

**Distributional Approximation of the Classification Accuracy
and Gaussian Mixture Models for Deep Learning**

by

Ehsan Variani

A dissertation submitted to The Johns Hopkins University in conformity with the
requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

April, 2015

© Ehsan Variani 2015

All rights reserved

Abstract

While the Deep Neural Networks (DNNs) have led significant improvement boost in many recognition tasks, their inherit structure and training algorithm has not significantly changed from traditinal Neural Networks. Sequence of linear and non-linear layers are trained together using the Back Propagation algorithm.

The idea of deep learning can be compared to the idea of flow diagrams and pipelines in the engineering design. From this point of view, the deep net components are not restricted to be simple linear or non-linear functions. More complicated components can be designed for DNNs and *jointly* trained within the deep structure. This thesis introduces two of such components: the Gaussian Mixture Model (GMM) layer and the a novel Classification layer.

Each neuron in the Gaussian Mixture Model layer outputs a GMM likelihood. This provides many new possibilities for deep learning. First, it provides a deep density estimation using a mixture of Gaussian distributions. In addition, it brings the pos-

ABSTRACT

sibility of having trainable non-linearities which have parameters trained through the coarse of training.

The Classification layer preserves a simple idea of classification objective; finding parameters which minimize the probability of classification error. In the proposed two step process, the distribution of classification accuracy is first approximated and then used to compute the classification error. The classification parameters are then computed to minimize this approximated error. Using the Central Limit Theorem, it is shown that the classification layer parameters obey a closed form equation for the case of binary classification. For the multiclass, the parameters are trained using the back propagation algorithm.

Both layers are experimented in several recognition tasks such as speech, image and diseases recognition. Competitive results are demonstrated using these components with state of the art recognition techniques in these tasks. Furthermore, significant part of this thesis discussed the theoretical aspects of the proposed components.

Primary Reader: Bruno M. Jedynek

Secondary Reader: Gerard G. L. Meyer

Dedication

This thesis is dedicated to my parents.

Contents

Abstract	ii
List of Tables	ix
List of Figures	x
1 Introduction	1
1.1 Pattern Recognition Systems	2
1.1.1 Feature Engineering	2
1.1.2 Classification	3
1.1.3 Separate vs. Joint Training	4
1.2 Deep Neural Networks	6
1.2.1 Linear and Non-Linear Layers	7
1.2.2 Final Hidden Layer	9
1.2.3 Softmax	10
1.2.4 Cross Entropy Loss	12

CONTENTS

1.2.5	Discussion	13
1.3	Gaussian Mixture Model Layer (GMM Layer)	15
1.3.1	Deep Gaussian Mixture Model	15
1.3.2	Density Estimation	16
1.3.3	Trainable Non-linearities	17
1.4	Distributional Approximation of the Classification Accuracy (DACA)	17
1.4.1	Closed Form Equation	19
1.4.2	Deep Neural Network with DACA loss	20
2	Deep Gaussian Mixture Models	23
2.1	Introduction	24
2.2	A GMM layer for deep learning	27
2.3	Training a Neural Network with GMM layer	31
2.4	Experiments	32
2.5	Conclusions	40
3	Toward Minimization of the Classification Error	41
3.1	Introduction	42
3.2	$C^{-1}\mu$ Classifier	46
3.2.1	Central Limit Theorem Assumptions	51
3.2.2	Estimation of the Covariance Matrix	58
3.2.3	Properties	60

CONTENTS

3.2.4	A Toy Example	61
3.3	Non-linear Extension	64
3.3.1	Neural Networks	67
3.3.2	Reproducing Kernel Hilbert Space (RKHS)	67
3.4	Experiments	68
3.4.1	Data	69
3.4.2	Parkinson Recognition	70
3.4.3	Character Recognition	71
3.4.4	Error Prediction	73
3.5	Conclusion	73
4	Multiclass Classification with $C^{-1}\mu$	75
4.1	Introduction	76
4.2	$C^{-1}\mu$: Multi-Class Extension	77
4.2.1	Notations	78
4.2.2	Probability of the Misclassification	81
4.2.3	Bounds on Probability of Error	81
4.3	Experiments	88
4.3.1	PCA + $C^{-1}\mu$	89
4.3.2	Neural Network + $C^{-1}\mu$	93
4.4	Conclusion	96

CONTENTS

5	Conclusion	97
5.0.1	Deep Gaussian Mixture Model	97
5.0.2	Distributional Approximation of the Classification Accuracy .	98
	Vita	111

List of Tables

2.1	Separate versus joint optimization of model and features in deep architectures (is-is).	34
2.2	Separate versus joint optimization of features and classification for shallow GMMs (is-is).	35
2.3	Effect of depth on DGMM performance (is-is).	36
2.4	NN and GMM results using shallow and deep structures, with matched # of parameters, and no pre-training (is-is).	37
2.5	Results for pre-trained DNNs and DGMMs.	38
2.6	Deep Subspace GMM experiments (is-is).	39
3.1	Comparison of the misclassification rate for the separate and joint recognition system.	72
3.2	Comparison of the misclassification rate for the separate and joint recognition system using kernel method.	72
3.3	Comparison of the misclassification rate and the prediction rate for three systems.	73

List of Figures

1.1	Pattern recognition systems.	5
1.2	The layering architecture of the Gaussian Mixture Model (GMM) for speech recognition.	6
1.3	Deep Neural Network.	7
1.4	The topology of the final layers of a DNN. These layers can be considered as classification layers.	9
1.5	Each row corresponds to a classifier. The upper classifier is perfect while the lower one has 50% classification accuracy.	13
1.6	Description of the Distributional Approximation the Classification Accuracy (DACA)	19
1.7	The classification layers with Distributional Approximation the Classification Accuracy (DACA) classifier.	21
1.8	Deep Neural Network with DACA loss.	22
2.1	The Gaussian Mixture Model (GMM) layer: n nodes with g Gaussian components per node, and associated μ , Σ & ω parameters.	25
2.2	GMM optimization within overall NN structure.	30
2.3	Integrating a Subspace Gaussian Mixture Model (SGMM) into a DGMM architecture: the Deep SGMM (DSGMM).	38
3.1	The $C^{-1}\mu$ classifier and the LDA classifier agree perfectly in this toy example except for the constant parameter b_0 which is plotted for both estimators. Note that the two estimators are close when the prior for class 1, π_1 is close to 0.5. however, they disagree when the prior is very skewed.	62
3.2	Misclassification rate versus number of principle components.	70
4.1	Effect of regularization term for shrinking the covaraince matrix. PCA with 10 dimension is used for feature extraction.	90

LIST OF FIGURES

4.2	The proposed upper bound is minimized over number of iterations. The PCA dimension of 40 is used as feature with $\lambda = 0$	91
4.3	The probability of error versus number of PCA dimension.	93
4.4	The probability of error versus number of iterations for single linear hidden layer and the $C^{-1}\mu$ loss.	94
4.5	The probability of error versus number of iterations for single sigmoid hidden layer and the $C^{-1}\mu$ loss.	95

Chapter 1

Introduction

In other engineering areas, the idea of using pipelines, flow diagrams and layered architectures to build complex systems is quite well entrenched, and our field should be working (inter alia) on principles for building such systems. The word deep just means that to me: layering (and I hope that the language eventually evolves toward such drier words). I hope and expect to see more people developing architectures that use other kinds of modules and pipelines, not restricting themselves to layers of neurons.

(Michael Jordan on deep learning)

1.1 Pattern Recognition Systems

Pattern recognition is the problem of *feature extraction* and *classification*. Informative features as well as efficient classification techniques are required together to design an accurate pattern recognition system.

1.1.1 Feature Engineering

The *feature extraction* is the first module upstream of any pattern recognition system, see Figure 1.1. Starting with a set of raw measurements, the objective of a feature engineering technique is deriving *features* X , that intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps.

Significant amount of research has been devoted toward *feature engineering* techniques. Sometimes the technique is motivated by investigating how human perception process the pattern to be recognized. The Mel Frequency Cepstral Coefficients (MFCCs) is one example of such feature engineering techniques widely used in Automatic Speech Recognition (ASR) tasks, [1]. There also exists feature engineering techniques which are designed by the constraints imposed by the recognition problem. Many *dimensionality reduction* techniques such as Principal Component Analysis (PCA) can be viewed as such methods, [2]. For high dimensional tasks

CHAPTER 1. INTRODUCTION

where the number of samples is significantly smaller than the number of measurements, the dimensionality reduction techniques are required to only extract the few most informative features. In the case of PCA these informative features are the ones corresponding to the PCA components with the largest eigen-values [2, 3].

1.1.2 Classification

The main objective of a recognition system is to *recognize* the feature pattern by *classifying* it into one out of many predefined classes. For a feature vector $X \in \mathbb{X}$, the most relevant class $\hat{Y} \in \mathbb{Y}$ is selected among all the $C = |\mathbb{Y}|$ classes through a decision function $\mathbb{f} : \mathbb{X} \rightarrow \mathbb{Y}$. Depending on the assumptions the decision function $\mathbb{f}$ is founded on, classifiers can be categorized into parametric and non-parametric methods [4, 5].

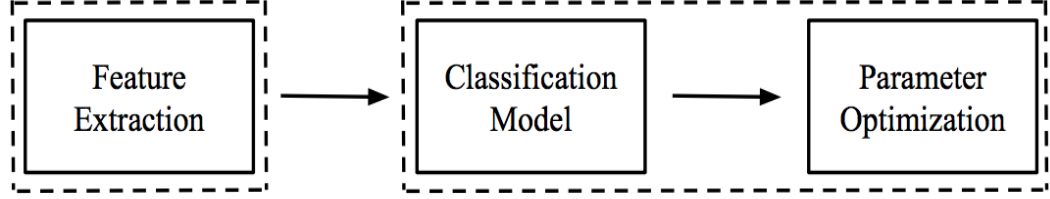
The parametric methods make assumptions about the underlying probabilistic properties of feature space, $P(X)$ or the conditional probability of class given feature, $P(Y|X)$. The Naive Bayes classifier assumes the value of a particular feature is independent of the value of any other feature, given the class variable, [6]. The Linear Discriminant Analysis (LDA) assumes the conditional distribution $P(X|Y)$ is distributed under a multi-variate Normal distribution with equal covariance matrices [7]. The Logistic Regression classifiers assumes the logarithm of the conditional,

CHAPTER 1. INTRODUCTION

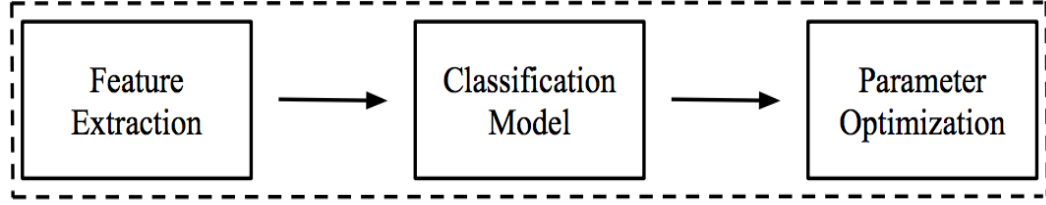
$\log(P(Y|X))$, to be a linear function of the feature vector, X , [8]. In some cases like LDA, the probabilistic assumptions can be further used to derive an estimation of the probability of misclassification error [9, 10]. However, this holds if the earlier Normal assumption of $P(X|Y)$ holds at the first place. The non-parametric methods do not make any probabilistic assumptions about features or conditional distribution of classes given features. The Perceptron [8] and Support Vector Machines [11] learn the linear mapping directly from training data are two well known examples of such methods.

