Learning from Geometry

by

Jiaji Huang

Department of Electrical and Computer Engineering
Duke University

Date: _____

Approved:

_____
Robert Calderbank, Supervisor

_____
Lawrence Carin

_____
Ingrid Daubechies

_____
Gallen Reeves

_____
Guillermo Sapiro

Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical and Computer Engineering in the Graduate School of Duke University
2016

Abstract

Learning from Geometry

by

Jiaji Huang

Department of Electrical and Computer Engineering
Duke University

Date: _____

Approved:

_____
Robert Calderbank, Supervisor

_____
Lawrence Carin

_____
Ingrid Daubechies

_____
Gallen Reeves

_____
Guillermo Sapiro

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical and Computer Engineering
in the Graduate School of Duke University
2016

Abstract

Subspaces and manifolds are two powerful models for high dimensional signals. Subspaces model linear correlation and are a good fit to signals generated by physical systems, such as frontal images of human faces and multiple sources impinging at an antenna array. Manifolds model sources that are not linearly correlated, but where signals are determined by a small number of parameters. Examples are images of human faces under different poses or expressions, and handwritten digits with varying styles. However, there will always be some degree of model mismatch between the subspace or manifold model and the true statistics of the source. This dissertation exploits subspace and manifold models as prior information in various signal processing and machine learning tasks.

A near-low-rank Gaussian mixture model measures proximity to a union of linear or affine subspaces. This simple model can effectively capture the signal distribution when each class is near a subspace. This dissertation studies how the pairwise geometry between these subspaces affects classification performance. When model mismatch is vanishingly small, the probability of misclassification is determined by the product of the sines of the *principal angles* between subspaces. When the model mismatch is more significant, the probability of misclassification is determined by the sum of the squares of the sines of the principal angles. Reliability of classification is derived in terms of the distribution of signal energy across principal vectors. Larger principal angles lead to smaller classification error, motivating a linear transform

that optimizes principal angles. This linear transformation, termed TRAIT, also preserves some specific features in each class, being complementary to a recently developed Low Rank Transform (LRT). Moreover, when the model mismatch is more significant, TRAIT shows superior performance compared to LRT.

The manifold model enforces a constraint on the freedom of data variation. Learning features that are robust to data variation is very important, especially when the size of the training set is small. A learning machine with large numbers of parameters, e.g., deep neural network, can well describe a very complicated data distribution. However, it is also more likely to be sensitive to small perturbations of the data, and to suffer from suffer from degraded performance when generalizing to unseen (test) data. From the perspective of complexity of function classes, such a learning machine has a huge capacity (complexity), which tends to overfit. The manifold model provides us with a way of regularizing the learning machine, so as to reduce the *generalization error*, therefore mitigate overfiting. Two different overfiting-preventing approaches are proposed, one from the perspective of data variation, the other from capacity/complexity control. In the first approach, the learning machine is encouraged to make decisions that vary smoothly for data points in local neighborhoods on the manifold. In the second approach, a graph adjacency matrix is derived for the manifold, and the learned features are encouraged to be aligned with the principal components of this adjacency matrix. Experimental results on benchmark datasets are demonstrated, showing an obvious advantage of the proposed approaches when the training set is small.

Stochastic optimization makes it possible to track a slowly varying subspace underlying streaming data. By approximating local neighborhoods using affine subspaces, a slowly varying manifold can be efficiently tracked as well, even with corrupted and noisy data. The more the local neighborhoods, the better the approximation, but the higher the computational complexity. A multiscale approximation

scheme is proposed, where the local approximating subspaces are organized in a tree structure. Splitting and merging of the tree nodes then allows efficient control of the number of neighborhoods. Deviation (of each datum) from the learned model is estimated, yielding a series of statistics for anomaly detection. This framework extends the classical *changepoint detection* technique, which only works for one dimensional signals. Simulations and experiments highlight the robustness and efficacy of the proposed approach in detecting an abrupt change in an otherwise slowly varying low-dimensional manifold.

Contents

Abstract **iv**

List of Tables **xii**

List of Figures **xiii**

List of Abbreviations and Symbols **xvi**

Acknowledgements **xvii**

1 Introduction **1**

2 Subspace Model and Linear Feature Learning **3**

2.1 Near Low-rank Gaussian Mixture Model 4

2.2 Geometric Framework 6

2.3 The MAP Classifier for a GMM 7

2.3.1 The High SNR Regime 8

2.3.2 The Low SNR Regime 10

2.3.3 The Moderate SNR Regime 11

2.3.4 Numerical Analysis of Synthetic Data 13

2.4 Nearest Subspace Classifier: extending GMM 14

2.4.1 Derivation of the Upper Bound 15

2.4.2 Numerical Analysis of Synthetic Data 18

2.5 TRAIT: Tunable Recognition Adapted to Intra-class Target 19

2.5.1 Related Methods 21

2.5.2 Two Properties of the TRAIT Transform 22
2.5.3 Robustness to Model Mismatch 25
2.6 Conclusion 28
3 Local Structure and Robust Feature Learning 30
3.1 Learning Robust Features from a Small Training Set 31
3.2 Problem Formulation 32
3.2.1 Motivation 32
3.2.2 Formulation 33
3.3 Theoretical Analysis 34
3.3.1 Theoretical Framework 35
3.3.2 (K, ϵ)-robustness and Covering Number 36
3.4 An Illustrative Realization of DRT 39
3.4.1 Other Distance 40
3.5 Experimental Results 41
3.5.1 Toy Example 41
3.5.2 MNIST Classfication Using a Very Small Training Set 42
3.5.3 Face Verification on LFW 44
3.6 Conclusion 46
4 *GraphConnect*: where Manifold Models Meet Deep Learning 47
4.1 Generalization Error of Deep Neural Networks 48
4.2 *GraphConnect*: A Motivating Example 50
4.3 A Theoretical Perspective 52
4.3.1 Analysis: Regularizing a Linear Layer 54
4.3.2 Analysis: Regularizing Multiple Layers 57
4.4 Algorithmic Details 58

4.4.1 Choice of Bandwidth σ 60
4.5 Experiments 61
4.5.1 MNIST Revisited 61
4.5.2 Comparison on SVHN and CIFAR-10 62
4.5.3 Face Verification on LFW 63
4.6 Conclusion 67
5 Connecting Subspace and Manifold **68**
5.1 Motivating Application: Changepoint Detection 69
5.2 Problem Formulation 70
5.3 Multiscale Online Union of Subspace Estimation (MOUSSE) 72
5.3.1 Multiscale union of subspaces model 73
5.3.2 MOUSSE Algorithm 75
5.3.3 Distances for MOUSSE 75
5.3.4 Update subset parameters 78
5.3.5 Tree structure update 82
5.3.6 Initialization 83
5.3.7 Choice of parameters 84
5.4 Changepoint detection 84
5.4.1 CUSUM procedure 84
5.4.2 Distribution of e_t 85
5.5 Performance Analysis 86
5.5.1 Optimality of estimator for $\mathbf{c}^*$ 87
5.5.2 MOUSSE tracking error scaling with level 88
5.5.3 Choice of threshold for changepoint detection 88
5.6 Numerical Examples 89

5.6.1 Tracking a static submanifold 89
5.6.2 Tracking a slowly time varying submanifold 90
5.6.3 Comparison of tracking algorithms 92
5.6.4 Changepoint detection example 93
5.6.5 Real data 94
5.7 Conclusions 98
A Supplementary Proofs for Chapter 2 **99**
A.1 Proof of high SNR case 99
A.2 Proof of Low SNR case 100
A.3 Proof of Moderate SNR Case 104
A.4 Analysis of NSC 106
A.5 Proof of Proposition 1 108
Bibliography **109**
Biography **117**

List of Tables

2.1 NSC accuracy on original and 1000 dimensional (compressed) extracted features 28

3.1 Varying λ on a toy dataset. 41

3.2 Implementation details of the neural network for MNIST classification. 43

3.3 Classification error on MNIST. 44

3.4 Verification accuracy and AUCs on LFW 45

4.1 Network architecture in the MNIST experiments, where layer 7 and 8 constitute the softmax classifier. 50

4.2 Network common to SVHN and CIFAR-10 experiments. 64

4.3 SVHN: test accuracy when individual regularizer is used. 64

4.4 CIFAR-10: test accuracy as size of the training set varies. 64

4.5 Fully connected network for face verification. 65

4.6 Verification accuracies and AUCs when using a training set of size 64,000 65

5.1 Average run length (ARL) $\mathbb{E}^{\infty}\{T\}$. 94

5.2 Detection delay when jump of γ_t is $\Delta_\gamma = 0.05$. 95

5.3 Detection delay when jump of γ_t is $\Delta_\gamma = 0.03$. 95

List of Figures

2.1 Error probability as a function of the degree of mismatch. Dashed lines represent empirical estimates, and solid lines represent upper bounds. In the low SNR regime the two upper bounds coincide. 14

2.2 Lines on which $\mathcal{E}$ is constant for the two case studies introduced in section 2.3.4. 17

2.3 Comparison of empirical NSC classification error with the upper bound obtained by numerical integration. (a) Larger principal angles reduce classification error; (b) Disproportionate assignment of signal energy to larger principal angles reduces classification error. 19

2.4 Embeddings of original and transformed data. 24

2.5 MAP classifier's P_e on transformed data. Note that TRAIT (blue) and LRT (red) almost overlap. 25

2.6 NSC's P_e on original/transformed face images. Concatenation of TRAIT and LRT features (TRAIT+LRT) provides superior results 25

2.7 Comparison of original images (top) with TRAIT transformed images (middle) and LRT transformed images (bottom). Red circles indicate structure that is present in both the original and the TRAIT transformed image. 26

2.8 NSC performance on TRAIT and LRT features under different SNR . 27

2.9 From top to bottom row: subjects in PIE, UMIST and ORL database, taken under different poses 27

3.1 (K, ϵ)-robustness: Here $d = \rho(\mathbf{x}_1, \mathbf{x}_2)$, $d' = \rho(\mathbf{x}'_1, \mathbf{x}'_2)$, $e = \rho(f_{\boldsymbol{\alpha}}(\mathbf{x}_1), f_{\boldsymbol{\alpha}}(\mathbf{x}_2))$, and $e' = \rho(f_{\boldsymbol{\alpha}}(\mathbf{x}'_1), f_{\boldsymbol{\alpha}}(\mathbf{x}'_2))$. The difference $|e - e'|$ cannot deviate too much from $|d - d'|$. 33

3.2 Proof without words. 38

3.3 Original and transformed training/testing samples embedded in 2-dimensional space with different colors representing different classes. . 42

3.4 MNIST test: with only 30 training samples per class. We vary λ and assess (a) R_{emp}; (b) generalization error; and (c) 1-nn classification accuracy. Peak accuracy is achieved at $\lambda = 0.25$. 43

3.5 Comparison of ROCs for all methods 45

3.6 Verification accuracy of Euc-DRT as λ varies 45

4.1 Comparing GraphConnect against weight decay on the MNIST dataset 51

4.2 Evaluation of the upper bound (Theorem 12) on the Rademacher complexity for the MNIST benchmark. 51

4.3 (a) *GraphConnect-One* regularizes individual linear layers so that individual outputs align with a graph $\mathbf{W}$; (b) *GraphConnect-All* regularizes the final output features to align with a graph $\mathbf{W}$. 59

4.4 Embedding of initial and transformed test samples with different colors representing different classes. All networks are learned from the same set of 500 training samples. In (d), we observe that numbers with curly strokes are clustered on the left, whereas those with straight strokes are on the right. 61

4.5 Comparing *GraphConnect* against weight decay on the MNIST dataset. 62

4.6 (a) Verification accuracy of GraphConnect and weight decay as a function of the size of the training set; (b) ROC curves when using 64,000 training samples. 66

5.1 Approximation of MOUSSE at $t = 250$ (upper) and $t = 1150$ (lower) of a 100-dimensional submanifold. In this figure we project everything into three-dimensional space. The blue curve corresponds to true submanifold, the dots are noisy samples from the submanifold (the lighter dots are more dated than the darker dots), and the red line segments are the approximation with MOUSSE. As the curvature of the submanifold increases, MOUSSE also adapts in the number of line segments. 73

5.2 Illustration of tree structure for subspaces. The subspaces used in our approximation are $\{\mathcal{S}_{1,0,t} \cup \mathcal{S}_{2,2,t} \cup \mathcal{S}_{2,3,t}\}$. 75

5.3 Q-Q plot of e_t, for a $D = 100$ submanifold. 86

5.4 MOUSSE tracking a static submanifold with $D = 100$ and $d = 1$. . . 90

5.5 MOUSSE tracking a slowly evolving submanifold with $D = 100$ and $d = 1$. Dashed red line depicts CUSUM theoretical threshold calculated for ARL = 1000. 91

5.6 MOUSSE tracking a slowly varying submanifold using: (a) GROUSE, (b) PETRELS-GS and (c) PETRELS-FO. Horizontal axis corresponds to rate of submanifold's change and vertical axis corresponds to fraction of data missing. Brightness corresponds to $\mathbb{E}\{e_t\}$. 92

5.7 Detection of solar flare at $t = 227$: (a) snapshot of original SDO data at $t = 227$; (b) MOUSSE residual $\hat{e}_t$, which clearly identifies an outburst of solar flare; (c) single subspace tracking residual $\hat{e}_t$, which gives a poor indication of the flare; (d) $e(t)$ for MOUSSE which peaks near the flare around $t = 227$; (e) the CUSUM statistic for MOUSSE; (f) $e(t)$ for single subspace tracking; (g) the CUSUM statistic for single subspace tracking. Using a single subspace gives much less reliable estimates of significant changes in the statistics of the frames. 96

5.8 Credit card user data experiments. (a) Number of leaves used by MOUSSE. (b) MOUSSE residual norm. (c) MOUSSE CUSUM statistic (solid blue line) and CUSUM theoretical threshold calculated for ARL = 1000 (dashed red line). (d) Ground truth transaction label with changepoint near where CUSUM statistic starts increasing. . . . 97

List of Abbreviations and Symbols

Symbols

$\mathbb{R}$	real number
$\mathcal{X}, \mathcal{Y}$	subspaces, or a space of data (will be clear in the context)
$\mathcal{F}$	feature space
$\mathbf{x}$,$\mathbf{y}$	data vectors
$\mathbf{U}, \mathbf{V}, \mathbf{X}, \mathbf{Y}$	basis matrices, or a set of vectors (will be clear in the context)
$\mathbf{x}^\top, \mathbf{X}^\top$	transpose a vector or matrix
$\mathbf{X}^\dagger$	pseudo-inverse or matrix
$\mathbf{A}$	linear transform matrix
$\mathbf{I}$	identity matrix
$\mathbf{\Lambda}$	diagonal matrix whose diagonal are the eigenvalues
$\mathbf{\Sigma}$	covariance matrix
θ_i	the i-th principal angle
$\mathbf{x}_i$	the i-th datum
y_i	class label of the i-th datum
$\mathcal{N}$	normal distribution
H_1, H_2	two hypotheses
P_e	classification error
$\mathbb{P}\mathrm{r}(\cdot)$	probability of an event
$\mathbb{E}(\cdot)$	expectation of an event

Acknowledgements

I deeply thank my advisor, Professor Robert Calderbank, who has created a fantastic environment for me to learn and explore. Robert leads the Information Initiative at Duke (IID), which is a very collaborative group of people and interaction between them leads to lots of new ideas. During my PhD, I have collaborated with researchers from several different backgrounds and there were lots of synergies. This could not have been achieved without Robert's help.

Robert is very encouraging. I remember once I got stuck in my research for a long time. Making no progress, I was very worried at that point. In a meeting with Robert, he made the metaphor that research is like flying from New York to Boston with a stop at California. He said trials that have been made would not be lost and would show their value in the long run. I felt much better after getting his encouraging feedback and stopped unnecessary worry. When our papers got accepted, Robert was always the first to congrat me about the good news. When there were negative comments from reviewers, Robert often tried to discover the positive side and motivated me to address the comments in a proper way.

Robert often provides me insights into a problem that I had not thought about before. For example, interpreting the role of principal angles and interpreting the complexity of function class, etc. He has the wisdom of refine complex math into something intuitive yet highly explanatory. This deepens my understanding of the problem.

Robert really cares for students' career. Though occupied by all sorts of management tasks, he made time to attend my rehearsal job talks. He gave me very useful feedbacks, that improved the quality of my talk significantly. I feel very lucky to have Robert as my advisor, who helped me develop a persistent interest in a wide range of research problems.

I am also very grateful to professor Guillermo Sapiro and his postdoc, Dr. Qiang Qiu. They have been very supportive during our collaboration. Guillermo often had very sharp insight towards the research topics, and helped me a lot in writing papers. On the other hand, the frequent discussion with Qiang has been a way of avoiding unnecessary trials and errors. They helped me work in a more efficient and effective way. More than research, they also gave me useful advice towards my career. Guillermo referred me to a very nice research position in MSKCC. Qiang has been encouraging me to work as a productive researcher even after graduation.

I also want to acknowledge Professor Lawrence Carin, Rebecca Willett, Miguel Rodrigues, Yao Xie, Matthew Nokleby, Andrew Thompson, and postdocs Dr. Alireza Vahid, Xin Yuan and Liming Wang. All of them are very brilliant, and I have learned a lot from working with them. There are many more colleagues in IID that I would like to thank. They are all very friendly and highly motivated researchers. I enjoyed the lunches we had together, where we had lots of interesting conversations about research.

Finally, I owe my highest gratitude to my parents. They are hardworking and caring, serving as my role models in the past decades. Their emotional support has been encouraging me to follow my interest and pursue a PhD degree. This dissertation would not have been completed without their support.

1

Introduction

Signals that are nominally high dimensional often exhibit a low dimensional geometric structure. Subspaces provide a powerful way to model these signals. For example, fixed-pose images of human faces are recorded using more than 1000 pixels, but can be represented by a 9-dimensional harmonic subspace [9]. Motion trajectories of a rigid body might be recorded by hundreds of sensors, but must intrinsically be represented by a 4-dimensional subspace [66]. There are many more examples where a low-dimensional subspace model captures intrinsic geometric structure, ranging from user ratings in a recommendation system [69] to signals emitted by multiple sources impinging at an antenna array [55]. Subspace geometry has assisted tasks of interest to both signal processing [88, 22] and machine learning communities[30, 63].

On the other hand, there are cases where a linear or affine subspace is not enough to capture the distribution of the signal. Then one may consider the signal as lying near a manifold [79, 70, 12, 24, 14, 81], which is nonlinearly determined by very few parameters. For example, non-frontal human face images form a manifold that is parameterized by different pitches and yaws [70]. Written digit images are parameterized by different styles of strokes [31]. As these underlying "semantic" information

evolves, the corresponding signal/data form a low dimensional manifold that is embedded in a high dimensional ambient space.

Standard manifold learning methods often estimate pairwise affinity between data samples, and use spectral decomposition to get a representation of the original signal[79, 70, 12]. This type of learning technique is transductive, *i.e.*, a new datum has to be placed together with the training data in order to get its representation. In other words, the training data has to be retained. In contrast, an inductive model removes this burden by learning an explicit mapping from the original data/signal to feature. A linear mapping [40] may be the simplest choice, but may not have enough power. Instead, one could learn a union of subspaces [2, 88], each as a "local" approximation of the manifold. Then, the new datum is projected onto the "local" disk to obtain a representation in the "local" coordinate. The other approach is by using a parameterized nonlinear mapping, e.g., (deep) neural network [41, 37].

This dissertation addresses the inductive feature learning problem for classification and anomaly detection tasks. We show how to leverage the low dimensional geometry, via (union of) subspace(s) or manifold model, to help extract discriminative and robust features. This dissertation is organized as follows. Chapter 2 considers feature extraction for classification, under the assumption that each class lies near a subspace. Chapter 3 uses local distance preservation as a regularization of supervised learning, leading to a feature transform that is discriminative and robust to data variation. Chapter 4 further studies the overfitting problem within deep neural networks and proposes a data dependent regularization by preserving manifold structure. Chapter 5 proposes to approximate a manifold by a union of affine subspaces, from which statistics are extracted for anomaly detection tasks.

2

Subspace Model and Linear Feature Learning

Subspace models play an important role in a wide range of signal processing tasks, and this chapter explores how the pairwise geometry of subspaces influences the probability of misclassification. When the mismatch between the signal and the model is vanishingly small, the probability of misclassification is determined by the product of the sines of the principal angles between subspaces. When the mismatch is more significant, the probability of misclassification is determined by the sum of the squares of the sines of the principal angles. Reliability of classification is derived in terms of the distribution of signal energy across principal vectors. Larger principal angles lead to smaller classification error, motivating a linear transform that optimizes principal angles. The transform presented here (TRAIT) preserves some specific characteristic of each individual class, and this approach is shown to be complementary to a previously developed transform (LRT) that enlarges inter-class distance while suppressing intra-class dispersion. Theoretical results are supported by demonstration of superior classification accuracy on synthetic and measured data even in the presence of significant model mismatch.

2.1 Near Low-rank Gaussian Mixture Model

A Gaussian Mixture Model (GMM) measures proximity to a union of linear or affine subspaces, by imposing a low-rank structure on the covariance of each mixture component. It can be used to approximate a nonlinear manifold by fitting mixture components to local patches of the manifold [21, 88], hence providing a high fidelity representation of a wide variety of signal geometries. The simplicity of the model facilitates signal reconstruction [47, 93, 94, 68], making GMMs a very attractive signal source model in compressed sensing. The value of low-rank GMMs extends to classification, where each class is modeled as a low-rank mixture component, and classes are identified by their projections onto linear features. Optimal feature design is addressed in [20, 63].

The GMM is usually only an approximation to the truth. For example, the full spectrum associated with a face image follows a power law distribution, and when we truncate to the first 9 harmonic dimensions, the residual energy will be a source of error in classification. Even if the true model were a GMM, we can only learn an approximation to the true model from training data. The more data we see, the better is the fit of our empirical model, but some degree of mismatch is unavoidable. If we treat this mismatch as a form of noise, then we can use information theory to derive fundamental limits on the number of classes that can be discerned (see [59] for more details).

This chapter explores how the pairwise geometry of subspaces influences the probability of misclassification. There are parallels with non-coherent wireless communication [43], where information is encoded as a subspace drawn from a fixed alphabet, and the function of the receiver is to distinguish the transmitted subspace. When each component is perfectly modeled as a Gaussian, the performance of the MAP classifier can be analyzed using the Chernoff Bound [28]. When fidelity is per-

fect, there is no mismatch, and fundamental limits on performance are determined by the rank of the intersection of the classes [67, 59].

In this chapter, we further consider how best to discriminate classes, when the alignment between the GMM model and the data is only approximate. We make three main contributions in this chapter:

1. We express the probability of pairwise misclassification in terms of the principal angles between the corresponding subspaces. This expression depends on the mismatch between the signal and the model. Interpreting this mismatch as noise, we provided analysis of the low, moderate, and high SNR regimes. This improves upon [67], in the sense that we have a more explicit expression of the "measurement gain" proposed in [67].

2. We characterize the probability of misclassification for more general distributions near subspaces. This is motivated by the case where training samples per class are insufficient for a reliable estimate of covariance. In these cases, we have very little knowledge about the signal distribution and a MAP classifier is not good fit.

   The Nearest Subspace Classifier (NSC) provides an alternative and we use the NSC classifier rather than the MAP to bound the probability of misclassification.

3. We develop a feature extraction method, TRAIT, that effectively enlarges principal angles between different subspaces and preserves intra-class structure. We demonstrate superior classification accuracy on synthetic and measured data, particularly in the presence of significant model mismatch.

This chapter is organized as follows. Section 2.2 presents the subspace geometry framework. Section 2.3 analyzes the Maximum a Posteriori (MAP) classifier under

the GMM assumption. Section 2.4 analyzes the performance of Nearest Subspace Classifer (NSC), which relaxes the GMM assumption. Section 2.5 proposes a feature extraction method, TRAIT, that exploits subspace geometry, and presents experimental results for both synthetic and measured datasets. Section 2.6 provides a final summary.

2.2 Geometric Framework

Consider two subspaces $\mathcal{X}$ and $\mathcal{Y}$ of $\mathbb{R}^n$ with dimensions ℓ and s respectively, where $\ell \leq s$. The principal angles between $\mathcal{X}$ and $\mathcal{Y}$, denoted as $\theta_1, \ldots, \theta_\ell$, are defined recursively as follows

$$\begin{aligned}
\theta_1 &= \min_{\mathbf{x}_1 \in \mathcal{X}, \mathbf{y}_1 \in \mathcal{Y}} \arccos\left(\frac{\mathbf{x}_1^\top \mathbf{y}_1}{\|\mathbf{x}_1\| \|\mathbf{y}_1\|}\right), \\
&\vdots \\
\theta_j &= \min_{\substack{\mathbf{x}_j \in \mathcal{X}, \mathbf{y}_j \in \mathcal{Y} \\ \mathbf{x}_j \perp \mathbf{x}_1, \ldots, \mathbf{x}_{j-1} \\ \mathbf{y}_j \perp \mathbf{y}_1, \ldots, \mathbf{y}_{j-1}}} \arccos\left(\frac{\mathbf{x}_j^\top \mathbf{y}_j}{\|\mathbf{x}_j\| \|\mathbf{y}_j\|}\right), \quad j = 2, \ldots, \ell.
\end{aligned}$$

The vectors $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$ and $\mathbf{y}_1, \ldots, \mathbf{y}_\ell$, are called principal vectors. The dimension of $\mathcal{X} \cap \mathcal{Y}$ is the multiplicity of zero as a principal angle. It is straightforward to compute the principal angles by calculating the singular values of $\mathbf{X}^\top \mathbf{Y}$, where $\mathbf{X}$ and $\mathbf{Y}$ are orthonormal bases for $\mathcal{X}$ and $\mathcal{Y}$ respectively. The singular values of $\mathbf{X}^\top \mathbf{Y}$ are then $\cos\theta_1, \ldots, \cos\theta_\ell$.

Let $\ell = s$. The principal angles induce several distance metrics on the Grassmann manifold, of which the most widely used is the (squared) chordal distance $\mathcal{D}_c^2(\mathcal{X}, \mathcal{Y})$ [29], given by

$$\mathcal{D}_c^2(\mathcal{X}, \mathcal{Y}) = \sum_{i=1}^{s} \sin^2\theta_i.$$

The chordal distance is an aggregate, and in the following sections we will see how probability of misclassification depends, not so much on this aggregate, but on the individual principal angles.

2.3 The MAP Classifier for a GMM

We begin by considering the MAP classifier, which is optimal when the signal distribution is known. We focus on binary classification, where the two classes are equiprobable, since the generalization from two to many classes is well understood [86, 67].

We model each class as zero mean Gaussian distributed, where the covariance is near low-rank. Classification can be formulated as the following binary hypothesis testing problem

$$\begin{array}{rcl} H_1 : \mathbf{x} & \sim & \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_1) \\ H_2 : \mathbf{x} & \sim & \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_2). \end{array} \tag{2.1}$$

We justify the zero-mean assumption by observing that in applications such as face recognition [87], or motion trajectory segmentation [66], the actual mean is considered as a nuisance parameter, and is removed prior to processing. Given the near-subspace assumption, we model the two covariances as

$$\begin{aligned} \mathbf{\Sigma}_1 &= \mathbf{U}_1 \mathbf{\Lambda}_1 \mathbf{U}_1^\top + \sigma^2 \mathbf{I} \\ \mathbf{\Sigma}_2 &= \mathbf{U}_2 \mathbf{\Lambda}_2 \mathbf{U}_2^\top + \sigma^2 \mathbf{I}. \end{aligned} \tag{2.2}$$

where $\mathbf{U}_1, \mathbf{U}_2 \in \mathbb{R}^{n \times d}$ are the orthonormal bases for the two signal subspaces, denoted by $\mathcal{X}_1$ and $\mathcal{X}_2$. Typically $n \gg d$. $\mathbf{\Lambda}_1, \mathbf{\Lambda}_2 \in \mathbb{R}^{d \times d}$ are diagonal matrices of eigenvalues. We assume that the two subspaces have the same dimension d, and that the diagonal elements of $\mathbf{\Lambda}_1$, $\mathbf{\Lambda}_2$ are arranged in descending order. In the application to motion trajectories we take $d = 4$, and in the application to face recognition we might take $d = 9$. Denote the i-th largest eigenvalue of $\mathbf{\Lambda}_j$ by $\lambda_{j,i}$. Finally let σ^2 be the variance of the noise, which quantifies the degree of mismatch between the subspace model and the data.

Denote the probability of mistaking hypothesis 2 for hypothesis 1 by $\Pr(H_2|H_1)$, and define $\Pr(H_1|H_2)$ similarly. Under the assumption that the two hypotheses are

equiprobable, the error probability P_e of a MAP (optimal) classifier is

$$\begin{aligned} P_e &= \frac{1}{2}[\mathbb{P}\mathrm{r}(H_2|H_1) + \mathbb{P}\mathrm{r}(H_1|H_2)] \\ &= \frac{1}{2}\int \min(\mathbb{P}\mathrm{r}(\mathbf{x}|H_1), \mathbb{P}\mathrm{r}(\mathbf{x}|H_2))d\mathbf{x} \end{aligned} \tag{2.3}$$

Since this integral does not admit a closed form solution, we study the Bhattacharyya upper bound [16] to P_e instead. This bound is a special case of the Chernoff bound [28] derived using the observation $\min(a, b) \leq \sqrt{ab}$. The Bhattacharyya bound gives

$$P_e \leq \frac{1}{2}e^{-K}, \text{ where } K = \frac{1}{2}\ln\frac{\det\left(\frac{\mathbf{\Sigma}_1+\mathbf{\Sigma}_2}{2}\right)}{\sqrt{\det\mathbf{\Sigma}_1 \cdot \det\mathbf{\Sigma}_2}}. \tag{2.4}$$

The numerator inside the logarithm measures the volume of space occupied by both subspaces together, and the denominator measures the volumes occupied separately. These quantities depend on the principal angles, and we now study the performance of the Bhattacharyya bound in the high, low and moderate SNR regimes.

2.3.1 The High SNR Regime

We first consider the case when $\sigma^2 \to 0$, which means that the mismatch between the signal and the model becomes vanishingly small. Since the intersection $\mathcal{X}_1 \cap \mathcal{X}_2$ between the two subspaces plays a special role, we write the two covariances as

$$\begin{aligned} \mathbf{\Sigma}_1 &= \mathbf{U}_{1,\cap}\mathbf{\Lambda}_{1,\cap}\mathbf{U}_{1,\cap}^\top + \mathbf{U}_{1,\setminus}\mathbf{\Lambda}_{1,\setminus}\mathbf{U}_{1,\setminus}^\top + \sigma^2\mathbf{I}, \\ \mathbf{\Sigma}_2 &= \mathbf{U}_{2,\cap}\mathbf{\Lambda}_{2,\cap}\mathbf{U}_{2,\cap}^\top + \mathbf{U}_{2,\setminus}\mathbf{\Lambda}_{2,\setminus}\mathbf{U}_{2,\setminus}^\top + \sigma^2\mathbf{I} \end{aligned} \tag{2.5}$$

Here both $\mathbf{U}_{1,\cap} \in \mathbb{R}^{n\times r}$ and $\mathbf{U}_{2,\cap} \in \mathbb{R}^{n\times r}$ span $\mathcal{X}_1 \cap \mathcal{X}_2$ with singular values $\mathbf{\Lambda}_{1,\cap}$ and $\mathbf{\Lambda}_{2,\cap}$ respectively. $\mathbf{U}_{1,\setminus} \in \mathbb{R}^{n\times(d-r)}$ spans $\mathcal{X}_1\setminus\mathcal{X}_2$ with singular values $\mathbf{\Lambda}_{1,\setminus}$. And $\mathbf{U}_{2,\setminus} \in \mathbb{R}^{n\times(d-r)}$ spans $\mathcal{X}_2\setminus\mathcal{X}_1$ with singular values $\mathbf{\Lambda}_{2,\setminus}$.

The following theorem bounds the classification error in the high SNR regime.

Theorem 1. *Assume $n \geq 2(d-r)$. As $\sigma^2 \to 0$, the classification error is upper bounded as*

$$P_e \leq c_1 (\sigma^2)^{\frac{d-r}{2}} \left(\prod_{i=r+1}^{d} \sin^2 \theta_i \right)^{-\frac{1}{2}} + o\left((\sigma^2)^{\frac{d-r}{2}} \right)$$

where "$g(\sigma^2) = o(f(\sigma^2))$" stands for $\lim_{\sigma^2 \to 0} \frac{g(\sigma^2)}{f(\sigma^2)} = 0$. The constant c_1 is given by,

$$c_1 = 2^{\frac{2d-r}{2}-1} \left[\frac{\mathrm{pdet}(\mathbf{U}_{1,\cap} \mathbf{\Lambda}_{1,\cap} \mathbf{U}_{1,\cap}^\top + \mathbf{U}_{2,\cap} \mathbf{\Lambda}_{2,\cap} \mathbf{U}_{2,\cap}^\top)}{\sqrt{\prod_{i=1}^{r} \lambda_{1,\cap,i} \cdot \prod_{i=1}^{r} \lambda_{2,\cap,i}}} \cdot \prod_{i=1}^{d-r} \sqrt{\lambda_{1,\setminus,i} \cdot \lambda_{2,\setminus,i}} \right]^{-\frac{1}{2}},$$

where pdet *denotes the pseudo-determinant.*

Proof. The method is to expand the Bhattacharyya bound in terms of principal angles, and the details are provided in Appendix A.1. □

Remark 1.

1. *Typically $n \gg d$ for measured data, so the condition $n \geq 2(d-r)$ is usually satisfied.*
2. *The classification error is upper bounded by $(\sigma^2)^{\frac{d-r}{2}}$; the smaller the overlap between subspaces, the easier it is to discriminate between classes. When two subspaces overlap completely, there is an error floor.*

There is a duality between the GMM classification problem and multiple antenna communication [78]. In multiple antenna communications, a codeword is a $d \times n$ array, where the rows are indexed by transmit antennas, the columns are indexed by time slots in a data frame, and the entries are the symbols to be transmitted. The probability of mistaking codeword C_i for codeword C_j, $\mathbb{P}\mathrm{r}(i \to j)$, satisfies

$$\mathbb{P}\mathrm{r}(i \to j) \leq (\sigma^2/2)^k (1/\lambda_1^2 \ldots \lambda_k^2),$$

where k is the rank of $C_i - C_j$, whose singular values are $\lambda_1, \ldots, \lambda_k$. The primary objective in code design for multiple antenna wireless communication is to maximize

the minimum rank of the difference between distinct codewords. If the minimum rank is k, the code is said to achieve a diversity gain of k.

An important secondary objective in code design for multiple antenna wireless communication is to maximize the minimum product of the singular values of the difference between distinct codewords. This minimum product determines the coding gain.

The counterpart of coding gain in classification is the product of sines of the principal angles. This quantity determines the intercept of the error exponent with the vertical axis. The smaller the energy in the intersection of the subspaces, the smaller is the classification error. The larger the principal angles, the smaller is the classification error.

2.3.2 The Low SNR Regime

This is the case where the noise variance σ^2 and the singular values are commensurable; in other words, the mismatch between the signal and the empirical model cannot be neglected. The MAP classifier in this case is characterized by the following theorem.

