$ be a sequence of functions on the flow line of $V_{tot,\lambda,m,\boldsymbol{f},\phi}$ starting with $\boldsymbol{f}_{1,\lambda,m}$ with the flow time $t_n \to \infty$ as $n \to \infty$. Then $R_P(\boldsymbol{f}_{n,\lambda,m,\phi}) \xrightarrow[\lambda\to 0,\phi\to\delta_{\mathcal{D}}]{m,n\to\infty} R^*_P$ in probability for all distributions P on $\mathcal{X} \times \mathcal{Y}$, if $k/m \to 0$ as $m \to \infty$.*

Proof. If $\boldsymbol{f}_{D,\mathcal{T}_m,\lambda}$ is a global minimum for $R_{D,\mathcal{T}_m,\lambda}$, then outside of $\mathcal{D}$, $\boldsymbol{f}_{D,\mathcal{T}_m,\lambda}$ is the limit of critical points for the negative flow of $V_{tot,\lambda,m,\boldsymbol{f},\phi}$ as $\phi \to \delta_{\mathcal{D}}$. To see this, fix an ϵ_i neighborhood $\mathcal{D}_{\epsilon_i}$ of $\mathcal{D}$. For a sequence $\phi_j \to \delta_{\mathcal{D}}$, $V_{tot,\lambda,m,f,\phi_j}$ is independent of $j \geq j(\epsilon_i)$ on $\mathcal{X} \setminus \mathcal{D}_{\epsilon_i}$, so we find a function $\boldsymbol{f}_i$, a critical point of $V_{tot,\lambda,m,\boldsymbol{f},\phi_{j(\epsilon_i)}}$, equal to $\boldsymbol{f}_{D,\mathcal{T}_m,\lambda}$ on $\mathcal{X} \setminus \mathcal{D}_{\epsilon_i}$. Since any $\boldsymbol{x} \notin \mathcal{D}$ lies outside some $\mathcal{D}_{\epsilon_i}$, the sequence $\boldsymbol{f}_i$ converges at $\boldsymbol{x}$ if we let $\epsilon_i \to 0$. Thus, we can ignore the choice of ϕ in our proof, and drop ϕ from the notation.

For our algorithm, for fixed λ, m, we have as above $\lim\limits_{n\to\infty} \boldsymbol{f}_{n,\lambda,m} = \boldsymbol{f}_{D,\mathcal{T}_m,\lambda}$, so

$$\lim_{n\to\infty} R_{D,\mathcal{T}_m,\lambda}(\boldsymbol{f}_{n,\lambda,m}) = R_{D,\mathcal{T}_m,\lambda}(\boldsymbol{f}_{D,\mathcal{T}_m,\lambda}), \tag{5.7}$$

for $\boldsymbol{f}_1 \in \mathcal{S}_{\boldsymbol{f}_{D,\mathcal{T}_m,\lambda}}$. By Step 2, it suffices to show $R_{D,P}(\boldsymbol{f}_{D,\mathcal{T}_m,\lambda}) \xrightarrow[\lambda\to 0]{m\to\infty} 0$. In probability, we have $\forall\delta > 0, \exists m > 0$ such that

$$\begin{aligned}
0 &\leq R_{D,P}(\boldsymbol{f}_{D,\mathcal{T}_m,\lambda}) \\
&\leq R_{D,P}(\boldsymbol{f}_{D,\mathcal{T}_m,\lambda}) + \lambda\mathcal{P}_G(\boldsymbol{f}_{D,\mathcal{T}_m,\lambda}) \\
&\leq R_{D,\mathcal{T}_m}(\boldsymbol{f}_{D,\mathcal{T}_m,\lambda}) + \lambda\mathcal{P}_G(\boldsymbol{f}_{D,\mathcal{T}_m,\lambda}) + \frac{\delta}{3} \quad \text{(Step 3)} \\
&= R_{D,\mathcal{T}_m,\lambda}(\boldsymbol{f}_{D,\mathcal{T}_m,\lambda}) + \frac{\delta}{3} \\
&\leq R_{D,\mathcal{T}_m,\lambda}(\boldsymbol{f}_{D,P,\lambda}) + \frac{\delta}{3} \quad \text{(minimality of } \boldsymbol{f}_{D,\mathcal{T}_m,\lambda}) \\
&= R_{D,\mathcal{T}_m}(\boldsymbol{f}_{D,P,\lambda}) + \lambda\mathcal{P}_G(\boldsymbol{f}_{D,P,\lambda}) + \frac{\delta}{3} \\
&\leq R_{D,P}(\boldsymbol{f}_{D,P,\lambda}) + \lambda\mathcal{P}_G(\boldsymbol{f}_{D,P,\lambda}) + \frac{2\delta}{3} \text{ (Step 3)} \\
&= R_{D,P,\lambda}(\boldsymbol{f}_{D,P,\lambda}) + \frac{2\delta}{3} = R^*_{D,P,\lambda} + \frac{2\delta}{3} \\
&\leq \delta, \quad \text{(Step 1)}
\end{aligned} \tag{5.8}$$

for λ close to zero. □

The following subsections for proving the three steps are taken from the supplemental materials of [5].

5.1.1 Step 1

Lemma 3. *(Step 1)* $\lim_{\lambda\to 0} R^*_{D,P,\lambda} = 0$.

Proof. After the smoothing procedure in §3.1 for the distance penalty term, the function $R_{D,P,\lambda} : \mathcal{M} \to \mathbb{R}$ is continuous in the Fréchet topology on $\mathcal{M}$. We check that the functions $R_{D,P,\lambda} : \mathcal{M} \to \mathbb{R}$ are equicontinuous in λ: for fixed $\boldsymbol{f}_0 \in \mathcal{M}$ and $\epsilon > 0$, there exists $\delta = \delta(\boldsymbol{f}_0, \epsilon)$ such that $|\lambda - \lambda'| < \delta \Rightarrow |R_{D,P,\lambda}(\boldsymbol{f}_0) - R_{D,P,\lambda'}(\boldsymbol{f}_0)| < \epsilon$.

This is immediate:

$$|R_{D,P,\lambda}(\boldsymbol{f}_0) - R_{D,P,\lambda'}(\boldsymbol{f}_0)| = |(\lambda - \lambda')\mathcal{P}_G(\boldsymbol{f}_0)| < \epsilon, \tag{5.9}$$

if $\delta < \epsilon/|\mathcal{P}_G(\boldsymbol{f}_0)|$. It is standard that the infimum $\inf R_\lambda$ of an equicontinuous family of functions is continuous in λ, so $\lim_{\lambda\to 0} R^*_{D,P,\lambda} = R^*_{D,P,\lambda=0} = R_{D,P}(\boldsymbol{\eta}) = 0$. □

5.1.2 Step 2

We assume that the class probability function $\boldsymbol{\eta}(\boldsymbol{x}) : \mathbb{R}^N \to \mathbb{R}^L$ is smooth, and that the marginal distribution $\mu(\boldsymbol{x})$ is continuous. We also let μ denote the corresponding measure on $\mathcal{X}$.

Denote: $h_{\boldsymbol{f}}(\boldsymbol{x}) = \text{argmax}\{f^\ell(\boldsymbol{x}), \ell \in \mathcal{Y}\}$, and,

$$\mathbb{1}_{h_{\boldsymbol{f}}(\boldsymbol{x})\neq y} = \begin{cases} 1, & h_{\boldsymbol{f}}(\boldsymbol{x}) \neq y, \\ 0, & h_{\boldsymbol{f}}(\boldsymbol{x}) = y. \end{cases} \tag{5.10}$$

Lemma 4. *(Step 2 for a subsequence)*

$$\lim_{n\to\infty} R_{D,P}(\boldsymbol{f}_n) = 0 \Rightarrow \lim_{i\to\infty} R_P(\boldsymbol{f}_{n_i}) = R^*_P$$

for some subsequence $\{\boldsymbol{f}_{n_i}\}_{i=1}^\infty$ *of* $\{\boldsymbol{f}_n\}$.

Proof. The left hand side of the Lemma is

$$\int_{\mathcal{X}} d^2(\boldsymbol{f}_n(\boldsymbol{x}), \boldsymbol{\eta}(\boldsymbol{x}))d\boldsymbol{x} \to 0, \tag{5.11}$$

which is equivalent to

$$\int_{\mathcal{X}} d^2(\boldsymbol{f}_n(\boldsymbol{x}), \boldsymbol{\eta}(\boldsymbol{x}))\mu(\boldsymbol{x})d\boldsymbol{x} \to 0, \tag{5.12}$$

since $\mathcal{X}$ is compact and μ is continuous. Therefore, it suffices to show

$$\begin{aligned} &\int_{\mathcal{X}} d^2(\boldsymbol{f}_n(\boldsymbol{x}), \boldsymbol{\eta}(\boldsymbol{x}))\mu(\boldsymbol{x})d\boldsymbol{x} \to 0 \\ &\implies \mathbb{E}_P[\mathbb{1}_{h_{\boldsymbol{f}_n}(\boldsymbol{x})\neq y}] \to \mathbb{E}_P[\mathbb{1}_{h_{\boldsymbol{\eta}}(\boldsymbol{x})\neq y}]. \end{aligned} \tag{5.13}$$

We recall that L^2 convergence implies pointwise convergence a.e, so (5.12) implies that a subsequence of $\boldsymbol{f}_n$, also denoted $\boldsymbol{f}_n$, has $\boldsymbol{f}_n \to \boldsymbol{\eta}(\boldsymbol{x})$ pointwise a.e. on $\mathcal{X}$. (By our assumption on $\mu(\boldsymbol{x})$, these statements hold for either μ or Lebesgue measure.) By Egorov's theorem, for any $\epsilon > 0$, there exists a set $B_\epsilon \subset \mathcal{X}$ with $\mu(B_\epsilon) < \epsilon$ such that $\boldsymbol{f}_n \to \boldsymbol{\eta}(\boldsymbol{x})$ uniformly on $\mathcal{X} \setminus B_\epsilon$.

Fix $\delta > 0$ and set

$$Z_\delta = \{\boldsymbol{x} \in \mathcal{X} : \#\{\operatorname*{argmax}_{\ell\in\mathcal{Y}} \eta^\ell(\boldsymbol{x})\} = 1, |\max_{\ell\in\mathcal{Y}} \eta^\ell(\boldsymbol{x}) - \operatorname*{submax}_{\ell\in\mathcal{Y}} \eta^\ell(\boldsymbol{x})| < \delta\}, \tag{5.14}$$

where $\operatorname*{submax}_{\ell\in\mathcal{Y}}$ denotes the second largest element in $\{\eta^1(\boldsymbol{x}), \ldots, \eta^L(\boldsymbol{x})\}$. For the moment, assume that $Z_0 = \{\boldsymbol{x} \in \mathcal{X} : \#\{\operatorname*{argmax}_{\ell\in\mathcal{Y}} \eta^\ell(\boldsymbol{x})\} > 1\}$ has $\mu(Z_0) = 0$.

It follows easily[1] that $\mu(Z_\delta) \to 0$ as $\delta \to 0$. On $\mathcal{X} \setminus (Z_\delta \cup B_\epsilon)$, we have $\mathbb{1}_{h_{\boldsymbol{f}_n}(\boldsymbol{x})\neq y} =$

---

[1] Let A_k be sets with $A_{k+1} \subset A_k$ and with $\mu(\cap_{k=1}^\infty A_k) = 0$. If $\mu(A_k) \not\to 0$, then there exists a subsequence, also called A_k, with $\mu(A_k) > K > 0$ for some K. We claim $\mu(\cap A_k) \geq K$, a contradiction. For the claim, let $Z = \cap A_k$. If $\mu(Z) \geq \mu(A_k)$ for all k, we are done. If not, since the A_k are nested, we can replace A_k by a set, also called A_k, of measure K and such that the new A_k are still nested. For the relabeled $Z = \cap A_k$, $Z \subset A_k$ for all k, and we may assume $\mu(Z) < K$. Thus there exists $Z' \subset A_1$ with $Z' \cap Z = \emptyset$ and $\mu(Z') > 0$. Since $\mu(A_i) = K$, we must have $A_i \cap Z' \neq \emptyset$ for all i. Thus $\cap A_i$ is strictly larger than Z, a contradiction. In summary, the claim must hold, so we get a contradiction to assuming $\mu(A_k) \not\to 0$.

$\mathbb{1}_{h_{\boldsymbol{\eta}}(\boldsymbol{x})\neq y}$ for $n > N_\delta$. Thus

$$\mathbb{E}_P[\mathbb{1}_{\mathcal{X}\setminus(Z_\delta\cup B_\epsilon)}\mathbb{1}_{h_{\boldsymbol{f}_n}(\boldsymbol{x})\neq y}] = \mathbb{E}_P[\mathbb{1}_{\mathcal{X}\setminus(Z_\delta\cup B_\epsilon)}\mathbb{1}_{h_{\boldsymbol{\eta}}(\boldsymbol{x})\neq y}]. \tag{5.15}$$

(Here $\mathbb{1}_A$ is the characteristic function of a set A.)

As $\delta \to 0$,

$$\mathbb{E}_P[\mathbb{1}_{\mathcal{X}\setminus(Z_\delta\cup B_\epsilon)}\mathbb{1}_{h_{\boldsymbol{f}_n}(\boldsymbol{x})\neq y}] \to \mathbb{E}_P[\mathbb{1}_{\mathcal{X}\setminus B_\epsilon}\mathbb{1}_{h_{\boldsymbol{f}_n}(\boldsymbol{x})\neq y}]. \tag{5.16}$$

and similarly for $\boldsymbol{f}_n$ replaced by $\boldsymbol{\eta}(\boldsymbol{x})$. During this process, N_δ presumably goes to ∞, but that precisely means

$$\lim_{n\to\infty} \mathbb{E}_P[\mathbb{1}_{\mathcal{X}\setminus B_\epsilon}\mathbb{1}_{h_{\boldsymbol{f}_n}(\boldsymbol{x})\neq y}] = \mathbb{E}_P[\mathbb{1}_{\mathcal{X}\setminus B_\epsilon}\mathbb{1}_{h_{\boldsymbol{\eta}}(\boldsymbol{x})\neq y}]. \tag{5.17}$$

Since

$$\left|\mathbb{E}_P[\mathbb{1}_{\mathcal{X}\setminus B_\epsilon}\mathbb{1}_{h_{\boldsymbol{f}_n}(\boldsymbol{x})\neq y}] - \mathbb{E}_P[\mathbb{1}_{h_{\boldsymbol{f}_n}(\boldsymbol{x})\neq y}]\right| < \epsilon, \tag{5.18}$$

and similarly for $\boldsymbol{\eta}(\boldsymbol{x})$, we get

$$\begin{aligned}
&\left|\lim_{n\to\infty} \mathbb{E}_P[\mathbb{1}_{h_{\boldsymbol{f}_n}(\boldsymbol{x})\neq y}] - \mathbb{E}_P[\mathbb{1}_{h_{\boldsymbol{\eta}}(\boldsymbol{x})\neq y}]\right| \\
\leq\ & \left|\lim_{n\to\infty} \mathbb{E}_P[\mathbb{1}_{h_{\boldsymbol{f}_n}(\boldsymbol{x})\neq y}] - \lim_{n\to\infty} \mathbb{E}_P[\mathbb{1}_{\mathcal{X}\setminus B_\epsilon}\mathbb{1}_{h_{\boldsymbol{f}_n}(\boldsymbol{x})\neq y}]\right| \\
& \quad + \left|\lim_{n\to\infty} \mathbb{E}_P[\mathbb{1}_{\mathcal{X}\setminus B_\epsilon}\mathbb{1}_{h_{\boldsymbol{f}_n}(\boldsymbol{x})\neq y}] - \mathbb{E}_P[\mathbb{1}_{\mathcal{X}\setminus B_\epsilon}\mathbb{1}_{h_{\boldsymbol{\eta}}(\boldsymbol{x})\neq y}]\right| \\
& \quad + \left|\lim_{n\to\infty} \mathbb{E}_P[\mathbb{1}_{\mathcal{X}\setminus B_\epsilon}\mathbb{1}_{h_{\boldsymbol{\eta}}(\boldsymbol{x})\neq y}] - \mathbb{E}_P[\mathbb{1}_{h_{\boldsymbol{\eta}}(\boldsymbol{x})\neq y}]\right| \\
\leq\ & 3\epsilon.
\end{aligned} \tag{5.19}$$

(Strictly speaking, $\lim_{n\to\infty} \mathbb{E}_P[\mathbb{1}_{h_{\boldsymbol{f}_n}(\boldsymbol{x})\neq y}]$ is first $\limsup$ and then $\liminf$ to show that the limit exists.) Since ϵ is arbitrary, the proof is complete if $\mu(Z_0) = 0$.

If $\mu(Z_0) > 0$, we rerun the proof with $\mathcal{X}$ replaced by Z_0. As above, $\boldsymbol{f}_n|_{Z_0}$ converges uniformly to $\boldsymbol{\eta}(\boldsymbol{x})$ off a set of measure ϵ. The argument above, without the set Z_δ, gives

$$\int_{Z_0} \mathbb{1}_{h_{\boldsymbol{f}_n}(\boldsymbol{x})\neq y}\mu(\boldsymbol{x})d\boldsymbol{x} \to \int_{Z_0} \mathbb{1}_{h_{\boldsymbol{\eta}}(\boldsymbol{x})\neq y}\mu(\boldsymbol{x})d\boldsymbol{x}. \tag{5.20}$$

We then proceed with the proof above on $\mathcal{X} \setminus Z_0$. □

Corollary 5. *(Step 2 in general) For our algorithm,* $\lim_{n\to\infty} R_{D,P}(\boldsymbol{f}_{n,\lambda,m}) = 0 \Rightarrow \lim_{i\to\infty} R_P(\boldsymbol{f}_{n,\lambda,m}) = R_P^*$.

Proof. Choose $\boldsymbol{f}_{1,\lambda,m}$ as in Theorem 2. Since $V_{tot,\lambda,m,\boldsymbol{f}_{n,\lambda,m}}$ has pointwise length going to zero as $n \to \infty$, $\{\boldsymbol{f}_{n,\lambda,m}(\boldsymbol{x})\}$ is a Cauchy sequence for all $\boldsymbol{x}$. This implies that $\boldsymbol{f}_{n,\lambda,m}$, and not just a subsequence, converges pointwise to $\boldsymbol{\eta}$. □

5.1.3 Step 3

Lemma 6. *(Step 3) If* $k \to \infty$ *and* $k/m \to 0$ *as* $m \to \infty$*, then for* $\boldsymbol{f} \in \mathrm{Maps}(\mathcal{X}, \Delta^{L-1})$,

$$|R_{D,\mathcal{T}_m}(\boldsymbol{f}) - R_{D,P}(\boldsymbol{f})| \xrightarrow{m\to\infty} 0 \quad \textit{in probability},$$

for all distributions P *that generate* $\mathcal{T}_m$.