1.1.3 Separate vs. Joint Training

The final stage of the pattern recognition module is the parameter optimization. Both feature extraction and the classification module have set of parameters which required to be optimized for the desired recognition task. This optimization is usually done separately. In conventional systems of Figure 1.1a, the features are first extracted *independent* of the downstream classification. The classifier parameters are then optimized with the *supervision* of the sample labels. Until most recently, practical speech recognition systems were categorized among the separate recognition systems. Figure 1.2 presents a conventional ASR system with the feature extraction module at the first stage and the Gaussian Mixture Model at the final stage. In ASR, the aim of designing features which are suitable for GMM modeling has been a major research



(a) Conventional recognition systems with separate optimization.



(b) Discriminative recognition systems with joint optimization.

Figure 1.1: Pattern recognition systems.

stream [12]. The MFCC features are widely used for this purpose [13, 14]. The GMM parameters were then trained using Maximum Likelihood optimization methods such as Expected Maximization algorithm, [15]. Another example of such separate systems is the Hybrid approach in ASR, which use the feature extracted from a separately trained Neural Network to train a GMM model [12]. The fact that features were extracted separately from the recognition task, brought a lot of additional feature engineering methods in the separately designed recognition systems. Adaptation is one example of such feature engineering methods [16].

In contrast to separate recognition systems, the *joint* recognition systems are designed to optimize the feature extraction and the classification module parameters all together, *jointly* as shown in Figure1.1b. To the best knowledge of the author, the first example of such method is the general concept of Discriminative Feature

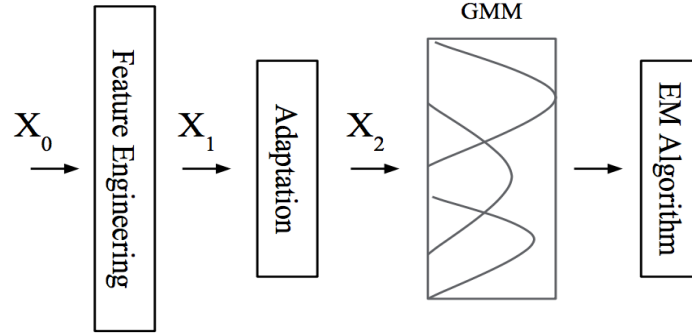


Figure 1.2: The layering architecture of the Gaussian Mixture Model (GMM) for speech recognition.

Extraction (DFE), in which both the feature extraction and classification modules are optimized jointly [17, 18].

The concept of Deep Neural Network can also be viewed from the joint vs separate recognition perspectives. These models usually do not require any further feature engineering. The raw data can be fed directly to these models and all parameters of the model are then optimized together. This conjecture has been extensively experimented in Chapter 2.

1.2 Deep Neural Networks

Recently, the Deep Neural Network (DNN) has been introduced as a recognition paradigm that beats many other recognition techniques by large margin [19, 20, 21, 22]. The Deep Neural Networks consist of a hierarchy of linear and non-linear layers

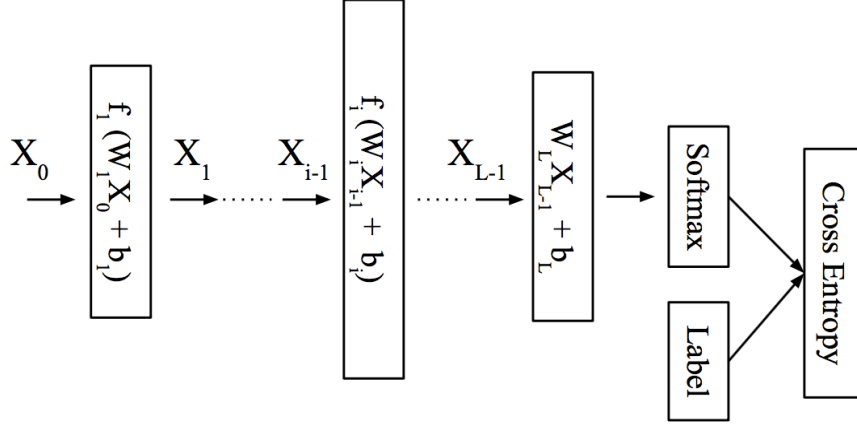


Figure 1.3: Deep Neural Network.

stacked together to model the conditional density $P(Y|X)$ from raw measurements X . The structure of such system is shown in Figure 1.3. These models have been widely used recently because of their superior performance, no need of extensive feature engineering and the availability of vast amount of training data and computational resources [23, 24]. Here, we briefly describe the main components of a deep neural network.

1.2.1 Linear and Non-Linear Layers

Deep neural networks consist of *linear* and *non-linear* layers. A linear layer is an affine transform of its input X_{i-1} :

$$X_i = W_iX_{i-1} + b_{i-1} \quad (1.1)$$

CHAPTER 1. INTRODUCTION

A non-linear layer, is equivalent to a non-linear function f , such that $X_i = f_i(X_{i-1})$. Different non-linearities has shown to provide performance gains on different tasks [25, 26]. Most non-linearities are functions with preset parameters. However, it is also possible to design *trainable non-linearities* which themselves are trained during the training phase. However, such non-linearities can be easily modeled by a linear layer together with non-linear layer right after it.

Figure 1.3 show the general structure of Deep Neural Network. The deep structure can contain sequence of linear and non-linear layers in different order and dimensionality.

While there is no limitation on size or form of the initial hidden layers of DNN, the final layers need to obey some specific structure. These layers consist of a *linear* layer, *softmax* layer and finally the *cross entropy* loss. The topology of the final layers of DNN is shown in Figure 1.4. The linear layer transforms the final hidden layer of dimension d to a C dimensional vector with a $C \times d$ dimensional weight matrix W_{L+1} . The softmax activations are then used to transforms the input to a vector which has the probability properties, namely positivity and sum to one. This is then compared with the true label codeword using the cross entropy loss.

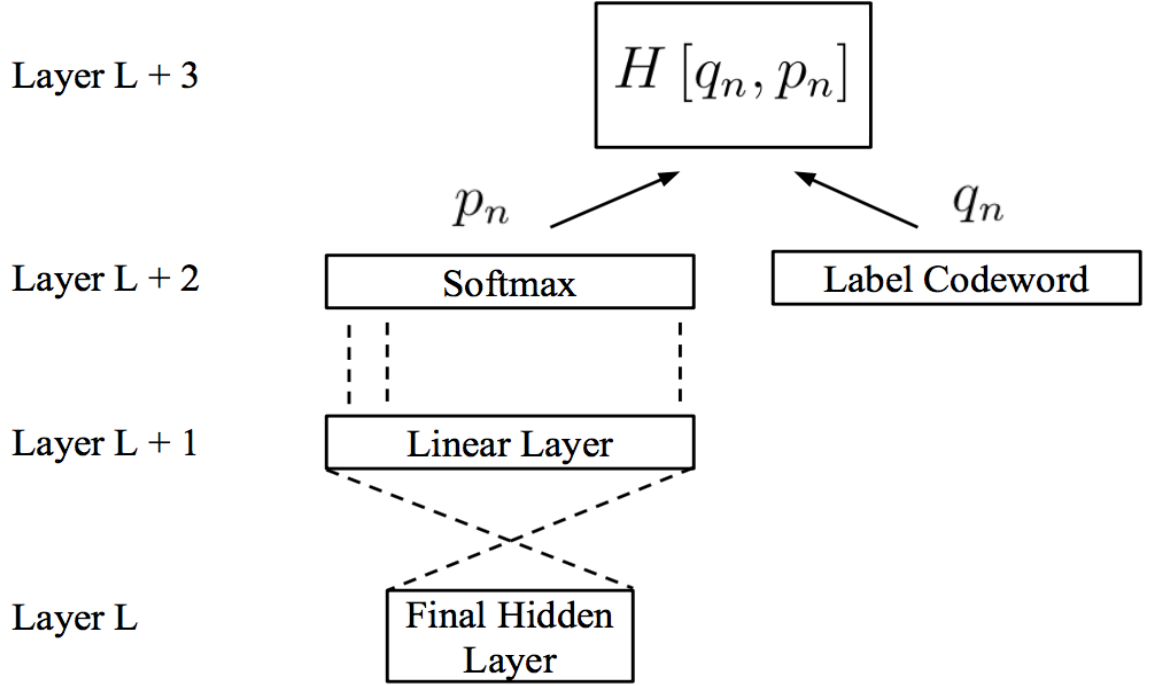


Figure 1.4: The topology of the final layers of a DNN. These layers can be considered as classification layers.

1.2.2 Final Hidden Layer

The layer immediately after the final hidden layer of a deep neural network is a linear layer. This layer connects a previous layer X_L to the softmax layer which is explained in the next subsection. Assuming the dimension of the X_L is d , the number of parameters of this linear layer is $C \times d$, where C is the number of classes. In other words, each class, has d parameters which is not shared with other classes. This is the beauty of the deep structure. All the parameters up to this layer are shared among all classes while the final hidden layer provide parameters per class which is

CHAPTER 1. INTRODUCTION

not shared among the other classes.

1.2.3 Softmax

The normalized exponential or *softmax* activation first introduced by [27]. For an input $X_{L+1} \in \mathbb{R}^C$, this non-linear function, outputs:

$$P(c|X_{L+1} = x) = \frac{\exp(x_c)}{\sum_{c'=1}^C \exp(x_{c'})} \quad (1.2)$$

Winner takes all

Softmax is a smooth version of the *winner-takes-all* activation model [28] in which the unit with the largest input has the output +1 while other units have output 0. If the numerator of the above function is replace by $\exp(\alpha x_c)$, then the winner-takes-all model is recovered as $\alpha \rightarrow \infty$.