Theorem 2. *When σ^2 is sufficiently large, the Bhattacharyya upper bound is sandwiched between*

$$\underline{P_e}^{UB} = \frac{1}{2}\exp\left\{-\frac{1}{\sigma^4}\left(c_2 - \frac{1}{16}\lambda_{1,1}\lambda_{2,1}\sum_{i=1}^{d}\cos^2\theta_i\right)\right\}$$

and

$$\overline{P_e}^{UB} = \frac{1}{2}\exp\left\{-\frac{1}{\sigma^4}\left(c_3 - \frac{1}{8}\lambda_{1,1}\lambda_{2,1}\sum_{i=1}^{d}\cos^2\theta_i\right)\right\},$$

where $\overline{P_e}^{UB} > \underline{P_e}^{UB}$. *And the constants* c_2 *and* c_3 *are given by*

$$c_2 = \frac{\sigma^4}{4}\left[\sum_{i=1}^{d}\frac{\lambda_{1,i}}{\sigma^2} - \frac{1}{2}\sum_{i=1}^{d}\left(\frac{\lambda_{1,i}}{2\sigma^2}\right)^2 - \sum_{i=1}^{d}\ln\left(1+\frac{\lambda_{1,i}}{\sigma^2}\right)\right]$$
$$+ \frac{\sigma^4}{4}\left[\sum_{i=1}^{d}\frac{\lambda_{2,i}}{\sigma^2} - \frac{1}{2}\sum_{i=1}^{d}\left(\frac{\lambda_{2,i}}{2\sigma^2}\right)^2 - \sum_{i=1}^{d}\ln\left(1+\frac{\lambda_{2,i}}{\sigma^2}\right)\right]$$

$$c_3 = \frac{\sigma^4}{4}\left[\sum_{i=1}^{d}\frac{\lambda_{1,i}}{\sigma^2} - \sum_{i=1}^{d}\left(\frac{\lambda_{1,i}}{2\sigma^2}\right)^2 - \sum_{i=1}^{d}\ln\left(1+\frac{\lambda_{1,i}}{\sigma^2}\right)\right]$$
$$+ \frac{\sigma^4}{4}\left[\sum_{i=1}^{d}\frac{\lambda_{2,i}}{\sigma^2} - \sum_{i=1}^{d}\left(\frac{\lambda_{2,i}}{2\sigma^2}\right)^2 - \sum_{i=1}^{d}\ln\left(1+\frac{\lambda_{2,i}}{\sigma^2}\right)\right].$$

Proof. The details are given in appendix A.2. □

Remark 2. *The dimension of the overlap between the two subspaces plays a less important role in the low SNR regime, and classification error is a function of chordal distance. This gives rise to an interesting duality between GMM model based classification and the space-time decoding [4], where error probability is influenced by product or sum diversity in high or low SNR regime respectively.*

2.3.3 The Moderate SNR Regime

We now consider a moderate noise/mismatch regime, where $\frac{p}{c(p)} \leq \frac{\lambda_{1,j}}{\sigma^2}, \frac{\lambda_{2,j}}{\sigma^2} \leq p$ for $j = 1, \ldots, d$ and $p > 1, c(p) > 1$. Moderate SNR also implies that p is not very large.

The most important element in the analysis of classification error is to lower bound the term $\ln\det\left(\frac{\mathbf{\Sigma}_1+\mathbf{\Sigma}_2}{2}\right)$ in Eq. (2.4),

$$\ln\det\left(\frac{\mathbf{\Sigma}_1+\mathbf{\Sigma}_2}{2}\right) = \ln\det\left(\mathbf{I} + \frac{\mathbf{U}_1\mathbf{\Lambda}_1\mathbf{U}_1^\top + \mathbf{U}_2\mathbf{\Lambda}_2\mathbf{U}_2^\top}{2\sigma^2}\right) + n\ln\sigma^2.$$

Denote the non-zero singular values of $\mathbf{D} \triangleq \frac{1}{2\sigma^2}(\mathbf{U}_1\mathbf{\Lambda}_1\mathbf{U}_1^\top + \mathbf{U}_2\mathbf{\Lambda}_2\mathbf{U}_2^\top)$ by $\lambda_1, \ldots, \lambda_{2d-r}$.

Then

$$\ln\det\left(\frac{\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2}{2}\right)=\sum_{i=1}^{2d-r}\ln(1+\lambda_i)+n\ln(\sigma^2). \tag{2.6}$$

The following lemma provides a lower bound on $\ln(1+\lambda_i)$.

Lemma 3. *There exists* $0\le L<\frac{p-1}{2}$ *such that for any* $\lambda_i\in[L,p]$,

$$\ln(1+\lambda_i)\ge\ln(1+p)+\frac{1}{1+p}(\lambda_i-p)-\frac{1}{(1+p)^2}(\lambda_i-p)^2. \tag{2.7}$$

Proof. See Appendix A.3. □

Let $L(p)$ be the smallest possible value of L, define $c(p)=\frac{p}{2L(p)}$ if $L(p)>0$ and $c(p)=+\infty$ if $L(p)=0$. Note that $c(p)>1$ since $L(p)<\frac{p-1}{2}$.

Theorem 4. *If* $\frac{p}{c(p)}\le\frac{\lambda_{1,i}}{\sigma^2},\frac{\lambda_{2,i}}{\sigma^2}\le p$, *then the classification error is upper bounded as*

$$P_e\le\frac{1}{2}\exp\left\{-c_4(2d-r)+\frac{\lambda_{1,1}\lambda_{2,1}}{4\sigma^4(1+p)^2}\sum_i\cos^2\theta_i+c_5\right\},$$

where $c_4=\frac{1}{2}\left[\ln(1+p)-\frac{p}{1+p}-\frac{p^2}{(1+p)^2}\right]$ *and* c_5 *depends on* p *and* $\frac{\lambda_{1,i}}{\sigma^2}$, $\frac{\lambda_{2,i}}{\sigma^2}$.

Proof. See Appendix A.3. □

Remark 3. *It is straightforward to show numerically that* $c(p)=3.44,\ 2.79$ *for* $p=4,5$ *respectively, that* $c(p)\ge 2.02$ *for* $p\le 10$, *and that* $c(p)\ge 1.61$ *for* $p\le 100$. *The form of the upper bound suggests that in the moderate SNR regime, the role of chordal distance is more important than the product of the sines of the principal angles.*

2.3.4 Numerical Analysis of Synthetic Data

We explore the difference between classification in the low and high SNR regimes through a simple numerical example. Consider the following pairs of subspaces:
case 1:

$$\mathbf{U}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}^\top \quad \mathbf{U}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}^\top.$$

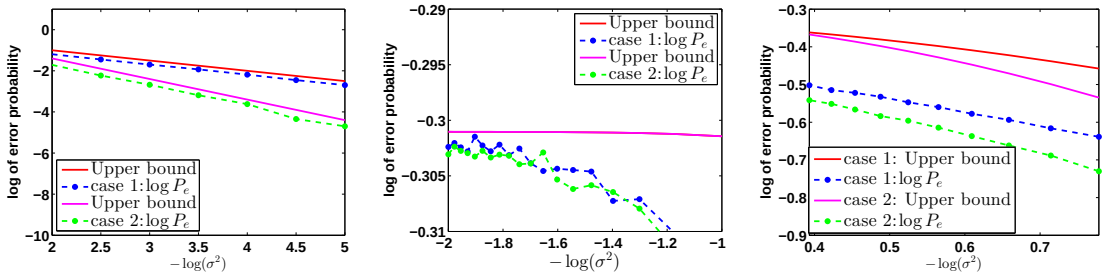
case 2:

$$\mathbf{U}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}^\top \quad \mathbf{U}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 1 & 0 \end{bmatrix}^\top.$$

We set $\mathbf{\Lambda}_1 = \mathbf{\Lambda}_2 = \mathbf{I}$ for both cases. In case 1, the two principal angles are $\theta_1 = 0, \theta_2 = \pi/2$ and in case 2, the two principal angles are $\theta_1 = \pi/4, \theta_2 = \pi/4$. The chordal distances in these two cases are the same, but in case 1 the product of sines of non-zero principal angles is 1, whereas in case 2 it is 1/2. However, there is a nontrivial intersection dimension in case 1. The product of nonzero sine principal angles is 1 for case 1, and $\frac{1}{2}$ for case 2.

We vary the degree of mismatch σ^2, and evaluate the bounds developed in the above three theorems. In the high SNR regime, we plot the empirical misclassification probability P_e with the value $c_1(\sigma^2)^{\frac{d-r}{2}} \left(\prod_{i=r+1}^{d} \sin^2\theta_i\right)^{-\frac{1}{2}}$ given in Theorem 1. In the low SNR regime, we plot the upper bound $\overline{P_e}^{UB}$ in Theorem 2. In the moderate SNR regime, we take $p = 6$, and we vary σ^2 between $\frac{1}{p}$ and $\frac{c(p)}{p}$, so that $\frac{p}{c(p)} \leq \frac{\lambda_{1,i}}{\sigma^2}, \frac{\lambda_{2,i}}{\sigma^2} \leq p$. We then plot the upper bound in Theorem 4, against the empirical classification error. In the high SNR regime (Fig. 2.1a), the classification error decays faster in Case 2 than in Case 1, consistent with Theorem 1. In the low SNR regime (Fig. 2.1b), there is little difference in classification error between the two cases, consistent with Theorem 2. In the moderate SNR regime (Fig. 2.1b), classification performance in case 1 is inferior to that in case2, because there is a shared 1-dimensional subspace,

and this is predicted by Theorem 4.



(a) high SNR regime (b) low SNR regime (c) moderate SNR regime

FIGURE 2.1: Error probability as a function of the degree of mismatch. Dashed lines represent empirical estimates, and solid lines represent upper bounds. In the low SNR regime the two upper bounds coincide.

Concluding this section, we have characterized the pair-wise classification error using the principal angles between a pair of subspaces. The union bound then makes it possible to derive an upper bound on classification error for multiple classes.

2.4 Nearest Subspace Classifier: extending GMM

If the class distribution is known (for example through its covariance) then the MAP classifier is optimal. If however we only know that each class is near a known low-dimensional subspace (possibly inferred from less training data) then we can substitute a Nearest Subspace Classifier (NSC) for the MAP. This Section connects performance of the NSC with principal angles, and for simplicity we focus on discriminating pairs of classes, given that the extension to multiple classes is straightforward.

Consider two classes, labeled C_1 and C_2, distributed near two subspaces with orthonormal bases $\mathbf{U}_1, \mathbf{U}_2 \in \mathbb{R}^{n \times d}$. The NSC determines the class label of a test sample $\mathbf{x}$, $\hat{\mathcal{C}}$, by comparing the norms of the projections onto $\mathbf{U}_1$ and $\mathbf{U}_2$.

$$\hat{\mathcal{C}} = \begin{cases} \mathcal{C}_1 & \|\mathbf{U}_1^\top \mathbf{x}\|^2 \geq \|\mathbf{U}_2^\top \mathbf{x}\|^2 \\ \mathcal{C}_2 & \text{otherwise} \end{cases} . \tag{2.8}$$

The preferred class label has a basis that is better aligned to the signal.

2.4.1 Derivation of the Upper Bound

Starting from the projection onto each subspace, we model the distribution of these two classes as

$$\begin{aligned} p(\mathbf{x}|\mathcal{C}_1) &= \int p(\mathbf{x}|\boldsymbol{\alpha}, \mathcal{C}_1) p(\boldsymbol{\alpha}) d\alpha = \int \mathcal{N}(\mathbf{x}; \mathbf{U}_1\boldsymbol{\alpha}, \sigma^2\mathbf{I}) p(\boldsymbol{\alpha}) d\boldsymbol{\alpha} \\ p(\mathbf{x}|\mathcal{C}_2) &= \int p(\mathbf{x}|\boldsymbol{\alpha}, \mathcal{C}_2) q(\boldsymbol{\alpha}) d\alpha = \int \mathcal{N}(\mathbf{x}; \mathbf{U}_2\boldsymbol{\alpha}, \sigma^2\mathbf{I}) q(\boldsymbol{\alpha}) d\boldsymbol{\alpha}. \end{aligned} \tag{2.9}$$

The NSC knows $\mathbf{U}_1$ and $\mathbf{U}_2$, but is blind to $p(\boldsymbol{\alpha})$ and $q(\boldsymbol{\alpha})$, where $\boldsymbol{\alpha}$ is the expansion of the projection $\mathbf{U}_i^\top \mathbf{x}$ in the basis $\mathbf{U}_i$. Note that since we are not assuming a GMM, the vector $\boldsymbol{\alpha}$ need not be multivariate normal.

Let $\mathbf{V}\,\mathrm{diag}\{\cos\theta_1, \ldots, \cos\theta_d\}\mathbf{W}^\top$ be the singular value decomposition of $\mathbf{U}_1^\top\mathbf{U}_2$, where $\mathbf{V}$, $\mathbf{W}$ are unitary, and the principal angles $\{\theta_1, \ldots, \theta_d\}$ are taken in ascending order. We may, absorb $\mathbf{V}$, $\mathbf{W}$ into $\mathbf{U}_1$, $\mathbf{U}_2$ at the cost of redefining $p(\boldsymbol{\alpha})$, $q(\boldsymbol{\alpha})$. Thus we may without loss of generality assume $\mathbf{V} = \mathbf{W} = \mathbf{I}$, *i.e.*,

$$\mathbf{U}_1^\top\mathbf{U}_2 = \mathrm{diag}\{\cos\theta_1, \ldots, \cos\theta_d\} \triangleq \mathbf{C}. \tag{2.10}$$

Define $\mathbb{P}\mathrm{r}(\mathcal{C}_2|\mathcal{C}_1)$ as the probability of mistaking $\mathcal{C}_2$ for $\mathcal{C}_1$ and define $\mathbb{P}\mathrm{r}(\mathcal{C}_1|\mathcal{C}_2)$ similarly. Then the classification error is

$$P_e = \frac{1}{2}\mathbb{P}\mathrm{r}(\mathcal{C}_2|\mathcal{C}_1) + \frac{1}{2}\mathbb{P}\mathrm{r}(\mathcal{C}_1|\mathcal{C}_2). \tag{2.11}$$

We bound $\mathbb{P}\mathrm{r}(\mathcal{C}_2|\mathcal{C}_1)$ using principal angles, and $\mathbb{P}\mathrm{r}(\mathcal{C}_1|\mathcal{C}_2)$ can be analyzed in the same manner. We expand $\mathbb{P}\mathrm{r}(\mathcal{C}_2|\mathcal{C}_1)$ using Bayes rule as

$$\mathbb{P}\mathrm{r}(\mathcal{C}_2|\mathcal{C}_1) = \int \mathbb{P}\mathrm{r}(\mathcal{C}_2|\mathcal{C}_1, \boldsymbol{\alpha}) p(\boldsymbol{\alpha}) d\boldsymbol{\alpha}. \tag{2.12}$$

We bound $\mathbb{P}\mathrm{r}(\mathcal{C}_2|\mathcal{C}_1, \boldsymbol{\alpha})$ by writing $\mathbf{x} = \mathbf{U}_1\boldsymbol{\alpha} + \mathbf{n}$, where the noise $\mathbf{n} \sim \mathcal{N}(0, \sigma^2\mathbf{I})$.

$$\begin{aligned} \mathbb{P}\mathrm{r}(\mathcal{C}_2|\mathcal{C}_1, \boldsymbol{\alpha}) =& \mathbb{P}\mathrm{r}(\|\mathbf{U}_1^\top(\mathbf{U}_1\boldsymbol{\alpha} + \mathbf{n})\|^2 \leq \|\mathbf{U}_2^\top(\mathbf{U}_1\boldsymbol{\alpha} + \mathbf{n})\|^2) \\ =& \mathbb{P}\mathrm{r}(\|\boldsymbol{\alpha} + \mathbf{U}_1^\top\mathbf{n}\|^2 \leq \|C\boldsymbol{\alpha} + \mathbf{U}_2^\top\mathbf{n}\|^2), \end{aligned} \tag{2.13}$$

where the probability is taken w.r.t. $\mathbf{n}$. Denote the i-th column in $\mathbf{U}_1(\mathbf{U}_2)$ as $\mathbf{u}_{1,i}(\mathbf{u}_{2,i})$, and the i-th element of $\boldsymbol{\alpha}$ as α_i. It follows from Eq. (2.13) that

$$\mathbb{P}\mathrm{r}(\|\boldsymbol{\alpha}+\mathbf{U}_1^\top\mathbf{n}\|^2 \leq \|\mathbf{C}\boldsymbol{\alpha}+\mathbf{U}_2^\top\mathbf{n}\|^2) = \mathbb{P}\mathrm{r}\left(\sum_i(\alpha_i+\mathbf{u}_{1,i}^\top\mathbf{n})^2 \leq \sum_i(\cos\theta_i\alpha_i+\mathbf{u}_{2,i}^\top\mathbf{n})^2\right). \tag{2.14}$$

We now define $a_i \triangleq \alpha_i + \mathbf{u}_{1,i}^\top\mathbf{n}$ and $b_i \triangleq \cos\theta_i\alpha_i + \mathbf{u}_{2,i}^\top\mathbf{n}$. Then Eq. (2.14) simplifies to

$$\mathbb{P}\mathrm{r}\left(\sum_i(\alpha_i+\mathbf{u}_{1,i}^\top\mathbf{n})^2 \leq \sum_i(\cos\theta_i\alpha_i+\mathbf{u}_{2,i}^\top\mathbf{n})^2\right) = \mathbb{P}\mathrm{r}\left(\sum_i(a_i+b_i)(a_i-b_i) \leq 0\right). \tag{2.15}$$

Lemma 5. *Let a_i, b_i as defined as above. For any pair of i, j where $i \neq j$:*

1. *a_i is independent from a_j*
2. *b_i is independent from b_j*
3. *a_i is independent from b_j*
4. *$a_i + b_i$ is independent from $a_i - b_i$*

Proof. The proof is given in appendix A.4. □

It follows from Lemma 5 that $\sum_i(a_i+b_i)(a_i-b_i)$ is the sum of products of independently distributed normal random variables. However the product of independently distributed normal random variables need not be normal, and so we need to show that $(a_i+b_i)(a_i-b_i)$ is normally distributed.

Lemma 6 (product of normal random variable[3])**.** *Let $x \sim \mathcal{N}(\mu_x, \sigma_x^2)$ and $y \sim \mathcal{N}(\mu_y, \sigma_y^2)$ be two independent normal variables. If $\mu_x/\sigma_x \to \infty$ and $\mu_y/\sigma_y \to \infty$ in any manner, then the distribution of xy approaches normality with mean $\mu_x\mu_y$ and variance $\mu_x^2\sigma_y^2 + \mu_y^2\sigma_x^2 + \sigma_x^2\sigma_y^2$.*

Applying Lemma 6 and combining the independence stated in Lemma 5, we have

Lemma 7. *As* $\sigma \to 0$, $\sum_i (a_i+b_i)(a_i-b_i) \sim \mathcal{N}\left(\sum_i \sin^2\theta_i \alpha_i^2, 4\sigma^2 \sum_i \sin^2\theta_i(\alpha_i^2+\sigma^2)\right)$

Proof. The proof is given in appendix A.4. □

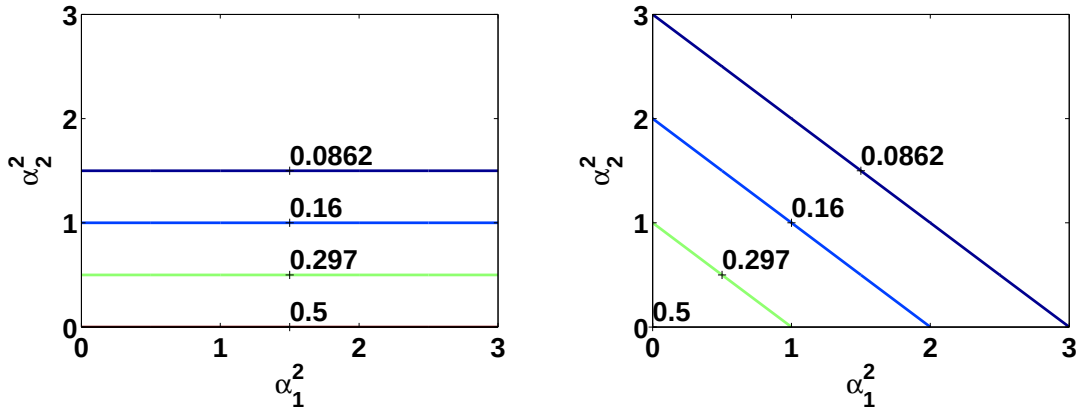
It follows that $\Pr\left(\sum_i (a_i+b_i)(a_i-b_i) \le 0\right)$ is the tail probability of a normal distribution. Applying the standard tail bound, we arrive at the following theorem.

Theorem 8. *As* $\sigma^2 \to 0$, *the classification error is upper bounded as*

$$P_e \le \int \mathcal{E}(\boldsymbol{\theta}, \boldsymbol{\alpha}, \sigma^2) \frac{p(\boldsymbol{\alpha}) + q(\boldsymbol{\alpha})}{2} d\alpha$$

where $\mathcal{E}(\boldsymbol{\theta}, \boldsymbol{\alpha}, \sigma^2) = \frac{1}{2} \exp\left[-\frac{\left(\sum_{i=1}^d \sin^2\theta_i \alpha_i^2\right)^2}{8\sigma^2 \sum_{i=1}^d \sin^2\theta_i(\alpha_i^2+\sigma^2)}\right]$.

Proof. The proof is given in appendix A.4. □



(a) case 1 (b) case 2

FIGURE 2.2: Lines on which $\mathcal{E}$ is constant for the two case studies introduced in section 2.3.4.

We return to the two case studies introduced in Section 2.3.4 to provide some intuition about the kernel $\mathcal{E}$. The principal angles are $[0, \pi/2]$ in Case 1, and $[\pi/4, \pi/4]$ in Case 2. In Case 1, the kernel is constant on horizontal lines, and in Case 2, it is

constant on lines of slope -1. These two cases are shown in Fig. 2.2, and we now make a number of general observations.

Remark 4. *1.* $\mathcal{E}(\boldsymbol{\theta}, \boldsymbol{\alpha}, \sigma^2)$ *is monotonically decreasing w.r.t.* $\sum_i \sin^2\theta_i\alpha_i^2$, *and monotonically increasing w.r.t.* σ^2. *Therefore, bigger principal angles or signal energy results in smaller classification error. Bigger noise results in bigger classification error. 2. Ignoring the higher order term of* σ^2 *in the denominator inside the* $\exp(\cdot)$, *we have*

$$\mathcal{E}(\boldsymbol{\theta}, \boldsymbol{\alpha}, \sigma^2) \approx \frac{1}{2}\exp\left(-\frac{\sum_i \sin^2\theta_i\alpha_i^2}{8\sigma^2}\right)$$

which clearly indicates that classification performance is a function of discernibility (the sine principal angles) weighted by signal energy (the α_i^2*'s). 3. For fixed energy, classification error is decreased by allocating larger* α_i^2 *to larger* θ_i.

2.4.2 Numerical Analysis of Synthetic Data

We now examine the agreement between empirical error and the upper bound given in Theorem 3. Set $n = 6$, $d = 2$,

$$\mathbf{U}_1 = \left[\mathbf{I}_2, \mathbf{0}_4\right]^\top,\ \mathbf{U}_2 = \begin{bmatrix} \cos\theta & 0 & 0 & 0 & \sin\theta & 0 \\ 0 & \cos\theta & 0 & 0 & 0 & \sin\theta \end{bmatrix}^\top,$$

so that the two principal angles between $\mathbf{U}_1$ and $\mathbf{U}_2$ are $\theta_1 = \theta_2 = \theta$. Set $p(\boldsymbol{\alpha}) = q(\boldsymbol{\alpha}) = \mathcal{N}(\boldsymbol{\alpha}; 0, \mathbf{I}_2)$, and vary σ^2 in $[0.01, 0.5]$. Fig. 2.3a considers three values of θ ($\pi/6$, $\pi/4$, and $\pi/3$), and shows that empirical NSC classification error tracks the upper bound obtained by numerical integration.

Next we examine the dependence of classification error on distribution of signal energy across the two modes. Set $n = 6$, $d = 2$, $\mathbf{U}_1 = \left[\mathbf{I}_2, \mathbf{0}_4\right]^\top$ and

$$\mathbf{U}_2 = \begin{bmatrix} \cos(\pi/6) & 0 & 0 & 0 & \sin(\pi/6) & 0 \\ 0 & \sin(\pi/6) & 0 & 0 & 0 & \cos(\pi/6) \end{bmatrix}^\top,$$

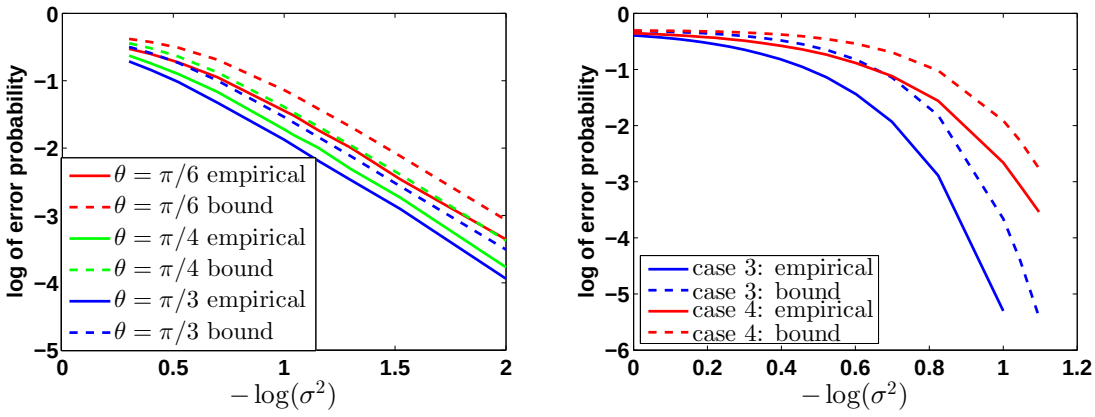


FIGURE 2.3: Comparison of empirical NSC classification error with the upper bound obtained by numerical integration. (a) Larger principal angles reduce classification error; (b) Disproportionate assignment of signal energy to larger principal angles reduces classification error.

so that the two principal angles are $\theta_1 = \pi/6$ and $\theta_2 = \pi/3$. Fix $\|\boldsymbol{\alpha}\|^2 = 1$, and compare the case when $\boldsymbol{\alpha}$ is distributed such that $|\alpha_1| < |\alpha_2|$ (Case 3 in Fig. 2.3b), with the case when $\boldsymbol{\alpha}$ is distributed such that $|\alpha_1| > |\alpha_2|$ (Case 4 in Fig. 2.3b). Empirical error is calculated for a range of noise variances, by randomly drawing 10,000 sample per class. Empirical NSC classification error tracks the upper bound given by numerical integration, with performance of Case 3 superior to that of Case 4.

2.5 TRAIT: Tunable Recognition Adapted to Intra-class Target

In the previous theorems, it is the principal angles that determine the performance of the classifiers in different SNR regimes. This suggests that we might improve classification by applying a linear transformation that optimizes principal angles, even at the cost of reducing dimensionality.

We denote the collection of all labeled training samples as $\mathbf{X} = [\mathbf{X}_1, \ldots, \mathbf{X}_K] \in \mathbb{R}^{n \times N}$, where columns in the submatrix $\mathbf{X}_k \in \mathbb{R}^{n \times N_k}$ are samples from the k-th

class. The signal subspace of $\mathbf{X}_k$ is spanned by the orthonormal basis $\mathbf{U}_k$ defined above. The linear transform $\mathbf{A} \in \mathbb{R}^{m \times n}$ $(m \leq n)$ is designed to maximize separation of the subspaces $\mathbf{AU}_1, \ldots, \mathbf{AU}_K$. The maximal separation is achieved when $(\mathbf{AU}_j)^\top (\mathbf{AU}_k) = 0$ for all $j \neq k$. In this case, all the principal angles are $\pi/2$. One approach is to use the SVD to compute the $\mathbf{U}_k$ and then to learn the linear transformation $\mathbf{A}$. However we may avoid pre-computing the $\mathbf{U}_k$ by simply encouraging $(\mathbf{AX}_j)^\top (\mathbf{AX}_k) = 0$ for all $j \neq k$.

We shall require that the transform $\mathbf{A}$ preserve some specific characteristic or trait of each individual class. For example, we may target $(\mathbf{AX}_k)^\top (\mathbf{AX}_k) = \mathbf{X}_k^\top \mathbf{X}_k$ for all k, so that the original intra-class data structure (with noise) is preserved. Given access to a denoised signal, $\hat{\mathbf{X}}_k$, we might instead target $(\mathbf{AX}_k)^\top (\mathbf{AX}_k) = \hat{\mathbf{X}}_k^\top \hat{\mathbf{X}}_k$ again for all k. In this case, the intra-class dispersion due to noise is suppressed. Thus, the Gram matrix $\mathbf{T}$ of the transformed signal can be designed to target preservation of particular intra-class structure. We formulate the optimization problem as

$$\min_{\mathbf{A} \in \mathbb{R}^{m \times n}} \frac{1}{N^2} \|(\mathbf{AX})^\top (\mathbf{AX}) - \mathbf{T}\|_F^2. \tag{2.16}$$

The block diagonal structure of the target Gram matrix $\mathbf{T}$ promotes larger principal angles between subspaces. At the same time the diagonal blocks can be tuned to different characteristics of individual classes. For example, when side information is available, we may consider incorporating it in diagonal blocks. Here we only consider

$$\mathbf{T} = \operatorname{diag}\{\mathbf{X}_1^\top \mathbf{X}_1, \ldots, \mathbf{X}_K^\top \mathbf{X}_K\}, \tag{2.17}$$

as a proof-of-concept. We refer to this approach as the TRAIT algorithm, where the acronym denotes **T**unable **R**ecognition **A**dapted to **I**ntra-class **T**argets.

It is possible to minimize the objective in E.q. (2.16) by first minimizing $\|\mathbf{X}^\top \mathbf{P} \mathbf{X} - \mathbf{T}\|^2$ for $\mathbf{P} \succeq 0$ (as Proposition 9), and then factoring $\mathbf{P}$ as $\mathbf{P} = \mathbf{A}^\top \mathbf{A}$ where $\mathbf{A} \in \mathbb{R}^{m \times n}$.

Proposition 9. *The minimizer of* $\|\mathbf{X}^\top\mathbf{P}\mathbf{X} - \mathbf{T}\|_F^2$ *where* $\mathbf{P} \succeq 0$*, is*

$$\mathbf{P}^\star = (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{T}\mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top)^{-1}.$$

Proof. Proof is detailed in appendix A.5. □

However when $m < n$, such a rank-m decomposition may not exist since this $\mathbf{P}$ is not guaranteed to be rank deficient. An alternative is to learn a rank deficient $\mathbf{P}$ by solving

$$\min_{P \succeq \mathbf{0}} \|\mathbf{X}^\top\mathbf{P}\mathbf{X} - \mathbf{T}\|_F^2 + \lambda\|\mathbf{P}\|_*,$$

where the nuclear norm $\|\mathbf{P}\|_*$ regularizes the rank of $\mathbf{P}$. However this approach requires careful tuning of λ, and it is computationally more complex since we work with a matrix $\mathbf{P}$ larger than $\mathbf{A}$. Given these considerations, we choose to solve (2.16) using gradient descent as described in Algorithm 1.

Algorithm 1 TRAIT for feature extraction

Input: labeled training samples $\mathbf{X} = [\mathbf{X}_1, \ldots, \mathbf{X}_K]$, target dimension m, $(m \leq n)$, target Gram matrix $\mathbf{T}$.
Output: feature extraction matrix (transform) $\mathbf{A} \in \mathbb{R}^{m \times n}$.

1: Initialize $\mathbf{A} = [\mathbf{e}_1, \ldots, \mathbf{e}_m]^\top$, where $\mathbf{e}_i$ is the i-th standard basis.
2: **while** stopping criteria not met **do**
3: Compute gradient

$$\mathbf{G} = \mathbf{A}(\mathbf{X}\mathbf{X}^\top\mathbf{A}^\top\mathbf{A}\mathbf{X}\mathbf{X}^\top - \mathbf{X}\mathbf{T}\mathbf{X}^\top).$$

4: Choose a positive step-size η and take a gradient step

$$\mathbf{A} \leftarrow \mathbf{A} - \eta\mathbf{G}.$$

5: **end while**

2.5.1 Related Methods

Linear Discriminant Analysis (LDA) is a classical feature extraction method which assumes each class to be Gaussian distributed. It achieves better performance on face recognition tasks than does PCA [11]. LDA does not assume near low-rank

structure of the covariances, and therefore considers a different data geometry than the one here studied.

Methods of feature extraction based on random projection have recently been developed and successfully applied to face recognition [87]. Random projection is designed to preserve pairwise distances between all data points uniformly across class labels [48].

More recently, the Low-Rank Transform (LRT) has been proposed as a method of extracting features [63]. It enlarges inter-class distance while suppressing intra-class dispersion. LRT uses the nuclear norm, $\|\mathbf{A}\mathbf{X}_i\|_*$, to measure the dispersion of the (transformed) data. The transform $\mathbf{A}$ is

$$\underset{\mathbf{A}\in\mathbb{R}^{m\times n}:\|\mathbf{A}\|_2\leq c}{\arg\min} \sum_{i=1}^{K} \|\mathbf{A}\mathbf{X}_i\|_* - \|\mathbf{A}\mathbf{X}\|_*.$$

What motivates the choice of the nuclear norm is that it is the convex relaxation of rank [63]. In the high SNR regime, Theorem 1 suggests that classification error decreases when the union of subspaces has large rank. LRT encourages the rank of the union to be large, and it works well in a regime where model mismatch is small. Experiments presented in Section 2.5.3 suggest that TRAIT may be more robust to model mismatch (Fig. 2.8).

2.5.2 Two Properties of the TRAIT Transform

On synthetic and measured data, we show that TRAIT effectively enlarges the angles between different subspaces and preserves intra-class structure. We also compare the classification accuracy of features extracted by TRAIT and the methods in Section 2.5.1. For synthetic data, the class distribution is known exactly, and the MAP classifier is used to measure classification accuracy. For measured data, the class distribution is unknown a priori, and the NSC classifier is employed instead.

Enlargement of the Principal angles

The synthetic dataset has parameters $n = 10$, $d = 1$ and $K = 3$.

$$\mathbf{\Sigma}_k = \mathbf{U}_k \mathbf{U}_k^\top + 10^{-2}\mathbf{I}(k = 1, 2, 3),$$

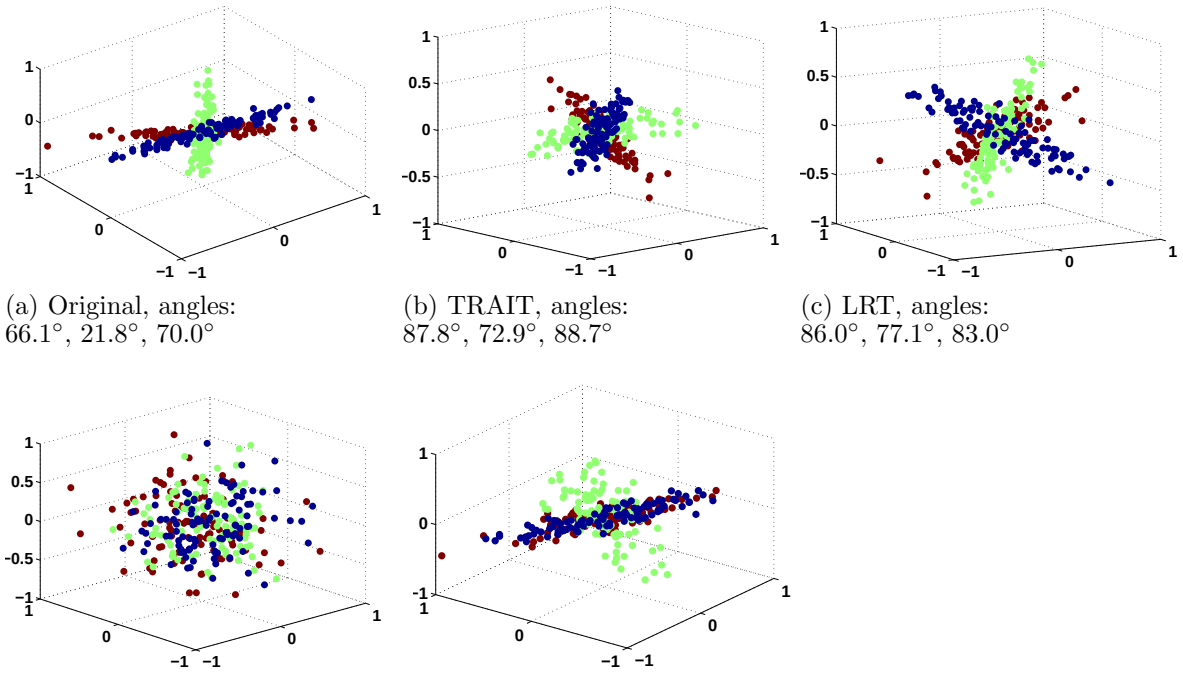
where $\mathbf{U}_k$ is a normalized n-vector with i.i.d. Gaussian random entries. Samples of the k-th class are i.i.d drawn from $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_k)$. For each class, 100 samples are used for learning the transform and 10000 are used for testing. On the training data, we learn the transform respectively via LDA, LRT, and TRAIT with target dimension $m = 3, \ldots, 10$. Then on each test datum, we apply the learned transforms as well as random projection (each entry drawn from $\mathcal{N}(0, 1)$) and classify using a MAP classifier.