Proof. Since $R_{D,P}(\boldsymbol{f})$ is a constant for fixed $\boldsymbol{f}$ and P, convergence in probability will follow from weak convergence, i.e.,

$$\mathbb{E}_{\mathcal{T}_m}[|R_{D,\mathcal{T}_m}(\boldsymbol{f}) - R_{D,P}(\boldsymbol{f})|] \xrightarrow{m\to\infty} 0. \tag{5.21}$$

We have

$$\begin{aligned} & |R_{D,\mathcal{T}_m}(\boldsymbol{f}) - R_{D,P}(\boldsymbol{f})| \\ = \; & \left| \int_{\mathcal{X}} \left[d^2\left(\boldsymbol{f}(\boldsymbol{x}), \frac{1}{k}\sum_{i=1}^{k} \tilde{\boldsymbol{z}}_i\right) - d^2(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{\eta}(\boldsymbol{x}))\right] d\boldsymbol{x} \right| \\ \leq \; & \int_{\mathcal{X}} \left| d^2\left(\boldsymbol{f}(\boldsymbol{x}), \frac{1}{k}\sum_{i=1}^{k} \tilde{\boldsymbol{z}}_i\right) - d^2(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{\eta}(\boldsymbol{x}))\right| d\boldsymbol{x}. \end{aligned} \tag{5.22}$$

Set $\boldsymbol{a} = \boldsymbol{f}(\boldsymbol{x}) - \frac{1}{k}\sum_{i=1}^{k} \tilde{\boldsymbol{z}}_i,\ \ \boldsymbol{b} = \boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{\eta}(\boldsymbol{x})$. Then

$$\begin{aligned}
& \left| \|\boldsymbol{a}\|_2^2 - \|\boldsymbol{b}\|_2^2 \right| \\
= & \left| \sum_{\ell=1}^{L} a_\ell^2 - \sum_{\ell=1}^{L} b_\ell^2 \right| = \left| \sum_{\ell=1}^{L} (a_\ell^2 - b_\ell^2) \right| \\
\leq & \sum_{\ell=1}^{L} |a_\ell^2 - b_\ell^2| \leq 2 \sum_{\ell=1}^{L} |a_\ell - b_\ell| \max\{|a_\ell|, |b_\ell|\} \\
\leq & \ 2 \sum_{\ell=1}^{L} |a_\ell - b_\ell|, \qquad (5.23)
\end{aligned}$$

since $f^\ell(\boldsymbol{x}), \frac{1}{k}\sum_i^k \tilde{z}_i^\ell, \eta^\ell(\boldsymbol{x}) \in [0,1]$. Therefore, it suffices to show that

$$\sum_{\ell=1}^{L} \mathbb{E}_{\mathcal{T}_m} \left[\int_{\mathcal{X}} \left| ((f^\ell(\boldsymbol{x}) - \frac{1}{k}\sum_i^k \tilde{z}_i^\ell) - (f^\ell(\boldsymbol{x}) - \eta^\ell(\boldsymbol{x})) \right| d\boldsymbol{x} \right] \xrightarrow{m\to\infty} 0, \qquad (5.24)$$

so the result follows if

$$\lim_{m\to\infty} \mathbb{E}_{\mathcal{T}_m,\boldsymbol{x}} \left[\left| \eta^\ell(\boldsymbol{x}) - \frac{1}{k}\sum_i^k \tilde{z}_i^\ell \right| \right] = 0 \text{ for all } \ell. \qquad (5.25)$$

By Jensen's inequality $(\mathbb{E}[f])^2 \leq \mathbb{E}[f^2]$, (5.25) follows if

$$\lim_{m\to\infty} \mathbb{E}_{\mathcal{T}_m,\boldsymbol{x}} \left[\left(\eta^\ell(\boldsymbol{x}) - \frac{1}{k}\sum_i^k \tilde{z}_i^\ell \right)^2 \right] = 0 \text{ for all } \ell. \qquad (5.26)$$

Let $\eta_{k,m}^\ell(\boldsymbol{x}) = \frac{1}{k}\sum_i^k \tilde{z}_i^\ell$. Then $\eta_{k,m}^\ell$ is actually an estimate of the class probability $\eta^\ell(\boldsymbol{x})$ by the k-Nearest Neighbor rule. Following the proof of Stone's Theorem [77, 23], if $k \xrightarrow{m\to\infty} \infty$ and $k/m \xrightarrow{m\to\infty} 0$, (5.26) holds for all distributions P. □

5.2 Riemannian Curvature Based Geometric Penalty

In defining the geometric penalty, we choose the volume of the functional graph as our complexity measure of the function, since it is a natural high-dimensional extension of the "surface energy" in a physical model related to the learning process. In the theory of Riemannian geometry, however, we could alternatively consider an L^2 measure of the Riemann curvature of $\mathrm{gr}(\boldsymbol{f})$, as the vanishing of this term gives optimal (i.e., locally distortion free) diffeomorphisms from $\mathrm{gr}(\boldsymbol{f})$ to $\mathbb{R}^N$. Given that it is complicated and inefficient to compute in practice, in this section, we discuss only the theory and formulations of the Riemannian curvature based geometric regularization, while leaving implementation concerns for future exploration.

We wish to penalize graphs for excessive curvature. Very sensitive measures are given by the L^2 norm of the length of the Riemann curvature tensor: $\int_{\mathcal{X}} |\mathcal{R}_{\boldsymbol{f}}|^2 \boldsymbol{f}^* \mathrm{dvol}$ or the L_∞ norm $\| \, |\mathcal{R}_{\boldsymbol{f}}| \, \|_\infty = \sup_{\boldsymbol{x} \in \mathcal{X}} |\mathcal{R}_{(\boldsymbol{x}, \boldsymbol{f}(\boldsymbol{x}))}|$. Minimization of Riemannian curvature corresponds to finding the graph with the least distortion of lengths and angles, i.e., minimizing $|d_{\boldsymbol{f}}((\boldsymbol{x}_1, \boldsymbol{f}(\boldsymbol{x}_1)), (\boldsymbol{x}_2, \boldsymbol{f}(\boldsymbol{x}_2))) - d_{\mathbb{R}^N}(\phi^{-1}(\boldsymbol{x}_1 \boldsymbol{f}(\boldsymbol{x}_1)), \phi^{-1}(\boldsymbol{x}_2, \boldsymbol{f}(\boldsymbol{x}_2))|$, where $d_{\boldsymbol{f}}$ is the geodesic distance on the graph of $\boldsymbol{f}$, $d_{\mathbb{R}^N}$ is the standard Euclidean distance, and ϕ ranges over all diffeomorphisms from $\mathbb{R}^N$ to $\mathrm{gr}(\boldsymbol{f})$. This is similar to the approach in [25].

We choose as curvature penalty function

$$\mathcal{P}_G(\boldsymbol{f}) = \int_{\mathcal{X}} |\mathcal{R}_{\boldsymbol{f}}|^2 \boldsymbol{f}^* \mathrm{dvol} = \int_{\mathrm{gr}(\boldsymbol{f})} |\mathcal{R}_{\boldsymbol{f}}|^2 \mathrm{dvol} = \int_{\mathrm{gr}(\boldsymbol{f})} \mathcal{R}_{ijkl} \mathcal{R}^{ijkl} \mathrm{dvol}, \tag{5.27}$$

where $\mathcal{R} = R_{ijkl} dx^i \otimes dx^j \otimes dx^k \otimes dx^l$ is the Riemann curvature tensor for the Riemannian metric on $\mathrm{gr}(\boldsymbol{f})$ induced from the Euclidean metric on $\mathbb{R}^{N+1}$, and dvol is the induced volume form. We always use summation convention on repeated indices. As with the original $\mathcal{P}_G$, this penalty term has the desirable property of being invariant under diffeomorphisms/reparametrizations of $\mathcal{X}$, while, *e.g.*, $\int_{\mathcal{X}} |\mathcal{R}_{\boldsymbol{f}}|^2 d\boldsymbol{x}$ is not.

We have the following theorem for the gradient vector field of $\mathcal{P}_G$. In this theorem, we work in a more general setup, with $\mathcal{X}$ replaced by a manifold M. (However, for simplicity we assume that M is compact without boundary; see [4] for the boundary terms in the case of a manifold with boundary and for the proof of the theorem below.)

Theorem 7. *Let $\phi = (\phi^1, \ldots, \phi^{N+L}) : M \to \mathbb{R}^{N+L}$ be an embedding. The gradient vector field for $\mathcal{P}_G$ at ϕ is the $\mathbb{R}^{N+L}$-valued vector field $Z = (Z^1, \ldots, Z^{N+L})$ defined on $\phi(M)$ with α component*

$$\begin{aligned} Z^\alpha = {} & 6\frac{1}{\sqrt{\det(g)}}\partial_r\left(\sqrt{\det(g)}R^r{}_{jkl}\phi^\alpha_i R^{ijkl}\right) \\ & +2\frac{1}{\sqrt{\det(g)}}\partial_z\Bigg(\Bigg[\frac{1}{\sqrt{\det(g)}}([\phi^\alpha_{lj}-\phi^\alpha_r\Gamma^r_{lj}]\partial_m(R^{mjlz}\sqrt{\det(g)})) \\ & +\left(\phi^\alpha_{ljm}-\phi^\alpha_{rm}\Gamma^r_{lj}-\phi^\alpha_r\partial_m(\Gamma^r_{lj})\right)R^{mjlz} \\ & +\Gamma^z_{nk}[\phi^\alpha_{lj}-\phi^\alpha_r\Gamma^r_{lj}]R^{njlk}\Bigg]\sqrt{\det(g)}\Bigg) \\ & -|\mathcal{R}|^2(\mathrm{Tr\ II})^\alpha - 2\langle\nabla\mathcal{R},\mathcal{R}\rangle^{\sharp,\alpha}. \end{aligned}$$

Here (i) $\det(g)$ is the determinant of the induced metric tensor in local coordinates on $\phi(M)$ coming from local coordinates on M composed with ϕ, (ii) $\phi^\alpha_i, \phi^\alpha_{rm}$, etc. denote partial derivatives of ϕ^α in these local coordinates and ∂_z is the z^{th} partial derivative, (iii) Γ^r_{lj} are the Christoffel symbols of the induced metric on $\phi(M)$, (iv) ∇R is the covariant derivative of R, (v) $\langle\nabla\mathcal{R},\mathcal{R}\rangle$ is the one-form on $\mathcal{M}$ given by contracting $\nabla\mathcal{R}$ with $\mathcal{R}$, (vi) $\langle\nabla\mathcal{R},\mathcal{R}\rangle^{\sharp,\alpha}$ is the α component of the dual vector field on M. In the graph case, the embedding ϕ associated to the function $\boldsymbol{f} : \mathcal{X} \to \mathbb{R}^L$

is the graph of $\boldsymbol{f}$: $\phi(\boldsymbol{x}) = (\boldsymbol{x}, \boldsymbol{f}(\boldsymbol{x}))$.

This formula is complicated, but the main point is that embeddings with minimal/vanishing geometric penalty are flat, i.e. the embedding admits distortion-free coordinate maps to $\mathbb{R}^N$. While this geometric penalty is more informative, the vector field Z involves fourth derivatives of the embedding ϕ, which are expensive to compute in practice.

5.3 Existence of Gradient Flow

In §5.1 where we discuss the Bayes consistency of our algorithm, we simply assume that gradient flow lines always exist. However, unlike the case of finite dimensions, the existence of flow lines on our functional space $\mathcal{M}' = \text{Maps}(\mathcal{X}, \mathbb{R}^L)$ is not automatic. We address this problem in this section. In particular, we discuss the existence of the solution of the following gradient flow equation, which is a ordinary differential equation (ODE) with initial condition on the functional space $\mathcal{M}'$,[2]

$$\begin{aligned} \frac{d\boldsymbol{f}_t}{dt} &= -\nabla \mathcal{P}_{\boldsymbol{f}_t}, \\ \boldsymbol{f}_0 &\equiv (\frac{1}{L}, \dots, \frac{1}{L}). \end{aligned} \tag{5.28}$$

In §5.3.1, we discuss the short time existence and uniqueness of solutions of ODE (5.28) under both Banach space norm and Fréchet space topology. We explain why the existence for general non-parametric form of $\boldsymbol{f}$ has no guarantee even for a short time. Then for parametric representation of $\boldsymbol{f}$, we show that the existence and uniqueness is guaranteed. In §5.3.2, we discuss the same ODE solution existence problem for the empirical term (standard classification losses), which again has no guarantee if $\boldsymbol{f}$ is

---

[2] Then we could use the simplex constraint of §4.2.3 to get the solution on $\mathcal{M} = \text{Maps}(\mathcal{X}, \Delta^{L-1})$.

expressed in the general non-parametric form, but has a straightforward guarantee if $\boldsymbol{f}$ is represented in a parametric form.

5.3.1 Existence and Uniqueness of The Geometric Flow

In ODE theory, the fundamental theorem on existence and uniqueness of solutions to first-order ODEs with initial condition is known as the Picard-Lindelöf Theorem [24, Chpater 10]. A proof for the situation of gradient flow equations can be found in [44, Chapter 17], where the arguments depend on two requirements, the *Lipschitz continuity* of the vector field and the Banach Fixed Point Theorem [24, Chapter 10.1]. As a result, to guarantee a short time[3] existence and uniqueness of the solution to our gradient flow equation (5.28), we have to ensure both requirements for our gradient flow setup.

To ensure that our functional space $\mathcal{M}'$ is a Banach space, we simply put the following Sobolev norm with sufficiently large s, and its induced topology on $\mathcal{M}'$,

$$\|\boldsymbol{f}\|_s = \left(\sum_{|\boldsymbol{\alpha}|<s} \int_{\mathcal{X}} |\partial^{\boldsymbol{\alpha}} \boldsymbol{f}|^2 d\boldsymbol{x} \right)^{\frac{1}{2}}. \tag{5.29}$$

Then $\mathcal{M}'$ becomes a s-Sobolev space, denoted as $\mathcal{H}^s$. Since each Sobolev space is a Banach space, the Banach Fixed Point Theorem applies to $\mathcal{M}'$. To determine a suitable choice of s, recall that the Sobolev embedding theorem [13] gives a continuous inclusion $i : \mathcal{H}^s \to C^{s-N/2-\delta}$ for any $\delta > 0$. Given that we only need up to second order derivatives to construct $\nabla \mathcal{P}_{\boldsymbol{f}_t}$ to solve the ODE (5.28), any integer $s > 2+N/2$ will work us.

On the other hand, the *Lipschitz continuity* of our geometric gradient vector

---

[3] i.e., for $\boldsymbol{f}_0$ given in (5.28), there exists some $\epsilon(\boldsymbol{f}_0)$, s.t. the solution to (5.28) exists for $|t| < \epsilon$.

field $\nabla\mathcal{P}_G$ is hard to show. By (4.6), $\nabla\mathcal{P}_G$ is the last L-components of Tr II.[4] Given $\boldsymbol{f},\boldsymbol{g}\in\mathcal{M}'$, such that $\|\boldsymbol{f}-\boldsymbol{g}\|_s<\epsilon$ for some $\epsilon>0$, then there must exist some $\epsilon'>0$, such that $\|\mathrm{Tr}\ \mathrm{II}_{\mathrm{gr}(\boldsymbol{f})}-\mathrm{Tr}\ \mathrm{II}_{\mathrm{gr}(\boldsymbol{g})}\|_{s-2}<\epsilon'$, with $\lim\limits_{\epsilon\to 0}\epsilon'=0$, since Tr II involves second partial derivatives. However, there is in general no guarantee that $\|\mathrm{Tr}\ \mathrm{II}_{\mathrm{gr}(\boldsymbol{f})}-\mathrm{Tr}\ \mathrm{II}_{\mathrm{gr}(\boldsymbol{g})}\|_s<\epsilon''$, for any $\epsilon''>0$. Therefore $\nabla\mathcal{P}_G$ is not continuous in the s-norm, so it is not *Lipschitz continuous*. As a result, the existence and uniqueness theorem of ODE does not apply directly to our regularization term $\mathcal{P}_G$, i.e., the short time existence of gradient flow lines is not automatic when we put Banach space norm on $\mathcal{M}'$.

Also note that if we just put the Fréchet topology on $\mathcal{M}'$, as we have done in §3.3.1, since we do not have a Fixed Point Theorem in Fréchet spaces, it is still unclear if the short time existence of gradient flow lines still holds there.

Parametric representation of $\boldsymbol{f}$

We now show that the *Lipschitz continuity* is not a concern for applying our algorithm in practice, where function $\boldsymbol{f}$ is always represented as some parametric form. To see this, consider some parametric representation $\boldsymbol{f}(\boldsymbol{x})=\boldsymbol{f}(\boldsymbol{w};\boldsymbol{x})$, where $\boldsymbol{w}\in W\subset\mathbb{R}^p$ is the vector of parameters in representing $\boldsymbol{f}$. Let $F:W\to\mathcal{M}'$ be the mapping from the parameter space W to the functional space $\mathcal{M}'$, i.e., $\boldsymbol{f}_{\boldsymbol{w}}=F(\boldsymbol{w})$, then we have the following commutative diagram,

$$\begin{tikzcd}
W \arrow[r, "F"] \arrow[dr, "\mathcal{P}'_G=\mathcal{P}_G\circ F"'] & \mathcal{M}' \arrow[d, "\mathcal{P}_G"] \\
 & \mathbb{R}
\end{tikzcd}$$

Let $\{\boldsymbol{e}^i\}=\{\frac{\partial}{\partial x^i}\}_{i=1}^p$ denotes the standard basis of $\mathbb{R}^p$, then the ODE (5.28) for

---

[4] Tr II is the trace of second fundamental form of $\mathrm{gr}(\boldsymbol{f})$.

parameter space W becomes

$$\begin{aligned} \frac{d\boldsymbol{w}_t}{dt} &= -\nabla \mathcal{P}'_{G,\boldsymbol{w}_t}, \\ \boldsymbol{f}_0 &= F(\boldsymbol{w}_0) \equiv (\frac{1}{L}, \ldots, \frac{1}{L}). \end{aligned} \tag{5.30}$$

From the above commutative diagram,

$$\nabla \mathcal{P}'_{G,\boldsymbol{w}} = \frac{\partial \mathcal{P}'_G}{\partial w^i} \boldsymbol{e}^i, \tag{5.31}$$

where $\frac{\partial \mathcal{P}'_G}{\partial w^i}$ can be computed by

$$\begin{aligned} \left.\frac{\partial \mathcal{P}'_G}{\partial w^i}\right|_{\boldsymbol{w}} &= \nabla \mathcal{P}'_{G,\boldsymbol{w}} \overset{\mathbb{R}^p}{\cdot} \boldsymbol{e}^i \\ &= d\mathcal{P}'_{G,\boldsymbol{w}}(\boldsymbol{e}^i) \\ &= d(\mathcal{P}_G \circ F)_{\boldsymbol{w}}(\boldsymbol{e}^i) \\ &= d\mathcal{P}_G\big(dF_{\boldsymbol{w}}(\boldsymbol{e}^i)\big) \\ &= \nabla \mathcal{P}_{G,\boldsymbol{f}_{\boldsymbol{w}}} \overset{\mathcal{M}'}{\cdot} F_* \boldsymbol{e}^i \\ &= \text{Tr II}^L_{\text{gr}(\boldsymbol{f}_{\boldsymbol{w}})} \overset{\mathcal{M}'}{\cdot} \left.\frac{d}{dt}\right|_{\boldsymbol{w}} \boldsymbol{f}_{\boldsymbol{w}+t\boldsymbol{e}^i}, \end{aligned} \tag{5.32}$$

where F_* denotes the pushforward associated with F, $\overset{\mathbb{R}^p}{\cdot}$ denotes the inner product on $\mathbb{R}^L$, and $\overset{\mathcal{M}'}{\cdot}$ denotes the L^2 metric on $\mathcal{M}'$. For compact W and smooth F, the component functions of $\nabla \mathcal{P}'_{G,\boldsymbol{w}}$ given by Eqn. (5.32) are *Lipschitz continuity.* As a result, by applying the Picard-Lindelöf Theorem, the existence and uniqueness of geometric gradient flow lines is guaranteed for parametric representation of $\boldsymbol{f}$.