Generalization of Sigmoid

The softmax activation function can be regarded as the generalization of the sigmoid function:

$$P(c|X_{L+1} = x) = \frac{1}{1 + \exp(-x'_c)} \quad (1.3)$$

CHAPTER 1. INTRODUCTION

where

$$x'_c = x_c - \log \left(\sum_{c' \neq c} \exp(x_{c'}) \right) \quad (1.4)$$

This generalization provide the intuition behind softmax. Similar to the sigmoid function in logistic regression, this can be interpreted as the posterior probability of class c given X .

Probabilistic Interpretation

The softmax is a multivariation function that map its input to an output with the same dimension for which, *all elements are positive* and further more *they sum to one*. However, are these enough to call softmax a probability function? The author answer is No.

Lets assume the class conditional densities belong to the family of the exponential family:

$$P(X_{L+1} = x|C = c) = \exp \left(A(\theta_c) + B(X_{L+1}, \phi) + \theta_c^T X_{L+1} \right) \quad (1.5)$$

using the Bayes' theorem,

$$P(C = c|X_{L+1} = x) = \frac{P(x|C = c)P(C = c)}{\sum_{c'} P(x|C = c')P(C = c')} \quad (1.6)$$

CHAPTER 1. INTRODUCTION

Substituting Eq 1.5 into Eq 1.6,

$$P(C = c|X_{L+1} = x) = \frac{\exp(a_k)}{\sum_{k'} \exp(a_{k'})} \quad (1.7)$$

where

$$a_k = \theta_k^T x + (A(\theta_k) + \log(P(C = c))) \quad (1.8)$$

which is the final layer of the neural network and show that given the exponential assumption, the outputs of softmax can be interpreted as probabilities of class membership, conditioned on the outputs of the hidden units. This is one of the main motivation behind proposing the GMM layer for DNN in this thesis.

1.2.4 Cross Entropy Loss

The cross entropy is the major loss used in practice to optimize the DNN parameters

$$H[q, p] = - \sum_{c=1}^C q_c \log(p_c) \quad (1.9)$$

where p is the posterior estimated from softmax layer and q is the codeword assigned to the class label y . This is a 1-hot vector, has one at index y and zero otherwise. The cross entropy is a divergence measure which is zero if and only if $p = q$. This means the minimum loss is achieved when the model is able to produce vectors which are exactly similar to the label codeword.

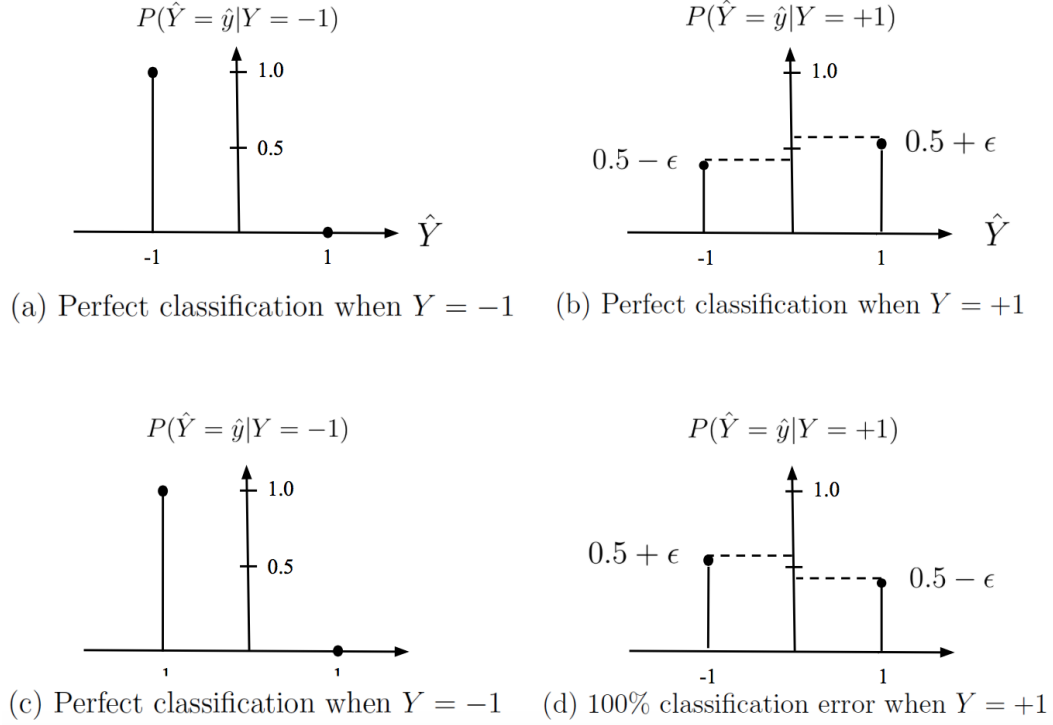


Figure 1.5: Each row corresponds to a classifier. The upper classifier is perfect while the lower one has 50% classification accuracy.

1.2.5 Discussion

The cross entropy loss takes its minimum value only if the distribution of p is exactly similar to the q . In the winner-takes-all scenario, there is only one element of q which wins everything while the other elements are all zero. However, is there any guarantee that the model learns such a vector for p ?

Figure 1.5 presents the softmax probabilities for two different classifiers. The y-axis of these curves is $P(\hat{Y} = \hat{y} | Y = y)$ which is the posterior of classification decision, $\hat{y}$, given the true class, y . Each row in this figure presents one classification scenario.

CHAPTER 1. INTRODUCTION

Based on the posteriors both classifiers can perfectly do the classification when $Y = -1$. The first classifier can also correctly classify all samples belong to the class $Y = +1$. However, the second classifier is not able to classify any samples corresponding to the class $Y = +1$ simply because the posteriors for class -1 is larger than posteriors for class $+1$. For both classifiers, the absolute difference $|P(\hat{Y} = +1|Y = +1) - P(\hat{Y} = -1|Y = +1)|$ is 2ϵ , where ϵ can take any positive value. Lets compute the cross entropy loss for these classifiers. We assume the prior is $P(Y = 1) = p$. For the first classifier

$$\begin{aligned} H_1 &= P(Y = 1)H \left[[1, 0], \left[\frac{1}{2} + \epsilon, \frac{1}{2} - \epsilon \right] \right] + P(Y = -1)H [[0, 1], [0, 1]] \\ &= -p (\log (0.5 + \epsilon) - \log (Z)) \end{aligned} \quad (1.10)$$

where $Z = \exp (0.5 + \epsilon) + \exp (0.5 - \epsilon)$. For the second classifier:

$$\begin{aligned} H_2 &= P(Y = 1)H \left[[1, 0], \left[\frac{1}{2} - \epsilon, \frac{1}{2} + \epsilon \right] \right] + P(Y = -1)H [[0, 1], [0, 1]] \\ &= -p (\log (0.5 - \epsilon) - \log (Z)) \end{aligned} \quad (1.11)$$

It is clear that for $\epsilon \rightarrow 0$ the two cross entropies H_1 and H_2 become equal. In terms of loss, both perfect and poor classifier lead to same average loss value! This clearly shows one concern with cross entropy loss. Better losses are required to handle this problem. This is the other contribution of this thesis in Chapter 3 and 4.

1.3 Gaussian Mixture Model Layer (GMM Layer)

The second chapter of this thesis is devoted to introduce the concept of the Gaussian Mixture Model Layer (GMM) layer [24]. The main contribution of this layer to the deep learning community will be summerized in the following subsections.

1.3.1 Deep Gaussian Mixture Model

Gaussian Mixture Model (GMM) represented as the last layer of a Deep Neural Network (DNN) architecture and jointly optimized with all previous layers using Asynchronous Stochastic Gradient Descent (ASGD). The resulting Deep GMM architecture was investigated with special attention to the following issues: (1) The extent to which joint optimization improves over separate optimization of the DNN-based feature extraction layers and the GMM layer; (2) The extent to which depth (measured in number of layers, for a matched total number of parameters) helps a deep generative model based on the GMM layer, compared to a vanilla DNN model; (3) Head-to-head performance of Deep GMM architectures vs. equivalent DNN architectures of comparable depth, using the same optimization criterion (frame-level Cross Entropy (CE)) and optimization method (ASGD); (4) Expanded possibilities for modeling of-

ferred by the Deep GMM generative model. The proposed Deep GMMs were found to yield Word Error Rates (WERs) competitive with state-of-the-art DNN systems, at the cost of pre-training using standard DNNs to initialize the Deep GMM feature extraction layers. An extension to Deep Subspace GMMs is described, resulting in additional gains.

1.3.2 Density Estimation

The GMM layer is easily configured to estimate the underlying density of any set feature through a mixture of gaussians. Using the GMM layer on the top of neural network, we propose a technique that estimate feature extraction as well as GMM parameters jointly together. This can be used for supervised and unsupervised tasks.

Due to the poor quality of features, most traditional Gaussian Mixture estimation methods require huge number of densities in the mixture. However, we show that our method require only few number of Gaussian mixture because of the deep power of feature extraction in the Deep GMM model.

1.3.3 Trainable Non-linearities

Non-linearities in neural network are the considered as major part of learning algorithm. The GMM layer can be used a non-linearity and with the parameters trained jointly in the model. Instead of huge amount of tuning effort, one can simply learn what need to be learned. This is addressed in Chapter 2.

1.4 Distributional Approximation of the Classification Accuracy (DACA)

A new classification technique is introduced in the Chapters 3 and 4. This classifier is based on the three simple steps.

- Modeling the accuracy even: A *detector* variable t is introduced which capture the event that accurate classification happen. This $C - 1$ dimensional vector is defined such that the error occurs if there is at least one element of t which is not positive. In other words, the event is classified as correct if $t \geq \mathbf{0}$, where $\mathbf{0}$ is also $C - 1$ dimensional zero vector and the inequality should hold for all elements.

CHAPTER 1. INTRODUCTION

- Approximating the accuracy density: The density of the variable t is estimated through an iterative process. Normal mixtures are considered for the density estimation. This involves the usage of the GMM layer.
- Computation and minimization of the probability of the error: Using the estimated density, we approximate the probability of error. The derivative with respect to model parameters are then derived and back-propagated through the model to optimize the parameters.

Figure 1.6 presents a schematic description of the proposed method. For the detector variable t , the empirical distribution of t is shown in the first figure. This empirical distribution is then estimated with a single Gaussian red curve. As the third figure shows, the error corresponds to the shaded area. The loss is the shaded area under the curve and the proposed classifier minimize this area.