We visualize original and transformed data via projection (PCA basis) into 3-dimensional Euclidean space. When the target feature dimension $m = 3$, the results are shown in Fig. 2.4. Each class is represented by a different color. After transforming the data, we use the SVD to calculate the basis vector ($d = 1$) that best describes each class, and we calculate the pairwise angles between basis vectors. The pairwise angles are significantly increased by both LRT and TRAIT. By contrast, neither LDA nor random projection increase separation between one-dimensional subspaces.

We now vary the feature dimension m, and compare the error probability of the MAP classifier across the different methods of extracting features. Fig. 2.5 shows that the performance of TRAIT and LRT are similar, and that both are superior to LDA and random projection. Note that after dimension reduction. TRAIT is still able to match error probabilities achieved with the original data.

Preservation of Intra-class Structure

When a convex body, e.g., human face, is illuminated, the resulting image is effectively represented by spherical harmonics. It has been shown that a 9-dimensional



(a) Original, angles: 66.1°, 21.8°, 70.0°

(b) TRAIT, angles: 87.8°, 72.9°, 88.7°

(c) LRT, angles: 86.0°, 77.1°, 83.0°

(d) LDA, angles: 47.0°, 86.4°, 87.8°

(e) Random, angles: 72.4°, 2.5°, 73.0°

FIGURE 2.4: Embeddings of original and transformed data.

subspace is sufficient to capture the geometry of an individual subject [9]. The extended Yale B face database includes 38 subjects, each with 64 images taken under different illumination conditions. We use a cropped version of this data set[1], where each image is of size $32 \times 32 = 1024$.

For each subject, we randomly select half of the 64 images for training, and retain the other half for testing. For all feature extraction methods, we vary the target dimension m, and apply the NSC to the transformed data. The NSC achieves much higher accuracy on features extracted by TRAIT and LRT (Fig. 2.6).

We also observe in Fig. 2.7 that the features extracted by TRAIT and LRT are quite different, suggesting that information present in one view is somewhat independent of information present in the other. This is confirmed by applying NSC to the concatenation of the two views (TRAIT+LRT in Fig. 2.6), and observing that

[1] http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html

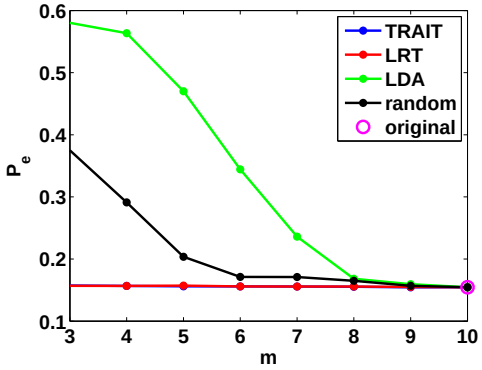


FIGURE 2.5: MAP classifier's P_e on transformed data. Note that TRAIT (blue) and LRT (red) almost overlap.

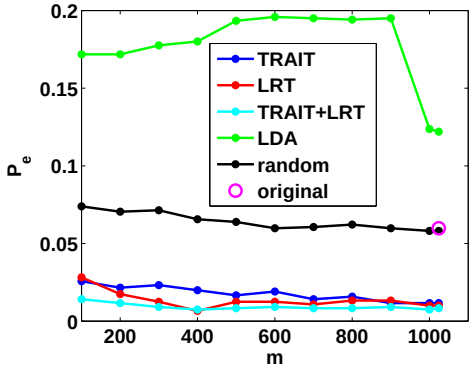


FIGURE 2.6: NSC's P_e on original/transformed face images. Concatenation of TRAIT and LRT features (TRAIT+LRT) provides superior results

classification accuracy is increased.

The intra-class structure preserving property of TRAIT is evident in Fig. 2.7 where we view transformed classes as faces in the original image domain. The original images of subject 10 are displayed together with their TRAIT and LRT transforms. TRAIT preserves a diversity of illumination conditions, whereas LRT blurs the differences between images. Classification performance is improved by using LRT and TRAIT features in combination.

2.5.3 Robustness to Model Mismatch

In the previous sections, we have demonstrated the effectiveness of TRAIT and LRT on both synthetic and real data. In this section, we present experiments showing that TRAIT is more robust with respect to model mismatch than is LRT. In many real world problems, data may not be exactly GMM distributed. Even if they are, there may not be sufficient training data to learn the covariances. Therefore, we use NSC throughout this section to assess the discriminability of the extracted features. Moreover, having seen the effectiveness of dimension reduction in previous sections, we turn to learning dimension reduced features, thereby saving computational cost

FIGURE 2.7: Comparison of original images (top) with TRAIT transformed images (middle) and LRT transformed images (bottom). Red circles indicate structure that is present in both the original and the TRAIT transformed image.

on measured datasets.

Synthetic Data

The synthetic data is a three-class dataset, where datum $\mathbf{x} \in \mathbb{R}^{100}$ in the k-th ($k = 1, 2, 3$) class is generated as

$$\mathbf{x} = \mathbf{U}_k \boldsymbol{\alpha} + \mathbf{n},$$

with $\mathbf{U}_k \in \mathbb{R}^{100 \times 5}$ and $\mathbf{U}_k^\top \mathbf{U}_k = \mathbf{I}$. $\boldsymbol{\alpha} \sim \text{Uniform}[-2, 2]$ and $\mathbf{n} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_{100})$. Note the data is not GMM distributed. Each class has 100 training samples and 10000 testing samples. We vary σ^2 and use NSC to classify TRAIT and LRT extracted features. Here we fix the extracted feature dimension to be 30.

Fig. 2.8 shows the NSC classification accuracy as a function of σ. Both TRAIT and LRT significantly improves classification performance compared with no transform. However, with increasing noise, TRAIT features outperform LRT features, showing greater robustness to model mismatch.

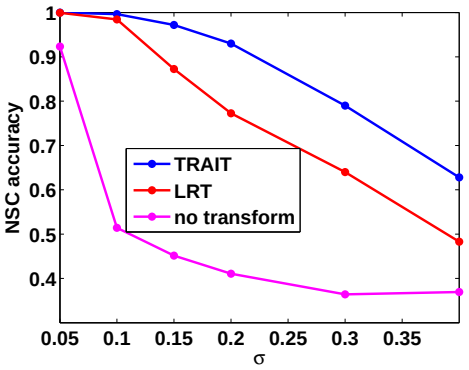


FIGURE 2.8: NSC performance on TRAIT and LRT features under different SNR

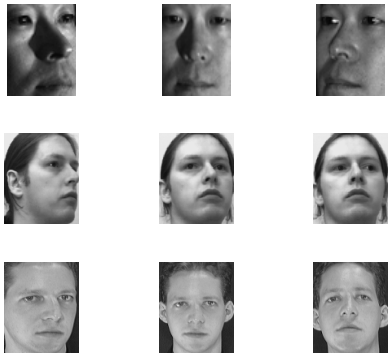


FIGURE 2.9: From top to bottom row: subjects in PIE, UMIST and ORL database, taken under different poses

Face Images with non-frontal Poses

It is known that human frontal face images are well modeled by subspaces. For example, the Yale-B face in section 2.5.2, where LRT slightly outperforms TRAIT. Now we further compare the performance of TRAIT and LRT in more mismatched cases by introducing non-frontal face images. We validate performance on three publicly available datasets, PIE [73], UMIST[2] and ORL[3]. All of them have a considerable number of non-frontal face images. Fig. 2.9 shows one subject from each database with different poses.

The PIE dataset includes 18562 64×48 images of 68 subjects. Each image is labeled with one of 13 different pose tags. We randomly select 7 pose tags and the images of these tags are used as training samples. The rest are used in testing. UMIST comprises 575 112×92 images of 20 subjects, and ORL comprises 400 112×92 images of 40 subjects. These two datasets have no pose tags. We split the UMIST and ORL datasets using the strategy followed for the Yale-B dataset in Section 2.5.2. We derive 1000-dimensional features for each of random projection, LDA, LRT and TRAIT. Table 2.1 lists accuracies of NSC classification for the different algorithms.

[2] http://www.sheffield.ac.uk/eee/research/iel/research/face

[3] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

Table 2.1: NSC accuracy on original and 1000 dimensional (compressed) extracted features

	PIE	UMIST	ORL
Original	74.57%	96.14%	95.50%
random	72.14%	95.44%	94.50%
LDA	40.10%	84.91%	92.00%
LRT	70.80%	96.84%	95.00%
TRAIT	**76.11%**	**97.90%**	**97.00%**

In all cases, TRAIT has the highest classification accuracy and outperforms LRT. LRT optimizes the rank (its convex relaxation), which is critical for reducing classification error in the high SNR regime. However, in this low SNR regime, TRAIT gains more discrimination via explicitly "orthogonalizing" between the classes. The criteria employed by TRAIT do not depend on the specific SNR regime and therefore are more robust.

2.6 Conclusion

In a low-rank Gaussian Mixture Model, we have explored how the probability of misclassification is governed by principal angles between subspaces. In the low-noise regime, the Bhattacharyya upper bound on misclassification is determined by the product of the sines of the principal angles. In the high/moderate-noise regime it is determined by the sum of the squares of the sines of the principal angles. Analysis of the Nearest Subspace Classifier connected reliability of classification to the distribution of signal energy across principal vectors. Classification was shown to be more reliable when more signal energy is associated with principal vectors corresponding to large principal angles. This observation motivated the design of a transform, TRAIT, that achieves superior classification performance by enlarging principal angles and preserving intra-class structure. Finally we showed that TRAIT complements a prior approach that enlarge inter-class distance while suppressing

intra-class dispersion, and that it is more robust to model mismatch.

3

Local Structure and Robust Feature Learning

The last section studies the linear subspace model, and proposes a linear feature extraction method, where angle preservation within classes introduces robustness with respect to model mismatch. In fact, the gain of structure preserving is more apparent for problems where only a small amount of labeled training data is available. A learned model with large number of parameters can well describe a very complicated data distribution. However, it is also more likely to overfit to noise, to be sensitive to small perturbation on the data, and to suffer from degraded performance when generalizing to unseen (test) data. This overfitting problem is especially prominent when training a "big" model while only a small training set is available.

This chapter proposes a framework for learning features that are robust to data variation. To be specific, we regularize the learning process by preserving local distances. The framework makes it possible to tradeoff the discriminative value of learned features against the generalization error of the learning algorithm. Robustness is achieved by encouraging the transform that maps data to features to be a local isometry. This geometric property is shown to improve (K, ϵ)-robustness, thereby providing theoretical justification for reductions in generalization error observed in

experiments. The proposed optimization framework is used to train standard learning algorithms such as deep neural networks. Experimental results obtained on benchmark datasets, such as labeled faces in the wild, demonstrate the value of being able to balance discrimination and robustness.

3.1 Learning Robust Features from a Small Training Set

Learning features that are able to discriminate is a classical problem in data analysis. The basic idea is to reduce the variance within a class while increasing it between classes. One way to implement this is by regularizing a certain measure of the variance, while assuming some prior knowledge about the data. For example, Linear Discriminant Analysis (LDA) [33] measures sample covariance and implicitly assumes that each class is Gaussian distributed. The Low Rank Transform (LRT) [63], instead uses nuclear norm to measure the variance and assumes that each class is near a low-rank subspace. A different approach is to regularize the pairwise distances between data points. Examples include the seminal work on metric learning [91] and its extensions [35, 34, 83].

While great attention has been paid to designing objectives to encourage discrimination, less effort has been made in understanding and encouraging robustness to data variation, which is especially important when a limited number of training samples are available. One exception is [96], which promotes robustness by regularizing the traditional metric learning objective using prior knowledge from an auxiliary unlabeled dataset.

In this chapter we develop a general framework for balancing discrimination and robustness. Robustness is achieved by encouraging the learned data-to-features transform to be locally an isometry within each class. We theoretically justify this approach using (K, ϵ)-robustness [92, 15] and give a concrete example of the proposed formulation, incorporating them in deep neural networks. Experiments validate the

capability to trade-off discrimination against robustness. Our main contributions are the following: 1) we prove that local near-isometry leads to robustness; 2) we propose a practical framework that allows to robustify a wide class of learned transforms, both linear and nonlinear; 3) we provide an explicit realization of the proposed framework, achieving competitive results on difficult face verification tasks.

3.2 Problem Formulation

Consider an L-way classification problem. The training set is denoted by $\mathcal{T} = \{(\mathbf{x}_i, y_i)\}$, where $\mathbf{x}_i \in \mathbb{R}^n$ is the data and $y_i \in \{1, \ldots, L\}$ is the class label. We want to learn a feature transform $f_{\boldsymbol{\alpha}}(\cdot)$ such that a datum $\mathbf{x}$ becomes more discriminative when it is transformed to feature $f_{\boldsymbol{\alpha}}(\mathbf{x})$.

The transform $f_{\boldsymbol{\alpha}}$ is parametrized by a vector $\boldsymbol{\alpha}$, a framework that includes linear transforms and neural networks where the entries of $\boldsymbol{\alpha}$ are the learned network parameters.

3.2.1 Motivation

The transform $f_{\boldsymbol{\alpha}}$ promotes discriminability by reducing intra-class variance and enlarging inter-class variance. This aim is expressed in the design of objective functions [34, 63] or the structure of the transform $f_{\boldsymbol{\alpha}}$ [74, 45]. However the robustness of the learned transform is an important issue that is often overlooked. When training samples are scarce, statistical learning theory [80] predicts overfitting to the training data. The result of overfitting is that discrimination achieved on test data will be significantly worse than that on training data. Our aim in this chapter is the design of robust transforms $f_{\boldsymbol{\alpha}}$ for which the training-to-testing degradation is small [92].

We formally measure robustness of the learned transform $f_{\boldsymbol{\alpha}}$ in terms of (K, ϵ)-robustness [15]. Given a distance metric ρ, a learning algorithm is said to be (K, ϵ)-robust if the input data space can be partitioned into K disjoint sets $\mathcal{S}_k, k = 1, ..., K$,

such that for all training sets $\mathcal{T}$, the learned parameter $\boldsymbol{\alpha}_{\mathcal{T}}$ determines a loss for which the value on pairs of training samples taken from different sets S_j and S_k is very close to the value of any pair of data samples taken from S_j and S_k.

(K, ϵ)-robustness is illustrated in Fig. 3.1, where S_1 and S_2 are both of diameter γ and

$$|e - e'| = |\rho(f_{\boldsymbol{\alpha}}(\mathbf{x}_1), f_{\boldsymbol{\alpha}}(\mathbf{x}_2)) - \rho(f_{\boldsymbol{\alpha}}(\mathbf{x}'_1), f_{\boldsymbol{\alpha}}(\mathbf{x}'_2))|.$$

If the transform $f_{\boldsymbol{\alpha}}$ preserves all distances within S_1 and S_2, then $|e - e'|$ cannot deviate much from $|d - d'| \leq 2\gamma$.

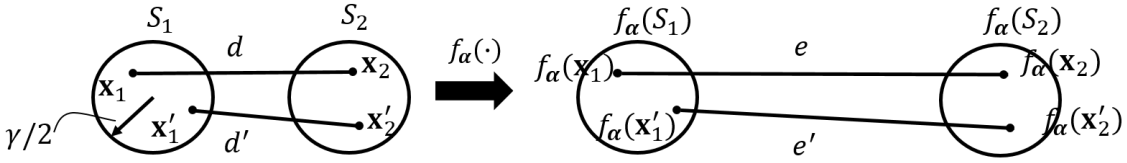


FIGURE 3.1: (K, ϵ)-robustness: Here $d = \rho(\mathbf{x}_1, \mathbf{x}_2)$, $d' = \rho(\mathbf{x}'_1, \mathbf{x}'_2)$, $e = \rho(f_{\boldsymbol{\alpha}}(\mathbf{x}_1), f_{\boldsymbol{\alpha}}(\mathbf{x}_2))$, and $e' = \rho(f_{\boldsymbol{\alpha}}(\mathbf{x}'_1), f_{\boldsymbol{\alpha}}(\mathbf{x}'_2))$. The difference $|e - e'|$ cannot deviate too much from $|d - d'|$.

3.2.2 Formulation

Motivated by the above reasoning, we now present our proposed framework. First we define a pair label $\ell_{i,j} \triangleq \begin{cases} 1 & \text{if } y_i = y_j \\ -1 & \text{otherwise} \end{cases}$. Given a metric ρ, we use the following hinge loss to encourage high inter-class distance and small intra-class distance.

$$\frac{1}{|\mathcal{P}|} \sum_{i,j \in \mathcal{P}} \max\left\{0, \ell_{i,j}\left[\rho\left(f_{\boldsymbol{\alpha}}(\mathbf{x}_i), f_{\boldsymbol{\alpha}}(\mathbf{x}_j)\right) - t(\ell_{i,j})\right]\right\}, \tag{3.1}$$

Here $\mathcal{P} = \{(i, j | i \neq j)\}$ is the set of all data pairs. $t(\ell_{i,j}) \geq 0$ is a function of $\ell_{i,j}$ and $t(1) < t(-1)$. Similar to metric learning [91], this loss function connects pairwise distance to discrimination. However traditional metric learning typically assumes squared Euclidean distance and here the metric ρ can be arbitrary.

For robustness, as discussed above, we may want $f_{\boldsymbol{\alpha}}(\cdot)$ to be distance-preserving within each small local region. In particular, we define the set of all local neighborhoods as

$$\mathcal{NB} \triangleq \{(i,j)|\ell_{i,j}=1, \rho(\mathbf{x}_i,\mathbf{x}_j)\leq\gamma\}\,.$$

Therefore, we minimize the following objective function

$$\frac{1}{|\mathcal{NB}|}\sum_{(i,j)\in\mathcal{NB}}\left|\rho(f_{\boldsymbol{\alpha}}(\mathbf{x}_i),f_{\boldsymbol{\alpha}}(\mathbf{x}_j))-\rho(\mathbf{x}_i,\mathbf{x}_j)\right|. \tag{3.2}$$

Note that we do not need to have the same metric in both the input and the feature space, they do not even have in general the same dimension. With a slight abuse of notation we use the same symbol to denote both metrics.

To achieve discrimination and robustness simultaneously, we formulate the objective function as a weighted linear combination of the two extreme cases in (3.1) and (3.2)

$$\frac{\lambda}{|\mathcal{P}|}\sum_{i,j\in\mathcal{P}}\max\left\{0,\ell_{i,j}\left[\rho\left(f_{\boldsymbol{\alpha}}(\mathbf{x}_i),f_{\boldsymbol{\alpha}}(\mathbf{x}_j)\right)-t(\ell_{i,j})\right]\right\}+\frac{1-\lambda}{|\mathcal{NB}|}\sum_{(i,j)\in\mathcal{NB}}\left|\rho(f_{\boldsymbol{\alpha}}(\mathbf{x}_i),f_{\boldsymbol{\alpha}}(\mathbf{x}_j))-\rho(\mathbf{x}_i,\mathbf{x}_j)\right| \tag{3.3}$$

where $\lambda\in[0,1]$.

The formulation (3.3) balances discrimination and robustness. When $\lambda=1$ it seeks discrimination, and as λ decreases it starts to encourage robustness. We shall refer to a transform that is learned by solving (3.3) as a Discriminative Robust Transform (DRT). The DRT framework provides opportunity to select both the distance measure and the transform family.

3.3 Theoretical Analysis

In this section, we provide a theoretical explanation for robustness. In particular, we show that if the solution to (3.1) yields a transform $f_{\boldsymbol{\alpha}}$ that is locally a near isometry,

then $f_{\boldsymbol{\alpha}}$ is robust.

3.3.1 Theoretical Framework

Let $\mathcal{X}$ denote the original data, let $\mathcal{Y} = \{1, ..., L\}$ denote the set of class labels, and let $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. The training samples are pairs $\mathbf{z}_i = (\mathbf{x}_i, y_i), i = 1, \ldots, n$ drawn from some unknown distribution $\mathcal{D}$ defined on $\mathcal{Z}$. The indicator function is defined as $\ell_{i,j} = 1$ if $y_i = y_j$ and -1 otherwise. Let $f_{\boldsymbol{\alpha}}$ be a transform that maps a low-level feature $\mathbf{x}$ to a more discriminative feature $f_{\boldsymbol{\alpha}}(\mathbf{x})$, and let $\mathcal{F}$ denote the space of transformed features.

For simplicity we consider an arbitrary metric ρ defined on both $\mathcal{X}$ and $\mathcal{F}$ (the general case of different metrics is a straightforward extension), and a loss function $g(\rho(f_{\boldsymbol{\alpha}}(\mathbf{x}_i), f_{\boldsymbol{\alpha}}(\mathbf{x}_j)), \ell_{i,j})$ that encourages $\rho(f_{\boldsymbol{\alpha}}(\mathbf{x}_i), f_{\boldsymbol{\alpha}}(\mathbf{x}_j))$ to be small (big) if $\ell_{i,j} = 1$ (-1). We shall require the Lipschtiz constant of $g(\cdot, 1)$ and $g(\cdot, -1)$ to be upper bounded by $A > 0$. Note that the loss function in Eq. (3.1) has a Lipschtiz constant of 1. We abbreviate

$$g(\rho(f_{\boldsymbol{\alpha}}(\mathbf{x}_i), f_{\boldsymbol{\alpha}}(\mathbf{x}_j)), \ell_{i,j}) \triangleq h_{\boldsymbol{\alpha}}(\mathbf{z}_i, \mathbf{z}_j).$$

The empirical loss on the training set is a function of $\boldsymbol{\alpha}$ given by

$$R_{emp}(\boldsymbol{\alpha}) \triangleq \frac{2}{n(n-1)} \sum_{\substack{i,j=1 \\ i \neq j}}^{n} h_{\boldsymbol{\alpha}}(\mathbf{z}_i, \mathbf{z}_j), \tag{3.4}$$

and the expected loss on the test data is given by

$$R(\boldsymbol{\alpha}) \triangleq \mathbb{E}_{\mathbf{z}'_1, \mathbf{z}'_2 \sim \mathcal{D}} \left[h_{\boldsymbol{\alpha}}(\mathbf{z}'_1, \mathbf{z}'_2) \right]. \tag{3.5}$$

The algorithm operates on pairs of training samples and finds parameters

$$\boldsymbol{\alpha}_{\mathcal{T}} \triangleq \arg\min_{\boldsymbol{\alpha}} R_{emp}(\boldsymbol{\alpha}), \tag{3.6}$$

that minimize the empirical loss on the training set $\mathcal{T}$. The difference $R_{emp} - R$ between expected loss on the test data and empirical loss on the training data is the generalization error of the algorithm.

3.3.2 (K, ϵ)-robustness and Covering Number

We work with the following definition of (K, ϵ)-robustness [15].

Definition 1. *A learning algorithm is (K, ϵ)-robust if $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ can be partitioned into K disjoint sets $\mathcal{Z}_k, k = 1, \ldots, K$ such that for all training sets $\mathcal{T} \in \mathcal{Z}^n$, the learned parameter $\boldsymbol{\alpha}_{\mathcal{T}}$ determines a loss function where the value on pairs of training samples taken from sets $\mathcal{Z}_p$ and $\mathcal{Z}_q$ is "very close" to the value of any pair of data samples taken from $\mathcal{Z}_p$ and $\mathcal{Z}_q$. Formally,*
assume $\mathbf{z}_i, \mathbf{z}_j \in \mathcal{T}$, with $\mathbf{z}_i \in \mathcal{Z}_p$ and $\mathbf{z}_j \in \mathcal{Z}_q$, if $\mathbf{z}'_i \in \mathcal{Z}_p$ and $\mathbf{z}'_j \in \mathcal{Z}_q$, then

$$\left| h_{\boldsymbol{\alpha}_{\mathcal{T}}}(\mathbf{z}_i, \mathbf{z}_j) - h_{\boldsymbol{\alpha}_{\mathcal{T}}}(\mathbf{z}'_i, \mathbf{z}'_j) \right| \leq \epsilon.$$

Remark 5. *(K, ϵ)-robustness means that the loss incurred by a testing pair $(\mathbf{z}'_i, \mathbf{z}'_j)$ in $\mathcal{Z}_p \times \mathcal{Z}_q$ is very close to the loss incurred by any training pair $(\mathbf{z}_i, \mathbf{z}_j)$ in $\mathcal{Z}_p \times \mathcal{Z}_q$. It is shown in [15] that the generalization error of (K, ϵ)-robust algorithms is bounded as*

$$R(\boldsymbol{\alpha}_{\mathcal{T}}) - R_{emp}(\boldsymbol{\alpha}_{\mathcal{T}}) \leq \epsilon + O\left(\sqrt{\frac{K}{n}}\right). \tag{3.7}$$

Therefore the smaller ϵ, the smaller is the generalization error, and the more robust is the learning algorithm.

Given a metric space, the covering number specifies how many balls of a given radius are needed to cover the space. The more complex the metric space, the more balls are needed to cover it. Covering number is formally defined as follows.

Definition 2 (Covering number)**.** *Given a metric space $(\mathcal{S}, \rho)$, we say that a subset $\hat{\mathcal{S}}$ of $\mathcal{S}$ is a γ-cover of $\mathcal{S}$, if for every element $\mathbf{s} \in \mathcal{S}$, there exists $\hat{\mathbf{s}} \in \hat{\mathcal{S}}$ such that $\rho(\mathbf{s}, \hat{\mathbf{s}}) \leq \gamma$. The γ-covering number of $\mathcal{S}$ is*

$$\mathcal{N}_\gamma(\mathcal{S}, \rho) = \min\{|\hat{\mathcal{S}}| : \hat{\mathcal{S}} \text{ is a } \gamma\text{-cover of } \mathcal{S}\}.$$

Remark 6. *The covering number is a measure of the geometric complexity of* $(\mathcal{S}, \rho)$. *A set* S *with covering number* $\mathcal{N}_{\gamma/2}(\mathcal{S}, \rho)$ *can be partitioned into* $\mathcal{N}_{\gamma/2}(\mathcal{S}, \rho)$ *disjoint subsets, such that any two points within the same subset are separated by no more than* γ.

Lemma 10. *The metric space* $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ *can be partitioned into* $L\mathcal{N}_{\gamma/2}(\mathcal{X}, \rho)$ *subsets, denoted as* $\mathcal{Z}_1, \ldots, \mathcal{Z}_{L\mathcal{N}_{\gamma/2}(\mathcal{X},\rho)}$, *such that any two points* $\mathbf{z}_1 \triangleq (\mathbf{x}_1, y_1), \mathbf{z}_2 \triangleq (\mathbf{x}_2, y_2)$ *in the same subset satisfy* $y_1 = y_2$ *and* $\rho(\mathbf{x}_1, \mathbf{x}_2) \leq \gamma$.

Proof. Assuming the metric space $(\mathcal{X}, \rho)$ is compact, we can partition $\mathcal{X}$ into $\mathcal{N}_{\gamma/2}(\mathcal{X}, \rho)$ subsets, each with diameter at most γ. Since $\mathcal{Y}$ is a finite set of size L, we can partition $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ into $L\mathcal{N}_{\gamma/2}(\mathcal{X}, \rho)$ subsets with the property that two samples $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2)$ in the same subset satisfy $y_1 = y_2$ and $\rho(\mathbf{x}_1, \mathbf{x}_2) \leq \gamma$. □

It follows from Lemma 10 that we may partition $\mathcal{X}$ into subsets $\mathcal{X}_1, \ldots, \mathcal{X}_{L\mathcal{N}_{\gamma/2}(\mathcal{X},\rho)}$, such that pairs of points $\mathbf{x}_1, \mathbf{x}_2$ from the same subset have the same label and satisfy $\rho(\mathbf{x}_i, \mathbf{x}_j) \leq \gamma$. Before we connect local geometry to robustness we need one more definition. We say that a learned transform $f_{\boldsymbol{\alpha}}$ is a δ-isometry if the metric is distorted by at most δ:

Definition 3 (δ-isometry)**.** *Let* $\mathcal{A}, \mathcal{B}$ *be metric spaces with metrics* $\rho_{\mathcal{A}}$ *and* $\rho_{\mathcal{B}}$. *A map* $f : \mathcal{A} \mapsto \mathcal{B}$ *is a* δ*-isometry if for any* $\mathbf{a}_1, \mathbf{a}_2 \in \mathcal{A}$, $|\rho_{\mathcal{A}}(f(\mathbf{a}_1), f(\mathbf{a}_2)) - \rho_{\mathcal{B}}(\mathbf{a}_1, \mathbf{a}_2)| \leq \delta$.

Theorem 11. *Let* $f_{\boldsymbol{\alpha}}$ *be a transform derived via Eq.* (3.6) *and let* $\mathcal{X}_1, \ldots, \mathcal{X}_{L\mathcal{N}_{\gamma/2}(\mathcal{X},\rho)}$ *be a cover of* $\mathcal{X}$ *as described above. If* $f_{\boldsymbol{\alpha}}$ *is a* δ*-isometry, then it is* $(L\mathcal{N}_{\gamma/2}(\mathcal{X}, \rho), 2A(\gamma + \delta))$*-robust.*

Proof sketch. Consider training samples $\mathbf{z}_i, \mathbf{z}_j$ and testing samples $\mathbf{z}'_i, \mathbf{z}'_j$ such that $\mathbf{z}_i, \mathbf{z}'_i \in \mathcal{Z}_p$ and $\mathbf{z}_j, \mathbf{z}'_j \in \mathcal{Z}_q$ for some $p, q \in \{1, \ldots, L\mathcal{N}_{\gamma/2}(\mathcal{X}, \rho)\}$. Then by Lemma 10,

$$\rho(\mathbf{x}_i, \mathbf{x}'_i) \leq \gamma \text{ and } \rho(\mathbf{x}_j, \mathbf{x}'_j) \leq \gamma, \qquad y_i = y'_i \text{ and } y_j = y'_j,$$

and $\mathbf{x}_i, \mathbf{x}'_i \in \mathbf{X}_p$ and $\mathbf{x}_j, \mathbf{x}'_j \in \mathbf{X}_q$. By definition of δ-isometry,

$$|\rho(f_{\boldsymbol{\alpha}_\mathcal{T}}(\mathbf{x}_i), f_{\boldsymbol{\alpha}_\mathcal{T}}(\mathbf{x}'_i)) - \rho(\mathbf{x}_i, \mathbf{x}'_i)| \leq \delta \text{ and } |\rho(f_{\boldsymbol{\alpha}_\mathcal{T}}(\mathbf{x}_j), f_{\boldsymbol{\alpha}_\mathcal{T}}(\mathbf{x}'_j)) - \rho(\mathbf{x}_j, \mathbf{x}'_j)| \leq \delta.$$

Rearranging the terms gives

$$\rho(f_{\boldsymbol{\alpha}_\mathcal{T}}(\mathbf{x}_i), f_{\boldsymbol{\alpha}_\mathcal{T}}(\mathbf{x}'_i)) \leq \rho(\mathbf{x}_i, \mathbf{x}'_i)+\delta \leq \gamma+\delta \text{ and } \rho(f_{\boldsymbol{\alpha}_\mathcal{T}}(\mathbf{x}_j), f_{\boldsymbol{\alpha}_\mathcal{T}}(\mathbf{x}'_j)) \leq \rho(\mathbf{x}_j, \mathbf{x}'_j)+\delta \leq \gamma+\delta.$$

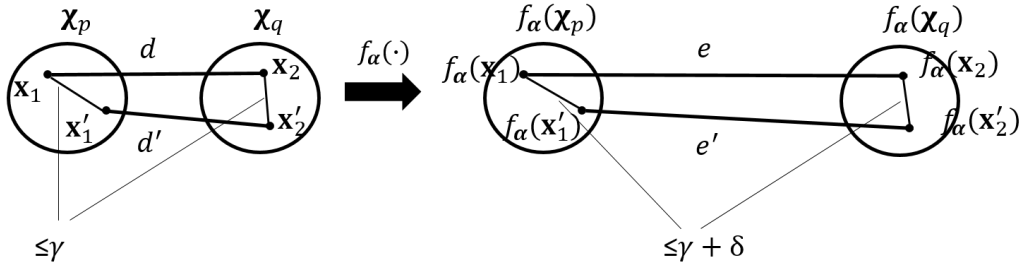


FIGURE 3.2: Proof without words.

In order to bound the generalization error, we need to bound the difference between $\rho(f_{\boldsymbol{\alpha}_\mathcal{T}}(\mathbf{x}_i), f_{\boldsymbol{\alpha}_\mathcal{T}}(\mathbf{x}_j))$ and $\rho(f_{\boldsymbol{\alpha}_\mathcal{T}}(\mathbf{x}'_i), f_{\boldsymbol{\alpha}_\mathcal{T}}(\mathbf{x}'_j))$. The details can be found in [46]; here we appeal to the proof schematic in Fig. 3.2. We need to bound $|e - e'|$ and it cannot exceed twice the diameter of a local region in the transformed domain. □

Robustness of the learning algorithm depends on the granularity of the cover and the degree to which the learned transform $f_{\boldsymbol{\alpha}}$ distorts distances between pairs of points in the same covering subset. The subsets in the cover constitute regions where the local geometry makes it possible to bound generalization error. It now follows from [15] that the generalization error satisfies $R(\boldsymbol{\alpha}_\mathcal{T}) - R_{emp}(\boldsymbol{\alpha}_\mathcal{T}) \leq 2A(\gamma + \delta) + O\left(\sqrt{\frac{K}{n}}\right)$.

The DRT proposed here is a particular example of a local isometry, and Theorem 11 explains why the generalization error is smaller than that of pure metric learning.

The transform described in [46] partitions the metric space $\mathcal{X}$ into exactly L subsets, one for each class. The experiments reported in Section 3.5 demonstrate

that the performance improvements derived from working with a finer partition can be worth the cost of learning finer grained local regions.

3.4 An Illustrative Realization of DRT

Having justified robustness, we now provide a realization of the proposed general DRT where the metric ρ is Euclidean distance. We use Gaussian random variables to initialize $\boldsymbol{\alpha}$, then, on the randomly transformed data, we set $t(1)$ ($t(-1)$) to be the average intra-class (inter-class) pairwise distance. In all our experiments, the solution satisfied the condition $t(1) < t(-1)$ required in Eq. (3.1). We calculate the diameter γ of the local regions $\mathcal{NB}$ indirectly, using the κ-nearest neighbors of each training sample to define a local neighborhood. We leave the question of how best to initialize the indicator t and the diameter γ for future research.