5.3.2 Existence for the Empirical Term

The empirical term is in general expressed as the following form,

$$\mathcal{P}_{\mathcal{T}_m}(\boldsymbol{f}) = \sum_{i=1}^{m} \ell(\boldsymbol{f}(\boldsymbol{x}_i), y_i), \tag{5.33}$$

where ℓ is some classification loss function. If $\boldsymbol{f}$ is in this general non-parametric form, the gradient flow equation (5.28) applies on the infinite dimensional function space $\mathcal{M}'$, then the *Lipschitz continuity* of the gradient vector field $\nabla\mathcal{P}_{\mathcal{T}_m}$ does not hold. To give an example, consider the following Quadratic loss defined in §4.2.1,

$$\mathcal{P}_{\mathcal{T}_m}(\boldsymbol{f}) = \sum_{i=1}^{m} \|\boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{z}_i\|^2, \tag{5.34}$$

where $\boldsymbol{z}_i$ is the one-hot vector corresponding to the ground truth label y_i. Then for any $\boldsymbol{v} \in T_{\boldsymbol{f}}\mathcal{M}'$, the differential $d\mathcal{P}_{\mathcal{T}_m,\boldsymbol{f}} : T_{\boldsymbol{f}}\mathcal{M}' \to \mathbb{R}$ is

$$\begin{aligned}
d\mathcal{P}_{\mathcal{T}_m,\boldsymbol{f}}(\boldsymbol{v}) &= \frac{d}{dt}\bigg|_{t=0} \mathcal{P}_{\mathcal{T}_m}(\boldsymbol{f} + t\boldsymbol{v}) \\
&= \frac{d}{dt}\bigg|_{t=0} \sum_{i=1}^{m}\sum_{j=1}^{L}\big((\boldsymbol{f} + t\boldsymbol{v})^j(\boldsymbol{x}_i) - z_i^j\big)^2 \\
&= 2\sum_{i,j}\big(f^j(\boldsymbol{x}_i) - z_i^j\big)\frac{d}{dt}\bigg|_{t=0} (\boldsymbol{f} + t\boldsymbol{v})^j(\boldsymbol{x}_i) \\
&= 2\sum_{i,j}\big(f^j(\boldsymbol{x}_i) - z_i^j\big)\boldsymbol{v}^j(\boldsymbol{x}_i) \\
&= 2\sum_{i=1}^{m}\big(\boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{z}_i\big) \cdot \boldsymbol{v}(\boldsymbol{x}_i) \\
&= 2\int_{\mathcal{X}}\sum_{i=1}^{m}\big(\boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{z}_i\big)\delta_{\boldsymbol{x}_i} \cdot \boldsymbol{v}(\boldsymbol{x})\mathrm{dvol}_{\boldsymbol{x}} \\
&= \Big\langle 2\sum_{i=1}^{m}\big(\boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{z}_i\big)\delta_{\boldsymbol{x}_i}, \boldsymbol{v}\Big\rangle,
\end{aligned} \tag{5.35}$$

where the last equality follows our choice of the Riemannian metric (3.8), and $\delta_{\boldsymbol{x}_i}$ is the delta function at $\boldsymbol{x}_i$. This gives

$$\nabla \mathcal{P}_{\mathcal{T}_m}(\boldsymbol{f}) = 2\sum_{i=1}^{m} \big(\boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{z}_i\big)\delta_{\boldsymbol{x}_i}. \tag{5.36}$$

It is obvious that $\nabla \mathcal{P}_{\mathcal{T}_m}$ is a discontinuous vector field. As a result, even short time existence of gradient flow lines for the empirical term $\mathcal{P}_{\mathcal{T}_m}$ is not guaranteed. This is in fact not surprising, given that the empirical loss term (5.33) is in general not designed for the infinite dimensional space of non-parametric functions, but for the finite dimensional vector space of parameters, by which the functions are represented. This is also one of the main reasons that we can only show consistency for a specific empirical loss in §5.1, where the existence of gradient flow lines is guaranteed.

However, this is not a concern for applying our algorithm in practice, where function $\boldsymbol{f}$ is always represented as some parametric form. To see this, consider some parametric representation $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{w};\boldsymbol{x})$, where $\boldsymbol{w} \in \mathbb{R}^p$ is the vector of parameters in representing $\boldsymbol{f}$. Then a direct computation gives the gradient of the empirical term (5.34) with respect to $\boldsymbol{w}$,

$$\nabla \mathcal{P}_{\mathcal{T}_m}(w_j) = \frac{\partial \mathcal{P}_{\mathcal{T}_m}}{\partial w_j} = 2\sum_{i=1}^{m} \big(\boldsymbol{f}(\boldsymbol{w};\boldsymbol{x}) - \boldsymbol{z}_i\big)\frac{\partial \boldsymbol{f}(\boldsymbol{w};\boldsymbol{x})}{\partial w_j}. \tag{5.37}$$

Then for $\mathcal{P}_{\mathcal{T}_m}$, the gradient flow equation (5.28) becomes a standard ODE on vector space $\mathbb{R}^p$, the gradient function $\nabla \mathcal{P}_{\mathcal{T}_m}$ defined above is *Lipschitz continuous*, so the existence and uniqueness of gradient flow lines is guaranteed.

Combining the analysis from §5.3.1, in our practical algorithms, where $\boldsymbol{f}$ is always represented as some parametric form, the short time existence and uniqueness of solutions of the gradient flow equation (5.28), for $\mathcal{P} = \mathcal{P}_{\mathcal{T}_m} + \lambda \mathcal{P}_G$, is always guaranteed. This lays the foundation for applying our regularization approach in practice, since algorithms applied to applications in the next chapter are therefore guaranteed to reflect our geometric intuition to fit the training data in a volume decreasing manner.

5.4 Extension to Nonlinear Input Spaces

To extend the input space from $\mathcal{X} \subset \mathbb{R}^N$ to a finite dimensional compact manifold $\mathcal{N}$, we can use coordinate charts and a partition of unity [44] to put a Sobolev norm on the space $\mathrm{Maps}(\mathcal{N}, \mathbb{R}^L)$. We still get a Banach space, and all the results and analysis of previous sections of this chapter remain. In fact, the key is that there exists a finite number of coordinate charts on $\mathcal{N}$. If this is true, even for non-compact manifolds, all previous results still hold. For instance, even non-compact Lie groups have a finite number of charts.

5.5 Summary

In this chapter, we discuss some theoretical aspects and extensions of our geometric approach. Firstly, with a particular empirical loss function, we establish Bayes consistency for our geometric regularization scheme, under some mild initialization assumptions, namely that the initial estimator lies in the stable manifold of a global minimum point of the vector field. Secondly, we discuss the theory and formulation of an alternative geometric regularizer based on Riemannian curvature. This has implications for a more sophisticated regularization method for future study. Thirdly, we discuss the short time existence and uniqueness of gradient flow lines for both the empirical term and the geometric regularization term. We point out that existence of the gradient flow, in a strict mathematical sense, is not automatic, and give a particular topology such that the short time existence and uniqueness of gradient flow lines for the regularization term has a theoretical guarantee. For the empirical term, we have shown that even short time existence may not hold in general for a nonparametric representation of $\boldsymbol{f}$. However, existence holds in general for parametric representations of $\boldsymbol{f}$. Combining the results on both terms, the short time existence and uniqueness of gradient flow lines of our regularized loss minimization formula is guaranteed in practice, where $\boldsymbol{f}$ is always represented as some parametric form. This provides a theoretical guarantee that our algorithm applied in applications, such as those reported in the next chapter, is applicable and matches the geometric intuition and theory we have proposed. Lastly, we discuss the extension of our geometric perspective and regularization approach to situations where the input space is a submanifold with local charts, rather than a Euclidean space. In particular, we point out that the key point for realizing this extension is the existence of a finite number of coordinate charts on the input submanifold.

Chapter 6

Applications

In previous chapters, we focus on introducing the intuition, methodology, and mathematical foundations of our geometric regularization scheme for supervised learning of classifiers. As a result, we have been treating the class probability estimator $\boldsymbol{f}$ as a general function from $\mathcal{X}$ to Δ^{L-1}, without specifying any particular representation of $\boldsymbol{f}$. In applications, however, we always need some particular representation model of $\boldsymbol{f}$ in order to design applicable algorithms. In this chapter, we discuss the application of the proposed geometric regularization approach in two representative classification models, i.e., the linear combination of radial basis functions (RBFs) and feedforward (deep) neural networks, such as convolutional neural networks. For each model, we introduce specific formulations for applying our regularization to that model, and design specific algorithms incorporating our regularization into the training process of that model. Implementation details and practical concerns are also discussed for both models. We test both models first on simple simulated examples, in order to validate our intuition and get visible insight about our method, and then on real-world benchmarks, in order to study the effectiveness of our method quantitatively.

Roadmap for this chapter

We present the application of our regularization approach to an RBF-based model in

§6.1 and to feedforward (deep) neural networks in §6.2. For §6.1, specific formulations and algorithms for the RBF-based model are introduced in §6.1.1, qualitative and quantitative experiments are then reported and discussed in detail in §6.1.2 and §6.1.3 respectively. For §6.2, specific formulations and algorithms for general feedforward (deep) neural networks are introduced in §6.2.1, followed by technical details for an efficient implementation in §6.2.2, and the qualitative and quantitative experiments are then reported and discussed in detail in §6.2.3 and §6.2.4 respectively.

6.1 RBF Representation

In this section, we illustrate our approach under an RBF representation of $\boldsymbol{f}$. Note that RBF's are also used by previous geometric classification methods [83, 46, 45].

6.1.1 Formulations and Algorithm

Given values of $\boldsymbol{f}$ are probabilistic vectors, it is common to represent $\boldsymbol{f}$ as a "softmax" output of RBFs, i.e.

$$f^j = \frac{e^{h^j}}{\sum_{l=1}^{L} e^{h^l}}, \text{ where } h^j = \sum_{i=1}^{m} a_i^j \varphi_i(\boldsymbol{x}), \text{ for } j = 1, \ldots, L, \tag{6.1}$$

where $\varphi_i(\boldsymbol{x}) = e^{-\frac{1}{c}\|\boldsymbol{x}-\boldsymbol{x}_i\|^2}$ is the RBF function centered at the training sample $\boldsymbol{x}_i$, with kernel width parameter c.

Estimating $\boldsymbol{f}$ becomes an optimization problem for the $m \times L$ coefficient matrix $A = (a_i^\ell)$. The following equation determines A:

$$[\boldsymbol{h}(\boldsymbol{x}_1), \ldots, \boldsymbol{h}(\boldsymbol{x}_m)]^T = GA, \text{ where } G_{ij} = \varphi_j(\boldsymbol{x}_i). \tag{6.2}$$

To plug this RBF representation into our gradient flow scheme, the gradient vector field $\nabla \mathcal{P}_{\boldsymbol{f}}$ is evaluated at each sample point $\boldsymbol{x}_i$, and A is updated by

$$A \leftarrow A - \tau G^{-1} \left[\nabla \mathcal{P}_{\boldsymbol{h}}(\boldsymbol{x}_1), \ldots, \nabla \mathcal{P}_{\boldsymbol{h}}(\boldsymbol{x}_m) \right]^T, \tag{6.3}$$

where τ is the step-size parameter, and

$$\nabla \mathcal{P}_{\boldsymbol{h}}(\boldsymbol{x}_i) = \left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{h}} \right]_{\boldsymbol{x}_i}^T \nabla \mathcal{P}_{\boldsymbol{f}}(\boldsymbol{x}_i). \tag{6.4}$$

Here $\nabla \mathcal{P}_{\boldsymbol{h}}(\boldsymbol{x}_i)$ denotes the gradient vector field with respect to $\boldsymbol{h}$ evaluated at $\boldsymbol{x}_i$, and the $L \times L$ Jacobian matrix $\left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{h}} \right]_{\boldsymbol{x}_i}$ can be obtained in closed form from (6.1). In the following subsections, we give exact forms of the empirical penalty $\mathcal{P}_{\mathcal{T}_m}$ and the geometric penalty $\mathcal{P}_G$, and discuss the computation of $\nabla \mathcal{P}_{\boldsymbol{h}}$ for both penalty terms.

6.1.1.1 The empirical penalty $\mathcal{P}_{\mathcal{T}_m}$

We give detailed formulas for gradient vectors of the two losses introduced in §4.2.1, under the RBF representation of $\boldsymbol{f}$.

Quadratic loss. As defined in Eqn. (4.2), the quadratic loss is given by

$$\mathcal{P}_{\mathcal{T}_m}(\boldsymbol{f}) = \sum_{i=1}^{m} \|\boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{z}_i\|^2, \tag{6.5}$$

where $\boldsymbol{z}_i$ is the one-hot vector corresponding to the ground truth label of $\boldsymbol{x}_i$. The gradient vector with respect to $\boldsymbol{f}$ evaluated at $\boldsymbol{x}_i$ is

$$\nabla \mathcal{P}_{\mathcal{T}_m, \boldsymbol{f}}(\boldsymbol{x}_i) = 2(\boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{z}_i). \tag{6.6}$$

Under the RBF representation (6.1), the gradient vector w.r.t. $\boldsymbol{h}$ evaluated at $\boldsymbol{x}_i$ is

$$\nabla \mathcal{P}_{\mathcal{T}_m,\boldsymbol{h}}(\boldsymbol{x}_i) = 2\left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{h}}\right]^T_{\boldsymbol{x}_i} (\boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{z}_i), \tag{6.7}$$

where $\left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{h}}\right]^T_{\boldsymbol{x}_i}$ is evaluated as in (6.4).

Cross-entropy loss. As defined in Eqn. (4.4), the cross-entropy loss is given by

$$\mathcal{P}_{\mathcal{T}_m}(\boldsymbol{f}) = -\sum_{i=1}^{m}\sum_{\ell=1}^{L} z_i^\ell \log f^\ell(\boldsymbol{x}_i). \tag{6.8}$$

Under the RBF representation (6.1), the gradient vector field with respect to $\boldsymbol{h}$ evaluated at $\boldsymbol{x}_i$ is

$$\nabla \mathcal{P}_{\mathcal{T}_m,\boldsymbol{h}}(\boldsymbol{x}_i) = \boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{z}_i. \tag{6.9}$$

6.1.1.2 The geometric penalty $\mathcal{P}_G$

As discussed in §4.2.2, the geometric penalty is

$$\mathcal{P}_G(\boldsymbol{f}) = \int_{\mathrm{gr}(\boldsymbol{f})} \mathrm{dvol} = \int_{\mathrm{gr}(\boldsymbol{f})} \sqrt{\det(g)}dx^1 \dots dx^N, \tag{6.10}$$

and the gradient is

$$\nabla \mathcal{P}_{G,\boldsymbol{f}} = V_{G,\boldsymbol{f}} = -\mathrm{Tr}\ \mathrm{II}^L. \tag{6.11}$$

Under the RBF representation (6.1), the geometric gradient with respect to $\boldsymbol{h}$ is

$$\nabla \mathcal{P}_{G,\boldsymbol{h}} = V_{G,\boldsymbol{h}} = -\left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{h}}\right]^T \mathrm{Tr}\ \mathrm{II}^L. \tag{6.12}$$

Evaluation of $\left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{h}}\right]$ and Tr II^L at $\boldsymbol{x}_i$ leads to $\nabla \mathcal{P}_{G,\boldsymbol{h}}(\boldsymbol{x}_i)$.

6.1.1.3 Algorithm summary

Algorithm 2 gives a summary of the classifier learning procedure. The input to the algorithm is the training set $\mathcal{T}_m$, the RBF kernel width c, the trade-off parameter λ, and the step-size parameter τ. For initialization, our algorithm first initializes the function values of $\boldsymbol{h}$ and $\boldsymbol{f}$ for every training point, and then constructs the matrix G and solves for A by (6.2). In the subsequent steps, at each iteration our algorithm first evaluates the gradient vector field $\nabla\mathcal{P}_{\boldsymbol{h}}$ at every training point, and then updates the coefficient matrix A by (6.3). For the overall penalty function $\mathcal{P} = \mathcal{P}_{\mathcal{T}_m} + \lambda\mathcal{P}_G$, we compute the total gradient vector field $\nabla\mathcal{P}_{\boldsymbol{h}}$ evaluated at $\boldsymbol{x}_i$ as follows.
For quadratic loss, it is:

$$\nabla\mathcal{P}_{\boldsymbol{h}}(\boldsymbol{x}_i) = \left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{h}}\right]^T_{\boldsymbol{x}_i} \Big(2(\boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{z}_i) - \lambda \mathrm{Tr}\ \mathrm{II}^L_{\boldsymbol{x}_i}\Big). \tag{6.13}$$

For cross-entropy loss, it is:

$$\nabla\mathcal{P}_{\boldsymbol{h}}(\boldsymbol{x}_i) = \boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{z}_i - \lambda \left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{h}}\right]^T_{\boldsymbol{x}_i} \mathrm{Tr}\ \mathrm{II}^L_{\boldsymbol{x}_i}. \tag{6.14}$$

Our algorithm iterates until it converges within a threshold or reaches the maximum iteration number.

The same algorithm applies to both the quadratic loss and the cross-entropy loss. To evaluate the total gradient vectors $\nabla\mathcal{P}_{\boldsymbol{h}}(\boldsymbol{x}_i)$ in each iteration, for the quadratic loss, we use (6.7) and (6.12) to compute the total gradient vector (6.13); for the cross-entropy loss, we use (6.9) and (6.12) instead to compute (6.14). The remaining steps of the procedure are exactly the same for both loss functions.