The classifier name comes from the abbreviation of the methodology behind the method: Distributional Approximation of the Classification Accuracy (DACA). The underlying distribution of the accuracy event is approximated which is used to minimize the probability of error.

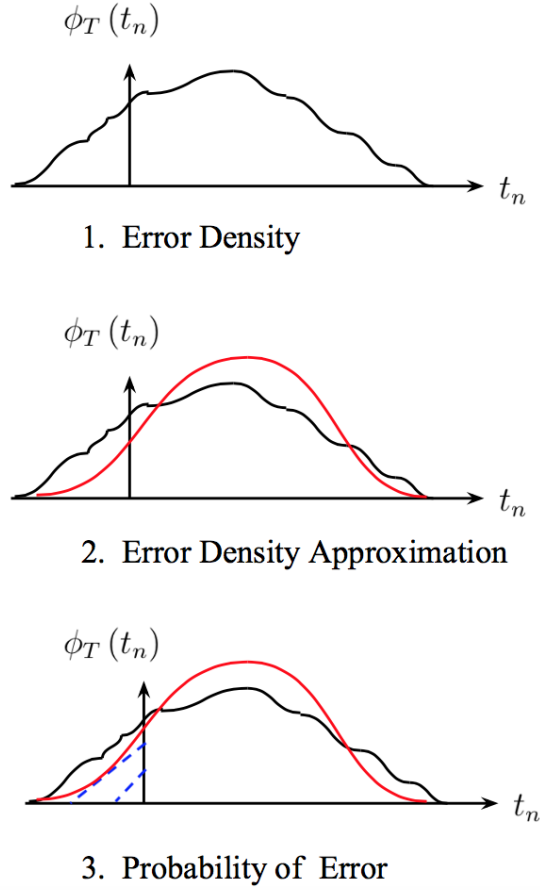


Figure 1.6: Description of the Distributional Approximation the Classification Accuracy (DACA)

1.4.1 Closed Form Equation

For the binary classification, we show that there exist a closed form equation for this classification technique. This has been proved thorough a Central Limit Theorem given in Chapter 3. It is shown that, under set of assumptions, this classifier is the best classification method that minimize the probability of error. In this chapter we also examined the performance of this classification against existitng methods in high

CHAPTER 1. INTRODUCTION

dimensional tasks.

The proposed classification technique is extended to multiclass classification. Same ideology is considered to generalize the method. This has been examined in Chapter 4.

1.4.2 Deep Neural Network with DACA loss

The configuration shown in Figure 1.7 is used as an alternative to the cross entropy loss on the top of Deep Neural Network. Similar to the conventional DNN architecture the layer $L + 1$ is a linear layer which transform the final hidden layer of a C dimensional layer. Instead of 1-hot codeword used for cross entropy, we propose a coding scheme which not only allow competition for the class itself but also consider the competition among classes. Using this coding, the detector variable is computed in layer $L + 2$. At each stage we have an approximate of the density of this parameter. Given the density $\phi_T(t)$, one can easily compute the probability of error which is $1 - \Phi(t \geq \mathbf{0})$. This is the DACA loss computed on the top of network. The normal Back-Propagation algorithm is then used to optimize all the model parameters jointly.

For further clarification of the proposed algorithm, Figure 1.8 presents the experimental results using the DACA loss for a task of binary character recognition. The MNIST dataset is used to classify characters larger than 4 from the one smaller or

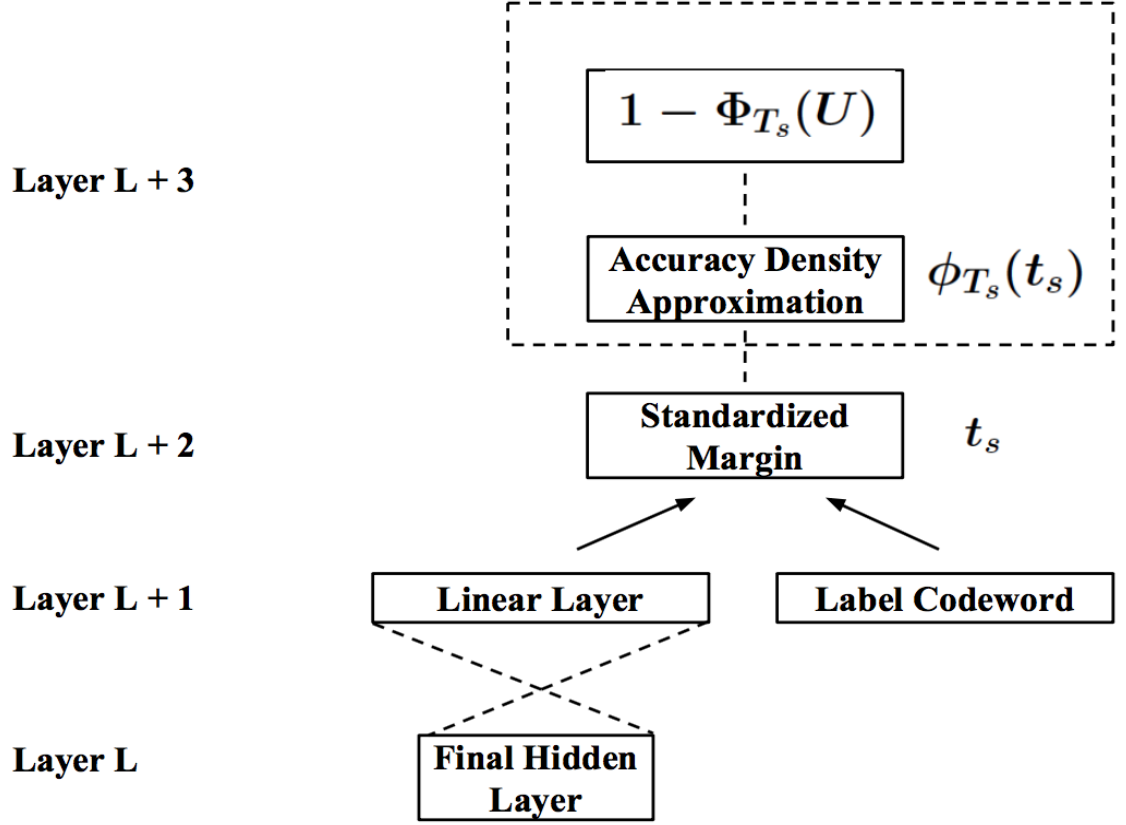
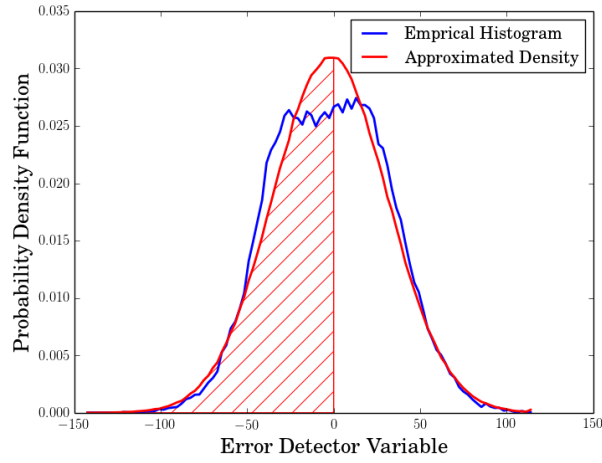


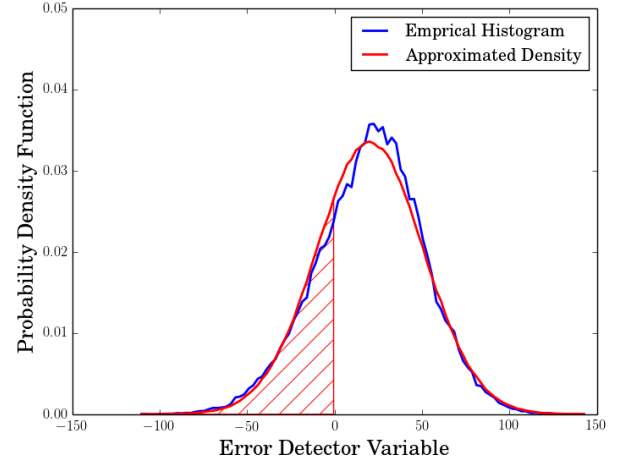
Figure 1.7: The classification layers with Distributional Approximation the Classification Accuracy (DACA) classifier.

equal to 4. As this figure shows, the model is able to have a good approximation of the accuracy density through the iterations. The approximation is more exact for the estimation of the error which is the shaded line under the curves. At the initialization step the error is 50%. After only 100 iterations, this error reduced to 22.70. After 1000 iteration, it reduces to 16.81 and finally after 10k steps, this gets to 14.04%.

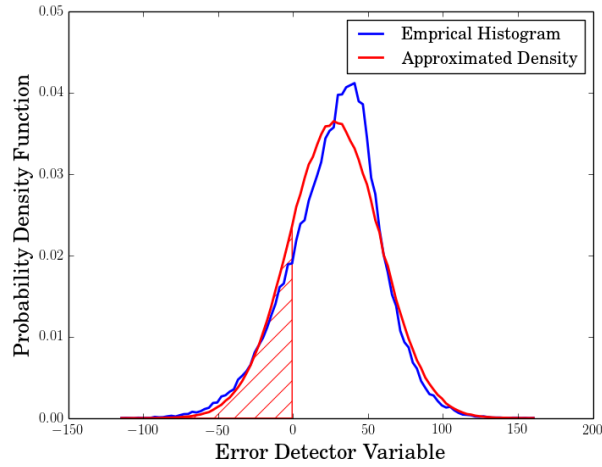
CHAPTER 1. INTRODUCTION



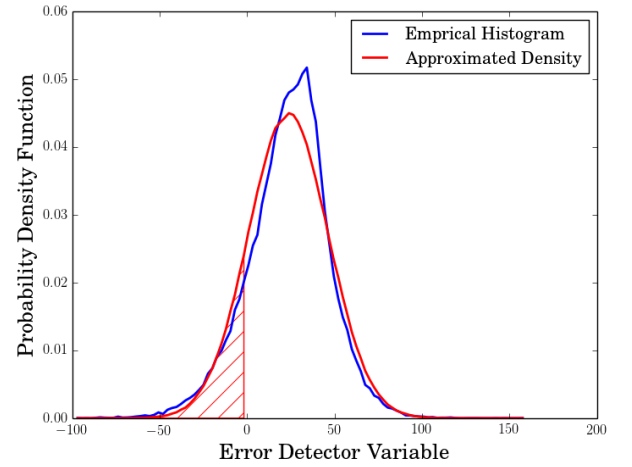
(a) Initialization.