We denote this particular example as Euc-DRT and use gradient descent to solve for $\boldsymbol{\alpha}$. Denoting the objective by J, we define $\mathbf{y}_i \triangleq f_{\boldsymbol{\alpha}}(\mathbf{x}_i)$, $\delta_{i,j} \triangleq f_{\boldsymbol{\alpha}}(\mathbf{x}_i) - f_{\boldsymbol{\alpha}}(\mathbf{x}_j)$, and $\rho^0_{i.j} \triangleq \|\mathbf{x}_i - \mathbf{x}_j\|$. Then

$$\frac{\partial J}{\partial \mathbf{y}_i} = \sum_{\substack{(i,j)\in\mathcal{P} \\ \ell_{i,j}(\|\delta_{i,j}\|-t(\ell_{i,j}))>0}} \frac{\lambda}{|\mathcal{P}|}\cdot\ell_{i,j}\cdot\frac{\delta_{i,j}}{\|\delta_{i,j}\|} + \sum_{(i,j)\in\mathcal{NB}} \frac{1-\lambda}{|\mathcal{NB}|}\cdot\mathrm{sgn}(\|\delta_{i,j}\|-\rho^0_{i,j})\cdot\frac{\delta_{i,j}}{\|\delta_{i,j}\|}. \quad (3.8)$$

In general, $f_{\boldsymbol{\alpha}}$ defines a D-layer neural network (when $D = 1$ it defines a linear transform). Let $\boldsymbol{\alpha}^{(d)}$ be the linear weights at the d-th layer, and let $\mathbf{x}^{(d)}$ be the output of the d-th layer, so that $\mathbf{y}_i = \mathbf{x}_i^{(D)}$. Then the gradients are computed as,

$$\frac{\partial J}{\partial \boldsymbol{\alpha}^{(D)}} = \sum_i \frac{\partial J}{\partial \mathbf{y}_i}\cdot\frac{\partial \mathbf{y}_i}{\partial \boldsymbol{\alpha}^{(D)}}, \text{ and } \frac{\partial J}{\partial \boldsymbol{\alpha}^{(d)}} = \sum_i \frac{\partial J}{\partial \mathbf{x}_i^{(d+1)}}\cdot\frac{\partial \mathbf{x}_i^{(d+1)}}{\partial \mathbf{x}_i^{(d)}}\cdot\frac{\partial \mathbf{x}_i^{(d)}}{\partial \boldsymbol{\alpha}^{(d)}} \text{ for } 1 \le d \le D-1. \quad (3.9)$$

Algorithm 2 provides a summary, and we note that the extension to stochastic training using min-batches is straightforward.

Algorithm 2 Gradient descent solver for Euc-DRT

Input: $\lambda \in [0,1]$, training pairs $\{(\mathbf{x}_i, \mathbf{x}_j, \ell_{i,j})\}$, a pre-defined D-layer network ($D = 1$ as linear transform), stepsize η, neighborhood size κ.
Output: $\boldsymbol{\alpha}$

Randomly initialize $\boldsymbol{\alpha}$, compute $\mathbf{y}_i = f_{\boldsymbol{\alpha}}(\mathbf{x}_i)$.
On the $\mathbf{y}_i$, compute the average intra and inter-class pairwise distances, assign to $t(1)$, $t(-1)$
For each training datum, find its κ nearest neighbor and define the set $\mathcal{NB}$.
while stable objective not achieved **do**
  Compute $\mathbf{y}_i = f_{\boldsymbol{\alpha}}(\mathbf{x}_i)$ by a forward pass.
  Compute objective J.
  Compute $\frac{\partial J}{\partial \mathbf{y}_i}$ as Eq. (3.8).
  for $l = D$ down to 1 **do**
    Compute $\frac{\partial J}{\partial \boldsymbol{\alpha}^{(d)}}$ as Eq. (3.9).
    $\boldsymbol{\alpha}^{(d)} \leftarrow \boldsymbol{\alpha}^{(d)} - \eta \frac{\partial J}{\partial \boldsymbol{\alpha}^{(d)}}$.
  end for
end while

3.4.1 Other Distance

In the above we use the Euclidean distance metric; we have also explored a cosine distance [46] based formulation, which takes the following form

$$\min_{\boldsymbol{\alpha}} \frac{1}{2} \sum_{i \neq j} \left(\frac{f_{\boldsymbol{\alpha}}(\mathbf{x}_i)^\top f_{\boldsymbol{\alpha}}(\mathbf{x}_j)}{\|f_{\boldsymbol{\alpha}}(\mathbf{x}_i)\| \cdot \|f_{\boldsymbol{\alpha}}(\mathbf{x}_j)\|} - t_{i,j} \right)^2, \tag{3.10}$$

where the indicator is

$$t_{i,j} = \begin{cases} \lambda + (1-\lambda) \frac{\mathbf{x}_i^\top \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|} & \text{if } \mathbf{x}_i, \mathbf{x}_j \in \text{ same class,} \\ -1 & \text{otherwise.} \end{cases}$$

and $\lambda \in [0,1]$. The rationale is as follows: with cosine distance, the most discriminative case is to make any two inter-class samples span an angle of π, and any two intra-class samples span an angle of zero. Meanwhile, preservation of intra-class angles is encouraged due to the robustness constraint. Cosine distance is not a metric, which does not fit gracefully into the theory. In later sections, we will focus on the Euclidean metric based method.

Table 3.1: Varying λ on a toy dataset.

λ	1	0.5	0.25
R_{emp}	1.5983	1.6025	1.9439
generalization error	10.5855	9.5071	8.8040
1-nn accuracy (original data 93.35%)	92.20%	**98.30%**	91.55%

3.5 Experimental Results

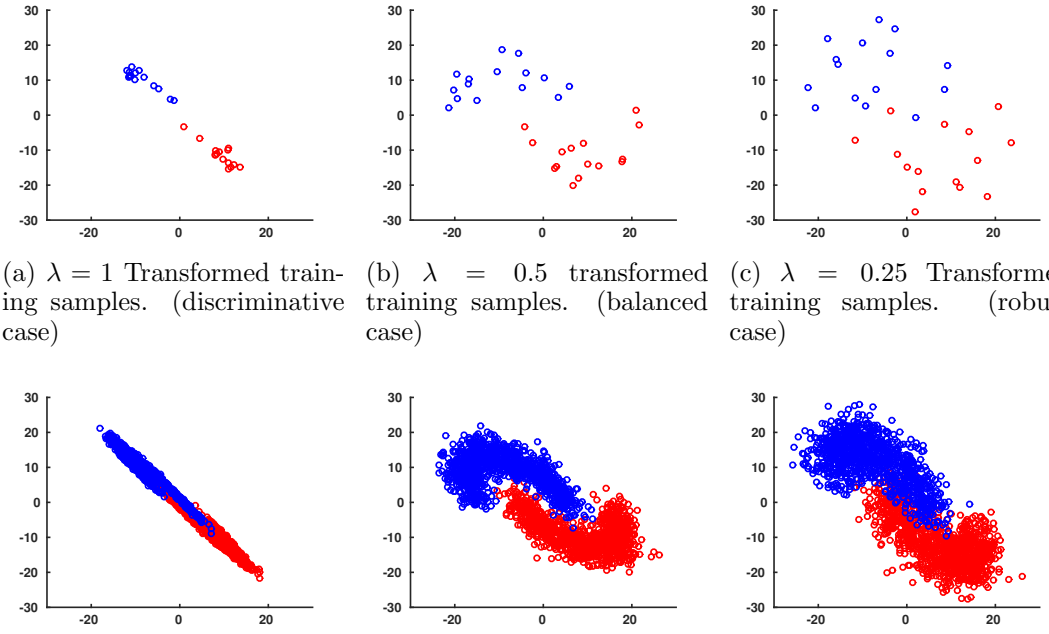
In this section we report on experiments that confirm robustness of Euc-DRT. Recall that empirical loss is given by Eq. (3.4) where $\boldsymbol{\alpha}$ is learned as $\boldsymbol{\alpha}_{\mathcal{T}}$ from the training set $\mathcal{T}$, and $|\mathcal{T}| = N$. The generalization error is $R - R_{emp}$ where the expected loss R is estimated using a large test set.

3.5.1 Toy Example

This illustrative example is motivated by the discussion in Section 3.2.1. We first generate a 2D dataset consisting of two noisy half-moons, then use a random 100×2 matrix to embed the data in a 100-dimensional space. We learn a linear transform $f_{\boldsymbol{\alpha}}$ that maps the 100 dimensional data to 2 dimensional features, and we use $\kappa = 5$ nearest neighbors to construct the set $\mathcal{NB}$. We consider $\lambda = 1, 0.5, 0.25$, representing the most discriminative, balanced, and more robust scenarios.

When $\lambda = 1$ the transformed training samples are rather discriminative (Fig. 3.3a), but when the transform is applied to testing data, the two classes are more mixed (Fig. 3.3d). When $\lambda = 0.5$, the transformed training data are more dispersed within each class (Fig. 3.3b), hence less easily separated than when $\lambda = 1$. However Fig. 3.3e shows that it is easier to separate the two classes on the test data. When $\lambda = 0.25$, robustness is preferred to discriminative power as shown in Figs. 3.3c and 3.3f.

Tab. 3.1 quantifies empirical loss R_{emp}, generalization error, and classification performance (by 1-nn) for $\lambda = 1, 0.5$ and 0.25. As λ decreases, R_{emp} increases,



(a) $\lambda = 1$ Transformed training samples. (discriminative case)

(b) $\lambda = 0.5$ transformed training samples. (balanced case)

(c) $\lambda = 0.25$ Transformed training samples. (robust case)

(d) $\lambda = 1$ Transformed testing samples. (discriminative case)

(e) $\lambda = 0.5$ transformed testing samples. (balanced case)

(f) $\lambda = 0.25$ Transformed testing samples. (robust case)

FIGURE 3.3: Original and transformed training/testing samples embedded in 2-dimensional space with different colors representing different classes.

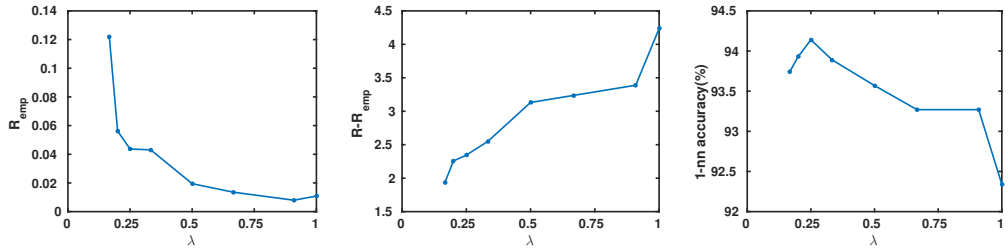
indicating loss of discrimination on the training set. However, generalization error decreases, implying more robustness. We conclude that by varying λ, we can balance discrimination and robustness.

3.5.2 MNIST Classfication Using a Very Small Training Set

The transform $f_{\boldsymbol{\alpha}}$ learned in the previous section was linear, and we now apply a more sophisticated convolutional neural network to the MNIST dataset. The network structure is similar to LeNet, and is made up of alternating convolutional layers and pooling layers, with parameters detailed in Table 3.2. We map the original 784-dimensional pixel values (28x28 image) to 128-dimensional features.

While state-of-art results often use the full training set (6,000 training samples per class), here we are interested in small training sets. We use only 30 training

samples per class, and we use $\kappa = 7$ nearest neighbors to define local regions in Euc-DRT. We vary λ and study empirical error, generalization error, and classification accuracy (1-nn). We observe in Fig. 3.4 that when λ decreases, the empirical error also decreases, but that the generalization error actually increases. By balancing between these two factors, a peak classification accuracy is achieved at $\lambda = 0.25$.



(a) (b) (c)

FIGURE 3.4: MNIST test: with only 30 training samples per class. We vary λ and assess (a) R_{emp}; (b) generalization error; and (c) 1-nn classification accuracy. Peak accuracy is achieved at $\lambda = 0.25$.

Table 3.2: Implementation details of the neural network for MNIST classification.

name	parameters
conv1	size: $5 \times 5 \times 1 \times 20$ stride: 1, pad: 0
pool1	size: 2×2
conv2	size: $5 \times 5 \times 20 \times 50$ stride: 1, pad: 0
pool2	size: 2×2
conv3	size: $4 \times 4 \times 50 \times 128$ stride: 1, pad: 0

Next, we use 30, 50, 70, 100 training samples per class and compare the performance of Euc-DRT with LeNet and Deep Metric Learning (DML) [45]. DML minimizes a hinge loss on the squared Euclidean distances. It shares the same spirit with our Euc-DRT using $\lambda = 1$. All methods use the same network structure, Tab. 3.2, to map to the features. For classification, LeNet uses a linear softmax

Table 3.3: Classification error on MNIST.

Training/class	30	50	70	100
original pixels	81.91%	86.18%	86.86%	88.49%
LeNet	87.51%	89.89%	91.24%	92.75%
DML	92.32%	94.45%	95.67%	96.19%
Euc-DRT	**94.14%**	**95.20%**	**96.05%**	**96.21%**

classifier on top of the "conv3" layer and minimizes the standard cross-entropy loss during training. DML and Euc-DRT both use a 1-nn classifier on the learned features. Classification accuracies are reported in Tab. 3.3. In Tab. 3.3, we see that all the learned features improve upon the original ones. DML is very discriminative and achieves higher accuracy than LeNet. However, when the training set is very small, robustness becomes more important and Euc-DRT significantly outperforms DML.

3.5.3 Face Verification on LFW

We now present face verification on the more challenging Labeled Faces in the Wild (LFW) benchmark, where our experiments will show that there is an advantage to balancing disciminability and robustness. Our goal is not to reproduce the success of deep learning in face verification [77, 45], but to stress the importance of robust training and to compare the proposed Euc-DRT objective with popular alternatives. Note also that it is difficult to compare with deep learning methods when training sets are proprietary [75, 76, 77].

We adopt the experimental framework used in [18], and train a deep network on the WDRef dataset, where each face is described using a high dimensional LBP feature [19] (available at [1]) that is reduced to a 5000-dimensional feature using PCA. The WDRef dataset is significantly smaller than the proprietary datasets typical of deep learning, such as the 4.4 million labeled faces from 4030 individuals in [77], or the 202,599 labeled faces from 10,177 individuals in [75]. It contains 2,995 subjects

[1] http://home.ustc.edu.cn/chendong/

Table 3.4: Verification accuracy and AUCs on LFW

Method	Accuracy (%)	AUC ($\times 10^{-2}$)
HD-LBP	74.73	82.22±1.00
deepFace	88.72	95.50± 0.29
DML	90.28	96.74±0.33
Euc-DRT	**92.33**	**97.77± 0.25**

with about 20 samples per subject.

We compare the Euc-DRT objective with DeepFace (DF) [77] and Deep Metric Learning (DML) [45], two state-of-the-art deep learning objectives. For a fair comparison, we employ the same network structure and train on the same input data. DeepFace feeds the output of the last network layer to an L-way soft-max to generate a probability distribution over L classes, then minimizes a cross entropy loss. The Euc-DRT feature $f_{\boldsymbol{\alpha}}$ is implemented as a two-layer fully connected network with *tanh* as the squash function. Weight decay (conventional Frobenius norm regularization) is employed in both DF and DML, and results are only reported for the best weight decay factor. After a network is trained on WDRef, it is tested on the LFW benchmark. Verification simply consists of comparing the cosine distance between a given pair of faces to a threshold.

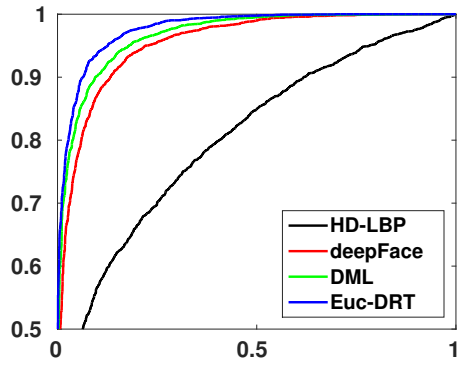


FIGURE 3.5: Comparison of ROCs for all methods

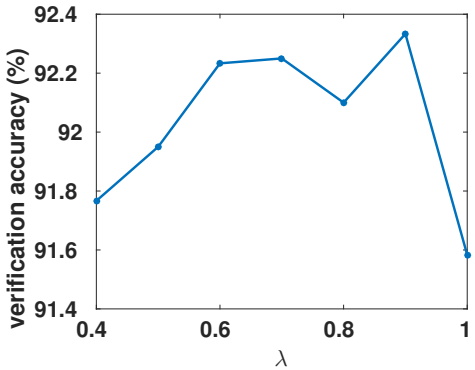


FIGURE 3.6: Verification accuracy of Euc-DRT as λ varies

Fig. 3.5 displays ROC curves and Table 3.4 reports area under the ROC curve (AUC) and verification accuracy. High-Dim LBP refers to verification using the initial LBP features. DeepFace (DF) optimizes for a classification objective by minimizing a softmax loss, and it successfully separates samples from different classes. However the constraint that assigns similar representations to the same class is weak, and this is reflected in the true positive rate displayed in Fig. 3.5. In Deep Metric Learning (DML) this same constraint is strong, but robustness is a concern when the training set is small. The proposed Euc-DRT improves upon both DF and DML by balancing disciminability and robustness. It is less conservative than DF for better discriminability, and more responsive to local geometry than DML for smaller generalization error. Face verification accuracy for Euc-DRT was obtained by varying the regularization parameter λ between 0.4 and 1 (as shown in Fig 3.6), then reporting the peak accuracy observed at $\lambda = 0.9$.

3.6 Conclusion

We have proposed an optimization framework within which it is possible to tradeoff the discriminative value of learned features with robustness of the learning algorithm. Improvements to generalization error predicted by theory are observed in experiments on benchmark datasets. Future work will investigate how to initialize and tune the optimization, also how the Euc-DRT algorithm compares with other methods that reduce generalization error.

4

GraphConnect: where Manifold Models Meet Deep Learning

This chapter continues to study the overfitting problem addressed in the last section. But now we restrict ourself to "deep" feature extractors. Deep neural networks have proved very successful in domains where large training sets are available, but when the number of training samples is small, their performance suffers from overfitting. Prior methods of reducing overfitting such as weight decay [7], *DropOut* [42] and *DropConnect* [82] are data-independent. Complementary to the above, this chapter proposes a new method for regularzing deep networks, *GraphConnect*, that is data-dependent, and is motivated by the observation that data of interest typically lie close to a manifold. This proposed method encourages the relationships between the learned decisions to resemble a graph representing the original manifold structure. Essentially *GraphConnect* is designed to learn attributes that are present in data samples in contrast to weight decay and *DropOut* which are simply designed to make it more difficult to fit to random error or noise. Analysis of empirical Rademacher complexity suggests that *GraphConnect* is stronger than weight decay

as a regularization. Experimental results on several benchmark datasets validate the theoretical analysis, and show that when the number of training samples is small, *GraphConnect* is able to significantly improve performance over weight decay, and competitive with *DropOut*.

4.1 Generalization Error of Deep Neural Networks

Neural networks have proved very successful in domains where large training sets are available; since their capacity can be increased by adding layers or by increasing the number of units in a layer [39]. When the number of training samples is small their performance suffers from overfitting. The degree of overfitting is measured by the generalization error, which is the difference between the loss on the training set and the loss on the test set. The best known bounds on the generalization error arise from the Vapnik-Chervonenkis (VC) dimension [80], which captures the inherent complexity of a family of classifiers. However, VC dimension is distribution agnostic, leading to a loose and often pessimistic upper bound [23].

In the last decade, distribution dependent measures of complexity have been developed, such as the Rademacher complexity [8], which can lead to tighter bounds on the generalization error. Rademacher complexity has been used to show that the generalization error of a neural network depends more on the size of the weights than on the size of the network [7]. This theoretical result supports a form of regularization called weight decay that simply encourages the ℓ-2 norm of all parameters to be small. A unified theoretical treatment of norm-based capacity control of deep neural networks is given in [58], and this theory is used to provide insights into max-out networks in [36].

DropOut is a new form of regularization, where for each training example, forward propagation involves randomly deleting some of the activations in each layer [42]. *DropConnect* is a recent generalization of *DropOut*, where a randomly selected subset

of weights within the network is set to zero [82]. Both methods introduce randomness into training so that the learned network is in some sense a statistical average of an ensemble of realizations. Adding a *DropConnect* layer to a neural network can reduce the Rademacher complexity by a factor of the *DropConnect* rate [82].

In this chapter, we propose a fundamentally different approach to control the complexity of a neural network, thereby preventing overfitting. Our approach differs from the aforementioned methods in that it is data dependent. It is motivated by the empirical observation that data of interest, such as images and audio, typically lie close to a manifold, an assumption that has previously assisted machine learning tasks such as nonlinear embedding [13], semi-supervised labeling [14], and multi-task classification [32]. The underlying idea in these works is to encourage the relationships between the learned decisions to resemble a graph representing the data manifold structure.

Graph/kernel regularized deep learning has been experimentally studied in [84, 95], where unlabeled testing samples are used to regularize training (semi-supervised). However, they did not explain how the approach might control capacity. In particular, it is not clear in these works whether graph-based regularization provides better generalization ability than standard data-independent approaches such as weight decay. In contrast, our work provides theoretical foundations and addresses a supervised learning problem. We first observe that the *GraphConnect* regularized network can achieve significantly better performance on the test data than the classical weight decay regularized network. Theoretical arguments are then provided to support the observation. Finally, extensive experiments are presented to further validate the effectiveness of *GraphConnect*.

4.2 *GraphConnect*: A Motivating Example

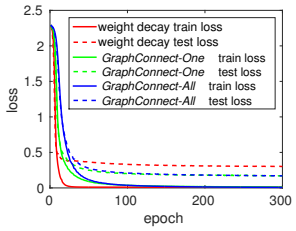
The MNIST dataset contains approximately 60,000 training images (28×28) and 10,000 test images. While state-of-the-art methods often use the entire training set, we are interested in quantifying what is possible with much smaller training sets. Table 4.1 describes the network architecture that we use to extract features and classify data. We begin by using 500 training samples (50 per class) to train the neural networks. The image mean is estimated from the training set and subtracted as a preprocessing step.

Table 4.1: Network architecture in the MNIST experiments, where layer 7 and 8 constitute the softmax classifier.

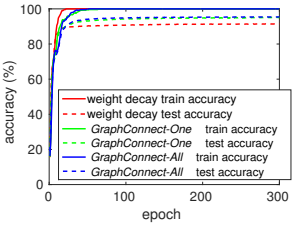
Layer	Type	Parameters
1	conv	size: $5 \times 5 \times 1 \times 20$ stride: 1, pad: 0
2	max pool	size: 2×2, stride: 2, pad: 0
3	conv	size: $5 \times 5 \times 20 \times 50$ stride: 1, pad: 0
4	max pool	size: 2×2, stride: 2, pad: 0
5	conv	size: $4 \times 4 \times 50 \times 500$ stride: 1, pad: 0
6	reLu	N/A
7	fully connected	500×10
8	softmaxloss	N/A

To prevent overfitting to the small training set, we may employ standard weight decay, *i.e.*, adding the ℓ_2 norm of all the weights to the softmax loss as a regularizer. Note that this regularizer does not exploit any "geometry" of the training data, therefore it is basically data-independent. On the other hand, in many applications, data from the same class tends to "concentrate" near a sub-manifold. Preserving the manifold structure is also a way of regularizing. An example is the success of unsupervised pre-training [41].

Motivated by this fact, we derive a graph that encodes the similarity of the



(a) Evolution of softmax loss on training and test set



(b) Evolution of classification accuracy on training and test set

FIGURE 4.1: Comparing GraphConnect against weight decay on the MNIST dataset

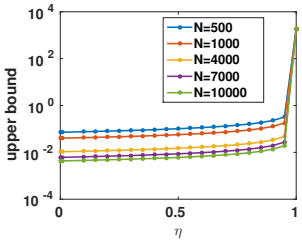


FIGURE 4.2: Evaluation of the upper bound (Theorem 12) on the Rademacher complexity for the MNIST benchmark.

training samples, and use it to regularize the intermediate or final learned features. The graph edge weights are

$$W_{i,j} = \begin{cases} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\right) & \text{if } \mathbf{x}_i, \mathbf{x}_j \in \text{ same class} \\ 0 & \text{otherwise} \end{cases}, \tag{4.1}$$

where $\mathbf{x}_i$ denotes the i-th training sample, and $i, j = 1 \ldots, m$, where m is the number of training samples. The choice of bandwidth σ will be discussed in Section 4.4.1. In particular, denote by $\mathbf{f}_i$ the feature that we want to regularize (corresponding to datum $\mathbf{x}_i$). Then we may add the following term as a regularizer to the softmax loss:

$$J = \sum_{i,j} W_{i,j} \|\mathbf{f}_i - \mathbf{f}_j\|^2. \tag{4.2}$$

This encourages neighboring pairs of data to remain close in the feature space. This general framework is termed *GraphConnect* regularization.

We now apply the *GraphConnect* regularizer to the architecture in Tab. 4.1. In particular, we study regularizing the linear layers (layers 3 and 5 in this case, denoted as *GraphConnect-One*), and regularizing the final learned feature (layer 6, denoted as *GraphConnect-All*). Mini-batch based stochastic gradient descent is adopted to train the network. After each training epoch, we calculate the softmax loss on the 500 training samples and the test set, as illustrated in Fig. 4.1a.

The gap between testing loss and training loss is called the *generalization error*. The bigger the generalization error, the more overfitting the model suffers. In Fig. 4.1a we observe that all the methods have near-zero loss on the training set. However, the two variants of *GraphConnect* have approximately the same generalization error and both are smaller than weight decay. Fig. 4.1b shows that all methods achieve 100% classification accuracy on the training set. However on the test set, *GraphConnect-One* and *GraphConnect-All* are superior to weight decay by a significant margin.

4.3 A Theoretical Perspective

The previous section suggests that *GraphConnect* better prevents overfitting than does weight decay. In this section, we give more theoretical insights into this phenomenon. We consider L-way classification using a deep neural network. Given a datum $\mathbf{x}$ with class label $y \in \{1, \ldots, L\}$, the network first learns a D-dimensional feature $\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}), \ldots, g_d(\mathbf{x}), \ldots, g_D(\mathbf{x})]^\top$, and then applies a softmax classifier to generate a probability distribution over the L classes. Thereafter the softmax loss is calculated as the negative log-probability of class y. Essentially, the entire deep network can be viewed as a function, mapping $(\mathbf{x}, y)$ to the softmax loss $\ell(\mathbf{g}(\mathbf{x}), y)$. Note that $\ell(\cdot, \cdot)$ and $g_d(\cdot)$ are parameterized by the weights in the network. Therefore, we let $\mathcal{L}$ denote a class of $\ell(\cdot, \cdot)$ defined by different network weights, and similarly, $\mathcal{G}$ denotes a class of $g_d(\cdot)$.

The average loss achieved on the training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ is the empirical loss ℓ_{emp}, given by

$$\ell_{emp} = \frac{1}{N} \sum_{i=1}^{N} \ell(\mathbf{g}(\mathbf{x}_i), y_i).$$

The expected loss $\mathbb{E}[\ell]$ is estimated from a large test set and is given by

$$\mathbb{E}[\ell] = \mathbb{E}_{(\mathbf{x},y)}[\ell(\mathbf{g}(\mathbf{x}), y)].$$

The difference $\mathbb{E}[\ell] - \ell_{emp}$ between the expected loss on the test data and the empirical loss on the training data is the *generalization error*. When training samples are scarce, statistical learning theory predicts overfitting to the training data [80]. The larger the generalization error, the more severe is the problem of overfitting. Analyzing the generalization error involves the Rademacher complexity of the function class $\mathcal{L}$ and $\mathcal{G}$.

Definition 4 (Empirical Rademacher Complexity)**.** *Let $\mathcal{D}$ be a probability distribution on a set $\mathcal{A}$ and assume that $\mathbf{a}_1, \ldots, \mathbf{a}_N$ are independent samples from $\mathcal{D}$. Let φ be a class of functions mapping from $\mathcal{A}$ to $\mathbb{R}$. The empirical Rademacher complexity of φ is*

$$\hat{\mathcal{R}}_N(\varphi) = \mathbb{E}_{\sigma_i}\left[\frac{2}{N}\sup_{\phi\in\varphi}\left|\sum_{i=1}^{N}\sigma_i\phi(\mathbf{a}_i)\right| : \mathbf{a}_1, \ldots, \mathbf{a}_N\right],$$

where the σ_i's are independent uniform $\{\pm 1\}$-valued random variables.

Remark 7. *Over-fitting occurs when a statistical model describes random error or noise instead of the underlying signal. We seek to minimize the empirical Rademacher complexity because it measures correlation with random errors. However we need to keep the objective of classification in mind, since we can reduce the empirical Rademacher complexity to zero by simply mapping every datum $\mathbf{x}$ to 0 without any discriminability. Therefore minimization of Rademacher complexity needs to be performed over a class of functions φ that is able to discriminate between classes.*

We recall from [52] that assuming $\ell(\mathbf{g}(\mathbf{x}), y)$ is bounded, the generalization error is with high probability bounded by the empirical Rademacher complexity [8] of function class $\mathcal{L}$, which is no more than the empirical Rademacher complexity of

$\mathcal{G}$ multiplied by some constant related to the softmax classifier [82]. Hence we can reduce the degree of overfitting by controlling the empirical Rademacher complexity of $\mathcal{G}$.

4.3.1 Analysis: Regularizing a Linear Layer

Since each $g_d(\mathbf{x})$ is a composition of linear layers and nonlinear activations, we consider regularizing the linear layers with *GraphConnect* first. Given an input $\mathbf{z}$ to a linear layer, we consider the linear map $f(\mathbf{z}) = \mathbf{v}^\top \mathbf{z}$, where the linear weights $\mathbf{v}$ are from a set $\mathcal{V}$ specified by *GraphConnect* regularization. More formally we consider the function family,

$$\mathcal{F} = \left\{ f(\mathbf{z}) | f(\mathbf{z}) = \mathbf{v}^\top \mathbf{z}, \mathbf{v} \in \mathcal{V} \right\}. \tag{4.3}$$

Suppose now that we have learned a graph where symmetric edge weights $\mathbf{W}$ encode the relationships between N input samples $\mathbf{Z} \stackrel{\text{def}}{=} [\mathbf{z}_1, \ldots, \mathbf{z}_N]$. The set $\mathcal{V}$ that defines the function family $\mathcal{F}$ is given by

$$\left\{ \mathbf{v} : \frac{(1-\eta) \sum_{i,j=1}^{N} W_{i,j} [f(\mathbf{z}_i) - f(\mathbf{z}_j)]^2 + \eta \|\mathbf{v}\|^2}{N} \leq \frac{B^2}{4} \right\}, \tag{4.4}$$

for some positive constant B, and for some $\eta \in (0, 1]$, and $W_{i,j}$ being as before the weights on the graph edges. When $\eta = 1$, this condition enforces conventional weight decay, and as η approaches 0, it enforces graph regularization.

Let $\mathbf{1}$ be the all-one vector, let $\mathbf{D}$ be the diagonal matrix with entries $\mathbf{W1}$, and let $\mathbf{L} = \mathbf{D} - \mathbf{W}$ be the graph Laplacian. It can be shown that

$$\sum_{i,j=1}^{N} W_{i,j} [f(\mathbf{z}_i) - f(\mathbf{z}_j)]^2 = \mathbf{v}^\top (\mathbf{Z}\mathbf{L}\mathbf{Z}^\top) \mathbf{v}, \tag{4.5}$$

Therefore by introducing the identity matrix $\mathbf{I}$, we can describe the set $\mathcal{V}$ very simply as

$$\mathcal{V} = \left\{ \mathbf{v} \,\middle|\, \mathbf{v}^\top \left((1-\eta) \mathbf{Z}\mathbf{L}\mathbf{Z}^\top + \eta \mathbf{I} \right) \mathbf{v} \leq \frac{N B^2}{4} \right\}. \tag{4.6}$$

Since the Laplacian is positive semidefinite and $\eta > 0$, the matrix $(1-\eta)\mathbf{Z}\mathbf{L}\mathbf{Z}^\top + \eta\mathbf{I}$ is positive definite. With these definitions, we now bound $\hat{\mathcal{R}}_N(\mathcal{F})$.

Theorem 12 (Empirical Rademacher Complexity of Regularizing a Linear Layer)**.** *Let $\mathcal{F}$ be the class of linear functions defined in Eq.* (4.3)*, where $\mathcal{V}$ is the set defined in Eq.* (4.6)*, and let $\mathbf{Z} \overset{\text{def}}{=} [\mathbf{z}_1, \ldots, \mathbf{z}_N] \in \mathbb{R}^{n \times N}$ be the sample set on which the Rademacher complexity $\hat{\mathcal{R}}_N(\mathcal{F})$ is evaluated. Then*

$$\hat{\mathcal{R}}_N(\mathcal{F}) \leq B\sqrt{\frac{\operatorname{tr}\left[\mathbf{Z}^\top \left((1-\eta)\mathbf{Z}\mathbf{L}\mathbf{Z}^\top + \eta\mathbf{I}\right)^{-1}\mathbf{Z}\right]}{N}}$$

Proof. Denote $(\mathbf{M}) \overset{\text{def}}{=} (1-\eta)\mathbf{Z}\mathbf{L}\mathbf{Z}^\top + \eta\mathbf{I}_n$ and $\boldsymbol{\sigma} \overset{\text{def}}{=} [\sigma_1, \ldots, \sigma_N]^\top$. Following the definition of empirical Rademacher complexity, we have

$$\begin{aligned}
\hat{\mathcal{R}}_N(\mathcal{F}) &= \mathbb{E}_{\boldsymbol{\sigma}}\left[\frac{2}{N}\sup_{\mathbf{v}\in\mathcal{V}}\left|\sum_{i=1}^N \sigma_i \mathbf{v}^\top \mathbf{z}_i\right| \middle| \mathbf{z}_1, \ldots, \mathbf{z}_N\right] \\
&= \mathbb{E}_{\boldsymbol{\sigma}}\left[\frac{2}{N}\sup_{\mathbf{v}\in\mathcal{V}}\left|\mathbf{v}^\top\left(\sum_{i=1}^N \sigma_i \mathbf{z}_i\right)\right| \middle| \mathbf{z}_1, \ldots, \mathbf{z}_N\right].
\end{aligned}$$

By Cauchy-Schwarz inequality, the right hand side (RHS)

$$\begin{aligned}
&\leq \mathbb{E}_{\boldsymbol{\sigma}}\left[\frac{2}{N}\sup_{\mathbf{v}\in\mathcal{V}}\|(\mathbf{M})^{\frac{1}{2}}\mathbf{v}\| \cdot \left\|(\mathbf{M})^{-\frac{1}{2}}\left(\sum_{i=1}^N \sigma_i \mathbf{z}_i\right)\right\| \middle| \mathbf{z}_1, \ldots, \mathbf{z}_N\right] \\
&= \left(\sup_{\mathbf{v}\in\mathcal{V}}\frac{2}{N}\|(\mathbf{M})^{\frac{1}{2}}\mathbf{v}\|\right) \cdot \mathbb{E}_{\boldsymbol{\sigma}}\left\|(\mathbf{M})^{-\frac{1}{2}}\left(\sum_{i=1}^N \sigma_i \mathbf{z}_i\right)\right\| \\
&\leq \left(\sup_{\mathbf{v}\in\mathcal{V}}\frac{2}{N}\|(\mathbf{M})^{\frac{1}{2}}\mathbf{v}\|\right) \cdot \left[\mathbb{E}_{\boldsymbol{\sigma}}\left\|(\mathbf{M})^{-\frac{1}{2}}\left(\sum_{i=1}^N \sigma_i \mathbf{z}_i\right)\right\|^2\right]^{\frac{1}{2}} \\
&= \left(\sup_{\mathbf{v}\in\mathcal{V}}\frac{2}{N}\|(\mathbf{M})^{\frac{1}{2}}\mathbf{v}\|\right) \cdot \left[\operatorname{tr}\left((\mathbf{M})^{-1}\mathbf{Z}\mathbb{E}[\boldsymbol{\sigma}\boldsymbol{\sigma}^\top]\mathbf{Z}^\top\right)\right]^{\frac{1}{2}} \\
&= \left(\sup_{\mathbf{v}\in\mathcal{V}}\frac{2}{N}\|(\mathbf{M})^{\frac{1}{2}}\mathbf{v}\|\right) \cdot \left[\operatorname{tr}\left((\mathbf{M})^{-1}\mathbf{Z}\mathbf{Z}^\top\right)\right]^{\frac{1}{2}}
\end{aligned}$$

By the constraint that $\mathbf{v}^\top(\mathbf{M})\mathbf{v} \leq \frac{NB^2}{4}$, we have $\frac{2}{N}\|(\mathbf{M})^{\frac{1}{2}}\mathbf{v}\| \leq \sqrt{\frac{1}{N}}B$. Therefore,

$$\hat{\mathcal{R}}_N(\mathcal{F}) \leq B\sqrt{\frac{\text{tr}\left(\mathbf{Z}^\top(\mathbf{M})^{-1}\mathbf{Z}\right)}{N}}.$$

□

In the definition of $\mathcal{V}$ and Theorem 12, we have excluded the value $\eta = 0$ so that the inverse $[(1-\eta)\mathbf{L}\mathbf{Z}\mathbf{Z}^\top + \eta\mathbf{I}]^{-1}$ exists. However, in our experiments (sections 4.2 and 4.5), with no weight decay($\eta = 0$), we have also observed strong generalization performance.