The final predictor learned by our algorithm is given by

$$F(\boldsymbol{x}) = \mathrm{argmax}\{f^{\ell}(\boldsymbol{x}), \ell \in \{1, 2, \cdots, L\}\}. \tag{6.15}$$

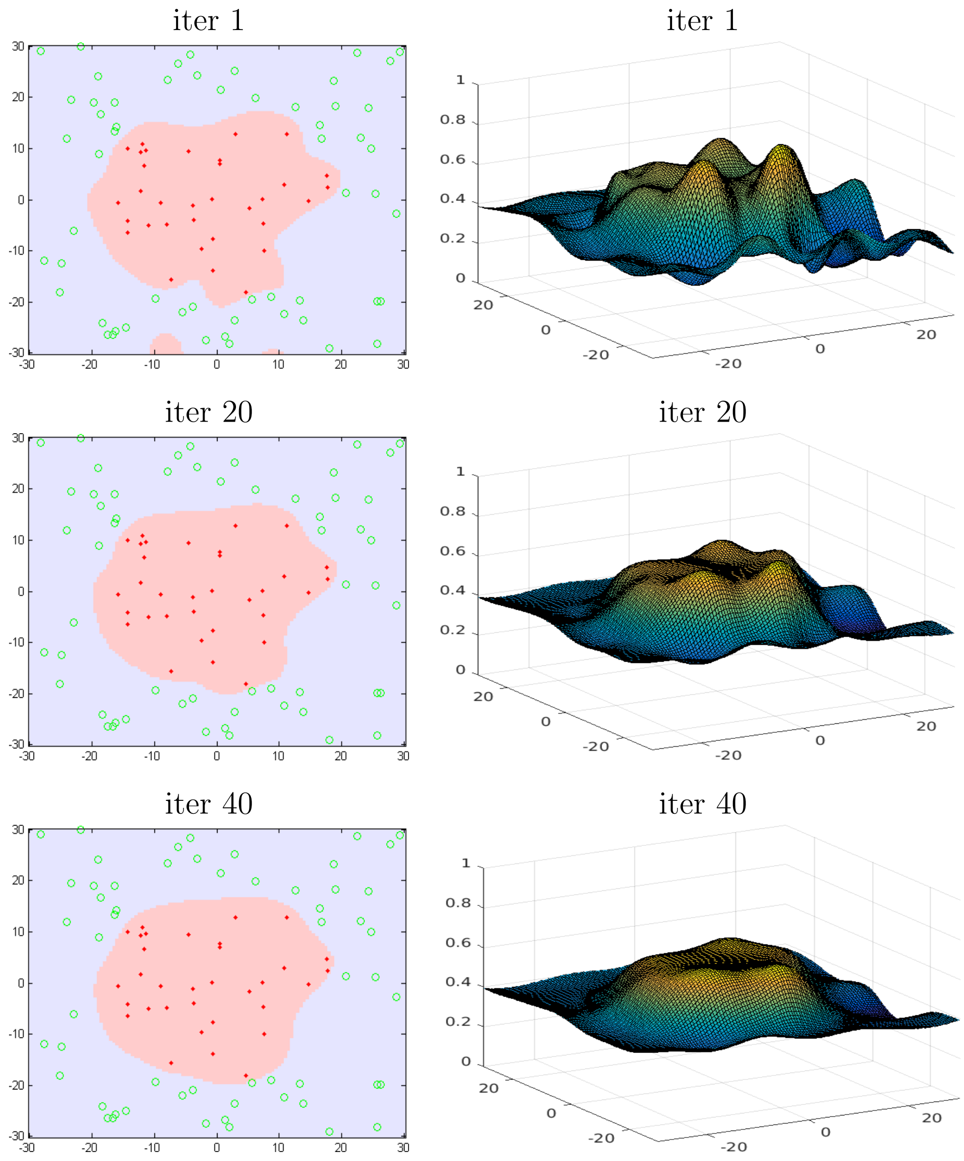
Algorithm 2 Geometric regularized classification for RBF representation

```
**Input:** training data $\mathcal{T}_m = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$, RBF kernel width $c$, trade-off parameter $\lambda$, step-size $\tau$
**Initialize:** $\boldsymbol{h}(\boldsymbol{x}_i) = (1, \dots, 1), \boldsymbol{f}(\boldsymbol{x}_i) = (\frac{1}{L}, \dots, \frac{1}{L})$, $\forall i \in \{1, \cdots, m\}$, construct matrix $G$ and solve $A$ by (6.2)
**for** $t = 1$ **to** $T$ **do**
    – Evaluate the total gradient vector $\nabla \mathcal{P}_{\boldsymbol{h}}(\boldsymbol{x}_i)$ at every training point according to (6.13) or (6.14).
    – Update the $A$ by (6.3).
**end for**
**Output:** class probability estimator $\boldsymbol{f}$ given by (6.1).
```

6.1.2 Qualitative Experiments on Synthetic Data

We first test our RBF-based implementation on the toy example of 2D classification that illustrates our physical model in §3.2, where training points are sampled uniformly within the region $[-30, 30] \times [-30, 30]$, and labeled by the function $y = \mathrm{sign}(20 - \|\boldsymbol{x}\|_2)$. As shown in Figure 6.1, under the RBF representation (6.1), our geometric regularization technique is effective in reducing the "local oscillations" of the surface (submanifold) corresponding to the class probability estimator, and as a byproduct, the decision boundary, which is a level set of the surface, is also getting smoother.

We also run an experiment under the same setup, but reverse the geometric gradient vector at every iteration. (Note that the same simplex constraint introduced in §4.2.3 is also enforced to make sure that the reversed gradient vector also lies in the simplex.) As shown in Figure 6.2, the reversed gradient vector does increase the


iter 1

iter 1

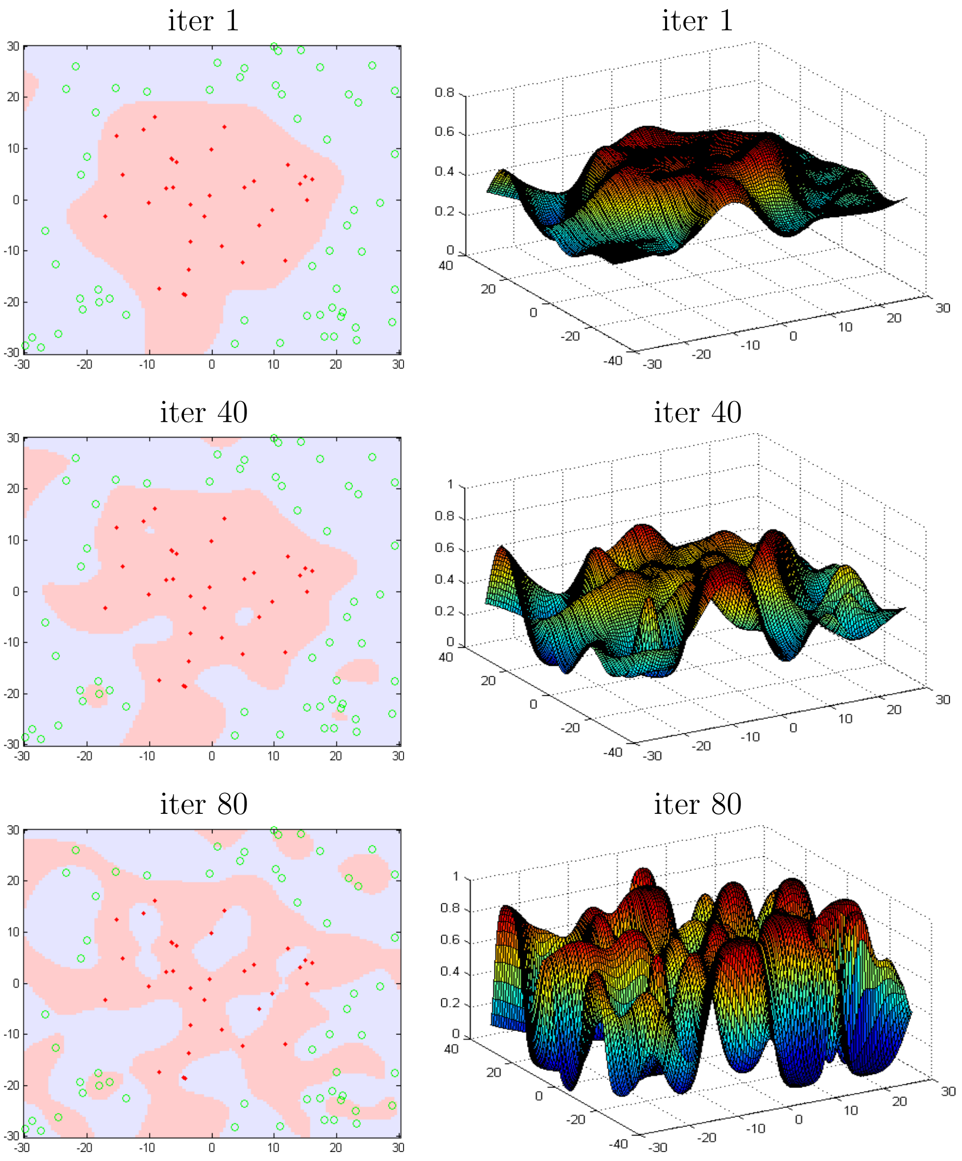
iter 20

iter 20

iter 40

iter 40

Figure 6.1: Example of binary learning with RBF-based implementation, where input space $\mathcal{X}$ is $2d$. Training points are sampled uniformly within the region $[-30, 30] \times [-30, 30]$, and labeled by the function $y = \mathrm{sign}(20 - \|\boldsymbol{x}\|_2)$. We plot the surface obtained by our method in the right column and the corresponding decision boundary in the left column. The vertical axis of the right image is the 1-simplex $\Delta^1 \subset \mathbb{R}^2$.



iter 1

iter 1

iter 40

iter 40

iter 80

iter 80

Figure 6.2: Example of binary learning with reversed geometric gradient under RBF representation, where input space $\mathcal{X}$ is $2d$. Training points are sampled uniformly within the region $[-30, 30] \times [-30, 30]$, and labeled by the function $y = \text{sign}(20 - \|\boldsymbol{x}\|_2)$. We plot the surface obtained by our method in the right column and the corresponding decision boundary in the left column. The vertical axis of the right image is the 1-simplex $\Delta^1 \subset \mathbb{R}^2$.

amount of local oscillations of the surface, and gradually tears apart the positive region of the input space, which is consistent with the cartoon motivation example of Figure 1.2 introduced in §1.2.

6.1.3 Quantitative Experiments on Benchmarks

To evaluate the effectiveness of the proposed regularization approach, we compare our RBF-based implementation with two groups of related classification methods. The first group of methods involve standard RBF-based methods that use regularizers different from ours. The second group of methods involve other geometric regularization methods.

In particular, the first group includes the Radial Basis Function Network (RBN), SVM with RBF kernel (SVM) and the Import Vector Machine (IVM) [91] (a greedy search variant of the standard RBF kernel logistic regression classifier). Note that both SVM and IVM use RKHS regularizers and the IVM also uses the similar cross-entropy loss as Ours-CE.

The second group includes the Level Learning Set classifier [15] (LLS), the Geometric Level Set classifier [83] (GLS) and the Euler's Elastica classifier [46, 45] (EE). Note that both GLS and EE use RBF representations and EE also uses the same quadratic distance loss as Ours-Q.

We test both the quadratic loss version (Ours-Q) and the cross-entropy loss version (Ours-CE) of our implementation.

6.1.3.1 UCI datasets

We tested our classification method on four binary classification datasets and four multiclass classification datasets. Given that [83] has covered several methods on our comparison list and their implementation is publicly available, we chose to use the same datasets as [83] and carefully followed their experimental setup. The tenfold cross-validation error is reported. For each of the ten trials, the kernel-width constant c and tradeoff parameter λ are found using fivefold cross-validation on the training folds. All dimensions of input sample points are normalized to a fixed range $[0, 1]$ throughout the experiments. We select c from the set of values $\{1/2^5, 1/2^4, 1/2^3, 1/2^2, 1/2, 1, 2, 4, 8\}$ and λ from the set of values $\{1/1.5^4, 1/1.5^3, 1/1.5^2, 1/1.5, 1, 1.5\}$ that minimize the fivefold cross-validation error. The step-size $\tau = 0.1$ and iteration number $T = 5$ are fixed over all datasets. We used the same settings for both loss functions.

Table 6.1: Tenfold cross-validation error rate (percent) on four binary and four multiclass classification datasets from the UCI machine learning repository. (L, N) denote the number of classes and input feature dimensions respectively. We compare both the quadratic loss version (Ours-Q) and the cross-entropy loss version (Ours-CE) of our method with 6 RBF-based classification methods and (or) geometric regularization methods: SVM with RBF kernel (SVM), Radial basis function network (RBN), Level learning set classifier [15] (LLS), Geometric level set classifier [83] (GLS), Import Vector Machine [91] (IVM), Euler's Elastica classifier [46, 45] (EE). The mean error rate averaged over all eight datasets is shown in the bottom row. Top performance for each dataset is shown in bold.

DATASET(L, N)	RBN	SVM	IVM	LLS	GLS	EE	OURS-Q	OURS-CE
PIMA(2,8)	24.60	24.12	24.11	29.94	25.94	**23.33**	23.98	24.51
WDBC(2,30)	5.79	2.81	3.16	6.50	4.40	**2.63**	**2.63**	**2.63**
LIVER(2,6)	35.65	28.66	29.25	37.39	37.61	26.33	**25.74**	26.31
IONOS.(2,34)	7.38	**3.99**	21.73	13.11	13.67	6.55	6.83	6.26
WINE(3,13)	1.70	1.11	1.67	5.03	3.92	0.56	**0.00**	**0.00**
IRIS(3,4)	4.67	**2.67**	4.00	3.33	6.00	4.00	3.33	3.33
GLASS(6,9)	34.50	31.77	**29.44**	38.77	36.95	32.28	29.87	**29.44**
SEGM.(7,19)	13.07	3.81	3.64	14.40	4.03	8.80	**2.47**	2.73
ALL-AVG	15.92	12.37	14.63	18.56	16.57	13.06	**11.86**	11.90

Table 6.1 reports the results of this experiment. The top performer for each

dataset is marked in bold, and the averaged performance of each method over all testing datasets is summarized in the bottom row. The numbers for RBN, LLS and GLS are copied from Table 1 of [83]. Results for SVM and IVM are obtained by running publicly available implementations for SVM [16] and IVM [64]. Results for EE are obtained by running an implementation provided by the authors of [46]. When running these implementations, we followed the same experimental setup as described above and exhaustively searched for the optimal range for the kernel bandwidth and the trade-off parameter via cross-validation.

As shown in the last row of Table 6.1, two versions of our approach are overall the top two performers among all reported methods. In particular, Ours-Q attains top performance on four out of the eight benchmarks, Ours-CE attains top performance on three out of the eight benchmarks. The performance of the two versions of our method are very close, which shows the robustness of our geometric regularization approach cross different loss functions for classification. Note that three pairs of comparisons, IVM vs Ours-CE, GLS vs Ours-Q/Ours-CE, and EE vs Ours-Q are of particular interest. We will discuss them in detail below.

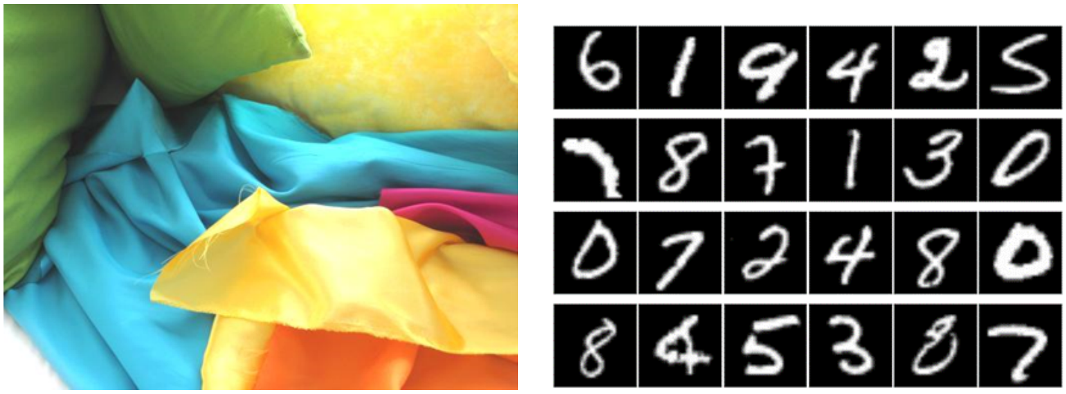
The IVM method of kernel logistic regression uses the same RBF-based implementation and very similar cross-entropy loss as our cross-entropy version Ours-CE, and both methods handle the multiclass case inherently. The main difference lies in regularization, i.e., the standard RKHS norm regularizer vs. our geometric regularizer. Ours-CE outperforms IVM on six of the eight benchmarks in Table 6.1, and achieves equal performance on one of the remaining two, and is only slightly behind on “PIMA”. The overall superior performance of Ours-CE demonstrates the advantage of the proposed geometric regularization over the standard RKHS norm regularization.

The GLS method uses the same RBF-based implementation as ours and also exploits volume geometry for regularization. However, there are key differences between the two regularization techniques. GLS measures the volume of the decision boundary supported in $\mathcal{X}$, while our approach measures the volume of a submanifold supported in $\mathcal{X} \times \Delta^{L-1}$ that corresponds to the class probability estimator. Our regularization technique handles the binary and multiclass cases in a unified framework, while the decision boundary based techniques, such as GLS (and EE), were inherently designed for the binary case and rely on a binary coding strategy to train $\log_2 L$ decision boundaries to generalize to the multiclass case. In our experiments, both Ours-Q and Ours-CE outperform GLS on all the benchmarks we have tested. This demonstrates the effectiveness of exploiting the geometry of the class probability in addressing the "small local oscillation" for classification.

The EE method of Euler's Elastica model uses the same RBF-based implementation and the same quadratic loss as our quadratic loss version Ours-Q. The main difference, again, lies in regularization, i.e., a combination of 1-Sobolev norm and curvature penalty on the decision boundary vs. our volume penalty on the submanifold corresponding to the class probability estimator. Since EE adopts a combination of sophisticated geometric measures on the decision boundary and level sets of the classification function, which specifically fits the binary case, it achieves top performance on binary datasets. However, the geometry of the class probability for general classification, which is captured by our approach, cannot be captured by decision boundary based techniques. That is the reason why Ours-Q, a general scheme for both the binary and multiclass case, outperforms EE on all four multiclass datasets, while it still achieves top performance on binary datasets. This again demonstrates our geometric perspective and regularization approach that exploits the geometry of the class probability.

6.1.3.2 Real-world datasets

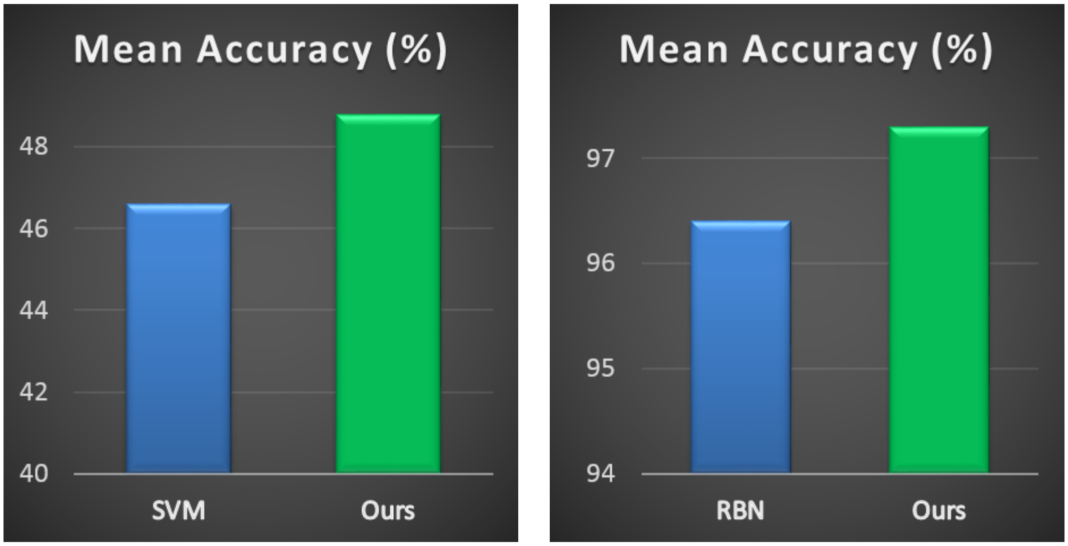
To test the scalability of our method to high-dimensional and large-scale problems, we also conducted experiments on two real-world datasets, i.e., the Flickr Material Database (FMD) [72] for image classification and the MNIST [43] Database of handwritten digits. Exemplar input images of both datasets are shown in Figure 6.3, and the results are shown in Figure 6.4.



(*a*) Flickr Material Database (*b*) MNIST handwritten digits

Figure 6.3: Exemplar input images from the Flickr Material Database and MNIST handwritten digits.

FMD (4096 dimensional). The FMD dataset contains 10 categories of images with 100 images per category. We extract image features using the SIFT descriptor augmented by its feature coordinates, implemented by the VLFeat library [84]. With this descriptor, Bag-of-visual-words uses 4096 vector-quantized visual words, histogram square rooting, followed by L^2 normalization. We compare our method with an SVM classifier with RBF kernels, using exactly the same 4096 dimensional feature. Our method achieves a correct classification rate of 48.8% while the SVM baseline achieves 46.4%. Note that while recent work (Qi et al., 2015; Cimpoi et



(a) Flickr Material Database (b) MNIST handwritten digits

Figure 6.4: Testing accuracy results on the Flickr Material Database and MNIST handwritten digits.

al., 2015) reports better performance on this dataset, their effort focuses on better feature design, not on the classifier itself. The features used in those works, such as local texture descriptors and CNN features, are more sophisticated than those used in our experiments.