(b) Iter 100.



(c) Iter 1000.



(d) Iter 10000.

Figure 1.8: Deep Neural Network with DACA loss.

Chapter 2

Deep Gaussian Mixture Models

This chapter proposes and evaluates a Gaussian Mixture Model (GMM) represented as the last layer of a Deep Neural Network (DNN) architecture and jointly optimized with all previous layers using Asynchronous Stochastic Gradient Descent (ASGD). The resulting “Deep GMM” architecture was investigated with special attention to the following issues: (1) The extent to which joint optimization improves over separate optimization of the DNN-based feature extraction layers and the GMM layer; (2) The extent to which depth (measured in number of layers, for a matched total number of parameters) helps a deep generative model based on the GMM layer, compared to a vanilla DNN model; (3) Head-to-head performance of Deep GMM architectures vs. equivalent DNN architectures of comparable depth, using the same optimization criterion (frame-level Cross Entropy (CE)) and optimization method (ASGD); (4)

Expanded possibilities for modeling offered by the Deep GMM generative model. The proposed Deep GMMs were found to yield Word Error Rates (WERs) competitive with state-of-the-art DNN systems, at the cost of pre-training using standard DNNs to initialize the Deep GMM feature extraction layers. An extension to Deep Subspace GMMs is described, resulting in additional gains.

2.1 Introduction

The recent gains in the field of ASR obtained from the use of DNNs rather than GMMs are typically attributed to the greater ability of DNNs to extract useful discriminative features automatically [23] [29]. In particular, [23] carefully examines the GMM vs. DNN picture as progressively more complex features are engineered into the GMM architecture. Eventually, the GMM results improve quite a bit compared to the no-engineering baseline, but still trail the DNN results. There is some evidence [30] that higher DNN layers represent features that are more invariant to speaker differences. These studies, among others, suggest that the fundamental breakthrough with DNNs derives from this superior feature extraction. However, there is no need to limit “deep” feature extraction to DNN architectures. Previous work proposed the general concept of *Discriminative Feature Extraction (DFE)*, in which both the feature extraction and classification modules are optimized jointly [17]. One can

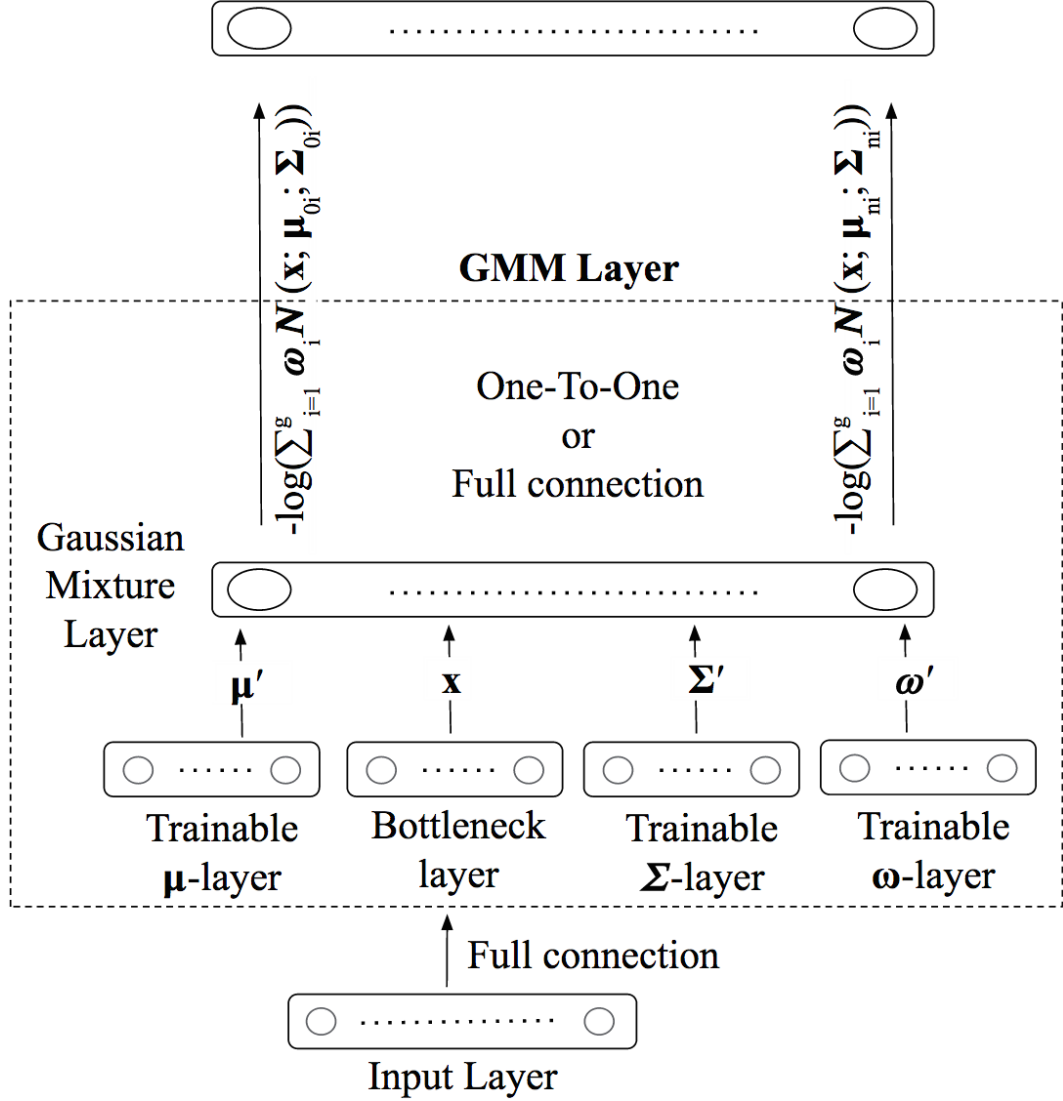


Figure 2.1: The Gaussian Mixture Model (GMM) layer: n nodes with g Gaussian components per node, and associated μ , Σ & ω parameters.

view the many successful DNN results as special cases of DFE, with the final DNN layer(s) being the classifier, and the preceding layers corresponding to the feature extractor. From this perspective, there is no reason to limit the classifier to be a DNN layer, itself closely related to a single Gaussian model with a globally pooled covariance matrix [31] [32]. This last observation makes the link to more structured

CHAPTER 2. DEEP GAUSSIAN MIXTURE MODEL

classifiers based on generative models such as GMMs. Previous work [33] [34] [35] explored this possibility by separately training DNNs to generate “bottleneck” features which are fed into independently trained GMM systems. This offers the possibility of preserving familiar GMM techniques for optimization (such as use of the Expectation-Maximization (EM) algorithm) and adaptation, on top of the richer “deep” features from the DNN. These hybrid approaches, however, did not address the possibility of joint optimization of the DNN features together with the GMM backend. That is exactly what the Deep GMM proposed here aims to address. The benefits of previously investigated hybrid systems still apply, but are now amplified by a new capacity for joint optimization.

We note that a related “Deep GMM” was proposed in [36] in which successive layers of log-linear models are built on *top* of GMM-derived posteriors; this work did not evaluate joint optimization of the GMM parameters. Moreover [37] investigated the joint training of the bottleneck features with sequence training objective, while the GMM parameters are separately trained using the EM algorithm. Finally, [38] proposed a deeply factored GMM, operating on original input features and not involving DNNs.

2.2 A GMM layer for deep learning

The GMM layer, shown in Fig. 2.1, embeds a set of Gaussian Mixture Models into a DNN. The layer is designed so that it can easily be imported to any part of the deep structure. Throughout this paper, the GMM layer is mainly used as the last layer. Each node in the *Gaussian Mixture sub-layer* has g Gaussian components. The number of GMM nodes in the layer is equal to the number of classes (or states). For each state s , the corresponding node in the GMM layer outputs the negative log likelihood $-\log(p(x|s))$, $p(x|s)$ being produced by a mixture of Gaussians,

$$p(x|s) = \sum_{i=1}^g \omega_{si} \mathcal{N}(x, \mu_{si}, \Sigma_{si}). \quad (2.1)$$

Here μ_{si} and Σ_{si} represent the mean vector and covariance matrix of the i^{th} Gaussian within state s , and ω_{si} is the corresponding weight.

The GMM layer has four additional sub-layers. A *linear bottleneck sub-layer* of d nodes feeds its output to all nodes in the GMM layer as the input feature x (of dimensionality d). The notation $GMM(d, g)$ is used for a GMM layer with given input dimensionality and number of Gaussians per state. Three parameter sub-layers, the μ -layer, Σ -layer, and ω -layer store real-valued parameters for each node, representing the standard GMM parameters, or transformed versions thereof. Each of these sub-layers has the same number of nodes as the GMM layer, to which they are connected one-to-one.

CHAPTER 2. DEEP GAUSSIAN MIXTURE MODEL

The actual model parameters are derived by transforming the stored values to respect the parameter constraints of a Gaussian Mixture distribution, detailed below. To avoid confusion between actual model parameters and stored values, the parameters in the trainable layers are referred to as μ' , Σ' and ω' . Each node stores a supervector corresponding to the model parameters. For a GMM(d, g) layer, each node in the μ -layer stores the supervector $\mu'_s = [\mu'_{s1}, \dots, \mu'_{sg}]$, where μ'_{si} is the vector stored for component i of state s . Similarly, Σ -layer and ω -layer store corresponding supervectors Σ'_s and ω'_s . In the forward pass, these parameters are transformed to the corresponding GMM parameters in the GMM layer. In the backward pass, the corresponding gradients are fed back to update the parameters.

Forward Pass. In the forward pass, the GMM layer outputs the negative log likelihood, $L(x, s) = -\log(p(x|s))$, where $p(x|s)$ is the Gaussian Mixture distribution from Eq. (2.1), with parameters μ , Σ , and ω . To ensure that the parameters of the GMM layer obey the constraints of a Gaussian mixture, appropriate activation functions are applied within the GMM layer for each state s and component i :

- μ -layer: There is no constraint on μ values, so $\mu_{si} = \mu'_{si}$.
- Σ -layer: The parameters stored in this layer, Σ' , correspond to the diagonal components of the covariance matrix. To impose the semi-positivity of these parameters, the exponential activation function was used, i.e. $\Sigma_{si} = \exp(\Sigma'_{si})$.