Reducing the Rademacher complexity of a single linear layer is an important step in reducing the Rademacher complexity of a multi-layer network, since adding one more layer just multiplies the empirical Rademacher complexity by some constant related to the new layer [82]. The bound in Theorem 12 depends on the eigenvalues of the Laplacian $\mathbf{L}$, which in turn depends on the edge weights $\mathbf{W}$ of the graph. We now show that the bound becomes lower as η goes to 0, suggesting *GraphConnect* is a more effective regularizer.

Example. Consider the MNIST dataset. There are 10 classes in the dataset, and samples are 784-dimensional (28×28 images). We randomly select N samples (N/10 samples per class), remove the sample mean, and form the $784 \times N$ matrix $\mathbf{Z}$. The edge weights $\mathbf{W}$ are given by $W_{i,j} = \exp\left(-\frac{\|\mathbf{z}_i - \mathbf{z}_j\|^2}{\gamma^2}\right)$, where γ is the average of all pairwise distances. We form the Laplacian $\mathbf{L}$ and evaluate the bound given in Theorem 12 for several values of N. We observe in Fig. 4.2 that the upper bound decreases significantly as η moves away from 1 (weight decay), showing the added value of graph regularization. As the sample size N increases, the bound decreases steadily, consistent with our intuition about the generalization error.

4.3.2 Analysis: Regularizing Multiple Layers

We now extend our analysis to include the effects in intermediate layers of pooling and of activation functions such as rectifiers. We consider a K-layer network that maps an input $\mathbf{x}$ onto a multidimensional feature $\mathbf{g}(\mathbf{x})$, and then restricts to a single dimension to obtain a scalar $g(\mathbf{x})$ given by

$$g(\mathbf{x}) = \mathbf{v}_K^\top s_{K-1}(\cdots s_2(\mathbf{V}_2 s_1(\mathbf{V}_1 \mathbf{x}))), \text{ where } \mathbf{v}_K, \mathbf{V}_i \in \mathscr{V}, \tag{4.7}$$

where the nonlinear mapping $s_k(\cdot)$ represents activation and pooling. $\mathbf{V}_1, \ldots, \mathbf{V}_{K-1}$ are matrices representing linear layers, and $\mathbf{v}_k$ is a vector that maps the input to a single coordinate in the final output feature. The $\mathbf{V}_1, \ldots, \mathbf{V}_{K-1}$ and $\mathbf{v}_K$ are taken from a set $\mathscr{V}$ that is defined by the property

$$\frac{1}{N} \sum_{i,j=1}^{N} W_{i,j}[g(\mathbf{x}_i) - g(\mathbf{x}_j)]^2 \leq \frac{B^2}{4}. \tag{4.8}$$

As before, the symmetric edge weights $\mathbf{W}$ encode the relationships between the N input samples samples $[\mathbf{x}_1, \ldots, \mathbf{x}_N]$.

Set $g(\mathbf{X}) = [g(\mathbf{x}_1), \ldots, g(\mathbf{x}_N)]^\top$ and recall that

$$\sum_{i,j=1}^{N} W_{i,j}[g(\mathbf{x}_i) - g(\mathbf{x}_j)]^2 = g(\mathbf{X}) \mathbf{L} g(\mathbf{X})^\top,$$

where $\mathbf{L}$ is as before the Lapacian of $\mathbf{W}$. As before, we want to work with a positive definite matrix so we add a small multiple of the identity matrix $\mathbf{I}_n$. We now derive an upper bound on the empirical Rademacher complexity for the function class $\mathcal{G}$ defined by

$$\mathcal{G} = \left\{ g(\mathbf{x}) | g(\mathbf{x}) = \mathbf{v}_K^\top s(\cdots s(\mathbf{V}_2 s(\mathbf{V}_1 \mathbf{x}))) \}, \text{ where } g(\mathbf{X})(\mathbf{L} + \epsilon \mathbf{I}) g(\mathbf{X})^\top \leq \frac{NB^2}{4} \right\}. \tag{4.9}$$

Theorem 13. $\hat{\mathcal{R}}_N(\mathcal{G}) \leq B\sqrt{\frac{\text{tr}[(\mathbf{L}+\epsilon\mathbf{I}_n)^{-1}]}{N}}$

Proof. Denote $\boldsymbol{\sigma} = [\sigma_1, \ldots, \sigma_N]^\top$, where σ_i's are i.i.d. uniformly distributed in $\{\pm 1\}$. We have

$$\begin{aligned}
\hat{\mathcal{R}}_N(\mathcal{G}) &= \mathbb{E}_{\sigma_i}\left[\sup_{\mathbf{v}_K, \mathbf{V}_i} \frac{2}{N} \left|g(\mathbf{X})\boldsymbol{\sigma}\right|\right] \\
&= \mathbb{E}_{\sigma_i}\left[\sup_{\mathbf{v}_K, \mathbf{V}_i} \frac{2}{N} \left|g(\mathbf{X})(\mathbf{L}+\epsilon\mathbf{I})^{\frac{1}{2}}(\mathbf{L}+\epsilon\mathbf{I})^{-\frac{1}{2}}\boldsymbol{\sigma}\right|\right] \\
&\leq \sup_{\mathbf{v}_K, \mathbf{V}_i} \frac{2}{N} \left\|(\mathbf{L}+\epsilon\mathbf{I})^{\frac{1}{2}} g(\mathbf{X})^\top\right\| \cdot \mathbb{E}_{\sigma_i}\left[\left\|(\mathbf{L}+\epsilon\mathbf{I})^{-\frac{1}{2}}\boldsymbol{\sigma}\right\|\right] \\
&\leq \frac{B}{\sqrt{N}} \cdot \left(\mathbb{E}_{\sigma_i}\left[\left\|(\mathbf{L}+\epsilon\mathbf{I})^{-\frac{1}{2}}\boldsymbol{\sigma}\right\|^2\right]\right)^{\frac{1}{2}} \\
&= B\sqrt{\frac{\text{tr}\left[(\mathbf{L}+\epsilon\mathbf{I})^{-1}\right]}{N}}
\end{aligned}$$

□

Remark 8. *Theorem 13 provides an upper bound on complexity that is not very sensitive to the number of layers in the network, in contrast to weight decay where complexity is exponential in the number of layers [92].*

To conclude, we have proposed bounds on the empirical Rademacher complexity of linear layers and the entire network, suggesting that *GraphConnect* may lead to smaller complexity than weight decay. The implication is that, *GraphConnect* may prevent overfitting better than weight decay.

4.4 Algorithmic Details

Following the theoretical analysis in the last section, we now describe two flavors of *GraphConnect* in Fig. 4.3, two different ways of using a graph to regularize a neural network. The first, *GraphConnect-One*, uses the graph to regularize individual layers,

and this method can be applied to all layers or to some layers but not others. The second, *GraphConnect-All*, uses the graph to regularize the final learned features. Implementation requires multiplying the *GraphConnect* regularizer by some $\lambda > 0$, then adding this quantity to the original objective function used to train the neural network. We show that both *GraphConnect* regularization schemes require only minor changes to the standard back-propagation algorithm.

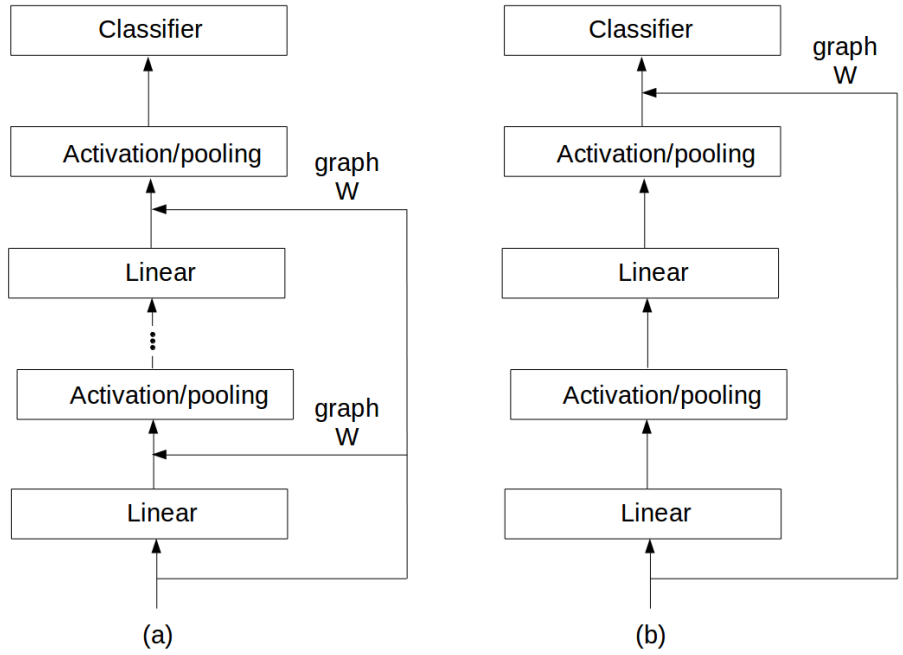


FIGURE 4.3: (a) *GraphConnect-One* regularizes individual linear layers so that individual outputs align with a graph $\mathbf{W}$; (b) *GraphConnect-All* regularizes the final output features to align with a graph $\mathbf{W}$.

Gradient Descent Solver for *GraphConnect-One*. Without loss of generality, we assume that the regularizer is imposed on the k-th linear layer with weights $\mathbf{V}_k$ and layer input $\mathbf{Z}^{(k-1)} = [\mathbf{z}_1^{(k-1)}, \ldots, \mathbf{z}_N^{(k-1)}]$. We seek to minimize

$$\ell_{emp} + \lambda J,$$

where J is the *GraphConnect* regularization on the linear transform given by

$$J = \sum_{i,j=1}^{N} W_{i,j} \|\mathbf{V}_k \mathbf{z}_i^{(k-1)} - \mathbf{V}_k \mathbf{z}_j^{(k-1)}\|^2$$

$$= \mathrm{tr}[\mathbf{V}_k \mathbf{Z}^{(k-1)} \mathbf{L} (\mathbf{V}_k \mathbf{Z}^{(k-1)})^\top].$$

The gradient of J w.r.t. $\mathbf{V}_k$ is

$$\frac{\partial J}{\partial \mathbf{V}_k} = 2\mathbf{V}_k \mathbf{Z}^{(k-1)} \mathbf{L} \mathbf{Z}^{(k-1)\top}. \tag{4.10}$$

The optimization then runs as normal back propagation, except that the gradient of $\mathbf{V}_k$ takes an additional term, Eq. (4.10).

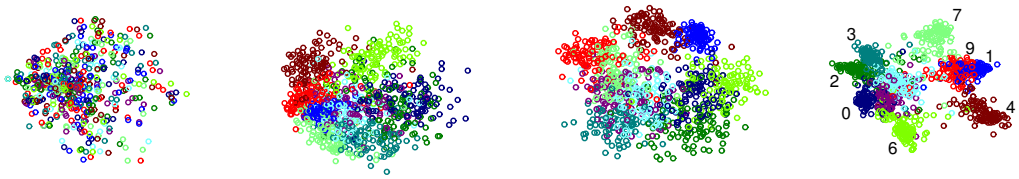
Gradient Descent Solver for *GraphConnect-All*. The objective function now takes an extra term $J = \lambda \,\mathrm{tr}(\mathbf{g}(\mathbf{X})\mathbf{L}\mathbf{g}(\mathbf{X})^\top)$. The gradient of J w.r.t. $\mathbf{g}(\mathbf{X})$ is

$$\frac{\partial J}{\partial \mathbf{g}(\mathbf{X})} = 2\lambda \mathbf{g}(\mathbf{X})\mathbf{L}.$$

So we just need to add an extra term $2\lambda\mathbf{g}(\mathbf{X})\mathbf{L}$ to the original gradient with respect to $\mathbf{g}(\mathbf{X})$. Then the gradient is back propagated as usual. *GraphConnect* regularization requires only minor modifications to standard back propagation algorithms and is very efficient in practice.

4.4.1 Choice of Bandwidth σ

Choice of the bandwidth in the matrix $\mathbf{W}$ is an open question. The graph matrix $\mathbf{W}$ implicitly defines a map, $\psi(\mathbf{x})$, from the data to a feature space, where $\psi(\mathbf{x}_i)^\top \psi(\mathbf{x}_j) = W_{i,j}$. In the extreme, when σ approaches zero, $W_{i,j} \to 0$, $\|\psi(\mathbf{x}_i) - \psi(\mathbf{x}_j)\|^2 \to 2$, meaning that all points are equally separated and no local geometrical information is preserved. In the extreme when σ approaches infinity, $\|\psi(\mathbf{x}_i) - \psi(\mathbf{x}_j)\|^2 = 2 - 2\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2) \to \frac{2}{\sigma^2}\|\mathbf{x}_i - \mathbf{x}_j\|^2$, and the feature space preserves all the



(a) 1000 test samples from MNIST

(b) Transformed test samples when the learned network is regularized by weight decay

(c) Transformed test samples when the learned network is regularized by *GraphConnect-One*

(d) Transformed test samples when the learned network is regularized by *GraphConnect-All*

FIGURE 4.4: Embedding of initial and transformed test samples with different colors representing different classes. All networks are learned from the same set of 500 training samples. In (d), we observe that numbers with curly strokes are clustered on the left, whereas those with straight strokes are on the right.

pairwise distances (up to a scale $\frac{2}{\sigma^2}$). However, since $\frac{2}{\sigma^2}$ approaches zero, the difference between points vanishes. The above reasoning suggests an intermediate choice of σ.

Essentially, the regularizer J is encouraging the learned features to "resemble" the implicitly defined feature space. In this work we use kernel alignment [25] to compute the bandwidth of the RBF kernel. Kernel alignment requires the computed RBF to resemble the pairwise label matrix. For further separation, after σ is computed, we enforce $W_{i,j} = 0$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are from different classes.

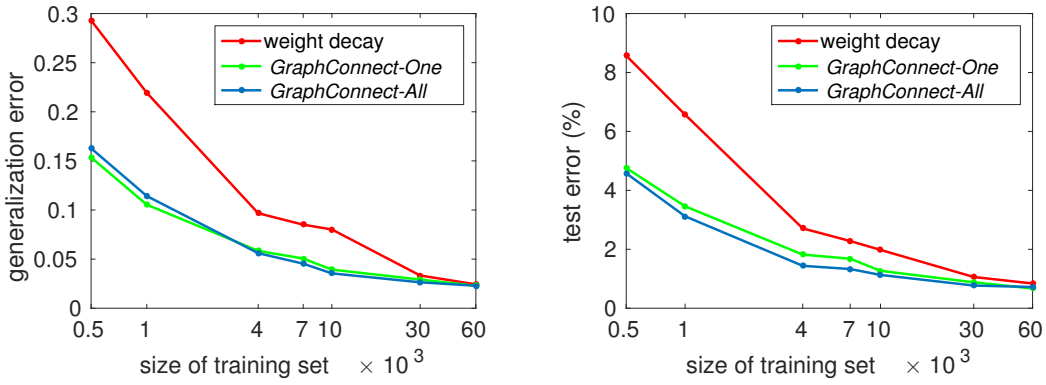
4.5 Experiments

In this section, we revisit the MNIST example in Section 4.2 to confirm the effectiveness of *GraphConnect*. Then, experiments on more challenging datasets are carried out to compare *GraphConnect* with weight decay and *DropOut*.

4.5.1 MNIST Revisited

We vary the size of the training set from 500 to 6,000, and repeat the experiment in Section 4.2. All hyper-parameters are optimized so that those reported are the best performance of each method. When the number of training samples is small, Fig.

4.5a shows *GraphConnect* yields a generalization error that is significantly smaller than that yielded by weight decay. Performance becomes broadly similar as the size of the training set increases. The same trend is evident in Fig. 4.5b, which compares the classification accuracy of the three methods.



(a) Dependence of generalization error on the size of the training set (after 100 iterations).

(b) Dependence of classification error on the size of the training set (after 100 iterations).

FIGURE 4.5: Comparing *GraphConnect* against weight decay on the MNIST dataset.

Fig. 4.4 shows the embedding of the learned features for 1,000 testing samples (network trained with 500 samples), the two variants of *GraphConnect* both yield more discriminative features than weight decay. Since performance of the two variants of *GraphConnect* is broadly similar, and since *GraphConnect-One* involves tuning multiple regularizers, we focus on *GraphConnect-All* in the sequel.

4.5.2 Comparison on SVHN and CIFAR-10

SVHN and CIFAR-10 are benchmark RGB image datasets, each containing 10 classes, that are more challenging than the MNIST benchmark because of more significant intra-class variation. On these two datasets, we compare different regularization techniques (*GraphConnect*, weight decay and *DropOut*). Table 4.2 specifies the network architecture (similar to [42]). When enabled, the *DropOut* layer is with a drop rate of 0.5, and applied to layers 11 and 13 as in table 4.2. All images are mean-subtracted

in a preprocessing step, and the graph weights $\mathbf{W}$ used in *GraphConnect-All* are computed in the same fashion as for the MNIST experiment. The network transforms sample images into 2,048-dimensional features that are input to a softmax classifier.

First, we compare each individual regularizer on SVHN, *i.e.*, *DropOut*, weight decay and *GraphConnect*. The number of training samples is varied from 50 to 700 per class, and the testing accuracies of all methods are evaluated as in Tab. 4.3. *GraphConnect* outperforms weight decay in all cases and is competitive with *DropOut*.

Since *DropOut* is widely adopted, it is important for a regularizer to be compatible with it. Next, we look at the compatibility of *GraphConnect* with *DropOut*. We apply weight decay and *DropOut* respectively to the network (Tab. 4.2) with the dropout layers enabled or disabled. The experiment is conducted on CIFAR-10 with 50 to 700 training samples per class, and testing accuracies are reported in Tab. 4.4. We observe that the performance is boosted when *DropOut* is adopted. However, either with or without *DropOut*, *GraphConnect* outperforms weight decay significantly.

4.5.3 Face Verification on LFW

We now evaluate *GraphConnect* on face verification, using the Labeled Faces in the Wild (LFW) benchmark dataset. The face verification task is to decide, when presented with a pair of facial images, whether the two images represent the same subject. Impressive verification accuracies are possible when deep neural networks are able to train on extremely large labeled training sets [75, 77]. The training sets are often proprietary, making it difficult to reproduce these successes, but that is not our aim in this work. Given the same network architecture, we seek to compare the performance of different regularizers.

We adopt the experimental framework used in [18], and train a deep network on the WDRef dataset, where each face is described using a high dimensional LBP

Table 4.2: Network common to SVHN and CIFAR-10 experiments.

Layer	Type	Parameters
1	conv	size: $5 \times 5 \times 3 \times 96$ stride: 1, pad: 2
2	ReLu	N/A
3	maxPool	size: 3×3, stride: 2, pad: 0
4	conv	size: $5 \times 5 \times 96 \times 128$ stride: 1, pad: 2
5	ReLu	N/A
6	maxPool	size: 3×3, stride: 2, pad: 0
7	conv	size: $4 \times 4 \times 50 \times 500$ stride: 1, pad: 0
8	ReLu	N/A
9	maxPool	size: 3×3, stride: 2, pad: 0
10	fully connected	#output: 2048
11	ReLu (w/ dropout)	N/A
12	fully connected	#output: 2048
13	ReLu (w/ dropout)	N/A

Table 4.3: SVHN: test accuracy when individual regularizer is used.

training per class	weight decay	*DropOut*	*GraphConnect*
50	72.58%	**73.54%**	73.13%
100	78.45%	**79.39%**	**79.39%**
400	87.20%	87.39%	**87.66%**
700	89.84%	89.77%	**90.13%**

Table 4.4: CIFAR-10: test accuracy as size of the training set varies.

	without *DropOut*		with *DropOut*	
training per class	weight decay	*GraphConnect*	weight decay	*GraphConnect*
50	34.78%	**38.65%**	37.31%	**40.47%**
100	42.17%	**45.07%**	44.99%	**46.11%**
400	56.14%	**57.39%**	58.81%	**59.48%**
700	61.83%	**62.17%**	64.03%	**64.26%**

feature (available at http://home.ustc.edu.cn/chendong/) that is reduced to a 5,000-dimensional feature using PCA. The WDRef dataset is significantly smaller than the

proprietary datasets in [75, 76, 77]. For example, [77] uses 4.4 million labeled faces from 4,030 individuals. [75] and [76] use 202,599 labeled faces from 10,177 individuals, while WDRef contains 2,995 subjects with only about 30 samples per subject, clearly a much more challenging task. We consider the two-layer fully connected net-

Table 4.5: Fully connected network for face verification.

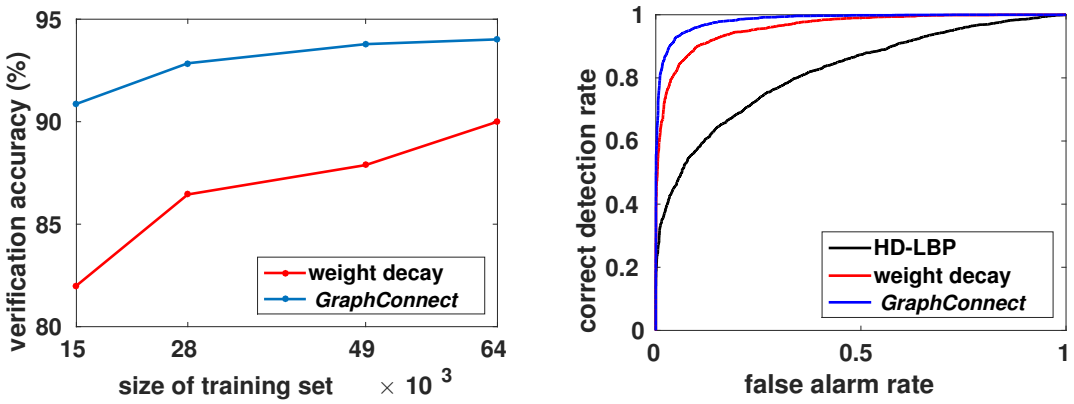
Layer	Type	Parameters
1	fully connected	#output: 2000
2	ReLu	N/A
3	fully connected	#output: 2000
4	ReLu	N/A

work described in Tab. 4.5, where the activation function is a rectifier. The network transforms a 5,000-dimensional input vector to a 2,000-dimensional feature vector, which is then input to a softmax classifier. The network parameters are learned using WDRef and the testing is carried out on the LFW dataset. Our focus is the expressiveness of the learned feature, so we do not employ advanced verification methods such as those used in [18] (those will make the study of the network itself very obscure). Instead, we simply compute the Euclidean distance between a pair of (learned) face features and compare it with a threshold to make a decision.

Table 4.6: Verification accuracies and AUCs when using a training set of size 64,000

Method	Accuracy (%)	AUC ($\times 10^{-2}$)
HD-LBP	74.73	82.22 ± 1.00
weight decay	90.00	96.14 ± 0.61
GraphConnect	**94.02**	$\mathbf{98.48 \pm 0.21}$

We vary the number of training samples per class and evaluate verification performance. We report results for the value of the regularization parameter that optimizes verification accuracy. Fig. 4.6a compares verification accuracies for *GraphConnect*



(a) (b)

FIGURE 4.6: (a) Verification accuracy of GraphConnect and weight decay as a function of the size of the training set; (b) ROC curves when using 64,000 training samples.

and weight decay as a function of the size of the training set. *GraphConnect* consistently outperforms weight decay. We also tried to apply *DropOut* to layers 2 and 4, but did not observe any gain. For example, when 15K training samples are used, weight decay+*DropOut* only gives a verification accuracy of 78.00% (significantly lower than the accuracy of weight decay without *DropOut*, 82.00%). This may due to the fact that the number of training samples here is too small such that the network overfits even with the noise introduced by *DropOut*. This phenomenon has been noticed in [42]. Therefore we leave out the experiment with *DropOut* here.

Fig. 4.6b compares the ROCs curves when a training set of size 64K is used. Corresponding Area Under Curves (AUCs) are reported in Tab. 4.6. As a baseline, we also evaluate the verification performance on the initial LBP features (without any learning). We observe from Fig. 4.6b and Tab. 4.6 that the learned features significantly outperform the initial LBP features, while *GraphConnect* further improves upon weight decay, validating the effectiveness of *GraphConnect* regularization when training set is small.

4.6 Conclusion

We have proposed *GraphConnect*, a data-dependent framework for regularizing deep neural networks, and we have compared performance against data-independent methods of regularization that are in widespread use. We observed that *GraphConnect* is superior to the classical weight decay and provided theoretical justification using the empirical Rademacher Complexity. We presented experimental results that validate our theoretical claims, showing that when the training set is small the improvement over weight decay is significant.

5

Connecting Subspace and Manifold

In the previous chapters, we have exploited the subspace and manifold model in various feature extraction tasks. In this chapter, we connect these two models by using affine subspaces to approximate a nonlinear manifold. More specifically, we assume streaming data lies near a low dimensional manifold evolving in a high dimensional ambient space. We learn the manifold structure by tracking a collection of affine subspaces that approximate the manifold. The proposed method can accommodate missing data, and has an efficient "online" updating scheme. Deviation (of each datum) from the learned model is estimated, yielding a series of statistics for anomaly detection. The proposed approach leverages several recent results in the field of high-dimensional data analysis, including subspace tracking with missing data, multiscale analysis techniques for point clouds, online optimization, and changepoint detection performance analysis. Simulations and experiments highlight the robustness and efficacy of the proposed approach in detecting an abrupt change in an otherwise slowly varying low-dimensional manifold.

5.1 Motivating Application: Changepoint Detection

Changepoint detection is a form of anomaly detection where the anomalies of interest are abrupt temporal changes in a stochastic process. In many applications, the stochastic process is non-stationary away from the changepoints and very high dimensional, resulting in significant statistical and computational challenges. For instance, we may wish to quickly identify changes in network traffic patterns [53], a timeseries of spectral images, social network interactions [65], surveillance video [54], or solar flare imagery collected by solar observatories [50, 64]. Classical methods for changepoint detection date back to the 1950s and are closely coupled with anomaly detection [10, 62]. A good changepoint detection algorithm will accept a sequence of random variables whose distribution may change abruptly at one time, detect such a change as soon as possible, and also have long period between false detections.

Traditional changepoint detection methods typically deal with a sequence of low-dimensional, often scalar, random variables. Naively applying these approaches to high-dimensional data is impractical because the underlying high-dimensional distribution cannot be accurately estimated and used for developing test statistics. This results in detection delays and false alarm rates that scale poorly with the dimensionality of the problem. Thus the primary challenge here is to develop a rigorous method for extracting meaningful low-dimensional statistics from the high-dimensional data stream without making restrictive modeling assumptions.

Our method addresses these challenges by using multiscale online manifold learning to extract univariate changepoint detection test statistics from high-dimensional data. We model the dynamic distribution underlying the data as lying close to a time-varying, low-dimensional submanifold embedded within the ambient observation space. This submanifold model, while non-parametric, allows us to generate meaningful test statistics for robust and reliable changepoint detection, and the mul-

tiscale structure allows for fast, memory-efficient computations. Furthermore, these statistics can be calculated even when elements are missing from the observation vector. The approach described in this chapter leverages several recent results in the field of high-dimensional data analysis, including subspace tracking with missing data, multiscale analysis techniques for point clouds, and online optimization.

While manifold learning has received significant attention in the machine learning literature [79, 70, 12, 14], online learning of a dynamic manifold remains a significant challenge, both algorithmically and statistically. Most existing methods are "batch", in that they are designed to process a collection of independent observations all lying near the same static submanifold, and all data is available for processing simultaneously.

In contrast, our interest lies with "online" algorithms, which accept streaming data and sequentially update an estimate of the underlying dynamic submanifold structure, and changepoint detection methods which identify significant changes in the submanifold structure rapidly and reliably. Recent progress towards this direction for a very special case of submanifolds appears in the context of subspace tracking. For example, the Grassmannian Rank-One Update Subspace Estimation (GROUSE) [5] and Parallel Estimation and Tracking by REcursive Least Squares (PETRELS) [22] effectively track a single subspace using incomplete data vectors. The subspace model used in these methods, however, provides a poor fit to data sampled from a manifold with non-negligible curvature.

5.2 Problem Formulation

Suppose we are given a sequence of data $\mathbf{x}_1, \mathbf{x}_2, \ldots,$, and for $t = 1, 2, \ldots$, $\mathbf{x}_t \in \mathbb{R}^D$, where D denotes the *ambient dimension*. The data are noisy measurements of points

lying on a submanifold $\mathbf{v}_t \in \mathcal{M}_t$:

$$\mathbf{x}_t = \mathbf{v}_t + \mathbf{w}_t. \tag{5.1}$$

The *intrinsic dimension* of the submanifold $\mathcal{M}_t$ is d. We assume $d \ll D$. The noise $\mathbf{w}_t$ is a zero mean white Gaussian random vector with covariance matrix $\sigma^2\mathbf{I}$. The underlying submanifold $\mathcal{M}_t$ may vary slowly with time. At each time t, we only observe a partial vector $\mathbf{x}_t$ at locations $\mathbf{\Omega}_t \in \{1, \ldots, D\}$. Let $\mathcal{P}_{\mathbf{\Omega}_t}$ represent the $|\mathbf{\Omega}_t| \times D$ matrix that selects the axes of $\mathbb{R}^D$ indexed by $\mathbf{\Omega}_t$; we observe $\mathcal{P}_{\mathbf{\Omega}_t}\mathbf{x}_t$.

Our goal is to design an online algorithm that generates a sequence of approximations $\widehat{\mathcal{M}}_t$ which tracks $\mathcal{M}_t$ when it varies slowly, and detects anomalies as soon as possible when the submanifold changes abruptly. The premise is that the statistical properties of the tracking error will be different when the submanifold varies slowly versus when it changes abruptly. In the following, we will present a new online submanifold learning algorithm that can effectively generate a sequence of tracking errors which are stationary when the submanifold is changing slowly, and then develop a changepoint detection method using the errors.

Define the operator

$$\mathcal{P}_{\mathcal{M}}\mathbf{x}_t = \arg\min_{\mathbf{x}\in\mathcal{M}} \|\mathbf{x} - \mathbf{x}_t\|^2 \tag{5.2}$$

as the projection of observation $\mathbf{x}_t$ on to $\mathcal{M}$. If we had access to all the data simultaneously without any memory constraints, we might solve the following batch optimization problem using all data up to time t for an approximation:

$$\widehat{\mathcal{M}}_t^{\circ} \triangleq \arg\min_{\mathcal{M}} \Big\{ \sum_{i=1}^{t} \alpha^{t-i} \|\mathcal{P}_{\mathbf{\Omega}_i}(\mathbf{x}_i - \mathcal{P}_{\mathcal{M}}\mathbf{x}_i)\|^2 + \mu \mathrm{pen}(\mathcal{M}) \Big\}, \tag{5.3}$$

where $\|\mathbf{x}\|$ denotes the Euclidean norm of a vector $\mathbf{x}$, $\mathrm{pen}(\mathcal{M})$ denotes a regularization term which penalizes the complexity of $\mathcal{M}$, $\alpha \in (0, 1]$ is a discounting factor on the

approximation error at each time t, and μ is a user-determined constant that specifies the relative weights of the data fit and regularization terms.

Note that (5.3) cannot be solved without retaining all previous data in memory, which is impractical for the applications of interest. To address this, we instead consider an approximation to the cost function in (5.3) of the form $F(\mathcal{M}) + \mu \text{pen}(\mathcal{M})$. To develop an online algorithm, instead of solving (5.3), we find a sequence of approximations $\widehat{\mathcal{M}}_1, \ldots, \widehat{\mathcal{M}}_t$ (without storing historic data), such that $\widehat{\mathcal{M}}_{t+1}$ is computed using $F(\mathcal{M})$ by updating the previous approximation $\widehat{\mathcal{M}}_t$ using the current datum $\mathbf{x}_{t+1}$. In Section 5.3, we will present several forms of $F(\mathcal{M})$ that lead to recursive updates and efficient tracking algorithms. One example of an approximation is illustrated in Figure 5.1; the context is described in more detail in Section 5.6.2.

Given the sequence of submanifold estimates $\widehat{\mathcal{M}}_1, \ldots, \widehat{\mathcal{M}}_t$, we can compute the distance of each $\mathbf{x}_t$ to $\widehat{\mathcal{M}}_t$, which we denote $\{e_t\}$. We then apply changepoint detection methods to the sequence of tracking errors $\{e_t\}$. In particular, we assume that when there is no anomaly, e_t are i.i.d. with distribution ν_0. When there is an anomaly, there exists an unknown time $\kappa < t$ such that before the changepoint $e_1, \ldots, e_\kappa$ are i.i.d. with distribution ν_0, and after the changepoint, $e_{\kappa+1}, \ldots$ are i.i.d. with distribution ν_1. Our goal is to detect the anomaly as quickly as possible after it occurs, and make as few false alarms as possible.

5.3 Multiscale Online Union of Subspace Estimation (MOUSSE)

In the following we describe the Multiscale Online Union of SubSpaces Estimation (MOUSSE) method, including the underlying multiscale model and online update approaches.

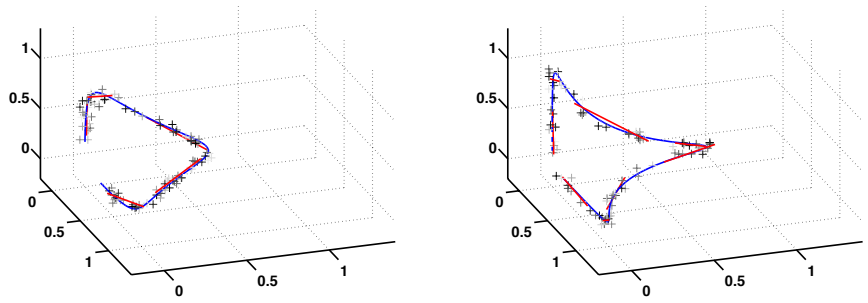


FIGURE 5.1: Approximation of MOUSSE at $t = 250$ (upper) and $t = 1150$ (lower) of a 100-dimensional submanifold. In this figure we project everything into three-dimensional space. The blue curve corresponds to true submanifold, the dots are noisy samples from the submanifold (the lighter dots are more dated than the darker dots), and the red line segments are the approximation with MOUSSE. As the curvature of the submanifold increases, MOUSSE also adapts in the number of line segments.

5.3.1 Multiscale union of subspaces model

MOUSSE uses a union of low-dimensional subsets $\widehat{\mathcal{M}}_t$ to approximate $\mathcal{M}_t$, and organizes these subsets using a tree structure. The idea for a multiscale tree structure is drawn from the multiscale harmonic analysis literature [27]. The leaves of the tree are subsets that are currently used for approximation. Each node in the tree represents a local approximation to the submanifold at one scale. The parent nodes are subspaces that contain coarser approximations to the submanifold than their children. The subset of the parent node roughly covers the subsets of its two children.