MNIST (60,000 samples). The MNIST dataset contains 10 classes ($0 \sim 9$) of handwritten digits with $60,000$ samples for training and $10,000$ samples for testing. Each sample is a 28×28 grey scale image. We use 1000 RBFs to represent our function $\boldsymbol{f}$, with RBF centers obtained by applying K-means clustering on the training set. Note that our learning and regularization approach still handles all the $60,000$ training samples as described by Algorithm 2. Our method achieves an error rate of 2.74%. While there are many results reported on this dataset, we feel that the most comparable method with our representation is the Radial Basis Function Network with 1000 RBF units [43], which achieves an error rate of 3.6%. This experiment shows the potential that our geometric regularization approach scales to larger datasets.

6.2 Deep Neural Networks

While deep neural networks have achieved great success in many machine learning tasks, some work [79, 56] shows that deep neural networks are vulnerable to certain perturbations of the input. We propose in this thesis a new geometric perspective on overfitting which leads to a regularization technique that exploits the geometry of the class probability estimator with "small local oscillations" in a neighborhood of the training data. Our motivation is closely related to "adversarial examples" [79, 56] and it is interesting to explore whether this geometric regularization scheme can be generalized to alleviate the vulnerability of deep neural networks to adversarial examples.

In this section, we derive specific formulations for applying our geometric regularization technique to general (deep) feedforward neural networks, including convolutional neural networks. In particular, we have designed a closed-form algorithm for deep architectures that incorporates the geometric regularization into the standard forward/backward procedure of network training. We test our implementation on state-of-the-art network models using classification benchmarks with and without adversarial examples.

6.2.1 Formulations and Algorithm

As in Figure. 6.5, denote the input to the network by $\boldsymbol{x} = (x^1, x^2, \ldots, x^d) \in \mathbb{R}^d$, and the activations of the last layer (before softmax) of the network by $\boldsymbol{z} = (z^1, \ldots, z^K)$. The softmax output is $\boldsymbol{f} = (f^1, f^2, \ldots, f^K)$, where $f^i = e^{z^i} / \sum_{j=1}^{K} e^{z^j}$.

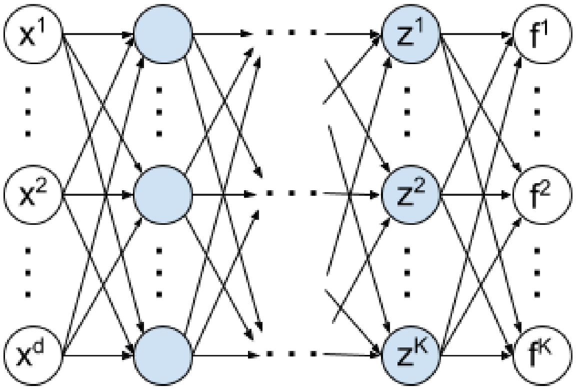


Figure 6.5: A feedforward neural network.

6.2.1.1 Formulation

Given a training set $X_n = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathcal{X}$ and corresponding label set $Y_n = \{y_1, \ldots, y_n\} \subset \mathcal{Y}$, we consider the following objective function for regularized network training,

$$\mathcal{P}(\boldsymbol{f}(\Theta); X_n, Y_n) = \mathcal{P}_L(\boldsymbol{f}(\Theta); X_n, Y_n) + \lambda \mathcal{P}_G(\boldsymbol{f}(\Theta)), \tag{6.16}$$

where Θ denotes the set of all network parameters, and $\mathcal{P}_L$ is some standard classification loss. Our regularization function for the network is

$$\mathcal{P}_G(\boldsymbol{f}) = \int_{\mathrm{gr}(\boldsymbol{f})} \mathrm{dvol}. \tag{6.17}$$

Minimization of $\mathcal{P}$ with respect to Θ can be solved by gradient descent methods, where $\nabla \mathcal{P}_\Theta$ needs to be computed at every gradient step. For a neural network architecture, this is done by back propagating the initial gradient $\nabla \mathcal{P}_{\boldsymbol{f}}$. The computation of $\nabla \mathcal{P}_{L,\boldsymbol{f}}$ is straightforward for standard classification loss, such as the cross-entropy losses. $\nabla\mathcal{P}_{G,\boldsymbol{f}}$ is given by Theorem 1:

$$\nabla\mathcal{P}_{G,\boldsymbol{f}} = -(g^{-1})^{ij}\Big(f^1_{ji} - (g^{-1})^{rs} f^k_{rs} f^k_i f^1_j, \ldots, f^K_{ji} - (g^{-1})^{rs} f^k_{rs} f^k_i f^K_j\Big), \tag{6.18}$$

where $f^k_i = \frac{\partial f^k}{\partial x^i}$, and the Jacobian of $\boldsymbol{f}$ is denoted by

$$J = \left[\frac{\partial f^k}{\partial x^i}\right]_{K\times d} = \left(f^k_i\right)^{k=1,\ldots,K}_{i=1,\ldots,d}, \tag{6.19}$$

$f^k_{ij} = \frac{\partial^2 f^k}{\partial x^i \partial x^j}$, and the Hessian of f^k for some fixed k is denoted by

$$H^k = \left[\frac{\partial^2 f^k}{\partial x^i \partial x^j}\right]_{d\times d} = \left(f^k_{ij}\right)_{i,j=1,\ldots,d}, \tag{6.20}$$

the Riemannian metric matrix is denoted by

$$G = (g_{ij})_{i,j=1,\ldots,d} = I + J^T J, \tag{6.21}$$

where $g_{ij} = \delta^i_j + f^k_i f^k_j$, and $(g^{-1})^{ij} = G^{-1}$.

From Eqns. (6.18) and (6.21), for the proposed regularized training of neural networks, the extra information needed is the Jacobian J and the Hessian H^k for all k. We describe in §6.2.2 an efficient method to compute both J and H^k for general feedforward networks.

6.2.1.2 Algorithm Summary

Algorithm 3 gives a summary of the procedure of (deep) network training with geometric regularization for a mini-batch. The input to the network $\mathcal{N}$ is a training batch $\mathcal{B}_m$, with current trade-off parameter λ and learning rate α. First, we do a standard batch forward to get the network output, and feed the network output together with ground truth labels into the loss layer to get the initial loss gradient $\nabla \mathcal{P}_{L,\boldsymbol{f}}(\mathcal{B}_m)$ for this batch. Then for each data point $\boldsymbol{x}_i$ in the batch, we need to get its Jacobian matrix $U_i = \left[\frac{\partial z^l}{\partial x^j}\right]_{\boldsymbol{x}=\boldsymbol{x}_i}$ and Hessian tensor $T_i = \left[\frac{\partial^2 z^l}{\partial x^j \partial x^k}\right]_{\boldsymbol{x}=\boldsymbol{x}_i}$. Feeding $\boldsymbol{z}_i$, U_i, and T_i into the regularization module, we obtain the corresponding initial gradient $\nabla \mathcal{P}_{G,\boldsymbol{f}}(\boldsymbol{x}_i)$, where $\nabla \mathcal{P}_{G,\boldsymbol{f}}(\boldsymbol{x}_i)$ is the i-th row of the regularization gradient $\nabla \mathcal{P}_{G,\boldsymbol{f}}(\mathcal{B}_m)$ for the whole batch. The last step is a standard batch back-propagation with an initial gradient matrix combining contributions from both the classification loss and the geometric regularization, i.e.$\nabla \mathcal{P}_{L,\boldsymbol{f}}(\mathcal{B}_m) + \lambda \nabla \mathcal{P}_{G,\boldsymbol{f}}(\mathcal{B}_m)$. The network is then updated following the standard stochastic gradient descent (SGD) strategy.

Computational details of the geometric regularization module will be discussed in next subsection.

Algorithm 3 Training with Geometric Regularization - for one mini-batch

```
**Input:** A mini-batch of training data $\mathcal{B}_m = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$, feedforward network $\mathcal{N}$, trade-off parameter $\lambda$, and learning rate $\tau$.
**Step 1:** Forward pass of $\mathcal{B}_m$ in $\mathcal{N}$ to get output matrix $Z = \big(\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_m\big)$.
**Step 2:** Feed $Z$ and the corresponding labels into the softmax + classification loss module to get the initial loss gradient $\nabla \mathcal{P}_{L,\boldsymbol{f}}(\mathcal{B}_m)$ for the whole batch.
**Step 3:**
    for $i = 1$ to $m$ do
        - Compute the Jacobian $U_i = \left[\frac{\partial z^l}{\partial x^j}\right]_{\boldsymbol{x}=\boldsymbol{x}_i}$ and Hessian tensor $T_i = \left[\frac{\partial^2 z^l}{\partial x^j \partial x^k}\right]_{\boldsymbol{x}=\boldsymbol{x}_i}$ through automatic differentiation.
        - Feed $\boldsymbol{z}_i$, $U_i$, and $T_i$ into the geometric regularization module to get the initial regularization gradient $\nabla \mathcal{P}_{G,\boldsymbol{f}}(\boldsymbol{x}_i)$.
    end for
**Step 4:** Back-propagate $\nabla \mathcal{P}_{L,\boldsymbol{f}}(\mathcal{B}_m) + \lambda \nabla \mathcal{P}_{G,\boldsymbol{f}}(\mathcal{B}_m)$ in $\mathcal{N}$ and update the weights of $\mathcal{N}$ using learning rate $\tau$.
```

6.2.2 Computational Details

We now discuss how to efficiently compute the gradient of the regularization term in order to initialize a standard back-propagation.

6.2.2.1 Jacobian and Hessian

First, it is straightforward to compute the softmax output $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{z}(\boldsymbol{x}))$,

Jacobian

$$\begin{aligned}
\frac{\partial f^k}{\partial x^i} &= \sum_{j=1}^{\frac{\partial f^k}{\partial z^j}} \frac{\partial z^j}{\partial x^i}, \\
\implies J &= \left[\frac{\partial f^k}{\partial x^i}\right]_{K\times d} = \left[\frac{\partial f^k}{\partial z^j}\right]_{K\times K} \left[\frac{\partial z^j}{\partial x^i}\right]_{K\times d} = \nabla U,
\end{aligned} \tag{6.22}$$

where $\nabla = \left[\frac{\partial f^k}{\partial z^j}\right]_{K\times K}$ is the Jacobian of softmax function $\boldsymbol{f}(\boldsymbol{z})$, and $U = \left[\frac{\partial z^j}{\partial x^i}\right]_{K\times d}$ is the Jacobian of the network function $\boldsymbol{z}(\boldsymbol{x})$. For fixed k,

$$J^k = \left[\frac{\partial f^k}{\partial z^j}\right]_{1\times K} \left[\frac{\partial z^j}{\partial x^i}\right]_{K\times d} = \nabla_k^T U, \tag{6.23}$$

where $\nabla_k^T = \left[\frac{\partial f^k}{\partial z^j}\right]_{1\times K}$ is the k-th row of ∇.

Hessian

$$\begin{aligned}
\frac{\partial^2 f^k}{\partial x^i \partial x^j} &= \frac{\partial}{\partial x^j}\left(\frac{\partial f^k}{\partial x^i}\right) = \frac{\partial}{\partial x^j}\left(\sum_{l=1}^{K} \frac{\partial f^k}{\partial z^l}\frac{\partial z^l}{\partial x^i}\right) \\
&= \sum_{l=1}^{K} \frac{\partial z^l}{\partial x^i}\frac{\partial}{\partial x^j}\left(\frac{\partial f^k}{\partial z^l}\right) + \sum_{l=1}^{K} \frac{\partial f^k}{\partial z^l}\frac{\partial^2 z^l}{\partial x^i \partial x^j} \\
&= \sum_{l=1}^{K} \frac{\partial z^l}{\partial x^i}\left(\sum_{m=1}^{K} \frac{\partial}{\partial z^m}(\frac{\partial f^k}{\partial z^l})\frac{\partial z^m}{\partial x^j}\right) + \sum_{l=1}^{K} \frac{\partial f^k}{\partial z^l}\frac{\partial^2 z^l}{\partial x^i \partial x^j}
\end{aligned}$$

$$\begin{aligned} &= \sum_{l=1}^{K} \frac{\partial z^l}{\partial x^i} \left(\sum_{m=1}^{K} \frac{\partial^2 f^k}{\partial z^l \partial z^m} \frac{\partial z^m}{\partial x^j} \right) + \sum_{l=1}^{K} \frac{\partial f^k}{\partial z^l} \frac{\partial^2 z^l}{\partial x^i \partial x^j} \\ \implies \forall k, H^k &= \left[\frac{\partial^2 f^k}{\partial x^i \partial x^j} \right]_{d\times d} = \left[\frac{\partial z^l}{\partial x^i} \right]^T_{d\times K} \left[\frac{\partial^2 f^k}{\partial z^l \partial z^m} \right]_{K\times K} \left[\frac{\partial z^m}{\partial x^j} \right]_{K\times d} \\ &\quad + \left[\frac{\partial f^k}{\partial z^l} \right]_{1\times K} \left[\frac{\partial^2 z^l}{\partial x^i \partial x^j} \right]_{K\times d\times d} \\ &= U^T \nabla_k^2 U + \nabla_k^T T, \end{aligned} \tag{6.24}$$

where $\nabla_k^2 = \left[\frac{\partial^2 f^k}{\partial z^l \partial z^m}\right]_{K\times K}$ is the Hessian of the k-th component of the softmax function $\boldsymbol{f}(\boldsymbol{z})$, and $T = \left[\frac{\partial^2 z^l}{\partial x^i \partial x^j}\right]_{K\times d\times d}$ is the Hessian of the network function $\boldsymbol{z}(\boldsymbol{x})$.

According to Eqn. (6.22) and (6.24), to compute J and H^k, we only need the matrix $U = \left[\frac{\partial z^m}{\partial x^j}\right]_{K\times d}$, the tensor $T = \left[\frac{\partial^2 z^l}{\partial x^i \partial x^j}\right]_{K\times d\times d}$, the softmax Jacobian $\nabla = \left[\frac{\partial f^k}{\partial z^j}\right]_{K\times K}$, and the softmax Hessian $\nabla_k^2 = \left[\frac{\partial^2 f^k}{\partial z^l \partial z^m}\right]_{K\times K}$.

Computation of ∇ and ∇_k^2

∇ and ∇_k^2 can be computed explicitly at the softmax layer.

$$\frac{\partial f^k}{\partial z^i} = \delta_i^k f^k - f^k f^i, \tag{6.25}$$

$$\frac{\partial^2 f^k}{\partial z^i \partial z^j} = \delta_{ij}^k f^k - \delta_j^i f^k f^i - \delta_i^k f^i f^j - \delta_j^k f^i f^j + 2 f^i f^j f^k, \tag{6.26}$$

where $\delta_{ij}^k = \mathbb{1}_{\{i=j=k\}}$.

Computation of U

U can be computed by a standard forward pass followed by a back propagation starting from the last layer of neurons (before softmax), where the input gradients to the last layer of neurons is a $K \times K$ identity matrix.

6.2.2.2 Efficient formula of $\nabla \mathcal{P}_{G,\boldsymbol{f}}$

Combining Eqn. (6.18), (6.22) and (6.24), the k-th component of $\nabla \mathcal{P}_{G,\boldsymbol{f}}$ can be computed by

$$\begin{aligned}\nabla \mathcal{P}_{G,\boldsymbol{f}}^{k} &= G^{-1} \bullet \big(U^T \nabla_k^2 U\big) - \sum_{l=1}^{K} \big(G^{-1} \bullet (U^T \nabla_l^2 U)\big)\big(\nabla_l^T U G^{-1} U^T \nabla_k\big) \\ &= \mathrm{Tr}\big(G^{-1} U^T \nabla_k^2 U + G^{-1} \nabla_k T\big) \\ &\quad - \sum_{l=1}^{K} \mathrm{Tr}\big(G^{-1} U^T \nabla_l^2 U + G^{-1} \nabla_l T\big) \cdot \big(\nabla_l^T U G^{-1} U^T \nabla_k\big), \qquad (6.27)\end{aligned}$$

where $\bullet$ denotes the matrix inner product, and Tr denotes the trace of the matrix.

Computation of $G^{-1}U^T$

There are two ways to speed up the computation of $G^{-1}U^T$, with the preferred method depending on the input and output dimension of the network.

The first way involves solving linear equations rather than directly computing G^{-1}. For $G^{-1}U^T = X$, i.e. $G^{-1}(\boldsymbol{u}_1, \dots, \boldsymbol{u}_K) = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_K)$, solve each $\boldsymbol{x}_i$ via the linear system

$$G^{-1}\boldsymbol{u}_i = \boldsymbol{x}_i \Longrightarrow \text{solve } G\boldsymbol{x}_i = \boldsymbol{u}_i. \qquad (6.28)$$

This way is preferred when both d and K are very large.

The second method uses the Woodbury formula on Eqn. (6.21) and (6.22): G^{-1} can be computed by

$$G^{-1} = I - J^T(I + JJ^T)^{-1}J. \qquad (6.29)$$

Although G is a $d \times d$ matrix, Eqn. (6.29) only involves the inversion of a $K \times K$ matrix, which is much more efficient if $K \ll d$.

Sampling Techniques for Computation of $\nabla_k^2 U$ and $G^{-1}\nabla_l T$

For matrices $(\nabla_k^2)_{K\times K}$, $U_{K\times d}$, computing the product $\nabla_k^2 U$ by definition requires

$O(dK^2)$ arithmetic operations. In order to speed it up, a variety of randomized algorithms to approximate the product have been proposed, depending on the scale of our problem and the available computational resources. We give one example of using such a sampling-based algorithm.

If we represent the two matrices in vector form, i.e., let $\nabla_k^2 = (\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_K)$ and $U = (\boldsymbol{u}_1^T, \boldsymbol{u}_2^T, ..., \boldsymbol{u}_K^T)^T$, then

$$\nabla_k^2 U = \sum_{i=1}^{K} \boldsymbol{a}_i \boldsymbol{u}_i^T, \tag{6.30}$$

where each summand is just the outer product of two vectors, which is cheap to compute. The basic idea is to do sampling from these vector outer products, rather than computing all of them.

We define the sampling probability distribution as follows: for each outer product pair $(\boldsymbol{a}_i, \boldsymbol{u}_i^T)$, let

$$p_i = \frac{\|\boldsymbol{a}_i\|^2 + \|\boldsymbol{u}_i\|^2}{\|\nabla_k^2\|_F^2 + \|U\|_F^2}, \tag{6.31}$$

where $\|\nabla_k^2\|_F^2 = \sum_{i=1}^{K} \|\boldsymbol{a}_i\|^2$ and $\|U\|_F^2 = \sum_{i=1}^{K} \|\boldsymbol{u}_i\|^2$. It is easy to check that the summation of the p_i equals 1, which shows it is indeed a distribution.

We sample $\frac{\boldsymbol{a}_i \boldsymbol{u}_i^T}{p_i}$ with probability p_i and repeat L times with replacement. Denote each sample as R_i. It is easy to see that $E[R_i] = \nabla_k^2 U$. For $R = \frac{\sum_{i=1}^{L} R_i}{L}$, it can be shown that even choosing L relatively small ($L \approx \ln K$), R is a good approximation of $\nabla_k^2 U$. For more details of this sampling algorithm, please refer to [81].