CHAPTER 2. DEEP GAUSSIAN MIXTURE MODEL

- ω -layer: To impose positivity and the sum-to-one property of the mixing weights, this layer stores real values but applies a *softmax* transform:

$$\omega_{si} = \frac{\exp(\omega'_{si})}{\sum_{i=1}^g \exp(\omega'_{si})}.$$

The input to the GMM layer is first fed to the bottleneck sub-layer. This layer is a linear layer which changes the dimensionality of input features and reduces the correlation between features. The latter is jointly enforced with the diagonal covariance matrix assumption. The bottleneck layer outputs a d -dimensional vector which will be fed to the GMM layer as input feature vector. The number of nodes in this layer can significantly affect the overall performance.

Backward Propagation of Gradients. The partial derivatives of each parameter back-propagated by the GMM to the rest of the network are as follows. The index i corresponds to the GMM component and index j denotes the dimension, so $i = 1, \dots, g$ and $j = 1, \dots, d$ ¹.

$$\begin{aligned} \frac{\partial L(x, s)}{\partial \mu'_{ij}} &= \pi_i(x, s) \left\{ \frac{x_j - \mu_{ij}}{\sigma_{ij}^2} \right\} \\ \frac{\partial L(x, s)}{\partial \sigma'_{ij}} &= \pi_i(x, s) \left\{ \left(\frac{x_j - \mu_{ij}}{\sigma_{ij}} \right)^2 - 1 \right\} \\ \frac{\partial L(x, s)}{\partial \omega'_i} &= \pi_i(x, s) - \omega_i \\ \frac{\partial L(x, s)}{\partial x} &= - \sum_{i=1}^g \frac{\partial L(x, s)}{\partial \mu_{ij}} \end{aligned} \tag{2.2}$$

with

¹The state dependency of variables is dropped to simplify the equations.

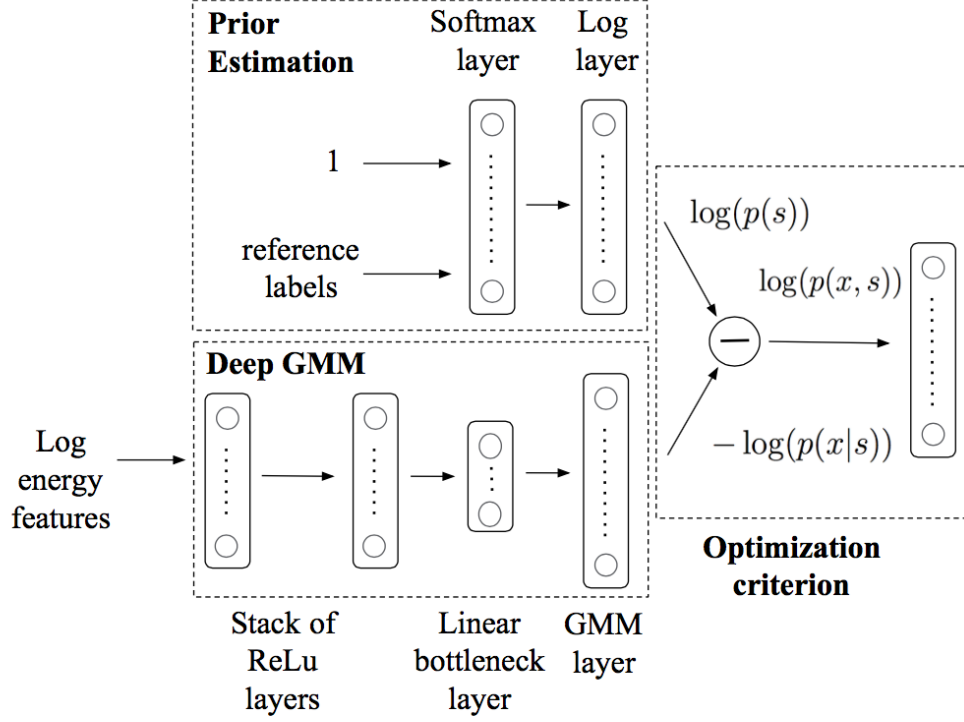


Figure 2.2: GMM optimization within overall NN structure.

$$\pi_i(x, s) = \frac{\omega_i \mathcal{N}(x, \mu_{ij}, \Sigma_{ij})}{\sum_{i=1}^g \omega_i \mathcal{N}(x, \mu_{ij}, \Sigma_{ij})} \quad (2.3)$$

which is the contribution of each Gaussian distribution in the mixture. Detailed derivation of the above equations can be found in [39, 40, 41, 42, 43].

2.3 Training a Neural Network with GMM layer

Fig. 2.2 shows the structure of the embedded GMM within a Neural Network (NN) architecture during optimization with Cross Entropy (CE) [29]. There are three modules in this figure. The top module estimates priors online during CE training. This is done by inputting a constant value of 1 and the reference frame label into the softmax. The logarithm of the estimated prior distribution, $\log(p(s))$, is then calculated.

The GMM layer is used as the last layer of the network. If the GMM layer is used after the stack of hidden layers, the corresponding structure is referred to as *Deep GMM* (DGMM), otherwise it will be called *Shallow GMM*. The GMM layer outputs $-\log(p(x|s))$ for each frame of the data. By subtracting this value from the logarithm of the prior, the logarithm of the joint distribution can be formed:

$$\log(p(x, s)) = \log(p(s)) - (-\log(p(x|s))). \quad (2.4)$$

The state posteriors for CE training can then be derived using

$$p(s|x) = \frac{\exp(\log(p(x, s)))}{\sum_s \exp(\log(p(x, s)))}. \quad (2.5)$$

In this paper all experiments used CE training. However, the proposed design allows

CHAPTER 2. DEEP GAUSSIAN MIXTURE MODEL

the use of any optimization criterion, such as Maximum Likelihood (ML), CE, or Discriminative Sequence Training [44] [45]. (Use of ML to optimize the entire structure requires care, since the Jacobian of the bottleneck features has to be handled; this is not detailed here).

The trainable sub-layers are initialized as follows. The μ -layer values are initialized to random numbers derived from a normal distribution, $\mathcal{N}(0, 1)$. All parameters in the Σ -layer are initialized with a constant value of 0 (corresponding to a transformed value of 1) and the parameters in the ω -layer are initialized with a uniform value of $1/g$. We considered the effect on training of each parameter separately. Our observation was that training all parameters jointly achieves the best performance, but requires more overall training steps than first training just the means and weights while fixing the variances, before then training all parameters together.

2.4 Experiments

The main concept behind the DGMM model is the *joint* optimization of *feature extraction* and *classification*. Since GMMs primarily function on the classification side, the “depth” of the model proposed here primarily operates on the feature extraction side. These two are then jointly optimized in the DGMM structure. The experiments are designed to carefully examine the effect of each of these components. DNNs can

CHAPTER 2. DEEP GAUSSIAN MIXTURE MODEL

also be seen as joint models with both deep structure and a final classification layer. Both jointly optimized GMMs and DNNs are compared in matched conditions for shallow and deep structures. Further on, the Subspace GMM (SGMM) model is investigated using the GMM layer.

Datasets. The experiments described here used two anonymized datasets: *Icelandic (is-is)*, consisting of 60 hours of read speech, simulating Voice Search queries, and *American English (en-us)*, consisting of 3000 hours of spontaneous speech from anonymized, human-transcribed Voice Search data. The test sets for each language are separate anonymized, human-transcribed Voice Search datasets of about 25 hours each.

The training examples are log energy features obtained from the concatenation of 26 frames, 20 left and 5 right. Frames consist of 40-dimensional log-energies extracted from a 25 ms window shifted over the audio samples in 10 ms increments. Each training step has a batch size of 200 examples. For both en-us and is-is, the DNN baseline uses 8 hidden layers, each with 2560 nodes. The is-is system uses 2432 context-dependent output states (resulting in about 55 M parameters in total) while the en-us system uses 13568 output states (for a total of about 85 M parameters). The DGMM model hidden layers are initialized using the baseline DNN model, unless explicitly mentioned otherwise. The rectified linear units were used for the hidden layers. ASGD training used 100 multi-core machines.

CHAPTER 2. DEEP GAUSSIAN MIXTURE MODEL

Effect of Joint Optimization. In this experiment we evaluated three configurations on is-is. The first configuration is the baseline DNN, trained to convergence. The second system is a DGMM with a GMM(272,5) layer and 7 hidden layers. The hidden layers are initialized using the corresponding baseline DNN model and then kept frozen during training. This simulates the *separate optimization* approach where feature extraction is done *independently* of classification. The third system, the *jointly optimized* DGMM, uses the same structure as the second one, with the difference that all the layers are trained together. Both DGMM systems have about 49 M parameters in all, but only 7.3 M of those are trained in the 2nd system. Table 2.1 summarizes the results of this experiment. The jointly optimized DGMM clearly outperforms the separately optimized DGMM.

System	baseline DNN	separately optimized DGMM	jointly optimized DGMM
state acc	60.75	59.01	62.66
WER	15.6	14.6	12.2

Table 2.1: Separate versus joint optimization of model and features in deep architectures (is-is).

The results for the two DGMM systems suggests that joint training successfully extracted features beneficial for GMM training. However, a counter-argument is that the gain is due to the greater number of trainable parameters in the jointly trained DGMM. To address this possibility, we compared the two approaches using a shallow structure. A GMM(39, 21) layer was used without any hidden layers. In the first experiment, we used PLP + LDA features extracted outside the model as input to

CHAPTER 2. DEEP GAUSSIAN MIXTURE MODEL

the GMM layer. In this case, the GMM layer doesn't have the bottleneck sub-layer. In the second experiment, we used log energies as the input features and a bottleneck sub-layer of size 39 as feature extractor. Each system here has about 4.0 M parameters in total. Table 2.2 shows the results, as well as the number of training steps and CE cost at the time of evaluation. The high WERs here suggest that optimization of shallow, randomly initialized GMMs using ASGD is not very effective compared to conventional GMM initialization techniques and ML-based training using the EM algorithm. Nonetheless, the joint system outperforms the separate system significantly, using a smaller number of steps. This argues against the number of trainable parameters having a dominant effect in the previous experiment, highlighting instead the importance of joint optimization.

Front Topology	CE cost	num steps	state acc	WER
PLP+LDA	10.12	33.92 M	33.35	51.7
Log Energy	8.92	17.8 M	44.7	29.3

Table 2.2: Separate versus joint optimization of features and classification for shallow GMMs (is-is).