More specifically, our approximation at each time t consists of a union of subspaces $\mathcal{S}_{j,k,t}$ that is organized using a tree structure. Here $j \in \{1, \ldots, J_t\}$ denotes the scale or level of the subset in the tree, where J_t is the tree depth at time t, and $k \in \{1, \ldots, 2^j\}$ denotes the index of the subset for that level. The approximation $\widehat{\mathcal{M}}_t$ at time t is given by:

$$\widehat{\mathcal{M}}_t = \bigcup_{(j,k) \in \mathcal{A}_t} \mathcal{S}_{j,k,t},$$

where $\mathcal{A}_t$ contains the indices of all leaf nodes used for approximation at time t. Also define $\mathcal{T}_t$ to be the set of indices of all nodes in the tree at time t, with

$$\mathcal{A}_t \subset \mathcal{T}_t.$$

Each of these subsets lies on a low-dimensional hyperplane with dimension d and is parameterized as

$$\mathcal{S}_{j,k,t} = \{\mathbf{v} \in \mathbb{R}^D : \mathbf{v} = \mathbf{U}_{j,k,t}\mathbf{z} + \mathbf{c}_{j,k,t}(\mathbf{z}^\top \Lambda_{j,k,t}^{-1}\mathbf{z}) \leq 1, \quad \mathbf{z} \in \mathbb{R}^d\}. \tag{5.4}$$

The matrix $\mathbf{U}_{j,k,t} \in \mathbb{R}^{D\times d}$ is the subspace basis, and $\mathbf{c}_{j,k,t} \in \mathbb{R}^D$ is the offset of the hyperplane from the origin. The diagonal matrix

$$\mathbf{\Lambda}_{j,k,t} \triangleq \text{diag}\{\lambda_{j,k,t}^{(1)}, \ldots, \lambda_{j,k,t}^{(d)}\} \in \mathbb{R}^{d\times d},$$

with $\lambda_{j,k,t}^{(1)} \geq \ldots \geq \lambda_{j,k,t}^{(d)} \geq 0$, contains eigenvalues of the covariance matrix of the projected data onto each subspace. This parameter specifies the shape of the ellipsoid by capturing the spread of the data within the subset. In summary, the parameters for $\mathcal{S}_{j,k,t}$ are

$$\{\mathbf{U}_{j,k,t}, \mathbf{c}_{j,k,t}, \mathbf{\Lambda}_{j,k,t}\}_{(j,k)\in\mathcal{T}_t},$$

and these parameters will be updated online.

In our tree structure, the leaf nodes of the tree also have two *virtual* children nodes that keep necessary information for when further partitioning is needed. The complexity of the approximation is defined to be the total number of subsets used for approximation at time t:

$$K_t \triangleq |\mathcal{A}_t|, \tag{5.5}$$

which is used as the complexity regularization term in (5.3)

$$\text{pen}(\widehat{\mathcal{M}}_t) \triangleq \log(K_t). \tag{5.6}$$

The tree structure is illustrated in Figure 5.2.

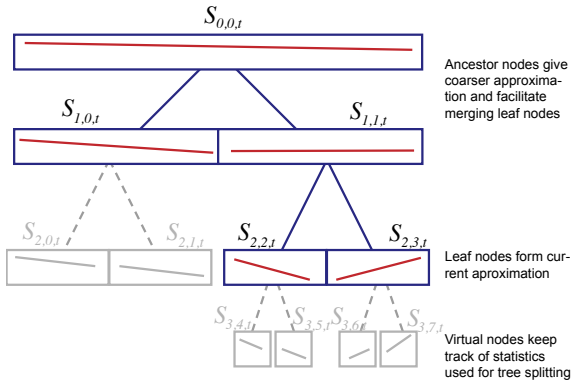


FIGURE 5.2: Illustration of tree structure for subspaces. The subspaces used in our approximation are $\{\mathcal{S}_{1,0,t} \cup \mathcal{S}_{2,2,t} \cup \mathcal{S}_{2,3,t}\}$.

5.3.2 MOUSSE Algorithm

When a new sample $\mathbf{x}_{t+1}$ becomes available, MOUSSE updates $\widehat{\mathcal{M}}_t$ to obtain $\widehat{\mathcal{M}}_{t+1}$. The update steps are presented in Algorithm 3; there are three main steps, detailed in the below subsections: (a) find the subset in the $\widehat{\mathcal{M}}_t$ which is closest to $\mathbf{x}_{t+1}$, (b) update a tracking estimate of that closest subset and its ancestors, and (c) grow or prune the tree structure to preserve a balance between fit to data and complexity. We use $[\mathbf{z}]_m$ to denote the m-th element of a vector $\mathbf{z}$.

5.3.3 Distances for MOUSSE

To update the submanifold approximation, we first determine the affinity of $\mathbf{x}_{t+1}$ to each subset. We may use Euclidean distance, though this distance is problematic since in our approximation each subspace is local with boundary defined by an ellipsoid. Hence, a point can be close to a hyperplane but far away from the center of the ellipsoid. An alternative choice is Mahalanobis distance, though this distance does not reflect the notion of low-dimensional subspaces since it does not depend on $\mathbf{U}_{j,k,t}$. Moreover, evaluating the Mahalanobis distance requires storing the entire covariance matrix and computing its inverse, which is computationally expensive for high-dimensional data.

Algorithm 3 MOUSSE

Input:
error tolerance ϵ, step size α, relative weight μ
Initialize tree structure, set $\epsilon_0 = 0$
for $t = 0, 1, \ldots$ **do**
  Given new data $\mathbf{x}_{t+1}$ and its support $\boldsymbol{\Omega}_{t+1}$, find the minimum distance set $\mathcal{S}_{j^*,k^*,t}$ according to $(j^*, k^*) = \arg\min_{(j,k)\in\mathcal{A}_t} \rho_{\delta_{j,k,t}}(\mathbf{x}_{t+1}, \mathcal{S}_{j,k,t})$ using (5.12)
  Update all ancestor nodes and closest virtual child node of (j^*, k^*) using Algorithm 4
  Calculate:
  $e_{t+1} = \rho_{\delta_{j^*,k^*,t}}(\mathbf{x}_{t+1}, \mathcal{S}_{j^*,k^*,t})$ using (5.12)
  $\epsilon_{t+1} = \alpha\epsilon_t + (1-\alpha)e_{t+1}$
  Denote parent node of (j^*, k^*) as $(j^* - 1, k_p)$ and virtual child node closest to $\mathbf{x}_{t+1}$ as $(j^* + 1, k_v)$
  if $\epsilon_{t+1} > \epsilon$ and
  $d(\mathbf{x}_{t+1}, \mathcal{S}_{j^*+1,k_v,t}) + \mu\log(K_t + 1) < e_{t+1} + \mu\log(K_t)$ **then**
    Split (j^*, k^*) using Algorithm 5, recalculate e_{t+1} and ϵ_{t+1}
  end if
  if $\epsilon_{t+1} < \epsilon$ and
  $d(\mathbf{x}_{t+1}, \mathcal{S}_{j^*-1,k_p,t}) + \mu\log(K_t - 1) < e_{t+1} + \mu\log(K_t)$ **then**
    Merge (j^*, k^*) and its sibling using Algorithm 6, recalculate e_{t+1} and ϵ_{t+1}
  end if
  Update $\mathcal{A}_t$ and $\mathcal{T}_t$
end for

Algorithm 4 Update node

Input: node index (j, k), α and subspace parameters
Calculate: β and $\beta_\perp$ using (5.8) and (5.9)
Update: $[\mathbf{c}_{j,k,t+1}]_m = \alpha[\mathbf{c}_{j,k,t}]_m + (1-\alpha)[\mathbf{x}_{t+1}]_m$, $m \in \Omega_{t+1}$
Update: $\lambda^{(m)}_{j,k,t+1} = \alpha\lambda^{(m)}_{j,k,t} + (1-\alpha)[\boldsymbol{\beta}]^2_m, m = 1, \ldots, d$
Update: $\delta_{j,k,t+1} = \alpha\delta_{j,k,t} + (1-\alpha)\|\boldsymbol{\beta}_\perp\|^2/(D-d)$
Update basis $\mathbf{U}_{j,k,t}$ using (modified) subspace tracking algorithm

Algorithm 5 Split node (j, k)

Turn two virtual children nodes $(j+1, 2k)$ and $(j+1, 2k+1)$ of node (j, k) into leaf nodes
Initialize virtual nodes $(j+1, 2k)$ and $(j+1, 2k+1)$:

$$k_1 = 2k,\ k_2 = 2k+1$$

$$\mathbf{c}_{j+1,k_1,t+1} = \mathbf{c}_{j,k,t} + \sqrt{\lambda^{(1)}_{j,k,t}}\mathbf{u}^{(1)}_{j,k,t}/2,\ \mathbf{c}_{j+1,k_2,t+1} = \mathbf{c}_{j,k,t} - \sqrt{\lambda^{(1)}_{j,k,t}}\mathbf{u}^{(1)}_{j,k,t}/2$$

$$\mathbf{U}_{j+1,k_i,t+1} = \mathbf{U}_{j,k,t}, \quad i = 1, 2$$

$$\lambda^{(1)}_{j+1,k_i,t+1} = \lambda^{(1)}_{j,k,t}/2, \quad i = 1, 2,\ \lambda^{(m)}_{j+1,k_i,t+1} = \lambda^{(m)}_{j,k,t}, \quad m = 2, \ldots, d, \quad i = 1, 2$$

Algorithm 6 Merge (j, k) and its sibling

1: Make the parent node of (j, k) into a leaf node
2: Make (j, k) and its sibling into virtual children nodes of the newly created leaf
3: Delete all four virtual children nodes of (j, k) and its sibling

To address these challenges, we introduce the *approximate Mahalanobis distance* of a point $\mathbf{x}$ to a subspace $\mathcal{S}$. Assume $\mathbf{x}$ with support $\mathbf{\Omega}$, parameter δ, and the parameters for a set $\mathcal{S}$ is given by $\{\mathbf{U}, \mathbf{c}, \mathbf{\Lambda}\}$. Define

$$\mathbf{U}_{\mathbf{\Omega}} \triangleq \mathcal{P}_{\mathbf{\Omega}}\mathbf{U} \in \mathbb{R}^{|\mathbf{\Omega}|\times d}, \quad \mathbf{c}_{\mathbf{\Omega}} \triangleq \mathcal{P}_{\mathbf{\Omega}}\mathbf{c} \in \mathbb{R}^{|\mathbf{\Omega}|}, \quad \mathbf{x}_{\mathbf{\Omega}} = \mathcal{P}_{\mathbf{\Omega}}\mathbf{x} \in \mathbb{R}^{|\mathbf{\Omega}|}.$$

Define the pseudoinverse operator that computes the coefficience a vector the subspace spanned by $\mathbf{V}$ as

$$\mathbf{V}^{\dagger} \triangleq (\mathbf{V}^{\top}\mathbf{V})^{-1}\mathbf{V}^{\top}. \tag{5.7}$$

Since $\mathbf{U}$ is an orthogonal matrix, we have $\mathbf{U}^{\dagger} \equiv \mathbf{U}^{\top}$, but in general $\mathbf{U}_{\mathbf{\Omega}}^{\dagger} \neq \mathbf{U}_{\mathbf{\Omega}}^{\top}$. Let

$$\boldsymbol{\beta} = \mathbf{U}_{\mathbf{\Omega}}^{\dagger}(\mathbf{x}_{\mathbf{\Omega}} - \mathbf{c}_{\mathbf{\Omega}}), \tag{5.8}$$

and

$$\boldsymbol{\beta}_{\perp} = (\mathbf{I} - \mathbf{U}_{\mathbf{\Omega}}\mathbf{U}_{\mathbf{\Omega}}^{\dagger})(\mathbf{x}_{\mathbf{\Omega}} - \mathbf{c}_{\mathbf{\Omega}}). \tag{5.9}$$

In this definition, $\boldsymbol{\beta}$ is the projection of $\mathbf{x}$ on $\mathbf{U}$, and $\|\boldsymbol{\beta}_{\perp}\|$ captures the energy of the projection residual. We denote Euclidean distance between $\mathbf{x}$ with support $\mathbf{\Omega}$ and the subspace where $\mathcal{S}$ lies on as

$$d(\mathbf{x}, \mathcal{S}) \triangleq \|\mathbf{x}_{\mathbf{\Omega}} - \mathbf{U}_{\mathbf{\Omega}}\mathbf{U}_{\mathbf{\Omega}}^{\dagger}(\mathbf{x}_{\mathbf{\Omega}} - \mathbf{c}_{\mathbf{\Omega}})\|^2 = \|\boldsymbol{\beta}_{\perp}\|^2. \tag{5.10}$$

Next we introduce the *approximate Mahalanobis distance*, which is a hybrid of Euclidean distance and Mahalanobis distance. Mahalanobis distance is commonly used for data classification, which measures the quadratic distance of $\mathbf{x}$ to a set $\mathcal{S}$ of data with mean $\mathbf{c} = \mathbb{E}\{\mathbf{x}\}$ and covariance $\mathbf{\Sigma} = \mathbb{E}\{(\mathbf{x} - \mathbf{c})(\mathbf{x} - \mathbf{c})^{\top}\}$. Specifically, the Mahalanobis distance is defined as

$$\varrho(\mathbf{x}, \mathcal{S}) = (\mathbf{x} - \mathbf{c})^{\top}\mathbf{\Sigma}^{-1}(\mathbf{x} - \mathbf{c}).$$

Assuming the covariance matrix has a low-rank structure with d large eigenvalues and $D-d$ small eigenvalues, we can write the eigendecomposition of the covariance matrix $\mathbf{\Sigma}$ as

$$\mathbf{\Sigma} \triangleq \begin{bmatrix}\mathbf{U} & \mathbf{U}_\perp\end{bmatrix} \mathbf{\Lambda} \begin{bmatrix}\mathbf{U} & \mathbf{U}_\perp\end{bmatrix}^\top = \mathbf{U}\mathbf{\Lambda}_1\mathbf{U}^\top + \mathbf{U}_\perp\mathbf{\Lambda}_2\mathbf{U}_\perp^\top,$$

where $\mathbf{\Lambda} = \text{diag}\{\lambda_1, \ldots, \lambda_D\}$, $\lambda_1 \geq \ldots \geq \lambda_D$, $\mathbf{\Lambda}_1 = \text{diag}\{\lambda_1, \ldots, \lambda_d\}$, and $\mathbf{\Lambda}_2 = \text{diag}\{\lambda_{d+1}, \ldots, \lambda_D\}$. If we further assume that the $D-d$ small eigenvalues are all approximately equal to δ, *i.e.* $\mathbf{\Lambda}_2 \approx \delta\mathbf{I}$, then

$$\varrho(\mathbf{x}, \mathcal{S}) \approx (\mathbf{x}-\mathbf{c})^\top\mathbf{U}\mathbf{\Lambda}_1^{-1}\mathbf{U}^\top(\mathbf{x}-\mathbf{c}) + \delta^{-1}\|\mathbf{U}_\perp^\top(\mathbf{x}-\mathbf{c})\|^2. \tag{5.11}$$

This motivates us to introduce the approximate Mahalanobis distance when the data is low-dimensional:

$$\rho_\delta(\mathbf{x}, \mathcal{S}) \triangleq \boldsymbol{\beta}^\top\mathbf{\Lambda}^{-1}\boldsymbol{\beta} + \delta^{-1}\|\boldsymbol{\beta}_\perp\|^2. \tag{5.12}$$

Note that when the data is complete, $\rho_\delta(\mathbf{x}, \mathcal{S})$ is equal to the right-hand-side of (5.11). With missing data, $\rho_\delta(x, \mathcal{S})$ is an approximation to $\varrho(\mathbf{x}, \mathcal{S})$.

5.3.4 Update subset parameters

When updating subspaces, we can update all subspaces in our multiscale representation and make the update step-size to be inversely proportional to the approximate Mahalanobis distance between the new sample and each subspace, which we refer to as the "update-all" approach. Alternatively, we can just update the subspace closest to $\mathbf{x}_{t+1}$, its virtual children, and all its ancestor nodes, which we refer to as the "update-nearest" approach. The update-all approach is computationally more expensive, especially for high dimensional problems, so we focus our attention on the greedy update-nearest approach. The below approaches extend readily to the update-all approach, however.

With the approximate Mahalanobis distance defined above, we can find the subset with minimum distance to the new datum $\mathbf{x}_t$:

$$(j^*, k^*) = \arg\min_{(j,k)} \rho_{\delta_{j,k,t}}(\mathbf{x}_t, \mathcal{S}_{j,k,t}),$$

and then update the parameters of that subset (and all its ancestors in the tree). The update algorithm is summarized in Algorithm 4 which denotes the parameters associated with $\mathcal{S}_{j^*,k^*,t}$ as $(\mathbf{c}, \mathbf{U}, \mathbf{\Lambda}, \delta)$, and drops the j^*, k^*, and t indices for simplicity of presentation. The update of the center $\mathbf{c}$, $\mathbf{\Lambda}$ and δ are straightforward.

Using the definition in (5.2), we have

$$\mathcal{P}_{\widehat{\mathcal{M}}_t}\mathbf{x} = \mathcal{P}_{\mathbf{\Omega}}^{\top}\mathbf{U}_{\mathbf{\Omega}}\mathbf{U}_{\mathbf{\Omega}}^{\dagger}(\mathbf{x}_{\mathbf{\Omega}} - \mathbf{c}_{\mathbf{\Omega}}) + \mathbf{c}; \tag{5.13}$$

that is, the projection onto the submanifold approximation is the projection onto the nearest subset. We further define the *instantaneous approximation error* of the submanifold at time t as:

$$e_t \triangleq \|\mathcal{P}_{\mathbf{\Omega}_t}(\mathbf{x}_t - \mathcal{P}_{\widehat{\mathcal{M}}_t}\mathbf{x}_t)\|^2, \tag{5.14}$$

and note that this is equivalent to the squared norm of the orthogonal projection of $\mathbf{x}_t$ onto $\mathcal{S}_{j^*,k^*,t}$, denoted $\boldsymbol{\beta}_{t,\perp}$; *i.e.*

$$e_t \equiv \|\boldsymbol{\beta}_{t,\perp}\|^2. \tag{5.15}$$

Next we will focus on three approaches to updating $\mathbf{U}$.

GROUSE

To use GROUSE subspace tracking in this context, we approximate the first term in (Eq. (5.3)) as

$$F(\mathcal{M}) = \sum_{i=1}^{t} \alpha^{t+1-i}\|\mathcal{P}_{\mathbf{\Omega}_i}(\mathbf{x}_i - \mathcal{P}_{\widehat{\mathcal{M}}_i}\mathbf{x}_i)\|^2 + \|\mathcal{P}_{\mathbf{\Omega}_{t+1}}(\mathbf{x}_{t+1} - \mathcal{P}_{\mathcal{M}}\mathbf{x}_{t+1})\|^2. \tag{5.16}$$

Note the first term is a constant with respect to $\mathcal{M}$, so we need only to consider the second term in computing an update. The basic idea is now to take a step in the direction of the instantaneous gradient of this cost function. Since $\mathcal{M}$ is constrained to be a union of subsets and the projection operator maps to the closest subset, this task corresponds to the basis update of GROUSE [5] with the cost function

$$f(U) \triangleq \min_{\mathbf{a}} \|\mathcal{P}_{\mathbf{\Omega}_{t+1}}(\mathbf{x}_{t+1} - \mathbf{U}\mathbf{a} - \mathbf{c})\|^2 \tag{5.17}$$

(assuming $\mathbf{U}$ is orthonormal and including the offset vector $\mathbf{c}$). Following the same derivation as in [5], we have that

$$\frac{df}{d\mathbf{U}} = -2\mathcal{P}_{\mathbf{\Omega}_{t+1}}(\mathbf{x}_{t+1} - \mathbf{c} - \mathbf{U}\boldsymbol{\beta})\boldsymbol{\beta}^\top \triangleq -2r\boldsymbol{\beta}^\top, \tag{5.18}$$

where $\boldsymbol{\beta}$ is defined in (5.8), and

$$\mathbf{r} = \mathcal{P}_{\mathbf{\Omega}_{t+1}}(\mathbf{x}_{t+1} - \mathbf{c} - \mathbf{U}\boldsymbol{\beta}).$$

Hence the gradient on the Grassmannian is given by

$$\nabla f = (\mathbf{I} - \mathbf{U}\mathbf{U}^\top)\frac{df}{d\mathbf{U}} = -2(\mathbf{I} - \mathbf{U}\mathbf{U}^\top)\mathbf{r}\boldsymbol{\beta}^\top = -2\mathbf{r}\boldsymbol{\beta}^\top,$$

since $\mathbf{U}^\top\mathbf{r} = 0$. We obtain that the update of $\mathbf{U}_t$ using the Grassmannian gradient is given by

$$\mathbf{U}_{t+1} = \mathbf{U}_t + \frac{\cos(\xi\eta) - 1}{\|\boldsymbol{\beta}\|^2}\mathbf{U}_t\boldsymbol{\beta}\boldsymbol{\beta}^\top + \sin(\xi\eta)\frac{\mathbf{r}}{\|\mathbf{r}\|}\frac{\boldsymbol{\beta}^\top}{\|\boldsymbol{\beta}\|}. \tag{5.19}$$

where $\eta > 0$ is the step-size, and $\xi = \|\mathbf{r}\|\|\mathbf{U}_t\boldsymbol{\beta}\|$. The step-size η is chosen to be $\eta = \eta_0/\|\mathbf{x}_{t+1}\|$, for a constant $\eta_0 > 0$.

PETRELS

Let $\mathbf{x}_{(1)}, \mathbf{x}_{(2)}, \mathbf{x}_{(3)}, \ldots$ denote a subsequence of the data such that each $x_{(i)}$ was drawn from the (j^*, k^*) node in our multiscale approximation, and let n_t denote the length

of this subsequence. Then we can approximate $F(\mathcal{M})$ as

$$F(\mathcal{M}) = \sum_{i=1}^{n_t} \alpha^{n_t - i} \min_{\mathbf{z}} \|\mathcal{P}_{\mathbf{\Omega}_{(i)}}(\mathbf{x}_{(i)} - \mathbf{c} - \mathbf{U}\mathbf{z})\|^2. \tag{5.20}$$

The minimization of $F(\mathcal{M})$ in (5.20) can be accomplished using the PETRELS algorithm [22], yielding a solution which can be expressed recursively as follows. Denote by $[\mathbf{U}]_m$ the m-th column of $\mathbf{U}$, we have the update of $\mathbf{U}$ given by

$$[\mathbf{U}_{t+1}]_m = [\mathbf{U}_t]_m + \mathbf{I}_{m \in \mathbf{\Omega}_t}([\mathbf{U}_t \mathbf{a}_{t+1}]_m - \mathbf{a}_{t+1}^\top [\mathbf{U}_t]_m)(\mathbf{R}_{m,t+1})^\dagger \mathbf{a}_{t+1}, \tag{5.21}$$

for $m = 1, \ldots, D$, where I_A is the indicator function for event A, and

$$\mathbf{a}_{t+1} = (\mathbf{U}_t^\top \mathcal{P}_{\mathbf{\Omega}_{t+1}} \mathbf{U}_t)^\dagger \mathbf{U}_t^\top \mathbf{x}_{t+1}.$$

The second-order information in $\mathbf{R}_{m,t+1}$ can be computed recursively as

$$(\mathbf{R}_{m,t+1})^\dagger = \alpha^{-1}(\mathbf{R}_{m,t})^\dagger + \frac{\alpha^{-2}\mathbf{p}_{m,t+1}}{1 + \alpha^{-1}\mathbf{a}_{t+1}^\top (\mathbf{R}_{m,t})^\dagger \mathbf{a}_{t+1}} (\mathbf{R}_{m,t})^\dagger \mathbf{a}_t \mathbf{a}_t^\top (\mathbf{R}_{m,t})^\dagger. \tag{5.22}$$

Note that PETRELS does not guarantee the orthogonality of $\mathbf{U}_{t+1}$, which is important for quickly computing projections onto our submanifold approximation. To obtain orthonormal $\mathbf{U}_{t+1}$, we may apply Gram-Schmidt orthonormalization after each update. We refer to this modification of PETRELS as *PETRELS-GS*. This orthogonalization requires an extra computational cost on the order of $\mathcal{O}(Dd^2)$ and may compromise the continuity of $\mathbf{U}_t$, *i.e.* $\|\mathbf{U}_{t+1} - \mathbf{U}_t\|$ after the orthogonalization may not be "small" [1]. As a result, this orthogonalization may change the optimality of $\mathbf{U}_t$. A faster orthonormalization (FO) strategy with less computation which also preserves the continuity of $\mathbf{U}_t$ is given in [1]. We refer to this FO strategy combined with PETRELS as *PETRELS-FO*.

Computational complexity

For each update with complete data (the maximum computational complexity), the computational complexity of GROUSE is on the order of $\mathcal{O}(Dd)$, PETRELS-GS is $\mathcal{O}(Dd^2)$, and PETRELS-FO is $\mathcal{O}(Dd)$. More details about the relative performance of these three subspace update methods can be found in Section 5.6.

5.3.5 Tree structure update

When the curvature of the submanifold changes and cannot be sufficiently characterized by the current subset approximations, we must perform adaptive model selection. This can be accomplished within our framework by updating the tree structure – growing the tree or pruning the tree, which we refer to as "splitting" and "merging" branches, respectively. Previous work has derived finite sample bounds and convergence rates of adaptive model selection in nonparametric time series prediction [56].

To decide whether to change the tree structure, we introduce the *average approximation error*:

$$\epsilon_t \triangleq \sum_{i=1}^{t} \alpha^{t-i} \|\mathcal{P}_{\mathbf{\Omega}_i}(\mathbf{x}_i - \mathcal{P}_{\widehat{\mathcal{M}}_i}\mathbf{x}_i)\|^2 = \alpha\epsilon_t + (1-\alpha)e_t. \tag{5.23}$$

This error is an approximation to the first term in (5.3), where we replace $\mathcal{P}_{\widehat{\mathcal{M}}}$ with the projection onto a sequence of approximations $\mathcal{P}_{\widehat{\mathcal{M}}_i}$. We will consider changing the tree structure when ϵ_t is greater than our prescribed error tolerance $\epsilon > 0$.

Splitting tree branches increases the resolution of the approximation at the cost of higher estimator complexity. Merging reduces resolution but lowers complexity. When making decisions on splitting or merging, we take into consideration the approximation errors as well as the model complexity (the number of subspaces K_t used in the approximation). This is related to complexity-regularized tree estima-

tion methods [17, 27, 85] and the notion of minimum description length (MDL) in compression theory [6, 57]. In particular, we use the sum of the average fitting error and a penalty proportional to the number of subspaces used for approximation as the cost function when deciding to split or merge. The splitting and merging are summarized in Algorithm 5 and Algorithm 6. The splitting process mimics the k-means algorithm. In these algorithms, note that for node (j, k) the parent is node $(j-1, \lfloor k/2 \rfloor)$ and the sibling node is $(j, k+1)$ for k even or $(j, k-1)$ for k odd.

5.3.6 Initialization

To initialize MOUSSE, we assume a small initial training set of samples, and perform a nested bi-partition of the training data set to form a tree structure, as shown in Figure 5.2. The root of the tree represents the entire data set, and the children of each node represent a bipartition of the data in the parent node. The bipartition of the data can be performed by the k-means algorithm. We start with the entire data, estimate the sample covariance matrix, perform an eigendecomposition, extract the d-largest eigenvectors and eigenvalues and use them for $\mathbf{U}_{1,1,0}$ and $\mathbf{\Lambda}_{1,1,0}$, respectively. The average of the $(D-d)$ minor eigenvalues are used for $\delta_{1,1,0}$. If the approximation error is greater than the prescribed error tolerance ϵ, we further partition the data into two clusters using k-means (for $k = 2$) and repeat the above process. We keep partitioning the data until $\delta_{\ell,n,0}$ of all leaf nodes are less than ϵ. Then we further partition the data one level down and form the virtual nodes. This tree construction is similar to that used in [2].

In principle, it is possible to bypass this training phase and just initialize the tree with a single root node and two random virtual children nodes. However, the training phase makes it much easier to select algorithm parameters such as ϵ and provides more meaningful initial virtual nodes, thereby shortening the "burn in" time of the algorithm.

5.3.7 Choice of parameters

In general, α should be close to 1, as in the Recursive Least Squares (RLS) algorithm [38]. In the case when the submanifold changes quickly, we would expect smaller weights for approximation based on historical data and thus a smaller α. In contrast, a slowly evolving submanifold requires a larger α. In our experiments, α ranges from 0.8 to 0.95. ϵ controls the data fit error, which varies from problem to problem according to the smoothness of the submanifold underlying the data and the noise variance. Since the tree's complexity is controlled and $\text{pen}(\mathcal{M})$ in (5.3) is roughly on the order of $\mathcal{O}(1)$, we usually set μ close to ϵ.

5.4 Changepoint detection

We are interested in detecting changes to the submanifold that arise abruptly and change the statistics of the data. When the submanifold varies slowly in time, MOUSSE described in Section 5.3 can track the submanifold and produce a sequence of stationary tracking errors. When an abrupt change occurs, MOUSSE loses track of the manifold and results in an abrupt increase in tracking errors. Hence, using tracking errors, we can develop a changepoint detection algorithm to detect abrupt changes in the submanifold.

5.4.1 CUSUM procedure

We adopt the widely used statistical CUSUM procedure [60, 62] for changepoint detection. In particular, we assume that ν_0 is a normal distribution with mean μ_0 and variance σ_0^2, and ν_1 is a normal distribution with mean μ_1 and the same variance σ_0^2. Then we can formulate the changepoint detection problem as the following

hypothesis test:

$$H_0: \quad e_1, \ldots, e_t \sim \mathcal{N}(\mu_0, \sigma_0^2)$$
$$H_1: \quad e_1, \ldots, e_\kappa \sim \mathcal{N}(\mu_0, \sigma_0^2), \quad e_{\kappa+1}, \ldots, e_t \sim \mathcal{N}(\mu_1, \sigma_0^2)$$

We assume μ_0 and σ_0^2 are known since typically there is enough normal data to estimate these parameters. (When the training phase is too short for this to be the case, these quantities can be estimated online, as described in [61].) However, we assume μ_1 is unknown since the magnitude of the changepoint can vary from one instance to another. In forming the detection statistic, we replace μ_1 by its maximum likelihood estimate (for each fixed changepoint time $\kappa = k$):

$$\hat{\mu}_1 = \frac{S_t - S_k}{t - k},$$

where

$$S_t \triangleq \sum_{i=1}^{t} e_i.$$

This leads to the generalized CUSUM procedure, which computes the CUSUM statistic at each time t and stops the first time when the statistic hits threshold b:

$$T = \inf \left\{ t \geq 1 : \max_{t-w \leq k < t} \frac{|(S_t - S_k) - \mu_0(t-k)|}{\sigma_0 \sqrt{t-k}} \geq b \right\}, \tag{5.24}$$

where w is a time-window length such that we only consider the most recent w errors for changepoint detection, and the threshold b is chosen to control the false-alarm-rate, which is characterized using average-run-length (ARL) in the changepoint detection literature [71]. This threshold choice is detailed in Section 5.5.3.

5.4.2 Distribution of e_t

In deriving the CUSUM statistics we have assumed that e_t are i.i.d. Gaussian distributed. A fair question to ask is whether e_t is truly Gaussian distributed, or even a

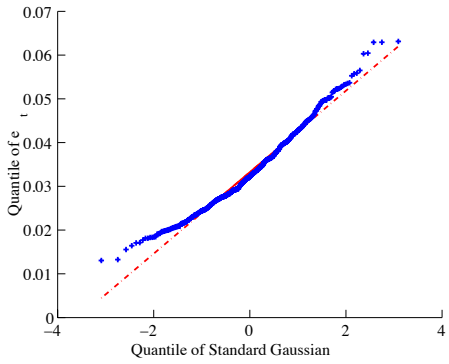


FIGURE 5.3: Q-Q plot of e_t, for a $D = 100$ submanifold.

good statistic to use. Consider the complete data case. Typically we can assume the high-dimensional tracking error is well-approximated by a Gaussian random vector:

$$\hat{e}_t \triangleq \mathcal{P}_{\widehat{\mathcal{M}}_t}(\mathbf{x}_t) - \mathbf{x}_t. \tag{5.25}$$

In general $\hat{e}_t$ has non-zero mean, since we approximate the manifold using a union of subspaces. Moreover, due to curvature of the manifold, in general each dimension of the error vector, $[\hat{e}_t]_m$, are not independent and the variances of $[\hat{e}_t]_m$ may not be identical. As a result, we cannot claim $e_t = \|\hat{e}_t\|^2$ is χ^2 or even non-central χ^2 distributed; in fact, no closed form expression for the distribution of e_t exists.

However, a Gaussian distribution is a good approximation for the distribution of e_t, since when D is large, e_t averages over errors across D dimensions. The QQ-plot of e_t from one of our numerical examples in Section 5.6 when $D = 100$ is shown in Figure 5.3. We will also demonstrate in Section 5.6.4 that the theoretical approximation for ARL using Gaussian assumption for e_t is quite accurate.

5.5 Performance Analysis

In this section, we first study the performance of MOUSSE, and then study the choice for the threshold parameter of the changepoint detection algorithm and pro-

vide theoretical approximations. A complete proof of convergence of MOUSSE (or GROUSE or PETRELS) is hard since the space of submanifold approximations we consider is non-convex. Nevertheless, we can still characterize several aspects of our approach.

In Sections 5.5.1 below, we assume that there is complete data, and we restrict our approximation to a single subspace so that $K_t = 1$. Assume the mean and covariance matrix of the data are given by $\mathbf{c}^*$ and $\mathbf{\Sigma}^*$, respectively. Assume the covariance matrix has low-rank structure: $\mathbf{\Sigma}^* = \text{diag}\{\lambda_1^*, \ldots, \lambda_D^*\}$ with $\lambda_m = \delta^*$ for $m = d+1, \ldots, D$.

5.5.1 Optimality of estimator for $\mathbf{c}^*$

When there is only one subspace and the data are complete, the cost function (5.3) without the penalty term becomes

$$\min_{\mathbf{U},\mathbf{c}} \sum_{i=1}^{t} \alpha^{t-i} \|(\mathbf{I} - \mathbf{U}\mathbf{U}^\top)(\mathbf{x}_i - \mathbf{c})\|^2. \tag{5.26}$$

By writing each term as $\|(\mathbf{I} - \mathbf{U}\mathbf{U}^\top)(\mathbf{x}_i - \mathbf{c})\|^2 = \|(\mathbf{I} - \mathbf{U}\mathbf{U}^\top)(\mathbf{x}_i - \bar{\mathbf{x}} + \bar{\mathbf{x}} - \mathbf{c})\|$, we can write the cost function as

$$\text{tr}[(I - \mathbf{U}\mathbf{U}^\top)\mathbf{S}(\mathbf{I} - \mathbf{U}\mathbf{U}^\top)] + \frac{1-\alpha^t}{1-\alpha}(\bar{\mathbf{x}} - \mathbf{c})^\top(\mathbf{I} - \mathbf{U}\mathbf{U}^\top)(\bar{\mathbf{x}} - \mathbf{c}), \tag{5.27}$$

where

$$\bar{\mathbf{x}} = \sum_{i=1}^{t} \alpha^{t-i}\mathbf{x}_i, \quad S = \sum_{i=1}^{t} \alpha^{t-i}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top. \tag{5.28}$$

Since the second term in (5.27) is quadratic in $\mathbf{c}$, it is minimized by $\mathbf{c} = \bar{\mathbf{x}}$. Also note that in this case the optimization problem (5.3) decouples in $\mathbf{U}$ and $\mathbf{c}$. Since our online update for $\mathbf{c}_t$ is given by $\mathbf{c}_{t+1} = \alpha\mathbf{c}_t + (1-\alpha)\mathbf{x}_t$, we have

$$\mathbf{c}_t = (1-\alpha)\sum_{i=1}^{t} \alpha^{t-i}\mathbf{x}_i + \alpha^t\mathbf{c}_0,$$

where the term $\alpha^t \mathbf{c}_0$ is a bias introduced by initial condition $\mathbf{c}_0$. Since $\mathbb{E}\{\mathbf{x}_t\} = \mathbf{c}^*$, $\mathbb{E}\{\mathbf{c}_t\} \to (1-\alpha) \cdot \frac{1}{1-\alpha}\mathbf{c}^* = \mathbf{c}^*$. Hence our estimator for $\mathbf{c}$ is proportional to the minimizer for (5.3) and is asymptotically unbiased.