The same technique also works for $G^{-1}\nabla_l T$, where both G^{-1} and $\nabla_l T$ are $d \times d$ matrices, and computing their product requires $O(d^3)$ arithmetic operations. Using the above approximation technique will reduce the number of arithmetic operations to $O(Ld^2)$, where $L \approx \ln d$.

6.2.2.3 Difficulties in Computing T

The rectified linear unit (ReLU) [55] function is the commonly-used activation function in state-of-the-art deep neural network models. There are two main advantages of using the ReLU function as activations in neural networks, compared with previous activations, such as the sigmoid and the tanh function. First, the ReLU induces sparsity of hidden units, which is found to be important for classification performance [49]. Second, it does not suffer from the gradient vanishing effect and thus can result in faster training [49]. For networks with ReLU activation functions, $\frac{\partial^2 z^l}{\partial x^i \partial x^j} = 0$ for all inputs $\boldsymbol{x}$ except for those that might hit the non-differentiable singular points of some ReLU functions. For a single ReLU function, the measure of the singular point is 0, which can be ignored in practical algorithms. However, for deeper networks, such a singularity effect can be serious. To intuitively explain the rationale, consider the following composition function of a linear transformation and a ReLU activation,

$$f(\boldsymbol{x}) = \max\{0, \boldsymbol{w}^T\boldsymbol{x} + b\}. \tag{6.32}$$

While the $\mathrm{ReLU}(x) = \max\{0, x\}$ has only one singular point $x = 0$, $f(\boldsymbol{x})$ as defined above has a line of singularity, i.e., $\{\boldsymbol{x} | \boldsymbol{w}^T\boldsymbol{x} + b = 0\}$. As a result, the singularity effect of ReLU activations will be "amplified" with the depth of the network and cannot be ignored.

To address this singularity issue, ideally, we should rely on the automatic differentiation approach [7, 8], such as back-propagation, to compute the first and second derivatives of $\boldsymbol{z}(\boldsymbol{x})$. In practice, however, this computation can be quite slow for second derivatives if the network is deep and the input dimension d is large. For our first implementation, we use instead an approximation method by substituting T

with $\alpha U^l \otimes U^l$, where $U^l = \left[\frac{\partial z^l}{\partial x^i}\right]_{d\times 1}$ is the l-th row of the network Jacobian U, and $\otimes$ denotes the outer product between two vectors. The scalar α is a small real number, which is fixed as 0.001 in our experiments. Using this approximation, the extra cost for our geometric regularization can be reduced to an extra back propagation pass plus a function based on Eqn. (6.27) for every training example.

6.2.3 Qualitative Experiments on Synthetic Data

We use again the toy example of 2D classification. For the representation of $\boldsymbol{f}$, we follow the structure of Figure 6.5, where we use a two-layer fully connected network to represent $\boldsymbol{z}(\boldsymbol{x})$. The training process follows Algorithm 3. Given the difficulties in computing T as described in §6.2.2.3, we have tried two different types of activation functions, the sigmoid activation, which is a smooth function and thus enables an exact computation of T, and the ReLU activation, which depends on the substitution approach described in §6.2.2.3.

Sigmoid activation

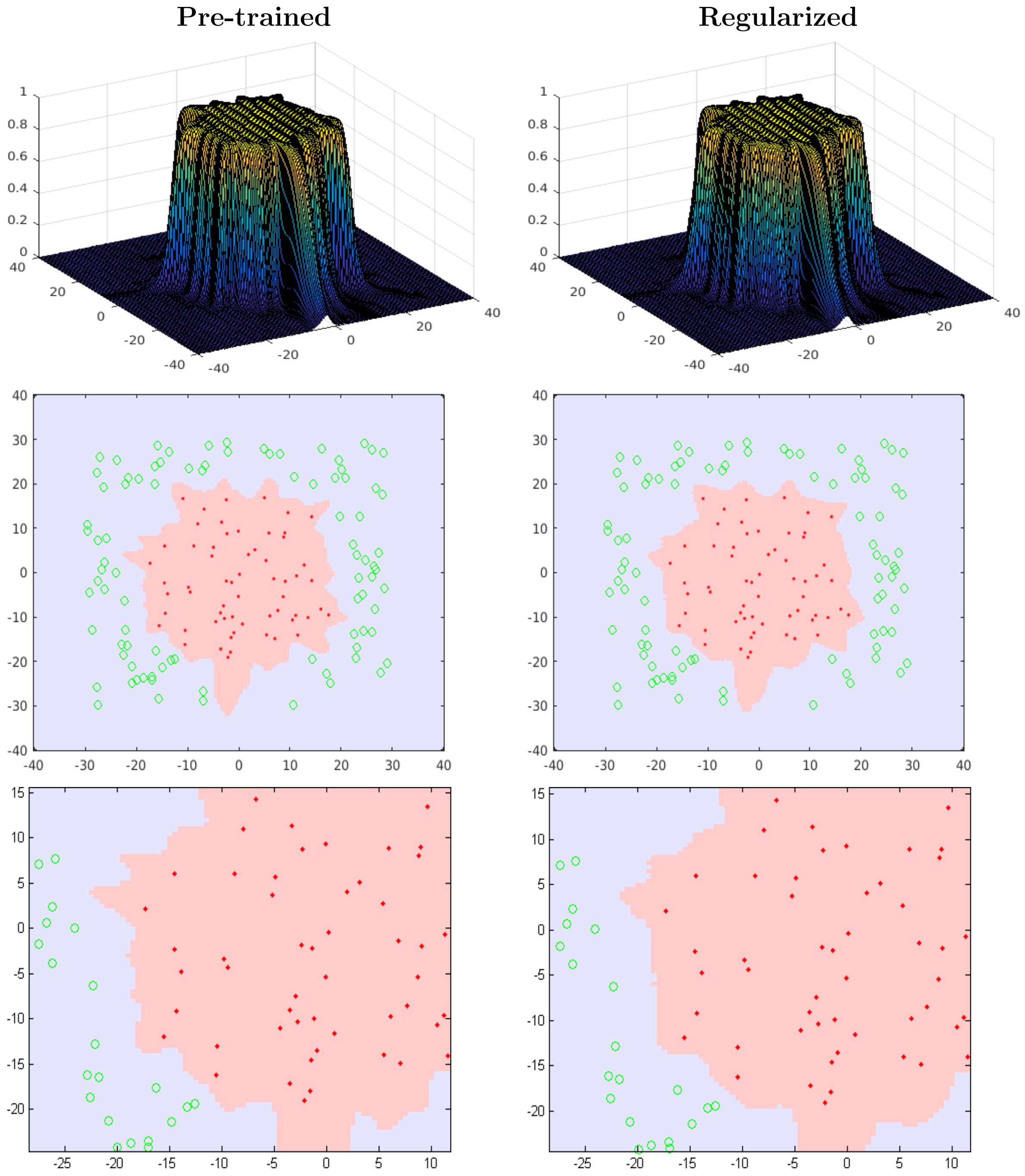
The purpose of testing on this toy example with sigmoid activations, is to provide visible insights on what our regularization approach actually does to the prediction functions represented by neural networks. As we will see, the neural network representation does have its specialties and challenges compared to "shallow" representations, such as the RBF representation in §6.1.

Firstly, we would like to isolate the effect of geometric regularization from any possible difficulties in the back-propagation process. For this purpose, we study the effect of one geometric gradient step applied directly on the network output, by subtracting the network output with the negative geometric gradient vector, i.e.,

$$\boldsymbol{f}(\boldsymbol{x}) \longleftarrow \boldsymbol{f}(\boldsymbol{x}) - \tau \nabla \mathcal{P}_{G,\boldsymbol{f}}(\boldsymbol{x}). \tag{6.33}$$

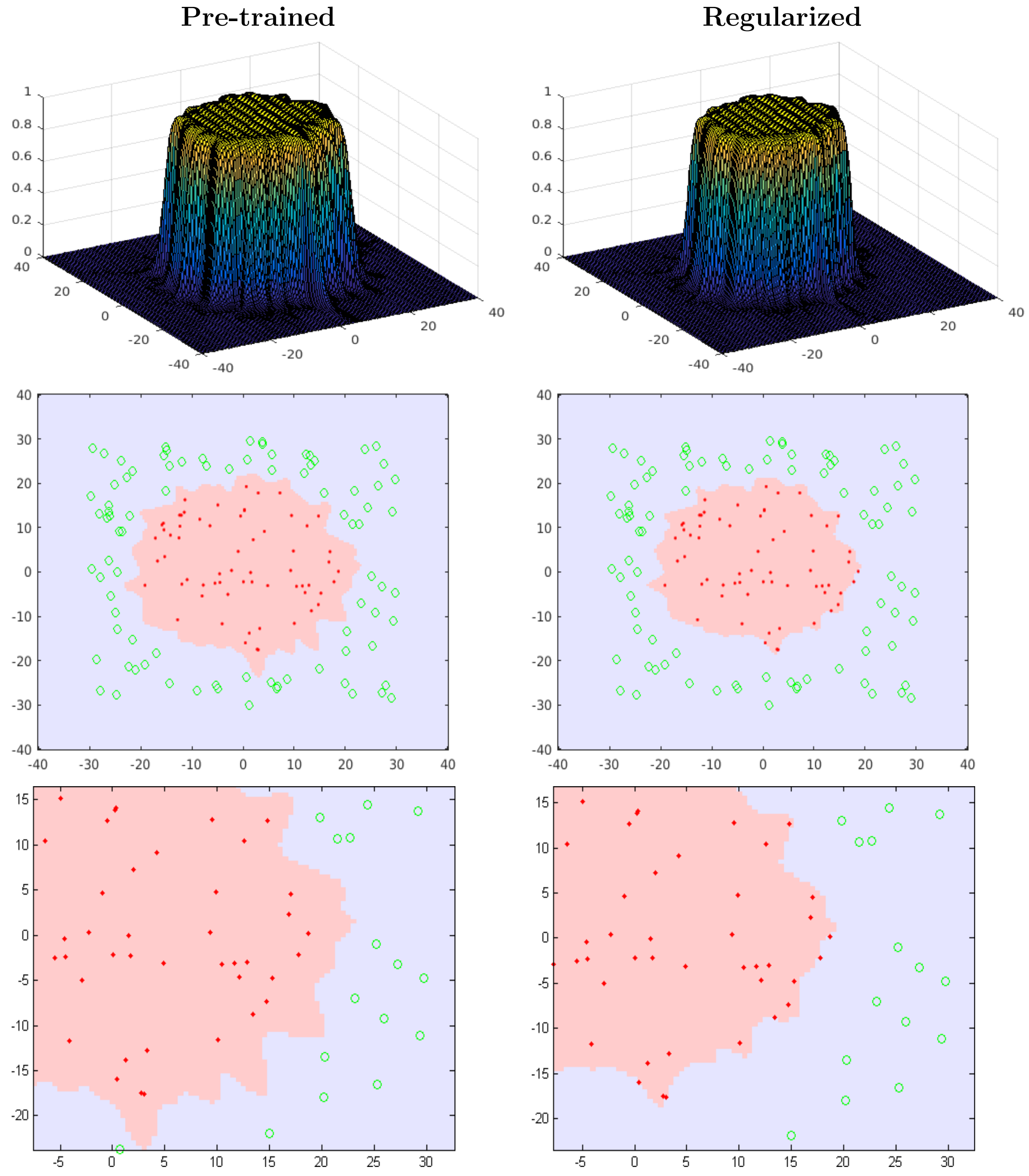
The functional graph (surface) and its corresponding decision boundary before and after one such gradient step is plotted in Figure 6.6. The smoothing effect on the decision boundary, especially in the third row of zoom-in plots, indicates that local oscillations of the surface corresponding to the network function are reduced, which shows how our geometric gradient vector can have the expected effect on the output function of networks, if the second derivatives of network functions can be computed accurately.

Secondly, we add the back-propagation and network update into our process, i.e., after computing the geometric gradient $\nabla \mathcal{P}_{G,\boldsymbol{f}}$, rather than directly updating the output of the network function as Eqn. (6.33), we apply a standard back-propagation of the network with initial gradient $\nabla \mathcal{P}_{G,\boldsymbol{f}}$, and update the network weights accordingly. As shown in Figure 6.7, the expected effect of reducing local oscillations is still observed in the updated network. However, it is not as faithful as that in Figure 6.6, where the gradient update is directly applied to the output of the network function without updating the network weights. The decision boundary in Figure 6.7 is slightly shrunken, while in Figure 6.6, only the smoothing effect of the decision boundary is observed. The implication is that after updating the network weights by back-propagating the geometric gradient vector, the updated network function might deviate from the ideal case of directly updating the output of the network function, as Eqn. (6.33). This problem is also related to widely observed difficulties in training neural networks, where the network function with some classification loss forms a highly nonconvex and nonlinear function over the parameter space.


Pre-trained

Regularized

Figure 6.6: Effect of one geometric gradient step directly applied on the output of the network function. The left column shows the functional graph of a learned two-layer network, its corresponding decision boundary, and a zoom-in plot of the decision boundary. The right column shows all three after one geometric gradient step (6.33). Note that no back-propagation or network update is involved in this example.



Pre-trained **Regularized**

Figure 6.7: Effect of geometric gradient update though back-propagation. The left column shows the functional graph of a learned two-layer network, its corresponding decision boundary, and a zoom-in plot of the decision boundary. The right column shows all three after one epoch of geometric gradient update by back-propagating the geometric gradient $\nabla \mathcal{P}_{G,\boldsymbol{f}}$.

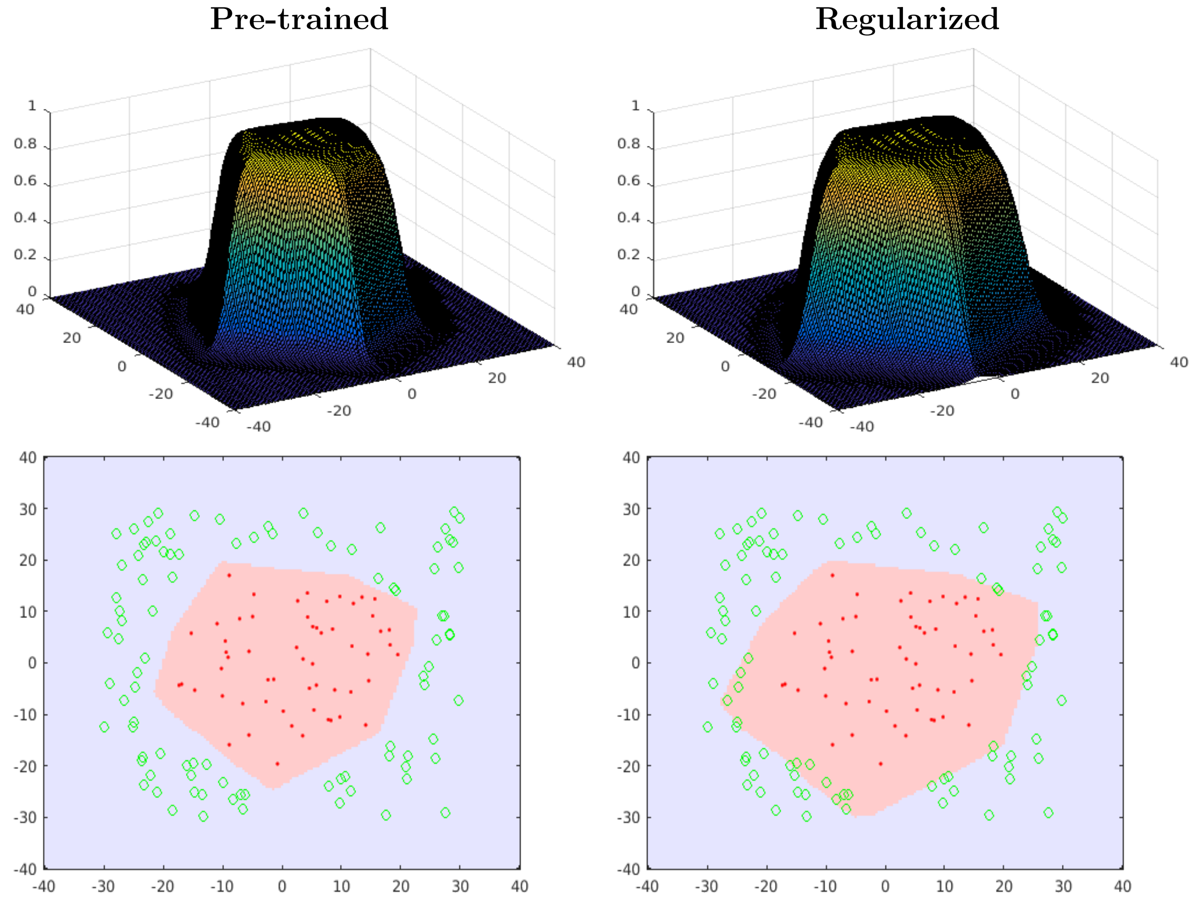


Figure 6.8: Example of binary learning using a two-layer network with ReLU activation, singularity problem of ReLU activations is naively ignored, i.e., $T \equiv 0$ for all input $\boldsymbol{x}$. The left column shows the functional graph of the trained network and its corresponding decision boundary. The right column shows both after 20 epochs of geometric gradient update by back-propagating the geometric gradient $\nabla \mathcal{P}_{G,\boldsymbol{f}}$.

ReLU activation

As explained in §6.2.2.3, for the ReLU activation, the computation of second derivatives of the network function encounters the singularity problem. If we just ignore this problem, i.e., naively setting $T \equiv 0$, then as shown in Figure 6.8, the geometric regularization has no effect on reducing local oscillations, but leads to expansion of the decision boundary. This is due to inaccurate computations of the gradient vectors because of the singularity problem.

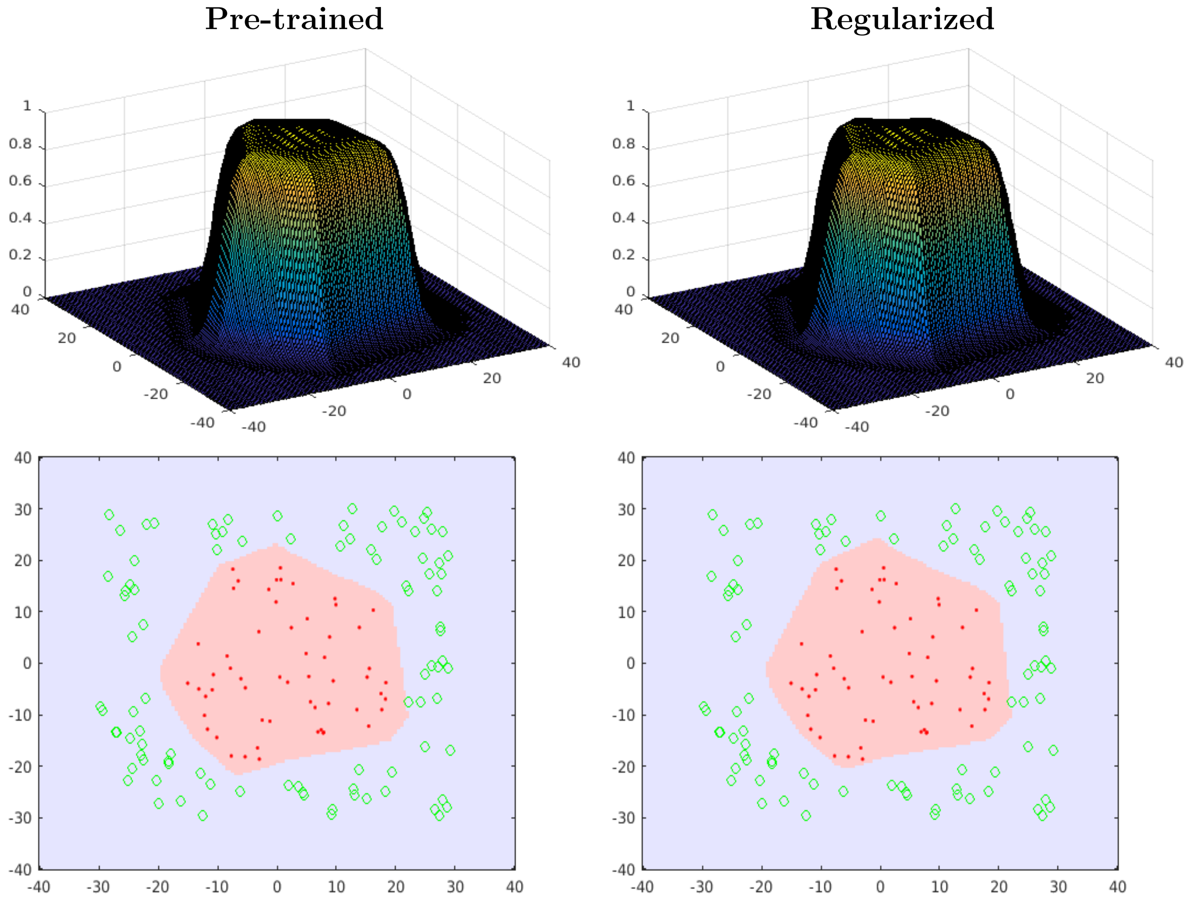


Figure 6.9: Example of binary learning using a two-layer network with ReLU activations, Second derivatives of the network function is substituted using the method introduced in §6.2.2.3. The left column shows the functional graph of the trained network and its corresponding decision boundary. The right column shows both after 20 epochs of geometric gradient update by back-propagation training.