Effect of DGMM depth. To examine the effect of depth more carefully we conducted another experiment comparing DGMM models with different numbers of hidden layers. Three structures with 1, 3 and 7 hidden layers, and 2560 nodes per layer, were considered. The GMM layer, GMM(272,5), is used right after the last hidden layer in each structure. Table 2.3 summarizes the results. Performance significantly improves with depth, suggesting that the deep structures possess superior feature extraction ability. In addition, the number of training steps significantly decreases as

CHAPTER 2. DEEP GAUSSIAN MIXTURE MODEL

# hidden layers	#params	CE cost	#steps	state acc	WER
1	10 M	10.02	19.74 M	47.43	32.4
5	36 M	8.00	2.75 M	59.44	14.8
7	49 M	7.62	1.4 M	62.66	12.2

Table 2.3: Effect of depth on DGMM performance (is-is).

the structure becomes deeper. This supports the discussion in [46] on the learning efficiency of deep structures over shallow ones. Note, however, that the total number of parameters is not matched across configurations.

Feature Extraction vs Classification Power in NNs & GMMs. In this section, the tradeoff between deep feature extraction capacity and classification power is examined by comparing GMMs and NNs with both shallow and deep architectures with matched total numbers of parameters. All four models have roughly the same number of parameters, 54 M. The shallow NN has 3936 hidden nodes in a single hidden layer while the deep NN is the baseline DNN described earlier. The shallow GMM has a GMM(272, 41) layer without any hidden layers beyond the 272-node linear bottleneck layer. For the DGMM, we used 7 hidden layers of 2560 nodes with a GMM(480, 5) layer. All four models were trained starting from randomly initialized weights. Table 2.4 summarizes the results. For both GMMs and NNs, the deep structure outperforms the corresponding shallow structure in terms of number of training steps as well as state accuracy and WER. The shallow NN outperforms the shallow GMM by a large margin. The performance difference between deep and shallow architectures is much larger for GMMs than for NNs. It might be that optimized deep

CHAPTER 2. DEEP GAUSSIAN MIXTURE MODEL

	Shallow		Deep	
	GMM	NN	GMM	NN
state acc	30.66	53.38	55.67	54.06
WER	71	18.2	15.6	16.1
CE cost	10.21	1.33	7.33	1.18
#training steps	9.4 M	23 M	4.2 M	8 M

Table 2.4: NN and GMM results using shallow and deep structures, with matched # of parameters, and no pre-training (is-is).

feature extraction is inherently more important for GMMs than for NNs. However, as observed earlier, the high WER for the shallow GMM suggests that optimizing shallow, randomly initialized GMMs using ASGD is not very effective. In any case, deep structure leads to much better optimization and/or feature extraction capability for the GMM system: the jointly optimized DGMM outperforms the shallow GMM structure significantly. The DGMM also outperforms the DNN.

Use of pre-training; large scale ASR results. Table 2.5 shows results comparing DNNs and DGMMs with matched numbers of parameters for the large-scale en-us task, as well as for is-is, with some use of pre-training. The DNN models have 8 hidden layers with 2560 nodes per layer; the is-is DNN was initialized cross-lingually from an existing en-us DNN. The DGMMs share the input layer and the first 7 hidden layers with their corresponding DNN model. A GMM(480, 5) layer was used for is-is and a GMM(80, 15) layer was used for en-us. The DGMM models are initialized from the corresponding DNN models. The DGMM training results for en-us are not fully converged. For both tasks, the proposed DGMMs yield WERs competitive with

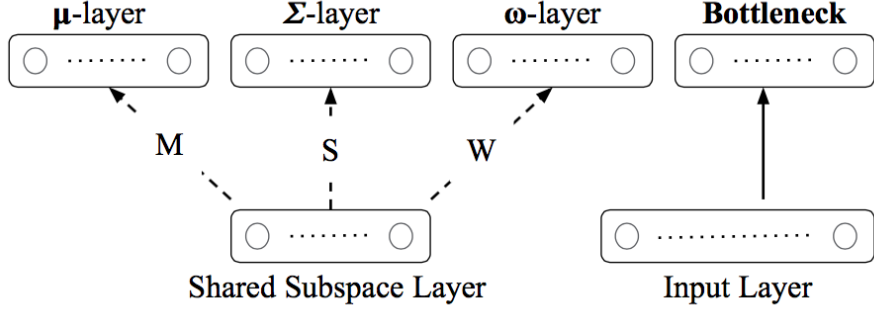


Figure 2.3: Integrating a Subspace Gaussian Mixture Model (SGMM) into a DGMM architecture: the Deep SGMM (DSGMM).

state-of-the-art DNN systems.

Dataset	is-is (60 hours)		en-us (3,000 hours)	
	DGMM	DNN	DGMM	DNN
state acc	61.69	60.75	57.4	59.2
WER	13.3	15.6	12.0	11.7

Table 2.5: Results for pre-trained DNNs and DGMMs.

Deep Subspace-GMM (DSGMM) The Subspace GMM (SGMM) extends GMMs by introducing a shared subspace, v_s , among mixture components and two linear transforms M and W which are *tied* among all mixtures [47]. For mixture s , the mean parameters are $\mu_s = Mv_s$ and the mixture weights are derived by applying the softmax activation on Wv_s . In [47], the variances are not modeled with a subspace, due to a cited difficulty with training.

The SGMM model can easily be applied to the GMM layer, as shown in Fig. 2.3. The *shared subspace layer* is a trainable layer that stores a shared subspace, v_s . There are three linear transforms, M , S , and W , corresponding to the three model

CHAPTER 2. DEEP GAUSSIAN MIXTURE MODEL

parameters, means, variances and mixing weights. These transforms are *tied* across states. The subspace layer can be *shared* by one, two, or all three sub-layers above. This is shown by dotted connections in the figure. We examined three configurations in Table 2.6. All these DSGMM models share the same deep structure of 7 hidden layers (baseline DNN) and a GMM(80, 5) layer. The subspace dimensionality is the same as that of the input feature, 80. The only difference is in the way that models share the subspace layer. The parameters that do not share the subspace have their own trainable layer instead. The experiments show that the DSGMM model outperforms the baseline DGMM model of similar size. The configuration that uses the subspace only for the mean parameters slightly outperforms other models in terms of state accuracy and WER. This required fewer training steps than in the other configurations. Sharing the subspace among all parameters also outperforms the baseline DGMM, but a significantly larger number of training steps is required than for other models. Sharing the subspace among just means and weights outperforms sharing the subspace among all parameters.

	baseline DGMM	shared mean	shared mean-weight	shared all
state acc	60.63	62.38	61.8	61.13
WER	14.9	12.7	13.3	13.7
# training steps	2.1 M	0.3 M	1.8 M	4.2 M

Table 2.6: Deep Subspace GMM experiments (is-is).

2.5 Conclusions

This paper proposed a GMM layer as an alternative to the softmax layer in a DNN structure. The resulting Deep GMM system combines the advantages of deep feature extraction and GMM representation. Joint training of this model with ASGD produced results competitive with DNNs on two Voice Search tasks, particularly for the smaller task. Extension of this model to Deep Subspace GMMs resulted in additional improvements.

Chapter 3

Toward Minimization of the Classification Error

This chapter considers the problem of binary classification of a long feature vector using a *linear* classifier with the constraint to minimize the *classification error*. The proposed linear classifier, notated as $C^{-1}\mu$, is derived under the assumption of Normal approximation of the classification error. The derived linear classifier is then extended to general *nonlinear* classification using two alternative techniques: a Neural Network architecture and Reproducing Kernel Hilbert Spaces (RKHS).

3.1 Introduction

At the root of machine learning is the problem of classification using a linear (affine) classifier where the goal is to minimize the classification error. A linear classifier maps the feature vector $X \in \mathbb{R}^d$ into a class label $\hat{Y}$ using set of real value weights, $w = (w_0, w_1)$:

$$\hat{Y} = f(w_0 + \langle w_1, X \rangle)$$

where, $w_0 \in \mathbb{R}$, $w_1 \in \mathbb{R}^d$ and $\langle \cdot, \cdot \rangle$ denotes the scalar dot product. The function f maps its argument score values to one of the classes. Often f is a simple function that maps score values into classes by comparing them with prespecified threshold value. For a task of binary classification, usually the threshold is set to be zero, which is equivalent to use of the sign function.

The classification error occurs when the decoded class $\hat{Y}$ differs from the true class, Y . For a binary classification, the probability of error is:

$$P(\hat{Y} \neq Y) = \mathbb{P}((w_0 + \langle w_1, X \rangle)Y < 0) \quad (3.1)$$

The main difficulty is that $\mathbb{P}$ is not necessarily known. Instead, using the training examples $\{(x^i, y^i)\}_{i=1}^n$, the above equation is rewritten in terms of the empirical risk

CHAPTER 3. MINIMIZATION OF THE CLASSIFICATION ERROR

[48]:

$$\sum_{i=1}^n \psi((w_0 + \langle w_1, x^i \rangle) y^i) \quad (3.2)$$

which is to be minimized subject to some constraints over weight vector w as well as the loss function ψ . Different choices for the loss function ψ and the constraints, lead to the most used linear classifiers including Support Vector Machines and Logistic Regression.

Due to the Bayes Error Rate theorem, the minimum classification error is achieved if the classifier decides $\hat{Y}$ that maximizes the posterior density $P(Y|X)$. Alternatively, using Bayes rule, the classifier can compute the product of likelihood $P(X|Y)$ with prior $P(Y)$. In practise neither $P(Y|X)$ nor $P(X|Y)$ is known. To overcome this issue most linear classifiers make assumptions on either of these distributions. In literature, the classifiers that make assumptions on the likelihood distribution $P(X|Y)$ is known as *generative* models [4, 5]. Examples of such algorithms include Linear Discriminant Analysis (LDA) and Naive Bayes. LDA, or Fisher Linear Discriminant, assumes Gaussian conditional density model [7] while the Naive Bayes classifier assumes multinomial or multivariate Bernoulli events models.

On the other hand, the *discriminative* models make no probabilistic assumptions on the conditional distribution $P(X|Y)$, [4, 5]. The discriminative methods themselves categorized into *probabilistic* and *non-probabilistic* methods. The Logistic Regression

CHAPTER 3. MINIMIZATION OF THE CLASSIFICATION ERROR

is an example of the discriminative methods which assumes the $\log(P(Y|X))$ to be a linear function of the feature vector X . Other methods like Perceptron [8] and Support Vector Machines [11] learn the linear mapping directly from training data without any further probabilistic assumptions.

Independent of generative or discriminative, none of the above methods *explicitly* minimize the classification error. The main idea is *estimating* the conditional distribution $P(Y|X)$ and then using the Bayes decision rule. However, these methods are highly dependent on how good the assumptions match with given training data.