5.5.2 MOUSSE tracking error scaling with level

As mentioned earlier, our multiscale subset model is closely related to geometric multiresolution analysis (GMRA) [2]. In that work, the authors characterize the favorable approximation capabilities of the proposed multiscale model. In particular, they prove that the magnitudes of the geometric wavelet coefficients associated with their algorithm decay asymptotically as a function of scale, so a collection of data lying on a smooth submanifold can be well-approximated with a small number (depending on the submanifold curvature) of relatively large geometric wavelets. These geometric wavelets are akin to the leaf nodes in our approximation, so the approximation results of [2] suggest that our model admits accurate approximations of data on smooth submanifolds with a small number of leafs.

5.5.3 Choice of threshold for changepoint detection

In accordance with standard changepoint detection notation, denote by $\mathbb{E}^\infty$ the expectation when there is no change, *i.e.*, $\mathbb{E}_{H_0}$, and by $\mathbb{E}^k$ the expectation when there is a changepoint at $\kappa = k$, *i.e.*, $\mathbb{E}_{H_1,\kappa=k}$. The performance metric for a changepoint detection algorithm is typically characterized by expected detection delay $\sup_{k\geq 0} \mathbb{E}^k\{T - k | T > k\}$ and the average-run-length (ARL) $\mathbb{E}^\infty\{T\}$ [71]. Typically we use $\mathbb{E}^0\{T\}$ as a performance metric since it is an upper bound for $\sup_{k\geq 0} \mathbb{E}^k\{T - k | T > k\}$. Note that the CUSUM statistic (5.24) is equivalent to

$$T = \inf\{t \geq 1 : \max_{t-w\leq k<t} \frac{|\tilde{S}_t - \tilde{S}_k|}{\sqrt{t-k}} \geq b, \} \tag{5.29}$$

where $\tilde{S}_t = \sum_{i=1}^{t}(e_i - \mu_0)/\sigma_0$. Under H_0, we have $(e_i - \mu_0)/\sigma_0$ i.i.d. Gaussian distributed with zero mean and unit variance. Using the results in [72], we have the following approximation. When $b \to \infty$,

$$\mathbb{E}^{\infty}\{T\} \sim \frac{(2\pi)^{1/2}\exp\{b^2/2\}}{b\int_0^b x\nu^2(x)dx}, \tag{5.30}$$

where $\nu(x) = \frac{(2/x)[\Phi(x/2)-0.5]}{(x/2)\Phi(x/2)+\phi(x)/2}$ [90], $\phi(x)$ and $\Phi(x)$ are the pdf and cdf of the normal random variable with zero mean and unit variance. We will demonstrate in Section 5.6.4 that this asymptotic approximation is fairly accurate even for finite b, which allows us to choose the changepoint detection threshold to achieve a target ARL without parameter tuning.

5.6 Numerical Examples

In this section, we present several numerical examples, first based on simulated data, and then real data, to demonstrate the performance of MOUSSE in tracking a submanifold and detecting changepoints. We also verify that our theoretical approximation to ARL in Section 5.5.3 is quite accurate.

5.6.1 Tracking a static submanifold

We first study the performance of MOUSSE tracking a static submanifold. The dimension of the submanifold is $D = 100$ and the intrinsic dimension is $d = 1$. Fixing $\theta \in [-2, 2]$, we define $\mathbf{v}(\theta) \in R^D$ with its n-th element

$$[\mathbf{v}(\theta)]_n = 1/\sqrt{2\pi}e^{-(z_n-\theta)^2/(2\gamma^2)}, \tag{5.31}$$

where $\gamma = 0.6$, and $z_n = -2 + 4n/D$, $n = 1, \ldots, 100$, corresponds to regularly spaced points between -2 and 2. This static submanifold is sampled by sampling different $\theta \in [-2, 2]$ and generating corresponding points on the submanifold according to

(5.31). The observation $\mathbf{x}_t$ is obtained from (5.1), where the noise variance is $\sigma^2 = 4 \times 10^{-4}$. We set parameter values as $\alpha = 0.95$, $\epsilon = 0.1$, $\mu = 0.1$, and use PETRELS-FO. Figure 5.4 demonstrates that MOUSSE is able to track a static submanifold and reach the steady state quickly from a coarse initialization. In Figure 5.4 and the following numerical examples, the expected instantaneous fitting error is evaluated using $N = 1200$ draws from $\mathcal{M}$, denoted $\mathbf{y}_1, \ldots, \mathbf{y}_N$ and computing the Monte Carlo estimate

$$\mathbb{E}\{e_t\} \approx \frac{1}{N} \sum_{i=1}^{N} d^2(\mathbf{y}_i, \mathcal{S}_i), \tag{5.32}$$

where $\mathcal{S}_i$ denotes the minimum distance subset to $\mathbf{y}_i$.

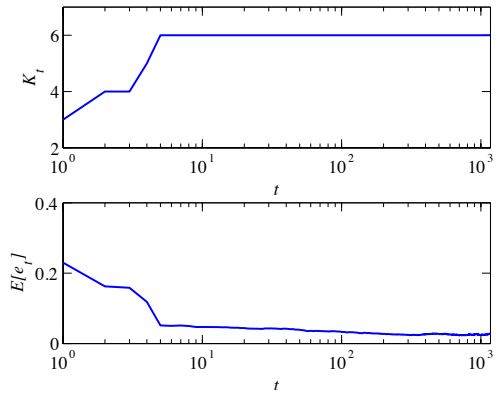


FIGURE 5.4: MOUSSE tracking a static submanifold with $D = 100$ and $d = 1$.

5.6.2 Tracking a slowly time varying submanifold

Next we track a slowly time varying submanifold using MOUSSE, in a similar setting to Section 5.6.1 where $D = 100$ and $d = 1$. The submanifold is also generated using (5.31), except that now we let γ to be time varying:

$$\gamma_t = \begin{cases} 0.6 - \gamma_0 t & t = 1, 2, \ldots, s \\ 0.6 - \gamma_0 (2s - t) & t = s+1, s+2, \ldots, 2s \end{cases} \tag{5.33}$$

where parameter γ_0 controls how fast the submanifold changes. We choose $\gamma_0 = 2 \times 10^{-4}$, $s = 1000$ with 40% missing data, and the parameters for MOUSSE are $\mu = 0.1$, $\epsilon = 0.1$, and $\alpha = 0.9$. The result of the tracking can be found in an illustrative video on Youtube. Snapshots of this video at time $t = 250$ and $t = 1150$ are shown in Figure 5.1. In this display, the dashed line corresponds to the true submanifold, the red lines correspond to the estimated union of subspaces, and the + signs correspond to the past 500 samples, with darker colors corresponding to more recent observations. From this video, it is clear that we are effectively tracking the dynamics of the submanifold, and keeping the representation parsimonious so the number of subspaces used by our model is proportional to the curvature of the submanifold, and as the curvature increases and decreases, the number of subspaces used in our approximation similarly increases and decreases. The number of subspaces K_t and fitting error as a function of time are shown in Figure 5.5. The red line in Figure 5.5 corresponds to ϵ. Note that MOUSSE is able to track the submanifold, in that it can maintain a stable number of leaf nodes in the approximation and meet the target error tolerance ϵ.

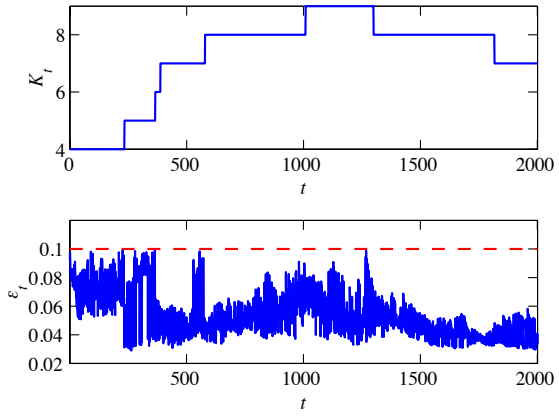
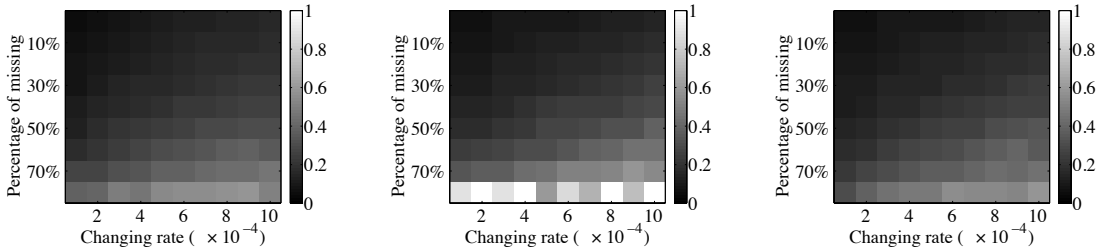


FIGURE 5.5: MOUSSE tracking a slowly evolving submanifold with $D = 100$ and $d = 1$. Dashed red line depicts CUSUM theoretical threshold calculated for ARL = 1000.

5.6.3 Comparison of tracking algorithms

We also compare the performance of different tracking algorithms presented in Section 5.3.4: GROUSE, PETRELS-GS and PETRELS-FO. We use $\mathbb{E}\{e_t\}$ defined in (5.32) as a comparison metric. In comparing the three methods, we set the parameters for each tracking algorithm such that the algorithm has the best possible performance. The comparison is displayed in Figure 5.6, where the horizontal axis is the submanifold changing rate γ, the vertical axis is the percentage of missing data, and the brightness of each block corresponds to $\mathbb{E}\{e_t\}$. In Figure 5.6, PETRELS-FO performs better than GROUSE and PETRELS-GS, especially with a large percentage of missing data. Also note that for PETRELS-FO, the optimal parameters are fairly stable for various combinations of submanifold changing rate and percentage of missing data: the optimal parameters are $\alpha \approx 0.9$, $\mu \approx 0.2$, and $\epsilon \approx 0.1$. Considering its lower computational cost and ease of parameter tuning, we use PETRELS-FO with MOUSSE for the remaining experiments in this chapter.



(a) $\mathbb{E}\{e_t\}$ of MOUSSE using GROUSE

(b) $\mathbb{E}\{e_t\}$ of MOUSSE using PETRELS-GS

(c) $\mathbb{E}\{e_t\}$ of MOUSSE using PETRELS-FO

FIGURE 5.6: MOUSSE tracking a slowly varying submanifold using: (a) GROUSE, (b) PETRELS-GS and (c) PETRELS-FO. Horizontal axis corresponds to rate of submanifold's change and vertical axis corresponds to fraction of data missing. Brightness corresponds to $\mathbb{E}\{e_t\}$.

5.6.4 *Changepoint detection example*

To verify our theoretical approximation for ARL, we perform Monte Carlo simulation. Direct simulation of T to obtain $\mathbb{E}^{\infty}\{T\}$ is very time-consuming because we typically want to choose a b such that $\mathbb{E}^{\infty}\{T\}$ is on the order of 10000. Hence we use an indirect simulation method commonly used in changepoint detection [90]. The indirect method is based on the fact that when there is no changepoint, for large b, T is typically exponentially distributed. Hence, we have under H_0,

$$\mathbb{P}\{T > m\} = e^{-m/E^{\infty}\{T\}},$$

which means we can simulate $\mathbb{P}\{T > m\}$ for fixed m and b, and then obtain under H_0

$$\mathbb{E}^{\infty}\{T\} = -m/\log \mathbb{P}\{T > m\}. \tag{5.34}$$

Using this formula, we generate 10000 Monte Carlo (MC) trials, each a slowly time-varying submanifold of duration $t = 500$. Then we use MOUSSE to track the submanifold, and obtain a sequence of errors. We form the CUSUM statistics using this sequence of errors, and find the fraction of sequences such that

$$\max_{1\le t\le 500} \max_{t-w\le k\le t} \frac{1}{\sigma_0} \frac{|(S_t - S_k) - \mu_0(t-k)|}{\sqrt{t-k}} \ge b,$$

which is the estimate for $\mathbb{P}\{T > 500\}$. Then we use the above formula to obtain $\mathbb{E}^{\infty}\{T\}$.

TABLE 5.1 shows the value of b suggested by theory for different ARLs and the value of b computed using the MC procedure described above. For comparison, we also obtain the ARL by treating this as a single subspace tracking problem for which PETRELS-FO is employed. These values of b are in parentheses.

To estimate the expected detection delay, we generate instances where the γ_t in

Table 5.1: Average run length (ARL) $\mathbb{E}^{\infty}\{T\}$.

ARL	b Theory	b MC (0% data missing)	b MC (20% data missing)	b MC (40% data missing)
1000	3.94	4.55 (4.28)	4.94 (4.32)	5.39 (4.38)
5000	4.35	5.26 (4.93)	5.56 (5.00)	6.00 (5.08)
10000	4.52	5.62 (5.20)	5.82 (5.27)	6.25 (5.37)

the model has an abrupt jump Δ_γ at time $t = 200$.

$$\gamma_t = \begin{cases} 0.6 - \gamma_0 t & t = 1, 2, \ldots, 199 \\ \gamma_{199} - \Delta_\gamma - \gamma_0 t & t = 200, 201, \ldots, 400 \end{cases} \tag{5.35}$$

Then we apply the CUSUM statistics and find the expected detection delay $\mathbb{E}^0\{T\}$ with respect to $t = 200$ using 10000 Monte Carlo trials. We compare the expected detection delay of MOUSSE and the single subspace tracking method. Results corresponding to big ($\Delta_\gamma = 0.05$) and small ($\Delta_\gamma = 0.03$) magnitude of jump of γ_t are given in TABLE 5.2, 5.3 respectively. Again, values in parenthesis are obtained by using single subspace tracking. The threshold b's are chosen according to the Monte Carlo results given in TABLE 5.1; e.g., for the cell corresponding to ARL = 1000 and 0% missing data in TABLE 5.2 or 5.3, b should be set as 4.55 for MOUSSE and 4.28 for the single subspace method. TABLE 5.2, 5.3 demonstrate that MOUSSE has much smaller expected detection delay than a single subspace method.

5.6.5 Real data

A video from the Solar Data Observatory, which demonstrates an abrupt emergence of a solar flare, can be found on Youtube. The Solar Object Locator for the original data is SOL2011-04-30T21-45-49L061C108. Also displayed is the residual $\hat{e}_t$ of (5.25) obtained using MOUSSE, which clearly shows peaks in the vicinity of the solar flare. A frame from this dataset during a solar flare is shown in Figure 5.7a. The frame

Table 5.2: Detection delay when jump of γ_t is $\Delta_\gamma = 0.05$.

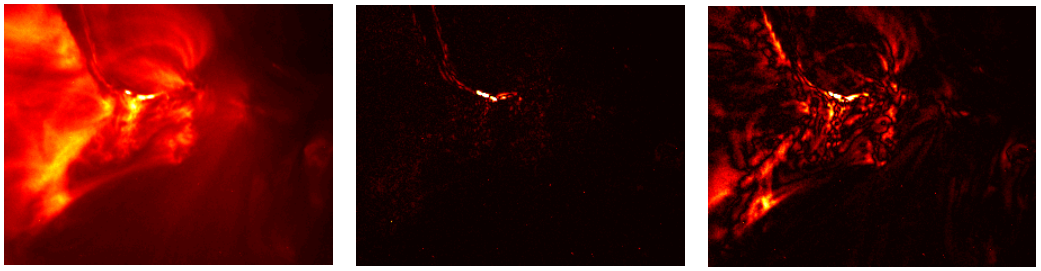
ARL	delay (0% data missing)	delay (20% data missing)	delay (40% data missing)
1000	2.28 (19.81)	2.43 (19.80)	2.87 (20.25)
5000	2.39 (24.70)	2.58 (24.70)	3.15 (25.22)
10000	2.46 (27.05)	2.65 (27.05)	3.28 (27.41)

Table 5.3: Detection delay when jump of γ_t is $\Delta_\gamma = 0.03$.

ARL	delay (0% data missing)	delay (20% data missing)	delay (40% data missing)
1000	3.41 (32.71)	4.53 (32.97)	6.84 (33.57)
5000	4.23 (43.25)	5.68 (44.02)	8.67 (44.10)
10000	4.79 (47.96)	6.27 (48.61)	9.52 (49.03)

is of size 232×292 resulting in 67744 dimensional online data. To reduce difficulty of parameter tuning, we scale the pixel intensities in the dataset by multiplying the data by a factor of 10^{-4} to be consistent with the scale of our simulated data experiments. The parameters we use are $\epsilon = 0.3$, $\mu = 0.3$, and $\alpha = 0.85$. The video and the snapshots in Figure 5.7 demonstrate that MOUSSE can not only detect the emergence of a solar flares but also localize the flare by presenting $\hat{e}_t$, and these tasks are accomplished far more effectively with MOUSSE (even with $d = 1$) than with a single subspace. Note that with the single subspace tracking, the residual norm $e(t)$ is not a stationary time series prior to the flare and thus poorly suited for changepoint detection. In the original images, the background solar images has bright spots that are slowly and changing shape, which makes detection based on simple background subtraction incapable of detecting small transient flares. In contrast, with our approach, with K_t around 10, the underlying manifold structure is better

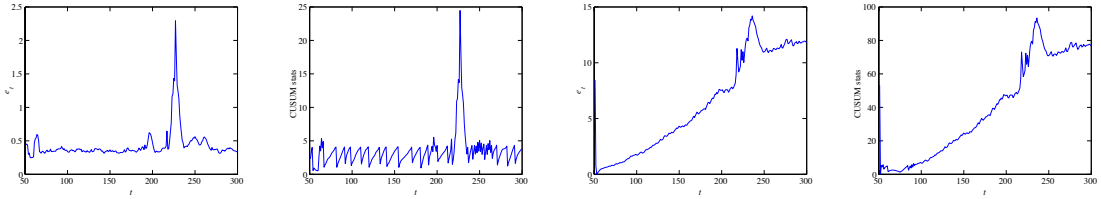
tracked and thus yields more obvious error $e(t)$ when anomaly occurs.



(a) Snapshot of original SDO data at $t = 227$

(b) MOUSSE residual at $t = 227$

(c) Single subspace tracking residual at $t = 227$



(d) e_t from MOUSSE

(e) CUSUM stats from MOUSSE

(f) e_t from single subspace tracking

(g) CUSUM stats from single subspace tracking

FIGURE 5.7: Detection of solar flare at $t = 227$: (a) snapshot of original SDO data at $t = 227$; (b) MOUSSE residual $\hat{e}_t$, which clearly identifies an outburst of solar flare; (c) single subspace tracking residual $\hat{e}_t$, which gives a poor indication of the flare; (d) $e(t)$ for MOUSSE which peaks near the flare around $t = 227$; (e) the CUSUM statistic for MOUSSE; (f) $e(t)$ for single subspace tracking; (g) the CUSUM statistic for single subspace tracking. Using a single subspace gives much less reliable estimates of significant changes in the statistics of the frames.

Our second real data example is related to automatic identity theft detection. The basic idea is that consumers have typical spending patterns which change abruptly after identity theft. Banks would like to identify these changes as quickly as possible without triggering numerous false alarms. To test MOUSSE on this high-dimensional changepoint detection problem, we examined the E-commerce transaction history of people in a dataset used for a 2008 UCSD data mining competition at `http://www.cs.purdue.edu/commugrate/data_access/all_data_sets_more.php?search_fd0=20`. For each person in this dataset, there is a time series of transactions. For each transaction we have a 31-dimensional real-valued feature vector and

a label of whether the transaction is "good" (0) or "bad" (1). The full dataset was generated for a generic anomaly detection problem, so it generally is not appropriate for our setting. However, some of these transaction timeseries show a clear changepoint in the labels, and we applied MOUSSE to these timeseries. In particular, we use MOUSSE to track the 31-dimensional feature vector and detect a changepoint, and compare this with the "ground truth" changepoint in the label timeseries. The effect of this for one person's transaction history is displayed in Figure 5.8. We see that while MOUSSE does not generate particularly large spikes in e_t, the associated CUSUM statistic shows a marked increase near $t = 70$, when the labels (not used by MOUSSE) change from 0 (good) to 1 (bad).

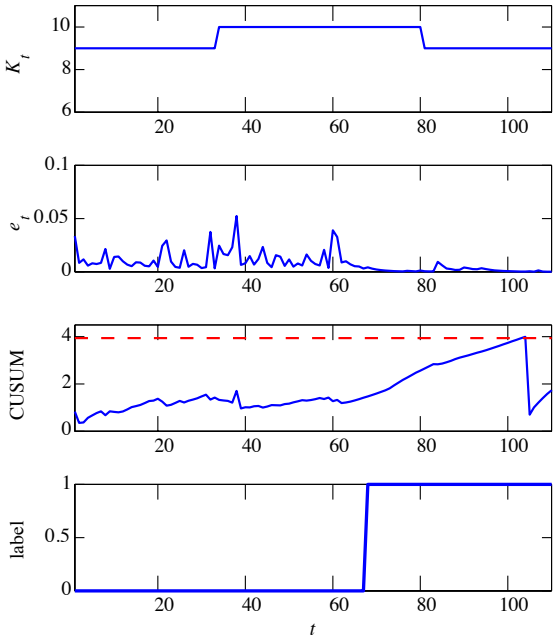


FIGURE 5.8: Credit card user data experiments. (a) Number of leaves used by MOUSSE. (b) MOUSSE residual norm. (c) MOUSSE CUSUM statistic (solid blue line) and CUSUM theoretical threshold calculated for ARL = 1000 (dashed red line). (d) Ground truth transaction label with changepoint near where CUSUM statistic starts increasing.

5.7 Conclusions

This chapter describes a novel multiscale method for online tracking of high-dimensional data on a low-dimensional submanifold, and using the tracking residuals to perform fast and robust changepoint detection. Changepoint detection is an important subset of anomaly detection problems due to the ever-increasing volume of streaming data which must be efficiently prioritized and analyzed. The multiscale structure at the heart of our method is based on a Geometric MultiResolution Analysis which facilitates low-complexity piecewise-linear approximations to a manifold. The multiscale structure allows for fast updates of the manifold estimate and flexible approximations which can adapt to the changing curvature of a dynamic submanifold. These ideas have the potential to play an important role in analyzing large volumes of streaming data, which arises in remote sensing, credit monitoring, and network traffic analysis.

While the algorithm proposed in this chapter has been focused on unions of subspaces, an important open question is whether similar techniques could be efficiently adopted based on sparse covariance matrix selection [26, 49]. The resulting approximation space may no longer correspond to a low-dimensional submanifold, but such structures provide good representations of high-dimensional data in many settings, and our future work includes tracking the evolution of a mixture of such structures. Issues related to non-Gaussian observation models, inverse problem settings, dynamical models, and optimal selection of the statistic used for changepoint detection (*i.e.* alternatives to e_t, as considered in [89]) all pose additional interesting open problems.

Appendix A

Supplementary Proofs for Chapter 2

A.1 Proof of high SNR case

Proof of Theorem 1 We have

$$\begin{aligned}\det \mathbf{\Sigma}_1 = &\ (\sigma^2)^{n-d} \textstyle\prod_{i=1}^{d} (\lambda_{1,i} + \sigma^2)\,, \\ \det \mathbf{\Sigma}_2 = &\ (\sigma^2)^{n-d} \textstyle\prod_{i=1}^{d} (\lambda_{2,i} + \sigma^2)\,.\end{aligned}$$

Let the SVD of $\mathbf{U}_{1,\cap}\mathbf{\Lambda}_{1,\cap}\mathbf{U}_{1,\cap}^\top + \mathbf{U}_{1,\setminus}\mathbf{\Lambda}_{1,\setminus}\mathbf{U}_{1,\setminus}^\top + \mathbf{U}_{2,\cap}\mathbf{\Lambda}_{2,\cap}\mathbf{U}_{2,\cap}^\top + \mathbf{U}_{2,\setminus}\mathbf{\Lambda}_{2,\setminus}\mathbf{U}_{2,\setminus}^\top$ be $\mathbf{Z}\mathbf{\Lambda}\mathbf{Z}^\top$, where $\mathbf{\Lambda} = \operatorname{diag}\{\lambda_1, \ldots, \lambda_{2d-r}\}$. Then,

$$\det\left(\frac{\mathbf{\Sigma}_1 + \mathbf{\Sigma}_2}{2}\right) = (\sigma^2)^{n-2d+r} \prod_{i=1}^{2d-r} \left(\frac{\lambda_i}{2} + \sigma^2\right).$$

Substituting the above into the Bhattacharyya bound, we have

$$\begin{aligned}P_e \leq &\frac{1}{2}(\sigma^2)^{\frac{d-r}{2}} \cdot \left[\frac{\sqrt{\prod_{i=1}^{d} (\lambda_{1,i} + \sigma^2) \prod_{i=1}^{d} (\lambda_{2,i} + \sigma^2)}}{\prod_{i=1}^{2d-r} \left(\frac{\lambda_i}{2} + \sigma^2\right)}\right]^{\frac{1}{2}} \\ = &(\sigma^2)^{\frac{d-r}{2}} \cdot 2^{\frac{2d-r}{2}-1} \left[\frac{\sqrt{\prod_{i=1}^{d} \lambda_{1,i} \prod_{i=1}^{d} \lambda_{2,i}}}{\prod_{i=1}^{2d-r} \lambda_i}\right]^{\frac{1}{2}} + o\left((\sigma^2)^{\frac{d-r}{2}}\right).\end{aligned} \tag{A.1}$$

Our objective is to expand $\prod_{i=1}^{2d-r} \lambda_i$ in terms of principal angles. Since the image of $\mathbf{U}_{1,\cap}$ (or $\mathbf{U}_{2,\cap}$) is orthogonal to $\mathbf{U}_{1,\setminus}$ and $\mathbf{U}_{2,\setminus}$,

$$\begin{aligned}\prod_{i=1}^{2d-r} \lambda_i &= \mathrm{pdet}(\mathbf{U}_{1,\cap}\mathbf{\Lambda}_{1,\cap}\mathbf{U}_{1,\cap}^\top + \mathbf{U}_{2,\cap}\mathbf{\Lambda}_{2,\cap}\mathbf{U}_{2,\cap}^\top) \cdot \mathrm{pdet}([\mathbf{U}_{1,\setminus}\mathbf{\Lambda}_{1,\setminus}^{\frac{1}{2}}\ \mathbf{U}_{2,\setminus}\mathbf{\Lambda}_{2,\setminus}^{\frac{1}{2}}][\mathbf{U}_{1,\setminus}\mathbf{\Lambda}_{1,\setminus}^{\frac{1}{2}}\ \mathbf{U}_{2,\setminus}\mathbf{\Lambda}_{2,\setminus}^{\frac{1}{2}}]^\top) \\ &= \mathrm{pdet}\left(\mathbf{U}_{1,\cap}\mathbf{\Lambda}_{1,\cap}\mathbf{U}_{1,\cap}^\top + \mathbf{U}_{2,\cap}\mathbf{\Lambda}_{2,\cap}\mathbf{U}_{2,\cap}^\top\right) \cdot \det([\mathbf{U}_{1,\setminus}\mathbf{\Lambda}_{1,\setminus}^{\frac{1}{2}}\ \mathbf{U}_{2,\setminus}\mathbf{\Lambda}_{2,\setminus}^{\frac{1}{2}}]^\top[\mathbf{U}_{1,\setminus}\mathbf{\Lambda}_{1,\setminus}^{\frac{1}{2}}\ \mathbf{U}_{2,\setminus}\mathbf{\Lambda}_{2,\setminus}^{\frac{1}{2}}]),\end{aligned}$$

where we assume $n \geq 2(d-r)$ in order to derive the second equality, which simplifies as follows:

$$\begin{aligned}&\det([\mathbf{U}_{1,\setminus}\mathbf{\Lambda}_{1,\setminus}^{\frac{1}{2}}\ \mathbf{U}_{2,\setminus}\mathbf{\Lambda}_{2,\setminus}^{\frac{1}{2}}]^\top[\mathbf{U}_{1,\setminus}\mathbf{\Lambda}_{1,\setminus}^{\frac{1}{2}}\ \mathbf{U}_{2,\setminus}\mathbf{\Lambda}_{2,\setminus}^{\frac{1}{2}}]) \\ &= \det\left(\left[\mathbf{\Lambda}_{1,\setminus}\mathbf{\Lambda}_{1,\setminus}^{\frac{1}{2}}\mathbf{U}_{1,\setminus}^\top\mathbf{U}_{2,\setminus}\mathbf{\Lambda}_{2,\setminus}^{\frac{1}{2}}\mathbf{\Lambda}_{2,\setminus}^{\frac{1}{2}}\mathbf{U}_{2,\setminus}^\top\mathbf{U}_{1,\setminus}\mathbf{\Lambda}_{1,\setminus}^{\frac{1}{2}}\mathbf{\Lambda}_{2,\setminus}\right]\right) \\ &= \det(\mathbf{\Lambda}_{1,\setminus})\det\left(\mathbf{\Lambda}_{2,\setminus} - \mathbf{\Lambda}_{2,\setminus}^{\frac{1}{2}}\mathbf{U}_{2,\setminus}^\top\mathbf{U}_{1,\setminus}\mathbf{\Lambda}_{1,\setminus}^{\frac{1}{2}}\mathbf{\Lambda}_{1,\setminus}^{-1}\mathbf{\Lambda}_{1,\setminus}^{\frac{1}{2}}\mathbf{U}_{1,\setminus}^\top\mathbf{U}_{2,\setminus}\mathbf{\Lambda}_{2,\setminus}^{\frac{1}{2}}\right) \\ &= \det(\mathbf{\Lambda}_{1,\setminus})\det\left(\mathbf{\Lambda}_{2,\setminus}^{\frac{1}{2}}(\mathbf{I} - \mathbf{U}_{2,\setminus}^\top\mathbf{U}_{1,\setminus}\mathbf{U}_{1,\setminus}^\top\mathbf{U}_{2,\setminus})\mathbf{\Lambda}_{2,\setminus}^{\frac{1}{2}}\right) \\ &= \prod_{i=1}^{d-r}\boldsymbol{\lambda}_{1,\setminus,i}\cdot\prod_{i=1}^{d-r}\boldsymbol{\lambda}_{2,\setminus,i}\cdot\prod_{i=r+1}^{d}\sin^2\theta_i.\end{aligned} \tag{A.2}$$

The last equality follows from the observation that the eigenvalues of $\mathbf{U}_{2,\setminus}^\top\mathbf{U}_{1,\setminus}\mathbf{U}_{1,\setminus}^\top\mathbf{U}_{2,\setminus}$ are $\cos^2\theta_{r+1},\ldots,\cos^2\theta_d$. The theorem now follows by substituting Eq. (A.2) into Eq. (A.1).

□

A.2 Proof of Low SNR case

We first state and prove (for completeness) two preliminary lemmas that are needed to characterize classification error.