On the other hand, if we use the substitution method introduced in §6.2.2.3 to approximately estimate the second derivatives, as shown in Figure 6.9, the geometric regularization still has little effect on reducing local oscillations, while the expansion effect becomes weaker than naively ignoring the singularities.

6.2.4 Quantitattive Experiments on Benchmarks

To evaluate the effectiveness of the proposed regularization technique on deep neural networks, we apply our approach to a representative network model for classification, i.e., the VGG network [74]. In particular, we use VGG-16 as the baseline and compare it against its counterpart models with two variants of the proposed geometric regularization scheme, i.e., Algorithm 3 and Algorithm 4. We use the substitution approach introduced in §6.2.2.3 to approximate second derivatives.

For evaluation benchmarks, we use the widely-used CIFAR-10 dataset [40] for multiclass image classification task, which contains a training set of 50,000 three channel color images of size 32×32, and a testing set of 10,000 images of the same format. The total number of classes is ten. We report two sets of experiments on CIFAR-10 in the following subsections. The first set of experiments focuses on the performance of a standard supervised learning task, following the standard setup. The second set of experiments focuses on the classifier's robustness with respect to adversarial examples, following the setup of [38] and [21].

6.2.4.1 Implementation details

Alternating training

Besides the regularized training algorithm as shown in Algorithm 3, we also implement a different strategy of incorporating our regularization process into the standard network training. As summarized in Algorithm 4, every epoch of the regularized training is split into one epoch of standard network training and another epoch of geometric regularization training. In other words, fitting the training data and penalizing excessive local oscillations are two alternating procedures within one epoch.

CUDA C implementation of Eqn. (6.27)

Algorithm 4 Alternating training with Geometric Regularization - for one epoch

```
Input: A set of training data $\mathcal{B}_m = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$, feedforward network $\mathcal{N}$, trade-off parameter $\lambda$, learning rate $\tau$, and a small parameter $\alpha$.
Step 1: Standard forward-backward training for one epoch and update the network weights.
Step 2:
    for every training example $\boldsymbol{x}_i$ do
        - Forward pass of $\boldsymbol{x}_i$ into $\mathcal{N}$ to get $\boldsymbol{z}_i$.
        - Compute the Jacobian $U_i = \frac{\partial \boldsymbol{z}_i}{\partial \boldsymbol{x}_i}$ by back-propagation, and substituting $T_i$ with $\alpha U_i^l \otimes U_i^l$.
        - Feed $\boldsymbol{z}_i$, $U_i$, and $T_i$ into the geometric regularization module to get the initial regularization gradient $\nabla \mathcal{P}_{G,\boldsymbol{f}}(\boldsymbol{x}_i)$.
        - Back-propagate $\nabla \mathcal{P}_{G,\boldsymbol{f}}(\boldsymbol{x}_i)$ in $\mathcal{N}$ and update the weights of $\mathcal{N}$ using learning rate $\tau$.
    end for
```

Even with the efficient formulas therein, computing Eqn. (6.27) could still be very slow: firstly, Eqn. (6.27) is the formula for computing only one component of $\nabla \mathcal{P}_{G,\boldsymbol{f}}$, we will need to compute K of them; secondly, in computing $\mathrm{Tr}\big(G^{-1}U^T\nabla_k^2 U\big)$, what we actually need is only the diagonal entries of the $d \times d$ matrix $G^{-1}U^T\nabla_k^2 U$, which involves Kd scalar multiplications given $G^{-1}U^T$ and $\nabla_k^2 U$, while the matrix multiplication $G^{-1}U^T \cdot \nabla_k^2 U$ involves Kd^2 scalar multiplications. Noticing that both computational bottlenecks can be parallelizable, we implement Eqn. (6.27) carefully in a CUDA C function, which achieves approximately 1000 times speedup on a single GPU when $d = 3072$.

6.2.4.2 Standard testing

For initial test, we sampled a subset of 1/10 of the total number of training examples for training, and test on the whole testing set containing 10,000 examples. We conduct five trials of experiments for all comparison models using the same training set without data augmentation. Batch normalization is used for all models.

Table 6.2: Classification performance on CIFAR-10. A same subset of 1/10 of the total number of training examples is used for training each model. Testing set is the default testing set of CIFAR-10, which contains $10,000$ examples and no training example is included. Mean and standard deviation of the testing accuracy are computed over five trials.

Model	# Training examples	Testing accuracy (%)
VGG-16 (baseline)	5000	65.54 ± 0.48
Ours (Algirithm 3)	5000	65.66 ± 0.43
Ours (Algorithm 4)	5000	65.52 ± 0.41

The classification accuracy averaged over five trials on the testing set is shown in Table 6.2. We compare two variants of our regularized learning algorithm, i.e., Algorithm 3 and 4, with the VGG-16 baseline implemented by [87]. From Table 6.2, both training schemes of our regularized approach are comparable, but do not seem to improve the performance over the baseline.

6.2.4.3 Adversarial testing

Recall the cartoon example of Figure 1.3 that motivates our study of the "small local oscillations" of the class probability, we expect that a class probability estimator is robust to reasonably small perturbations of the input. However, as reviewed in §2.6, state-of-the-art neural networks are vulnerable to certain perturbations of the input, known as adversarial examples. In this section, we conduct initial experiments to see if our geometric regularization approach can alleviate the vulnerability of neural networks to adversarial perturbations.

Following [38, 21], we test our implementation on the perturbation mechanism based on the classification loss and the ℓ_2 constraint. In particular, adversarial example perturbed from a given example $\boldsymbol{x}$ is generated by,

$$\boldsymbol{x}'(\boldsymbol{x}, \epsilon) = \boldsymbol{x} + \epsilon \frac{\nabla_{\boldsymbol{x}} \mathcal{P}_L(\boldsymbol{f}(\boldsymbol{x}), y)}{\|\nabla_{\boldsymbol{x}} \mathcal{P}_L(\boldsymbol{f}(\boldsymbol{x}), y)\|_2}, \tag{6.34}$$

where ϵ is the magnitude of the perturbation, and $\mathcal{P}_L$ is the cross-entropy loss for classification.

Adversarial examples are generated for each comparison model by applying Eqn. (6.34) on both the training set and the testing set. Classification accuracy on the perturbed training set under different perturbation magnitudes is shown in Table 6.3, and classification accuracy on the perturbed testing set under different perturbation magnitudes is shown in Table 6.4. While our implementation based on the alternating scheme, i.e., Algorithm 4, does not have any obvious improvement over the baseline, the implementation based on Algorithm 3 performs a bit better than the baseline. Given that our current implementation is based on a substitution method to estimate second derivatives of the network function, as shown in the qualitative experiments (§6.2.3), it might not be very effective in reducing the local oscillations of the functional graph. Thus it is as expected that our current implementation cannot achieve any substantial improvement over the baseline regarding robustness to adversarial examples. However, it is also as expected that our implementation is doing at least no worse than the baseline, as also shown in the toy example of Figure 6.9, our implementation will, at least, not make the surface any worse. Moreover, a bit of improvement from Algorithm 3 indicates that our approach still holds promise, and could be better if the estimation of second derivatives is more precise. It is also reasonable that Algorithm 3 works slightly better than Algorithm 4, since it is more faithful to the original regularized formulation (6.16) than the alternating scheme.

Table 6.3: Classification performance on perturbed training set (results are averaged over three trials).

Model	Averaged accuracy (%)		
	$\epsilon = 0.5$	$\epsilon = 1$	$\epsilon = 2.5$
VGG-16 (baseline)	97.16	79.14	37.34
Ours (Algirithm 3)	97.68	81.66	38.82
Ours (Algorithm 4)	97.84	79.36	35.04

Table 6.4: Classification performance on perturbed testing set (results are averaged over three trials).

Model	Averaged accuracy (%)		
	$\epsilon = 0.5$	$\epsilon = 1$	$\epsilon = 2.5$
VGG-16 (baseline)	57.78	48.99	27.59
Ours (Algorithm 3)	58.37	50.17	29.75
Ours (Algorithm 4)	60.02	50.73	25.77

6.2.4.4 Insight for future experiments

Based on both the qualitative and quantitative experiments as reported above, it is clear that further exploring the potential of our regularization approach for deep neural networks depends on a more accurate computation of second derivatives than our current substitution method. We propose the following experimental setup for next step.

Automatic differentiation (AD) approaches [7, 8] are the most powerful techniques to numerically evaluate the derivatives of any function specified by a computer program. It is especially suitable for computing derivatives of functions specified by a sequence of basic operations, such as deep neural networks. By repeatedly applying chain rule to the sequence of operations, derivatives of working precision can be obtained automatically and efficiently. For the next step, we plan to apply automatic differentiation approaches [7, 8] in our implementation to obtain a more accurate second derivative estimate for neural networks with ReLU activations. AD is already incorporated in the TensorFlow and Theano library for deep learning and there also exists an AD library [19] “Autograd” for torch. The difference is that TensorFlow and Theano will construct the computational graph for automatic differentiation before carrying out any actual computation, i.e., in a static way, while the “Autograd” library will expand the computational graph on-the-fly. Given that our current implementation is based on the Torch library, we plan to try “Autograd” first. However, given that on-the-fly AD is less efficient than the static way of carrying AD, we also plan to move our implementation to Theano if efficiency becomes a concern.

In case the precision obtained by AD is still insufficient for ReLU activations, which could be possible, given that we need second derivatives and ReLU has the singularity problem, we will replace the ReLU activation with the following softplus function:

$$f(x) = \ln(1 + e^x), \tag{6.35}$$

which is a smoothed version of ReLU and does not suffer from the singularity problem. For neural networks with softplus activations, the best way to compute the second derivatives is still by means of automatic differentiation, as discussed above.

With all of these possible implementation changes, we will re-run all previous experiments and more, with networks trained on the whole training set of CIFAR-10. We also plan to test our regularization approach on other state-of-the-art network architectures, such as the ResNet [35].

6.3 Summary

In this chapter, we study the applications of our geometric regularization principle and technique in two representative classification models, the linear combination of radial basis functions and feedforward (deep) neural networks. We obtain parallelizable algorithms and efficient implementations for training both models with our regularization scheme. In experiments with the RBF-based model, we demonstrate the effectiveness of our approach. In initial experiments with deep neural networks, we obtain a small, but nevertheless promising improvement over the baseline regarding robustness to adversarial examples, based on a naive substitution method for estimating second derivatives. We then analyze in detail the experimental results and suggest recipes that can compute second derivatives in a much more precise way. Such next-step implementation improvements have the potential of unlocking the full power of our geometric regularization approach in training deep neural networks.

Chapter 7

Conclusions and Future Work

In this final chapter, we first summarize the key contributions of this thesis: a new geometric perspective on overfitting in supervised learning of classifiers, the first regularization approach that exploits the geometry of the class probability estimator for classification, and an efficient algorithm for applying this regularization approach to feedforward (deep) neural networks. We then describe the major strengths and limitations of our work. Finally, we point out some interesting directions for future research.

7.1 Main Contributions

In this thesis, we study the problem of supervised learning of classifiers. Specifically, we focus on the overfitting problem of classification and the regularization technique to prevent overfitting during training. This problem is crucial for the generalization ability of the learned classifier and many successful regularization techniques have been proposed in the literature of machine learning. The new discovery of this thesis is that there is inherently a differential geometric structure in the class probability estimator, which is closely related to overfitting and the complexity measure of the classification function.

Our study draws insights from both observations in practice and principles in learning theory. In practice, it is widely observed that the class probability does not change dramatically with small perturbations in the input, which we refer to as the "small local oscillations" phenomenon. In learning theory, Occam's razor favors simpler models that explain the training data, and an optimization argument indicates that the regularizer should precisely encode our observed/believed prior of the data distribution. Therefore, our study focuses on investigating a complexity measure of the classification function that properly encodes the "small local oscillations" of the class probability.

To address this problem, we notice that there is a submanifold in the product space $\mathcal{X} \times \Delta^{L-1}$ inherently corresponding to the class probability estimator, and the Riemannian geometry of this submanifold carries some information that is closely related to the amount of "local oscillations" of the class probability estimator. We also notice, more interestingly, that there is an elastic model in physics that naturally corresponds to the learning process of a classifier, where the above Riemannian geometry of the class probability estimator corresponds to a natural extension of the surface energy in physics that regularizes the deformation of the elastic model.

All these studies point to a characterization of the complexity of the classification function in the language of differential geometry, which measures the volume of the functional graph of the class probability estimator. After carefully establishing the geometric foundation for this new perspective, such complexity measurement leads to a new geometric regularization approach for supervised learning of classifiers. In particular, our approach finds the functional graph of the class probability estimator by iteratively fitting the training data in a volume decreasing manner. Solving our variational formulation involves a mean curvature flow based algorithm, which is unified for both binary and multiclass classification and can be easily parallelizable.

For applications, we first apply our regularization technique to a RBF-based representation of the class probability estimator, where our implementation achieves favorable performance comparing with widely used regularization techniques for both binary and multiclass classification. We then develop specific formulations and algorithms to incorporate our regularization technique into the standard forward-backward training of deep neural networks. For theoretical analysis, we establish Bayes consistency for a specific loss function under some mild initialization assumptions, and discuss the extension of our approach to situations where the input space itself is a submanifold.

7.2 Strength and Limitations

The main strength of our approach is that it encodes some geometric information that is closely related to the widely observed "small local oscillation" prior of the underlying class probability of classification problems, and such information has been overlooked by previous regularization methods. The hints behind this strength lie in several areas. Firstly, there is much useful information for classification hidden in the class probability, which is also pointed out by other researchers, such as the "dark knowledge" by [36]. Secondly, designing more sophisticated regularization techniques based on carefully studying the nature of the classification problem is beneficial to classifier learning algorithms. Thirdly, differential geometric techniques have further potential to be investigated in machine learning problems, especially those involving high dimensional structures.

Other advantages of our approach include: it is inherently a unified framework for both binary and multiclass classification; It does not rely on any assumptions of the data manifold or the marginal distribution $P(\boldsymbol{x})$ of the data; and, it scales up linearly in the number of classes and quadratically in the input dimension of the problem, which is much more amenable than the exponential growth known as the curse of dimensionality.

The major drawbacks of our approach are in three areas. Firstly, it requires an accurate computation of up to second derivatives of the class probability estimator, which can be expensive for some classification models or undefined or numerically unstable in others. Secondly, it does not provide efficient formulations for processing the training data in a batch mode. Thirdly, the effectiveness of our approach is still restricted by the representation of the classification function, as with all other regularization techniques.

7.3 Interesting Directions for Future Research

In the last section of this thesis, we discuss some interesting directions that extend our current work.

Multi-label classification

> Multi-label classification is getting increasing attention in many real-world applications, where a single object might have multiple semantic meanings. For instance, in document classification, each document may involve multiple topics so that a document can be classified into multiple categories simultaneously. Other examples include protein function classification for genomics and pathology, music/movie categorization for recommendation systems, and semantic scene understanding for autonomous robots and augmented reality. Our regularization approach can be directly extended to multi-label classification by simply changing the output space from a probabilistic simplex (single label case) to a unit hyper-cube (multi-label case). All our geometric foundations,

formulations, and algorithms carry over.

Distilling neural networks

Hinton *et al.* [36] suggested that the “dark knowledge” in the class probability can be used to transfer knowledge from a powerful teacher network to a simpler student network. The teacher network could be an ensemble of different networks trained on the same dataset, which is inefficient to evaluate when applied in practice. Then training a single smaller network under the “guidance” of this teacher network is of practical interest. Another example is the Cross Quality Distillation [78], where high-quality data is available at training time but not at testing time, but the teacher network trained on high-quality data can guide the learning of a student network on low-quality data. Although promising results have been reported, regarding the methodology itself, the regularization technique adopted has not been specifically tailored for this distilling process. In particular, the class probability of training data is the teacher knowledge to guide the learning of the student network; however, the regularization technique remains the same as standard network training as if no such knowledge is provided. Given our regularization approach exploits the underlying geometry of the class probability estimator, it should be quite suitable for regularizing the learning process of the student network.

Auto-encoders for unsupervised learning

Auto-encoders have become the state-of-the-art approaches for unsupervised learning of generative models [37, 86, 3, 85]. The basic architecture is an encoder network followed by a decoder network, with the bottleneck layer in between. A major source of the variants and considerations focuses on constraints/regularizations of this bottleneck layer. Our geometric regularization approach also has a potential for improving the performance of this architecture. To give an example, Goroshin et al. [32] recently proposed a generative model for video frame prediction. One of the key components is a curvature measure computed among three successive frames at the bottleneck layer. It has been shown in [32] that adding this curvature regularization clearly improves the quality of predicted frames. However, the actual curvature computation in [32] is just a finite difference approximation of a quite weak curvature measure. Our regularization approach, instead, provides an exact computation of a more informative curvature measure, which has a potential to further improve the generative result.

More experiments with hyper-parameter tuning for deep neural networks

Our iterative algorithm for neural network training terminates when the gradient converges within a threshold or the maximum epoch number is reached. It is widely known [14] that gradient descent at every iteration is equivalent to the steepest descent with respect to the ℓ_2-norm in the parameter space, and the magnitude of the ℓ_2-norm is inversely related to the gradient descent step-size. Therefore, gradient descent methods implicitly impose regularization along the gradient flow line in the parameter space. It is interesting, then, to further study how such regularization affects our geometric regularization approach in the functional space. There also exist implicit regularization techniques other than explicit functional-norm based regularization methods. For instance, the dropout strategy [75] for network training implicitly approximates some sort of "geometric averaging" over a large ensemble of possible sub-networks, and early stopping [60] is another widely used heuristic to implicitly impose Occam's razor to prevent overfitting in the iterative learning procedure. It is therefore also interesting to empirically study how dropout and early stopping (based on validation error) would work together with our geometric regularization approach.