Lemma 14. *Let* $\mathbf{D} \in \mathbb{R}^{n\times n}$ *be any positive semi-definite matrix with all eigenvalues smaller than* 1*, then*

$$\operatorname{tr}(\mathbf{D}) - \frac{1}{2}\operatorname{tr}(\mathbf{D}^2) \leq \ln\det(\mathbf{I}_n + \mathbf{D}) \leq \operatorname{tr}(\mathbf{D}) - \frac{1}{4}\operatorname{tr}(\mathbf{D}^2).$$

Proof. Denote the nonnegative eigenvalues of $\mathbf{D} \succeq 0$ as $d_1, \ldots, d_n$, where $d_1, \ldots, d_n \leq 1$. Then $\ln\det(\mathbf{I}_n+\mathbf{D}) = \ln\prod_{i=1}^n(1+d_i) = \sum_{i=1}^n \ln(1+d_i)$. Since $x-\frac{x^2}{2} \leq \ln(1+x) \leq x - \frac{x^2}{4}$ for all $x \in [0,1]$, we obtain $\sum_i d_i - \frac{d_i^2}{2} \leq \ln\det(\mathbf{I}_n + \mathbf{D}) \leq \sum_i d_i - \frac{d_i^2}{4}$, which reduces to

$$\operatorname{tr}(\mathbf{D}) - \frac{1}{2}\operatorname{tr}(\mathbf{D}^2) \leq \ln\det(\mathbf{I}_n + \mathbf{D}) \leq \operatorname{tr}(\mathbf{D}) - \frac{1}{4}\operatorname{tr}(\mathbf{D}^2).$$

This bound is very tight when all the d_i's approach 0. □

Lemma 15. *Suppose* $\mathbf{U} \in \mathbb{R}^{n\times d}, \mathbf{V} \in \mathbb{R}^{n\times d}$ *are two orthonormal bases and that* $\mathbf{\Phi} \in \mathbb{R}^{d\times d}$, $\mathbf{\Psi} \in \mathbb{R}^{d\times d}$ *are diagonal with nonnegative decreasing diagonal elements* $\phi_1, \ldots, \phi_d$ *and* $\psi_1, \ldots, \psi_d$ *respectively. Denote the* i*-th principal angle between* $\mathbf{U}$ *and* $\mathbf{V}$ *as* θ_i *where* $i = 1, \ldots, d$*. Then*

$$\phi_d\psi_d \sum_i \cos^2\theta_i \leq \operatorname{tr}(\mathbf{U}\mathbf{\Phi}\mathbf{U}^\top\mathbf{V}\mathbf{\Psi}V^\top) \leq \phi_1\psi_1 \sum_i \cos^2\theta_i.$$

Proof. Let the Singular Value Decomposition of $\mathbf{U}^\top\mathbf{V}$ be $\mathbf{JCH}^\top$, then $\operatorname{tr}(\mathbf{U}^\top\mathbf{V}\mathbf{V}^\top\mathbf{U}) = \operatorname{tr}(\mathbf{C}^2) = \sum_i \cos^2\theta_i$. We have

$$\begin{aligned}\operatorname{tr}(\mathbf{U}\mathbf{\Phi}\mathbf{U}^\top\mathbf{V}\mathbf{\Psi}\mathbf{V}^\top) &= \operatorname{tr}(\mathbf{\Phi}\mathbf{U}^\top\mathbf{V}\mathbf{\Psi}\mathbf{V}^\top\mathbf{U}) \\ &= \operatorname{tr}(\mathbf{\Phi}\mathbf{J}\mathbf{C}\mathbf{H}^\top\mathbf{\Psi}\mathbf{H}\mathbf{C}\mathbf{J}^\top) = \operatorname{tr}(\mathbf{J}^\top\mathbf{\Phi}\mathbf{J}\mathbf{C}\mathbf{H}^\top\mathbf{\Psi}\mathbf{H}\mathbf{C}).\end{aligned}$$

For any two positive semidefinite matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m\times m}$, let the maximum and minimum eigenvalues of $\mathbf{A}$ be $\lambda_1(\mathbf{A}), \lambda_m(\mathbf{A})$ respectively, then by [51]

$$\lambda_m(\mathbf{A})\operatorname{tr}(\mathbf{B}) \leq \operatorname{tr}(\mathbf{A}\mathbf{B}) \leq \lambda_1(\mathbf{A})\operatorname{tr}(\mathbf{B}).$$

Hence,

$$\begin{aligned}\operatorname{tr}(\mathbf{U}\mathbf{\Phi}\mathbf{U}^\top\mathbf{V}\mathbf{\Psi}\mathbf{V}^\top) &\le \phi_1 \operatorname{tr}(\mathbf{C}\mathbf{H}^\top\mathbf{\Psi}\mathbf{H}\mathbf{C}) = \phi_1 \operatorname{tr}(\mathbf{H}^\top\mathbf{\Psi}\mathbf{H}\mathbf{C}^2) \\ &\le \phi_1\psi_1 \operatorname{tr}(\mathbf{C}^2) = \phi_1\psi_1 \sum_i \cos^2\theta_i.\end{aligned}$$

The lower bound can be proved in the same way. This bound becomes tight when the diagonal elements of $\mathbf{\Phi}$ and $\mathbf{\Psi}$ are uniform. □

Proof of Theorem 2 We are now ready to prove theorem 2. We expand K in Eq. (2.4) as

$$K = \frac{1}{2}\ln\det\left(\frac{\mathbf{\Sigma}_1+\mathbf{\Sigma}_2}{2}\right) - \frac{1}{4}(\ln\det\mathbf{\Sigma}_1 + \ln\det\mathbf{\Sigma}_2). \tag{A.3}$$

The second term becomes:

$$-\frac{1}{4}\left[\sum_{i=1}^{d}\ln\left(1+\frac{\lambda_{1,i}}{\sigma^2}\right)+\sum_{i=1}^{d}\ln\left(1+\frac{\lambda_{2,i}}{\sigma^2}\right)\right] - \frac{n}{2}\ln(\sigma^2), \tag{A.4}$$

and we use Lemma 14 to bound the first term. Note that

$$\begin{aligned}\frac{1}{2}\ln\det\left(\frac{\mathbf{\Sigma}_1+\mathbf{\Sigma}_2}{2}\right) &= \frac{1}{2}\ln\det\left[\sigma^2\left(\mathbf{I}+\frac{\mathbf{U}_1\Lambda_1\mathbf{U}_1^\top+\mathbf{U}_2\mathbf{\Lambda}_2\mathbf{U}_2^\top}{2\sigma^2}\right)\right] \\ &= \frac{n}{2}\ln(\sigma^2) + \frac{1}{2}\ln\det\left(\mathbf{I}+\frac{\mathbf{U}_1\mathbf{\Lambda}_1\mathbf{U}_1^\top+\mathbf{U}_2\mathbf{\Lambda}_2\mathbf{U}_2^\top}{2\sigma^2}\right).\end{aligned} \tag{A.5}$$

Let $\mathbf{D} \triangleq \frac{\mathbf{U}_1\mathbf{\Lambda}_1\mathbf{U}_1^\top+\mathbf{U}_2\mathbf{\Lambda}_2\mathbf{U}_2^\top}{2\sigma^2}$. We apply Lemma 14 to bound $\frac{1}{2}\ln\det\left(\frac{\mathbf{\Sigma}_1+\mathbf{\Sigma}_2}{2}\right)$:

$$\begin{aligned}&\frac{n}{2}\ln(\sigma^2) + \frac{1}{2}\left[\operatorname{tr}(\mathbf{D}) - \frac{1}{2}\operatorname{tr}(\mathbf{D}^2)\right] \le \frac{1}{2}\ln\det\left(\frac{\mathbf{\Sigma}_1+\mathbf{\Sigma}_2}{2}\right) \\ &\le \frac{n}{2}\ln(\sigma^2) + \frac{1}{2}\left[\operatorname{tr}(\mathbf{D}) - \frac{1}{4}\operatorname{tr}(\mathbf{D}^2)\right],\end{aligned} \tag{A.6}$$

Expanding $\mathrm{tr}(\mathbf{D})$ gives

$$\begin{aligned}
&\frac{n}{2}\ln(\sigma^2)+\frac{1}{4}\left[\sum_{i=1}^{d}\frac{\lambda_{1,i}}{\sigma^2}+\sum_{i=1}^{d}\frac{\lambda_{2,i}}{\sigma^2}\right]-\frac{1}{4}\mathrm{tr}(\mathbf{D}^2)\le\frac{1}{2}\ln\det\left(\frac{\mathbf{\Sigma}_1+\mathbf{\Sigma}_2}{2}\right)\\
&\le\frac{n}{2}\ln(\sigma^2)+\frac{1}{4}\left[\sum_{i=1}^{d}\frac{\lambda_{1,i}}{\sigma^2}+\sum_{i=1}^{d}\frac{\lambda_{2,i}}{\sigma^2}\right]-\frac{1}{8}\mathrm{tr}(\mathbf{D}^2).
\end{aligned}\tag{A.7}$$

Note that

$$\mathrm{tr}(\mathbf{D}^2)=\frac{1}{4\sigma^4}\bigg(\sum_{i=1}^{d}\lambda_{1,i}^2+\sum_{i=1}^{d}\lambda_{2,i}^2+2\,\mathrm{tr}(\mathbf{U}_1\mathbf{\Lambda}_1\mathbf{U}_1^\top\mathbf{U}_2\mathbf{\Lambda}_2\mathbf{U}_2^\top)\bigg).\tag{A.8}$$

Envoking Lemma 15 to bound the last term of the above:

$$\begin{aligned}
\mathrm{tr}(\mathbf{U}_1\mathbf{\Lambda}_1\mathbf{U}_1^\top\mathbf{U}_2\mathbf{\Lambda}_2\mathbf{U}_2^\top)&\ge\lambda_{1,d}\lambda_{2,d}\sum_i\cos^2\theta_i\\
\mathrm{tr}(\mathbf{U}_1\mathbf{\Lambda}_1\mathbf{U}_1^\top\mathbf{U}_2\mathbf{\Lambda}_2\mathbf{U}_2^\top)&\le\lambda_{1,1}\lambda_{2,1}\sum_i\cos^2\theta_i
\end{aligned}\tag{A.9}$$

Combining Eq. (A.4) to (A.9), we obtain upper and lower bounds on K,

$$\begin{aligned}
K\le&\frac{1}{4}\left[\sum_{i=1}^{d}\frac{\lambda_{1,i}}{\sigma^2}+\sum_{i=1}^{d}\frac{\lambda_{2,i}}{\sigma^2}\right]-\frac{1}{32\sigma^4}\bigg(\sum_{i=1}^{d}\lambda_{1,i}^2+\sum_{i=1}^{d}\lambda_{2,i}^2+2\lambda_{1,d}\lambda_{2,d}\sum_{i=1}^{d}\cos^2\theta_i\bigg)\\
&-\frac{1}{4}\left[\sum_{i=1}^{d}\ln\left(1+\frac{\lambda_{1,i}}{\sigma^2}\right)+\sum_{i=1}^{d}\ln\left(1+\frac{\lambda_{2,i}}{\sigma^2}\right)\right]\\
=&\frac{1}{4}\left[\sum_{i=1}^{d}\frac{\lambda_{1,i}}{\sigma^2}-\frac{1}{2}\sum_{i=1}^{d}\left(\frac{\lambda_{1,i}}{2\sigma^2}\right)^2-\sum_{i=1}^{d}\ln\left(1+\frac{\lambda_{1,i}}{\sigma^2}\right)\right]-\frac{1}{16\sigma^4}\lambda_{1,d}\lambda_{2,d}\sum_{i=1}^{d}\cos^2\theta_i\\
&+\frac{1}{4}\left[\sum_{i=1}^{d}\frac{\lambda_{2,i}}{\sigma^2}-\frac{1}{2}\sum_{i=1}^{d}\left(\frac{\lambda_{2,i}}{2\sigma^2}\right)^2-\sum_{i=1}^{d}\ln\left(1+\frac{\lambda_{2,i}}{\sigma^2}\right)\right]\\
\triangleq&\frac{1}{\sigma^4}\left(c_2-\frac{1}{16}\lambda_{1,d}\lambda_{2,d}\sum_{i=1}^{d}\cos^2\theta_i\right).
\end{aligned}\tag{A.10}$$

$$
\begin{aligned}
K \geq & \frac{1}{4}\left[\sum_{i=1}^{d}\frac{\lambda_{1,i}}{\sigma^2}+\sum_{i=1}^{d}\frac{\lambda_{2,i}}{\sigma^2}\right]-\frac{1}{16\sigma^4}\left(\sum_{i=1}^{d}\lambda_{1,i}^2+\sum_{i=1}^{d}\lambda_{2,i}^2+2\lambda_{1,1}\lambda_{2,1}\sum_{i=1}^{d}\cos^2\theta_i\right) \\
& -\frac{1}{4}\left[\sum_{i=1}^{d}\ln\left(1+\frac{\lambda_{1,i}}{\sigma^2}\right)+\sum_{i=1}^{d}\ln\left(1+\frac{\lambda_{2,i}}{\sigma^2}\right)\right] \\
= & \frac{1}{4}\left[\sum_{i=1}^{d}\frac{\lambda_{1,i}}{\sigma^2}-\sum_{i=1}^{d}\left(\frac{\lambda_{1,i}}{2\sigma^2}\right)^2-\sum_{i=1}^{d}\ln\left(1+\frac{\lambda_{1,i}}{\sigma^2}\right)\right]-\frac{1}{8\sigma^4}\lambda_{1,1}\lambda_{2,1}\sum_{i=1}^{d}\cos^2\theta_i \\
& +\frac{1}{4}\left[\sum_{i=1}^{d}\frac{\lambda_{2,i}}{\sigma^2}-\sum_{i=1}^{d}\left(\frac{\lambda_{2,i}}{2\sigma^2}\right)^2-\sum_{i=1}^{d}\ln\left(1+\frac{\lambda_{2,i}}{\sigma^2}\right)\right] \\
\triangleq & \frac{1}{\sigma^4}\left(c_3-\frac{1}{8}\lambda_{1,d}\lambda_{2,d}\sum_{i=1}^{d}\cos^2\theta_i\right).
\end{aligned}
\tag{A.11}
$$

Negating K and exponentiating gives theorem 2. □

A.3 Proof of Moderate SNR Case

Proof of Lemma 3 consider the function

$$
f(\lambda_i)=\ln(1+\lambda_i)-\ln(1+p)-\frac{1}{1+p}(\lambda_i-p)+\frac{1}{(1+p)^2}(\lambda_i-p)^2,
$$

defined in $[0,p]$. Its derivative is

$$
f'(\lambda_i)=\frac{1}{1+\lambda_i}-\frac{1}{1+p}+\frac{2(\lambda_i-p)}{(1+p)^2}=\frac{(p-\lambda_i)(p-1-2\lambda_i)}{(1+\lambda_i)(1+p)^2},
$$

which is positive in $\left[0,\frac{p-1}{2}\right)$ and negative in $\left(\frac{p-1}{2},p\right]$. Therefore, $f(\lambda_i)$ is monotonically increasing in $\left[0,\frac{p-1}{2}\right)$ and decreasing in $\left(\frac{p-1}{2},p\right]$. Further, $f(p)=0$ and $f(0)=-\ln(1+p)+\frac{p}{1+p}+\frac{p^2}{(1+p)^2}$ whose sign depends on the value of p. The shape of $f(\lambda_i)$ is now characterized. There exists $L<\frac{p-1}{2}$ such that $f(\lambda_i)\geq 0$ when $\lambda_i\in[L,p]$.

□

Before proving theorem 4, we need to bound λ_i using Weyl's inequality [44].

Lemma 16 (Weyl's inequality [44])**.** *Let* $\mathbf{M}$ *and* $\mathbf{P}$ *be two* $n \times n$ *Hermitian matrices, with eigenvalues* $\mu_1 \geq \cdots \geq \mu_n$ *and* $\nu_1 \geq \cdots \geq \nu_n$ *respectively. Denote the eigenvalues of* $\mathbf{M}+\mathbf{P}$ *by* $\gamma_1 \geq \cdots \geq \gamma_n$*. Then*

$$\max(\mu_i + \nu_n, \nu_i + \mu_n) \leq \gamma_i \leq \min(\mu_i + \nu_1, \nu_i + \mu_1).$$

Proof of Theorem 4 Since $\frac{p}{c(p)} \leq \frac{\lambda_{1,i}}{\sigma^2}, \frac{\lambda_{2,i}}{\sigma^2} \leq p$, by the Weyl's inequality, $\frac{p}{2c(p)} = \frac{p/c(p)+0}{2} \leq \lambda_i \leq \frac{p+p}{2} = p$. Further, since $1 \leq c(p) \leq \frac{p}{2L(p)}$, we have $\lambda_1, \ldots, \lambda_{2d-r} \in [L(p), p]$. By definition of $L(p)$, we can invoke Eq. (2.7) in Lemma 3 to obtain

$$\begin{aligned}
\ln\det\left(\frac{\mathbf{\Sigma}_1+\mathbf{\Sigma}_2}{2}\right) &= \sum_{i=1}^{2d-r}\ln(1+\lambda_i) + n\ln(\sigma^2) \\
&\geq (2d-r)\ln(1+p) + \frac{\operatorname{tr}\mathbf{D} - p(2d-r)}{1+p} - \frac{\operatorname{tr}\mathbf{D}^2 - 2p\operatorname{tr}\mathbf{D} + p^2(2d-r)}{(1+p)^2} + n\ln(\sigma^2).
\end{aligned} \tag{A.12}$$

Notice $\operatorname{tr}\mathbf{D} = \frac{1}{2}\sum_i\left(\frac{\lambda_{1,i}}{\sigma^2} + \frac{\lambda_{2,i}}{\sigma^2}\right)$, and by Eq. (A.8) and (A.9),

$$\operatorname{tr}\mathbf{D}^2 \leq \frac{1}{4\sigma^4}\left(\sum_i \lambda_{1,i}^2 + \lambda_{2,i}^2 + 2\lambda_{1,1}\lambda_{2,1}\sum_i\cos^2\theta_i\right).$$

Substituting these into Eq. (A.12), we get

$$\begin{aligned}
\ln\det\left(\frac{\mathbf{\Sigma}_1+\mathbf{\Sigma}_2}{2}\right) \geq& n\ln(\sigma^2) + (2d-r)\left[\ln(1+p) - \frac{p}{1+p} - \frac{p^2}{(1+p)^2}\right] \\
&+ \frac{1+3p}{2\sigma^2(1+p)^2}\left(\sum_i \lambda_{1,i} + \lambda_{2,i}\right) \\
&- \frac{1}{4\sigma^4(1+p)^2}\left(\sum_i \lambda_{1,i}^2 + \lambda_{2,i}^2 + 2\lambda_{1,1}\lambda_{2,1}\sum_i\cos^2\theta_i\right)
\end{aligned}$$

Substituting the above into the Bhattacharyya bound (2.4) yields an upper bound on P_e, of the form given in Theorem 4. In particular,

$$c_4 = \frac{1}{2}\left[\ln(1+p) - \frac{p}{1+p} - \frac{p^2}{(1+p)^2}\right],$$

and

$$\begin{aligned} c_5 = &- \frac{1+3p}{4\sigma^2(1+p)^2}\sum_i (\lambda_{1,i}+\lambda_{2,i}) + \frac{\sum_i \lambda_{1,i}^2 + \lambda_{2,i}^2}{8\sigma^4(1+p)^2} \\ &+ \frac{1}{4}\sum_i \left[\ln\left(1+\frac{\lambda_{1,i}}{\sigma^2}\right) + \ln\left(1+\frac{\lambda_{2,i}}{\sigma^2}\right)\right]. \end{aligned}$$

□

A.4 Analysis of NSC

Proof of Lemma 5 Since that the joint distribution of $[a_i\ a_j]^\top$, $[b_i\ b_j]^\top$, $[a_i\ b_j]^\top$ and $[a_i+b_i\ a_i-b_i]^\top$ are all Gaussian, it suffices to show that all covariance are diagonal. For any $i \neq j$,

$$\begin{bmatrix} a_i \\ a_j \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix}, \sigma^2\mathbf{I}_2\right) \quad \begin{bmatrix} b_i \\ b_j \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \cos\theta_i\alpha_i \\ \cos\theta_j\alpha_j \end{bmatrix}, \sigma^2\mathbf{I}_2\right)$$

$$\begin{bmatrix} a_i \\ b_j \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \alpha_i \\ \cos\theta_j\alpha_j \end{bmatrix}, \sigma^2\mathbf{I}_2\right).$$

For any i,

$$\begin{aligned} \begin{bmatrix} a_i \\ b_i \end{bmatrix} &\sim \mathcal{N}\left(\begin{bmatrix} \alpha_i \\ \cos\theta_i\alpha_i \end{bmatrix}, \sigma^2\begin{bmatrix} 1 & \cos\theta_i \\ \cos\theta_i & 1 \end{bmatrix}\right) \\ \begin{bmatrix} a_i+b_i \\ a_i-b_i \end{bmatrix} &\sim \mathcal{N}\left(\begin{bmatrix} (1+\cos\theta_i)\alpha_i \\ (1-\cos\theta_i)\alpha_i \end{bmatrix}, \right. \\ &\qquad \left. 2\sigma^2\begin{bmatrix} (1+\cos\theta_i) & 0 \\ 0 & (1-\cos\theta_i) \end{bmatrix}\right), \end{aligned} \tag{A.13}$$

which concludes the proof. □

Proof of Lemma 7 As $\sigma^2 \to 0$, the mean-covariance ratios of both $a_i + b_i$ and $a_i - b_i$ tend to infinity. Therefore, applying Lemma 6 to Eq. (A.13) (see proof of Lemma 5), we have $(a_i + b_i)(a_i - b_i) \sim \mathcal{N}\left(\sin^2\theta_i\alpha_i^2, 4\sigma^2\sin^2\theta_i(\alpha_i^2+\sigma^2)\right)$. Applying the independence between $(a_i + b_i)(a_i - b_i)$ and $(a_j + b_j)(a_j - b_j)$ $(i \neq j)$, we obtain the desired result by summing the mean and variance over all i. □

Proof of Theorem 8 We prove the theorem by deriving upper bounds on $\mathbb{P}\mathrm{r}(\mathcal{C}_2|\mathcal{C}_1, \boldsymbol{\alpha})$ and $\mathbb{P}\mathrm{r}(\mathcal{C}_1|\mathcal{C}_2, \boldsymbol{\alpha})$.

$$\begin{aligned}\mathbb{P}\mathrm{r}(\mathcal{C}_2|\mathcal{C}_1, \boldsymbol{\alpha}) &= \mathbb{P}\mathrm{r}\left(\sum_i (a_i + b_i)(a_i - b_i) \leq 0\right) \\ &= \mathbb{P}\mathrm{r}\left(\frac{\sum_i (a_i + b_i)(a_i - b_i) - \sum_i \sin^2\theta_i\alpha_i^2}{2\sigma\sqrt{\sum_i \sin^2\theta_i(\alpha_i^2+\sigma^2)}} \leq -\frac{\sum_i \sin^2\theta_i\alpha_i^2}{2\sigma\sqrt{\sum_i \sin^2\theta_i(\alpha_i^2+\sigma^2)}}\right).\end{aligned} \tag{A.14}$$

As $\sigma \to 0$, the term to the left of "$\leq$" in the last line of Eq. (A.14) is standard normal distributed. Therefore we can invoke the Gaussian tail bound to obtain

$$\begin{aligned}&\mathbb{P}\mathrm{r}(\mathcal{C}_2|\mathcal{C}_1, \boldsymbol{\alpha}) \\ &= \mathbb{P}\mathrm{r}\left(\frac{\sum_i (a_i + b_i)(a_i - b_i) - \sum_i \sin^2\theta_i\alpha_i^2}{2\sigma\sqrt{\sum_i \sin^2\theta_i(\alpha_i^2+\sigma^2)}} \geq \frac{\sum_i \sin^2\theta_i\alpha_i^2}{2\sigma\sqrt{\sum_i \sin^2\theta_i(\alpha_i^2+\sigma^2)}}\right) \\ &\leq \frac{1}{2}\exp\left[-\frac{\left(\sum_i \sin^2\theta_i\alpha_i^2\right)^2}{8\sigma^2\sum_i \sin^2\theta_i(\alpha_i^2+\sigma^2)}\right].\end{aligned} \tag{A.15}$$

$\mathbb{P}\mathrm{r}(\mathcal{C}_1|\mathcal{C}_2, \boldsymbol{\alpha})$ can be upper bounded in the same manner:

$$\mathbb{P}\mathrm{r}(\mathcal{C}_1|\mathcal{C}_2, \boldsymbol{\alpha}) \leq \frac{1}{2}\exp\left[-\frac{\left(\sum_i \sin^2\theta_i\alpha_i^2\right)^2}{8\sigma^2\sum_i \sin^2\theta_i(\alpha_i^2+\sigma^2)}\right]. \tag{A.16}$$

Therefore,

$$\begin{aligned} P_e =& \frac{1}{2}\int \mathbb{P}\mathrm{r}(\mathcal{C}_2|\mathcal{C}_1,\boldsymbol{\alpha})p(\boldsymbol{\alpha})d\boldsymbol{\alpha} + \frac{1}{2}\int \mathbb{P}\mathrm{r}(\mathcal{C}_1|\mathcal{C}_2,\boldsymbol{\alpha})q(\boldsymbol{\alpha})d\boldsymbol{\alpha} \\ \leq& \int \frac{1}{2}\exp\left[-\frac{\left(\sum_i \sin^2\theta_i\alpha_i^2\right)^2}{8\sigma^2\sum_i \sin^2\theta_i(\alpha_i^2+\sigma^2)}\right]\frac{p(\boldsymbol{\alpha})+q(\boldsymbol{\alpha})}{2}d\boldsymbol{\alpha} \qquad \text{(A.17)} \\ \triangleq& \int \mathcal{E}(\theta,\boldsymbol{\alpha},\sigma)\frac{p(\boldsymbol{\alpha})+q(\boldsymbol{\alpha})}{2}d\boldsymbol{\alpha}, \end{aligned}$$

which concludes the proof. □

A.5 Proof of Proposition 1

Observe that

$$\|\mathbf{X}^\top\mathbf{P}\mathbf{X}-\mathbf{T}\|_F^2 = \|(\mathbf{X}^\top\otimes\mathbf{X}^\top)\,\mathrm{vec}(\mathbf{P})-\mathrm{vec}(\mathbf{T})\|_2^2,$$

is a least squares problem with minimizer

$$\mathrm{vec}(\mathbf{P}^\star) = (\mathbf{X}^\top\otimes\mathbf{X}^\top)^\dagger\,\mathrm{vec}(\mathbf{T}) = \mathbf{X}^\top,$$

which can be rearranged to give

$$\mathbf{P}^\star = (\mathbf{X}^\top)^\dagger\mathbf{T}[(\mathbf{X}^\top)^\dagger]^\top = (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{T}\mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top)^{-1} \succeq 0.$$

□

Bibliography

[1] K. Abed-Meraim, A. Chkeif, Y. Hua, and S. Attallah. On a class of orthonormal algorithms for principal and minor subspace tracking. *J. of VLSI Sig Proc.*, 31:57–70, 2002.

[2] W. Allard, G. Chen, and M. Maggioni. Multi-scale geometric methods for data sets II: Geometric multi-resolution analysis. *App. and Comp. Harmonic Ana.*, 32(3):435 – 462, May 2011.

[3] L. A. Aroian. The probability function of the product of two normally distributed variables. *The Annals of Mathematical Statistics*, pages 265–271, 1947.

[4] A. Ashikhmin and R. Calderbank. Grassmannian packings from operator reed-muller codes. *IEEE transactions on Information Theory*, 56(11):5689–5714, 2010.

[5] L. Balzano, R. Nowak, and B. Recht. Online identification and tracking of subspaces from highly incomplete information. In *Proc. Allerton Conf. on Comm., Control and Comp.*, pages 704 – 711, Sept. 2010.

[6] A. Barron, J. Rissanen, and B. Yu. Minimum description length principle in coding and modeling. *IEEE Trans. Info. Theory*, 44(6):2743–2760, October 1998.

[7] P. Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):525–536, 1998.

[8] Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

[9] R. Basri and D. W. Jacobs. Lambertian reflectance and linear subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):218–233, 2003.

[10] M. Basseville and I. V. Nikiforov. *Detection of abrupt changes: Theory and applications*. Prentice Hall, April 1993.

[11] P. N. Belhumeur, J. P. Hespanha, and D. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[12] M. Belkin. *Problems of Learning on Manifolds*. PhD thesis, University of Chicago, 2003.

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

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

[15] A. Bellet and A. Habrard. Robustness and generalization for metric learning. *Neurocomputing*, 151(14):259–267, 2015.

[16] A. Bhattacharyya. On a meausre of divergence between two statistical populations defined by their probability distributions. *Bulletin of the Calcutta Mathematical Society*, 45:99–109, 1943.

[17] L. Breiman, J. Friedman, R. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, CA, 1983.

[18] D. Chen, X. Cao, L. Wang, F. Wen, and J. Sun. Bayesian face revisited: A joint formulation. In *European Conference on Computer Vision (ECCV)*, 2012.

[19] D. Chen, X. Cao, F. Wen, and J. Sun. Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[20] M. Chen, W. Carson, M. Rodrigues, R. Calderbank, and L. Carin. Communications inspired linear discriminant analysis. In *International Conference of Machine Learning*, 2012.

[21] M. Chen, J. Silva, J. Paisley, C. Wang, D. Dunson, and L. Carin. Compressive sensing on manifolds using a nonparametric mixture of factor analyzers: Algorithm and performance bounds. *IEEE Transactions on Signal Processing*, 58(12):6140–6155, 2010.

[22] Y. Chi, Y. C. Eldar, and R. Calderbank. Petrels: Parallel subspace estimation and tracking using recursive least squares from partial observations. *IEEE Transaction on Signal Processing*, 61(23):5947–5959, 2013.

[23] D. Cohn and G. Tesauro. How tight are the vapnik-chervonenkis bounds? *Neural Computation*, 4(2):249–269, 1992.

[24] J. A. Costa and A. O. Hero. Geodesic entropic graphs for dimension and entropy estimation in manifold learning. *IEEE Trans. Sig. Proc.*, 25(8):2210–2221, 2004.

[25] N. Cristianini, A. Elisseeff, J. Shawe-Taylor, and J. Kandola. On kernel-target alignment. In *Advances in Neural Information Processing Systems*, 2001.

[26] A. d'Aspremont, O. Banerjee, and L. El Ghaoui. First-order methods for sparse covariance selection. *SIAM. J. Matrix Anal. & Appl*, 30(56), 2008.

[27] D. Donoho. Cart and best-ortho-basis selection: A connection. *Annals of Stat.*, 25:1870–1911, 1997.

[28] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. New York, NY: Wiley-Interscience, 2000.

[29] A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

[30] E. Elhamifar and R. Vidal. Sparse subspace clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[31] E. Elhamifar and R. Vidal. Sparse manifold clustering and embedding. In *Advances in neural information processing systems*, 2011.

[32] T. Evgeniou, C. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

[33] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. San Diego: Academic Press, 1990.

[34] A. Globerson and S. Roweis. Metric learning by collapsing classes. In *Advances in Neural Information Processing Systems (NIPS)*, 2005.

[35] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems (NIPS)*, 2004.

[36] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio. Max-out networks. In *In Proceedings of the 30th International Conference on Machine Learning*, 2013.

[37] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *IEEE computer society conference on Computer vision and pattern recognition*, 2006.

[38] S. Haykin. *Adaptive Filter Theory*. Prentice Hall, 4th edition, 2001.

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

[40] X. He and P. Niyogi. Locality preserving projections. In *Neural information processing systems*, 2004.

[41] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504507, 2006.

[42] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[43] B. M. Hochwald, T. L. Marzetta, T. J. Richardson, W. Sweldens, and R. Urbanke. Systematic design of unitary space-time constellations. *IEEE Transactions on Information Theory*, 46(6):1962–1973, 2000.

[44] R. A. Horn and C. R. Johnson. *Matrix analysis*. Cambridge University Press, 2012.

[45] J. Hu, J. Lu, and Y. Tan. Discriminative deep metric learning for face verification in the wild. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1875–1882, 2014.

[46] J. Huang, Q. Qiu, R. Calderbank, and G. Sapiro. Geometry-aware deep transform. In *International Conference on Computer Vision*, 2015.

[47] S. Ji, Y. Xue, and L. Carin. Bayesian compressive sensing. *IEEE Transactions on Signal Processing*, 56(6):2346–2356, 2008.

[48] W. Johnson and J. Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. In *Conference in Modern Analysis and Probability*, 1982.

[49] S. M. Kakade, O. Shamir, K. Sridharan, and A. Tewari. Learning exponential families in high-dimensionals: Strong convenxity and sparsity. In *Proc. of Int. Conf. on Artificial Intel. and Stats*, pages 381 – 388, 2010.

[50] J. Kappenman. A perfect storm of planetary proportions. *IEEE Spectrum*, 49(2):26 – 31, Feb. 2012.

[51] D. L. Kleinman and M. Athans. The design of suboptimal linear time-varying systems. *IEEE Transaction on Automatic Control*, 13(2):150–159, 1968.

[52] V. Koltchinskii and D. Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. *Annals of Statistics*, 30(1):1–50, 2002.

[53] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *Proc. of SIGCOMM*, 2004.

[54] K.-C. Lee and D. Kriegman. Online learning of probabilistic appearance manifolds for video-based recognition and tracking. In *Proc. of CVPR*, pages 852 – 859, 2005.

[55] M. L. McCloud and L. L. Scharf. A new subspace identification algorithm for high-resolution doa estimation. *IEEE Transactions on Antennas and Propagation*, 50(10):1382–1390, 2002.

[56] R. Meir. Nonparametric time series prediction through adaptive model selection. *Machine Learning*, 39:5–34, 2000.

[57] N. Merhav and M. Feder. Universal prediction. *IEEE Trans. Info. Theory*, 44(6):2124–2147, October 1998.

[58] B. Neyshabur, R. Tomioka, and N. Srebro. Norm-based capacity control in neural networks arxiv preprint arxiv:1503.00036 (2015). In *The 28th Conference on Learning Theory (COLT)*, 2015.

[59] M. Nokleby, M. Rodrigues, and R. Calderbank. Discrimination on the grassmann manifold: Fundamental limits of subspace classifiers. *IEEE Transaction on Information Theory*, 61(4):2133–2147, 2015.

[60] E. S. Page. Continuous inspection scheme. *Biometrika*, 41(1/2):100 – 115, June 1954.

[61] M. Pollak and D. Siegmund. Sequential detection of a change in a normal mean when the initial value is unknown. *Annals of Stats.*, 19(1):394 – 416, 1991.

[62] H. V. Poor and O. Hadjiliadis. *Quickest detection*. Cambridge University Press, Dec. 2008.

[63] Q. Qiu and G. Sapiro. Learning transformations for clustering and classification. *Journal of Machine Learning Research*, 16:187–225, 2015.

[64] M. Qu, F. Y. Shih, J. Jing, and H. Wang. Automatic solar filament detection using image processing techniques. *Solar Physics*, (1-2):119–135, 2005. DOI: 10.1007/s11207-005-5780-1.

[65] M. Raginsky, R. Willett, C. Horn, J. Silva, and R. Marcia. Sequential anomaly detection in the presence of noise and limited feedback. *IEEE Trans. Info. Theory*, 58(8):5544 – 5562, Aug. 2012.

[66] S. R. Rao, R. Tron, R. Vidal, and Y. Ma. Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[67] H. Reboredo, F. Renna, R. Calderbank, and M. Rodrigues. Compressive classification of a mixture of gaussians: Analysis, designs and geometrical interpretation. arXiv preprint arXiv:1401.6962, 2014.

[68] F. Renna, R. Calderbank, L. Carin, and M. Rodrigues. Reconstruction of signals drawn from a gaussian mixture via noisy compressive measurements. *IEEE Transactions on Signal Processing*, 62(9):2265–2277, 2014.

[69] J. D. M. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *International Conference of Machine Learning*, 2005.

[70] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 22(5500):2323–2326, 2000.

[71] D. Siegmund. *Sequential Analysis: Test and Confidence Intervals*. Springer, Aug. 1985.

[72] D. Siegmund and E. S. Venkatraman. Using the generalized likelihood ratio statistic for sequential detection of a change-point. *Annals of Stat.*, 23(1):255 – 271, 1995.

[73] T. Sim, S. Baker, and M. Bsat. The cmu pose, illumination, and expression (pie) database. In *IEEE International Conference on Automatic Face and Gesture Recognition*, 2002.

[74] C. Sumit, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 539–546, 2005.

[75] Y. Sun, Y. Chen, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1988–1996, 2014.

[76] Y. Sun, X. Wang, and X. Tang. Deep learning face representation from predicting 10,000 classes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1891–1898, 2014.

[77] Y. Taigman, M. Yang, M. A. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1701–1708, 2014.

[78] V. Tarokh, N. Seshadri, and R. Calderbank. Space-time codes for high data rate wireless communication: Performance criterion and code construction. *IEEE Transactions on Information Theory*, 44(2):744–765, 1998.

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

[80] V. N. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999.

[81] M. B. Wakin. Manifold-based signal recovery and parameter estimation from compressive measurements. submitted, 2009.

[82] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus. Regularization of neural networks using dropconnect. In *In Proceedings of the 30th International Conference on Machine Learning*, 2013.

[83] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, 2009.

[84] J. Weston, F. Ratle, and R. Collobert. Deep learning via semi-supervised embedding. *International Conference on Machine Learning*, 2008.

[85] R. Willett and R. Nowak. Multiscale Poisson intensity and density estimation. *IEEE Trans. Info. Theory*, 53(9):3171–3187, 2007.

[86] T. Wimalajeewa, H. Chen, and P. K. Varshney. Performance limits of compressive sensing-based signal classification. *IEEE Transactions on Signal Processing*, 60(6):2758–2770, 2012.

[87] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Yi Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):210–227, 2009.

[88] Y. Xie, J. Huang, and R. Willett. Change-point detection for high-dimensional time series with missing data. *IEEE Journal of Selected Topics in Signal Processing*, 7(1):12–27, 2013.

[89] Y. Xie and D. Siegmund. Parallel sequential multisensor changepoint detection. In *Joint Stats. Meeting (JSM)*, San Diego, 2012.

[90] Y. Xie and D. Siegmund. Sequential multi-sensor change-point detection. *submitted to Annals of Statis*, June 2012.

[91] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems (NIPS)*, 2002.

[92] H. Xu and S. Mannor. Robustness and generalization. *Machine Learning*, 86(3):391–423, 2012.

[93] G. Yu and G. Sapiro. Statistical compressed sensing of gaussian mixture models. *IEEE Transactions on Signal Processing*, 59(12):5842–5858, 2011.

[94] G. Yu, G. Sapiro, and S. Mallat. Solving inverse problems with piecewise linear estimators: from gaussian mixture models to structured sparsity. *IEEE Transactions on Image Processing*, 21(5):2481–2499, 2012.

[95] K. Yu, W. Xu, and Y. Gong. Deep learning with kernel regularization for visual recognition. In *Advances in Neural Information Processing Systems*, 2009.

[96] Z. Zha, T. Mei, M. Wang, Z. Wang, and X. Hua. Robust distance metric learning with auxiliary knowledge. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

Biography

Jiaji Huang was born in Changzhou, Jiangsu, on July 27th, 1990. He received the B.S. degree in electrical engineering from the University of Science and Technology of China (USTC), Hefei, China, in 2011, and master's degree in electrical and computer engineering from Duke University, Durham, NC, in 2013. His research interest lies in the intersection of signal processing, machine learning and information theory. He is especially interested in understanding how the geometry of signals assists learning tasks. His representative publications include (but not limited to):

J. Huang, Q. Qiu and R. Calderbank. The Role of Principal Angles in Subspace Classification. IEEE transaction on Signal Processing, 64(8), 1933-1945, 2015.

L. Wang*, **J. Huang***, X. Yuan*, K. Krishnamurthy, J. Greenberg, V. Cevher, M. Rodrigues, D. Brady, R. Calderbank, and L. Carin. Signal Recovery and System Calibration from Multiple Compressive Poisson Measurements, SIAM Journal on Imaging Sciences (SIIMS), vol. 8, no. 3, 1923-1954. ($*$: equal contribution)

Y. Xie, **J. Huang**, and R. Willett. Changepoint detection for high-dimensional time series with missing data, IEEE Journal of Selected Topics on Signal Processing (J-STSP), vol. 7, no. 1, pp. 12-27. 2013.

J. Huang, Q. Qiu, R. Calderbank and G. Sapiro. *GraphConnect*: A Regularization Framework for Neural Networks. Submitted to ICML 2016.

J. Huang, Q. Qiu, R. Calderbank and G. Sapiro. Discriminative Robust Transformation Learning. Neural Information Processing Systems (NIPS), 2015.

J. Huang, Q. Qiu, R. Calderbank and G. Sapiro. Geometry-aware Deep Transform. International Conference on Computer Vision (ICCV), 2015.