Further study on inexact gradients

We have discussed the difficulties of accurately computing derivatives for networks with ReLU activations, and shown that it prevents our regularization method from exerting its full potential in network training. We also suggest recipes to improve the accuracy of this computation for future experiments. On the other hand, the problem of inexact gradients [29] is widely encountered in applications. From the application perspective, it would be interesting to study how commonly used tricks, such as stochastic subgradient [62, 73] at non-differentiable points, work with our algorithm. From the theoretical perspective, it would be worthwhile to study the possibility of bounding the prediction error if the actual gradient obtained by the algorithm is within an ϵ-ball of the underlying exact gradient [69].

A refined formula for gradient update of parametric representation

In §5.3.1, we have derived the following formula for computing the geometric gradient vector with respect to parameters of $\boldsymbol{f}$,

$$\left.\frac{\partial \mathcal{P}'_G}{\partial w^i}\right|_{\boldsymbol{w}} = \mathrm{Tr}\ \mathrm{II}^L_{\mathrm{gr}(\boldsymbol{f}_{\boldsymbol{w}})} \overset{\mathcal{M}'}{\cdot} F_* \boldsymbol{e}^i. \tag{7.1}$$

This formula involves an L^2 metric on $\mathcal{M}'$, which is an integration over the functional graph of $\boldsymbol{f}_{\boldsymbol{w}}$. In practice, this integration is replaced by a summation over a finite set of points $\{\boldsymbol{x}_i\} \subset \mathcal{X}$,

$$\mathrm{Tr}\ \mathrm{II}^L_{\mathrm{gr}(\boldsymbol{f}_{\boldsymbol{w}})} \overset{\mathcal{M}'}{\cdot} F_* \boldsymbol{e}^i$$

$$\begin{aligned} &= \int_{\mathcal{X}} \mathrm{Tr}\ \mathrm{II}^{L}_{\mathrm{gr}(\boldsymbol{f}_{\boldsymbol{w}})} \cdot F_{*}\boldsymbol{e}^{i}\boldsymbol{f}^{*}_{\boldsymbol{w}} \mathrm{dvol}_{\mathbb{R}^{L}} \\ &\approx \sum_{i} \mathrm{Tr}\ \mathrm{II}^{L}_{\mathrm{gr}(\boldsymbol{f}_{\boldsymbol{w}})}(\boldsymbol{x}_i) \cdot F_{*}\boldsymbol{e}^{i}(\boldsymbol{x}_i)\delta_{\boldsymbol{x}_i}\sqrt{\det(\boldsymbol{g}(\boldsymbol{x}_i))}. \end{aligned} \tag{7.2}$$

If we omit the term $\sqrt{\det(\boldsymbol{g}(\boldsymbol{x}_i))}$, Eqn. (7.2) is exactly the general formula for computing geometric gradient with respect to parameters of $\boldsymbol{f}$ used throughout Chapter 6, where we followed previous work using variational formulas to enable a fair comparison. The extra term $\sqrt{\det(\boldsymbol{g}(\boldsymbol{x}_i))}$ in Eqn. (7.2)can be regarded as a correction term caused by the distortion of the volume form on the functional graph of $\boldsymbol{f}_{\boldsymbol{w}}$. It would be interesting to test the effect of this correction term in practice. Note that there is almost no extra computational burden given that $\boldsymbol{g}(\boldsymbol{x}_i)$ is already computed for each $\boldsymbol{x}_i$ in the computation of Tr $\mathrm{II}^{L}_{\mathrm{gr}(\boldsymbol{f}_{\boldsymbol{w}})}(\boldsymbol{x}_i)$.

Generalization to discrete input sets

It would also be interesting to study the possibility of generalizing our framework to the discrete input case, for instance, the input space is a weighted undirected graph $G = (V, E, W)$. One such problem is studied in [41], where a value function $f : T \to \mathbb{R}$ is also given on a subset T of vertex set V. The goal is to learn a value function $\tilde{f}$ on V that agrees with f on T, such that values assigned by $\tilde{f}$ are as smooth as possible across edges. To enforce smoothness, functional norms [41], such as ℓ_0, ℓ_1, and the graph Laplacian norm [90] are widely used as a regularization term. If we can find a discrete approximation of the local chart at every vertex of G, where the dimensionality of the chart depends on the degree of the vertex, we should be able to study the geometry of the functional graph (in the geometric sense) of the value function of the graph (in the discrete sense). If first derivatives of the value function at each vertex can be approximated by operators depending on the weights of edges connecting each vertex, such as those used in the graph Laplacian [9], then our volume-based regularization approach can be introduced to this problem to learn a less oscillating value function on the graph. Moreover, given the discrete approximation in computing derivatives, it is possible that the regularized optimization problem will turn out to be a quadratic program which can be solved directly, without relying on the gradient flow procedure introduced in this thesis.

The structure of the loss surfaces of neural networks

Deep neural networks are traditionally considered to be very difficult to train because of the non-convexity. However, the great success in many applications indicates that training such a non-convex objective function might not be that difficult, given some special structure of loss surfaces of the deep networks. Some recent work [18, 68, 39] has already revealed some aspects of the specialties of the network structure that facilitate the training procedure. We also notice in our experiments with neural networks alternating between a standard classification loss and our regularization loss in gradient descent may sometimes quickly move the solution from one saddle point of the classification loss surface to another one. This raises an interesting topic/conjecture for future research that by carefully designing an extra complementary loss surface and alternating between the standard loss and the geometric regularization loss, the network training could be much improved.

The above gives some potentially fruitful directions for future investigation. Applications of our framework to solve open challenges in supervised learning in deep neural networks are particularly intriguing/exciting, as indicated by our pilot study in the application of our geometric flow framework to deep neural networks for image classification in Chapter 6. Section 6.2.4.4 describes natural next steps for extending and evaluating our formulation in this particular application setting.

Bibliography

[1] Audibert, J.-Y., and Tsybakov, A. Fast learning rates for plug-in classifiers. *Annals of Statistics 35*, 2 (2007), 608–633.

[2] Audin, M., and Damian, M. *Morse Theory and Floer Homology.* Universitext. Springer, London; EDP Sciences, Les Ulis, 2014.

[3] Badrinarayanan, V., Kendall, A., and Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv:1511.00561* (2015).

[4] Bai, Q., Gold, D., and Rosenberg, S. In preperation.

[5] Bai, Q., Rosenberg, S., Wu, Z., and Sclaroff, S. Differential geometric regularization for supervised learning of classifiers. *ICML* (2016).

[6] Bartlett, P. L., Jordan, M. I., and McAuliffe, J. D. Convexity, classification, and risk bounds. *Journal of the American Statistical Association 101*, 473 (2006), 138–156.

[7] Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I., Bergeron, A., Bouchard, N., Warde-Farley, D., and Bengio, Y. Theano: new features and speed improvements. *arXiv:1211.5590* (2012).

[8] Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. Automatic differentiation in machine learning: a survey. *arXiv:1502.05767* (2015).

[9] Belkin, M., and Niyogi, P. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Proc. Advances in Neural Information Processing Systems (NIPS)* (2001), vol. 14, pp. 585–591.

[10] Belkin, M., and Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation 15*, 6 (2003), 1373–1396.

[11] Belkin, M., Niyogi, P., and Sindhwani, V. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research 7* (2006), 2399–2434.

[12] Breiman, L. Random forests. *Machine Learning 45*, 1 (2001), 5–32.

[13] Brezis, H. *Functional Analysis, Sobolev Spaces and Partial Differential equations.* Springer Science & Business Media, 2010.

[14] Bubeck, S., et al. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning 8*, 3-4 (2015), 231–357.

[15] Cai, X., and Sowmya, A. Level learning set: A novel classifier based on active contour models. In *Proc. European Conf. on Machine Learning (ECML).* 2007, pp. 79–90.

[16] Chang, C.-C., and Lin, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology 2* (2011), 27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[17] Chen, Y.-G., Giga, Y., and Goto, S. Uniqueness and existence of viscosity solutions of generalized mean curvature flow equations. In *Fundamental Contributions to the Continuum Theory of Evolving Phase Interfaces in Solids.* Springer, Berlin, 1999, pp. 375–412.

[18] Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y. The loss surfaces of multilayer networks. In *Proc. International Conf. on Artificial Intelligence and Statistics* (2015).

[19] Cortex, T. Github project: Autograd for torch. `https://github.com/twitter/torch-autograd`, 2016.

[20] Cover, T., and Hart, P. Nearest neighbor pattern classification. *IEEE Trans. Information Theory 13*, 1 (1967), 21–27.

[21] Demyanov, S., Bailey, J., Kotagiri, R., and Leckie, C. Invariant backpropagation: how to train a transformation-invariant neural network. *arXiv:1502.04434* (2015).

[22] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2009), IEEE, pp. 248–255.

[23] Devroye, L., Györfi, L., and Lugosi, G. *A Probabilistic Theory of Pattern Recognition.* Springer, 1996.

[24] Dieudonné, J. *Foundations of Modern Analysis.* Read Books Ltd, 2013.

[25] Donoho, D., and Grimes, C. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences 100*, 10 (2003), 5591–5596.

[26] Fawzi, A., Moosavi-Dezfooli, S.-M., and Frossard, P. Robustness of classifiers: from adversarial to random noise. *arXiv:1608.08967* (2016).

[27] Fefferman, C., Mitter, S., and Narayanan, H. Testing the manifold hypothesis. *Journal of the American Mathematical Society* (2016).

[28] Freund, Y., and Schapire, R. E. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory* (1995), Springer, pp. 23–37.

[29] Fukuda, E. H., and Drummond, L. G. Inexact projected gradient method for vector optimization. *Computational Optimization and Applications 54*, 3 (2013), 473–493.

[30] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. Book in preparation for MIT Press, 2016.

[31] Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv:1412.6572* (2014).

[32] Goroshin, R., Mathieu, M. F., and LeCun, Y. Learning to linearize under uncertainty. In *Advances in Neural Information Processing Systems* (2015), pp. 1234–1242.

[33] Gu, S., and Rigazio, L. Towards deep neural network architectures robust to adversarial examples. *arXiv:1412.5068* (2014).

[34] Guckenheimer, J., and Worfolk, P. Dynamical systems: Some computational problems. In *Bifurcations and Periodic Orbits of Vector Fields*. Springer, 1993, pp. 241–277.

[35] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *arXiv:1512.03385* (2015).

[36] Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv:1503.02531* (2015).

[37] Hinton, G. E., Krizhevsky, A., and Wang, S. D. Transforming auto-encoders. In *Artificial Neural Networks and Machine Learning–ICANN 2011*. Springer, 2011, pp. 44–51.

[38] Huang, R., Xu, B., Schuurmans, D., and Szepesvári, C. Learning with a strong adversary. *CoRR, abs/1511.03034* (2015).

[39] Kawaguchi, K. Deep learning without poor local minima. *arXiv:1605.07110* (2016).

[40] Krizhevsky, A., and Hinton, G. Learning multiple layers of features from tiny images. *Citeseer* (2009).

[41] Kyng, R., Rao, A., Sachdeva, S., and Spielman, D. A. Algorithms for lipschitz learning on graphs. In *Proc. Conf. on Learning Theory (COLT)* (2015), pp. 1190–1223.

[42] Lawson, Jr., H. B. *Lectures on Minimal Submanifolds. Vol. I*, second ed., vol. 9 of *Mathematics Lecture Series*. Publish or Perish, Inc., Wilmington, Del., 1980.

[43] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 11 (1998), 2278–2324.

[44] Lee, J. *Introduction to Smooth Manifolds*, vol. 218. Springer Science & Business Media, 2012.

[45] Lin, T., Xue, H., Wang, L., Huang, B., and Zha, H. Supervised learning via euler's elastica models. *Journal of Machine Learning Research 16* (2015), 3637–3686.

[46] Lin, T., Xue, H., Wang, L., and Zha, H. Total variation and Euler's elastica for supervised learning. *Proc. International Conf. on Machine Learning (ICML)* (2012).

[47] Lin, T., and Zha, H. Riemannian manifold learning. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 30*, 5 (2008), 796–809.

[48] Lyu, C., Huang, K., and Liang, H.-N. A unified gradient regularization family for adversarial examples. In *IEEE International Conference on Data Mining (ICDM)* (2015), IEEE, pp. 301–309.

[49] Maas, A. L., Hannun, A. Y., and Ng, A. Y. Rectifier nonlinearities improve neural network acoustic models. In *Proc. International Conf. on Machine Learning (ICML)* (2013), vol. 30.

[50] MacKay, D. J. *Information Theory, Inference and Learning Algorithms.* Cambridge university press, 2003.

[51] Mantegazza, C. *Lecture Notes on Mean Curvature Flow*, vol. 290 of *Progress in Mathematics*. Birkhäuser/Springer Basel AG, Basel, 2011.

[52] Milnor, J. *Morse Theory.* Annals of Mathematics Studies, No. 51. Princeton University Press, Princeton, N.J., 1963.

[53] Miyato, T., Maeda, S. I., Koyama, M., Nakae, K., and Ishii, S. Distributional smoothing with virtual adversarial training. *Proc. International Conf. on Learning Representations (ICLR)* (2015).

[54] Mumford, D., and Shah, J. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics 42*, 5 (1989), 577–685.

[55] Nair, V., and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Proc. International Conf. on Machine Learning (ICML)* (2010), pp. 807–814.

[56] Nguyen, A., Yosinski, J., and Clune, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2015), IEEE, pp. 427–436.

[57] Nøkland, A. Improving back-propagation by adding an adversarial gradient. *arXiv:1510.04189* (2015).

[58] Ororbia, I., Alexander, G., Giles, C. L., and Kifer, D. Unifying adversarial training algorithms with flexible deep data gradient regularization. *arXiv:1601.07213* (2016).

[59] Osher, S., and Sethian, J. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics 79*, 1 (1988), 12–49.

[60] Prechelt, L. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks 11*, 4 (1998), 761–767.

[61] Quinlan, J. R. *C4. 5: programs for machine learning.* Elsevier, 2014.

[62] Ram, S. S., Nedić, A., and Veeravalli, V. V. Distributed stochastic subgradient projection algorithms for convex optimization. *Journal of optimization theory and applications 147*, 3 (2010), 516–545.

[63] Rish, I. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence* (2001), vol. 3, IBM New York, pp. 41–46.

[64] Roscher, R., Förstner, W., and Waske, B. I 2 vm: incremental import vector machines. *Image and Vision Computing 30*, 4 (2012), 263–278.

[65] Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review 65*, 6 (1958), 386.

[66] Roweis, S., and Saul, L. Nonlinear dimensionality reduction by locally linear embedding. *Science 290*, 5500 (2000), 2323–2326.

[67] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV) 115*, 3 (2015), 211–252.

[68] Safran, I., and Shamir, O. On the quality of the initial basin in overspecified neural networks. *Proc. International Conf. on Machine Learning (ICML)* (2016).

[69] Schmidt, M., Roux, N. L., and Bach, F. R. Convergence rates of inexact proximal-gradient methods for convex optimization. In *Advances in neural information processing systems* (2011), pp. 1458–1466.

[70] Schölkopf, B., and Smola, A. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT press, 2002.

[71] Sethian, J. A. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, vol. 3. Cambridge University Press, 1999.

[72] Sharan, L., Rosenholtz, R., and Adelson, E. Material perception: What can you see in a brief glance? *Journal of Vision 9*, 8 (2009), 784–784.

[73] Shor, N. Z. *Nondifferentiable optimization and polynomial problems*, vol. 24. Springer Science & Business Media, 2013.

[74] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556* (2014).

[75] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research 15*, 1 (2014), 1929–1958.

[76] Steinwart, I. Consistency of support vector machines and other regularized kernel classifiers. *IEEE Trans. Information Theory 51*, 1 (2005), 128–142.

[77] Stone, C. Consistent nonparametric regression. *Annals of Statistics* (1977), 595–620.

[78] Su, J.-C., and Maji, S. Cross quality distillation. *arXiv:1604.00433* (2016).

[79] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv:1312.6199* (2013).

[80] Tenenbaum, J., De Silva, V., and Langford, J. A global geometric framework for nonlinear dimensionality reduction. *Science 290*, 5500 (2000), 2319–2323.

[81] Tropp, J. A. An introduction to matrix concentration inequalities. *Foundations and Trends in Machine Learning 8*, 1-2 (2015), 1–230.

[82] Vapnik, V. N. *Statistical Learning Theory*, vol. 1. Wiley New York, 1998.

[83] Varshney, K., and Willsky, A. Classification using geometric level sets. *Journal of Machine Learning Research 11* (2010), 491–516.

[84] Vedaldi, A., and Fulkerson, B. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.

[85] Yang, J., Price, B., Cohen, S., Lee, H., and Yang, M.-H. Object contour detection with a fully convolutional encoder-decoder network. *arXiv:1603.04530* (2016).

[86] Yang, J., Reed, S. E., Yang, M.-H., and Lee, H. Weakly-supervised disentangling with recurrent transformations for 3d view synthesis. In *Proc. Advances in Neural Information Processing Systems (NIPS)* (2015), pp. 1099–1107.

[87] Zagoruyko, S. Github project: cifar.torch. `https://github.com/szagoruyko/cifar.torch/`, 2016.

[88] Zhang, Z., and Zha, H. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM Journal on Scientific Computing 26*, 1 (2005), 313–338.

[89] Zhou, D., and Schölkopf, B. Regularization on discrete spaces. In *Pattern Recognition*. Springer, 2005, pp. 361–368.

[90] Zhou, X., and Belkin, M. Semi-supervised learning by higher order regularization. In *Proc. International Conf. on Artificial Intelligence and Statistics* (2011), pp. 892–900.

[91] Zhu, J., and Hastie, T. Kernel logistic regression and the import vector machine. *Journal of Computational and Graphical Statistics* (2005).

Curriculum Vitae of Qinxun Bai

Address

MCS 210, 111 Cummington Mall
Department of Computer Science, Boston University
Boston, 02215, MA, US

Email qinxun@bu.edu

Website http://cs-people.bu.edu/qinxun/

[redacted] [redacted]

Education

Ph.D in Computer Science
· Boston University, Sep. 2010 – Oct. 2016
· Advisors: Prof. Stan Sclaroff and Prof. Steven Rosenberg

Master of Science in Pattern Recognition and Intelligent Systems
· Chinese Academy of Sciences, China, Sep. 2006 – May. 2010
· Advisor: Prof. Yihong Wu

Bachelor of Engineering in Electronic Engineering
· Tsinghua University, China, Sep. 2002 – June. 2006

Publications

Qinxun Bai, Steven Rosenberg, Zheng Wu, and Stan Sclaroff. Differential Geometric Regularization for Supervised Learning of Classifiers. In *Proceedings of International Conference on Machine Learning (ICML)*, 2016.

Qinxun Bai, Henry Lam, and Stan Sclaroff. A Bayesian Approach for Online Classifier Ensemble. arXiv:1507.02011, 2015.

Qinxun Bai, Henry Lam, and Stan Sclaroff. A Bayesian Framework for Online Classifier Ensemble. In *Proceedings of International Conference on Machine Learning (ICML)*, 2014.

Qinxun Bai, Zheng Wu, Stan Sclaroff, Margrit Betke, and Camille Monnier. Randomized Ensemble Tracking. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2013.

Qinxun Bai, Yihong Wu, and Lixin Fan. PCA-based Structure Refinement for Reconstruction of Urban Scene. In *Proceedings of IEEE International Conference on Image Processing (ICIP)*, 2010.

Zhen Lei, **Qinxun Bai**, Ran He, and Stan Z. Li. Face Shape Recovery from a Single Image Using CCA Mapping between Tensor Spaces. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.