In this chapter we explore an alternative path consisting in approximating Eq. 3.1 by a function which depends only weakly in $\mathbb{P}$. Specifically, we use a Normal approximation to this quantity. To simplify the discussion we introduce the variable $Z \in \mathbb{R}^{d+1}$ for which

$$Z_0 = Y$$

$$Z_i = X_i Y \quad 1 \leq i \leq d$$

By assuming $\langle w, Z \rangle = \sum_{i=0}^d w_i Z_i$ is Normally distributed, the probability of error in Eq. 3.1 is approximated by

$$P(\hat{Y} \neq Y) = \phi \left(-\frac{\langle w, \mu \rangle}{(w^T C w)^{\frac{1}{2}}} \right) \quad (3.3)$$

where ϕ is the cumulative distribution of the standard Normal distribution, μ is the expected value of Z , $\mu = \mathbb{E}[Z]$, and C is the covariance matrix of Z , $C = \text{Cov}[Z]$.

CHAPTER 3. MINIMIZATION OF THE CLASSIFICATION ERROR

When C is invertible, minimizing Eq. 3.3 provide the solution

$$w = C^{-1}\mu \tag{3.4}$$

up to a positive multiplicative constant, which gives the name to the proposed classifier family. Estimating the empirical $(d + 1) \times (d + 1)$ dimensional inverse covariance matrix C^{-1} , and the empirical $d + 1$ dimensional mean vector μ , using the training data provides the following binary classifier weight values:

$$w^* = \hat{C}^{-1}\hat{\mu} \tag{3.5}$$

The rest of this chapter is organized as follows: in Section 3.2 we present sufficiency conditions under which the approximation above is justified by the central limit theorem. We then discuss the invertibility of C and the existence of the $C^{-1}\mu$ classifier. We further review a collection of methods available for the estimation of μ and C using the training data. Section 3.2 is then concluded with a toy example: the case of a naive Bayes model. In section 3.3, we present two extensions of the $C^{-1}\mu$ linear classifier to non linear classifiers, first using a Neural network architecture and second using Reproducing Kernel Hilbert Spaces. Section ?? is devoted to experiments on real data. We consider first a digit recognition problem using the classic MNIST dataset and secondly an image classification problem where molecular imaging allows to identify subjects suffering Parkinson's disease versus healthy controls. Section 3.5 contains our concluding remarks.

3.2 $C^{-1}\mu$ Classifier

For a feature vector $X \in R^d$, we are looking for a linear (affine) classifier that minimizes the probability of misclassification. More precisely, if we denote binary classes by -1 and 1 , the decision rule is $f(X) = \text{sign}(w_0 + \langle X, w_1 \rangle)$ for a well chosen vector $w = (w_0, w_1)$. The choice of this vector is based on the following minimization problem:

$$w_{Bayes} = \underset{w \in R^{d+1}}{\text{argmin}} \mathbb{P}((w_0 + \langle X, w_1 \rangle)Y < 0) \quad (3.6)$$

Of course, with no further assumptions or knowledge of the joint distribution of (X, Y) , Eq. 3.6 cannot be solved exactly. Using the notation Z given in Eq. 3.3 and setting $\mu = \mathbb{E}(Z)$ and $C = \text{Cov}(Z)$, the above equation can be rewritten as follows:

$$\mathbb{P}((w_0 + \langle X, w_1 \rangle)Y < 0) = \mathbb{P}\left(\frac{\langle w, (Z - \mu) \rangle}{(w^T C w)^{\frac{1}{2}}} < \frac{-\langle w, \mu \rangle}{(w^T C w)^{\frac{1}{2}}}\right) \quad (3.7)$$

Nest, notate

$$T_d = \frac{\langle w, (Z - \mu) \rangle}{(w^T C w)^{\frac{1}{2}}} \quad (3.8)$$

In certain cases where d is not too small, and under further conditions discussed in next section, the central limit theorem assess that T_d converges in distribution to a standard Normal random variable. Thus performing the following approximation is

CHAPTER 3. MINIMIZATION OF THE CLASSIFICATION ERROR

justified:

$$\mathbb{P}\left(\frac{\langle w, (Z - \mu) \rangle}{(w^T C w)^{\frac{1}{2}}} < \frac{-\langle w, \mu \rangle}{(w^T C w)^{\frac{1}{2}}}\right) \sim \phi\left(\frac{-\langle w, \mu \rangle}{(w^T C w)^{\frac{1}{2}}}\right) \quad (3.9)$$

where ϕ is the cumulative distribution function of a standard Normal distribution.

When this approximation is valid, the optimization for w is reduced to the optimization problem:

$$\hat{w} = \operatorname{argmax}_{w \in \mathbb{R}^{d+1}} \frac{\langle w, \mu \rangle}{(w^T C w)^{\frac{1}{2}}} \quad (3.10)$$

There are many gradient based techniques that can be used to derive the weights w that minimize the above function. Fortunately, for the binary classifier, the above minimization problem can be solved analytically. Notate

$$g(w) = \frac{\langle w, \mu \rangle}{(w^T C w)^{\frac{1}{2}}} \quad (3.11)$$

we consider three cases based on the invertibility of C and whether μ is in $\operatorname{Im}(C)$ or not. The invertibility of C can be easily determined by evaluation of the determinant of C . There are many ways to check if μ is in $\operatorname{Im}(C)$ or not, one easy way is to determine if $CC^\dagger \mu = \mu$, where $C^\dagger$ is any psuedo inverse of C .

Case I: C is invertible

Since C is invertible, we use change of variable $y = C^{\frac{1}{2}}w$ in function $g(w)$ to derive

CHAPTER 3. MINIMIZATION OF THE CLASSIFICATION ERROR

the following function:

$$h(y) = \frac{\langle y, Ay \rangle}{||y||^2} \quad (3.12)$$

where $A = C^{-\frac{1}{2}}\mu\mu^tC^{-\frac{1}{2}}$ which itself can be written as $A = vv^T$, for $v = C^{-\frac{1}{2}}\mu$. Since C is invertible, there is a one to one relation between any maximizer of the function $h(\cdot)$ and the function $g(\cdot)$. The matrix A has rank one, so it assets only one strictly positive eigen value with corresponding eigen space, spanned by the vector v . Hence, any positive multiple of the vector v is a maximizer for $h(\cdot)$ which requires any positive multiple of the vector $w = C^{-1}\mu$ is a maximizer of $g(\cdot)$.

Case II: C is not invertible and $\mu \in \text{Im}(C)$

The weight vector w can be decomposed as follows:

$$w = w_{Im} + w_{Ker} \quad (3.13)$$

where w_{Im} and w_{Ker} are representations of w in the image and kernel space of matrix C , respectively. Using this, the numerator of the fuction $g(w)$ can be written as:

$$\begin{aligned} \langle w, \mu \rangle &= \langle w_{Im} + w_{Ker}, \mu \rangle \\ &= \langle w_{Im}, \mu \rangle + \langle w_{Ker}, \mu \rangle \\ &= \langle w_{Im}, \mu \rangle \end{aligned} \quad (3.14)$$

where the last equality holds because of the assumption $\mu \in \text{Im}(C)$. Similarly, the

CHAPTER 3. MINIMIZATION OF THE CLASSIFICATION ERROR

denominator of $g(w)$ can be written as:

$$\begin{aligned}
 \langle w, Cw \rangle &= \langle w_{Im} + w_{Ker}, Cw_{Im} + Cw_{Ker} \rangle \\
 &= \langle w_{Im}, Cw_{Im} \rangle + \langle w_{Ker}, Cw_{Ker} \rangle \\
 &= \langle w_{Im}, Cw_{Im} \rangle
 \end{aligned} \tag{3.15}$$

since $Im(C) \perp Ker(C)$, due to the symmetry of the covariance matrix C .

Using the above equalities, for each value w , the function $g(w)$ need only be computed at w_{Im} :

$$\begin{aligned}
 g(w) &= \frac{\langle w, \mu \rangle}{\langle w, Cw \rangle^{\frac{1}{2}}} \\
 &= \frac{\langle w_{Im}, \mu \rangle}{\langle w_{Im}, Cw_{Im} \rangle^{\frac{1}{2}}}
 \end{aligned} \tag{3.16}$$

So instead of maximizing g in whole space we only need to maximize g in the subspace $Im(C)$. Applying the discussion of the first case, for any w in $Im(C)$, we can pose $y = C^{\frac{1}{2}}b$ where $C^{\frac{1}{2}}$ is any square root of C **but also** $b = (C^\dagger)^{\frac{1}{2}}y$ where $C^\dagger$ is any pseudo-inverse of the covariance matrix C . The solution of the problem (3.10) will be then any positive multiple of $C^\dagger \mu$.

Case III: C is not invertible and $\mu \notin Im(C)$

In other words, the system $CX = \mu$ does not have any solution. Here, we show the existence of a sequence w_n for which $g(w_n) \rightarrow \infty$ as $n \rightarrow \infty$. Lets u and v be two eigen vectors of the covariance matrix C such that $u \in Im(C)$ and $v \in Ker(C)$, we

CHAPTER 3. MINIMIZATION OF THE CLASSIFICATION ERROR

define the sequence w_n as follows:

$$w_n = u + nv \quad (3.17)$$

Since $g(w)$ is an odd function, $g(w) = -g(-w)$, the w that maximizes $g(\cdot)$ maximizes $g^2(\cdot)$ as well. The numerator of the $g^2(\cdot)$ can be written as:

$$\langle w_n, \mu \rangle^2 = \langle u, \mu \rangle^2 + 2n\langle u, \mu \rangle \langle v, \mu \rangle + n^2 \langle v, \mu \rangle^2 \quad (3.18)$$

since $\mu \notin \text{Im}(C)$, it is possible to choose v such that $\langle v, \mu \rangle \neq 0$ so the numerator of $g^2(w_n)$ goes to ∞ as $n \rightarrow \infty$. On the other side, the denominator of this function is not function of n :

$$\begin{aligned} \langle w_n, Cw_n \rangle &= \langle u + nv, Cu + nCv \rangle \\ &= \langle u, Cu \rangle \end{aligned} \quad (3.19)$$

Since $u \perp v$.

Using the above equations for the numerator and the denominator, the limit of $g^2(w_n)$ can be derived as follows:

$$\begin{aligned} \lim_{n \rightarrow \infty} g(w_n) &= \lim_{n \rightarrow \infty} \frac{\langle u, \mu \rangle^2 + 2n\langle u, \mu \rangle \langle v, \mu \rangle + n^2 \langle v, \mu \rangle^2}{\langle u, Cu \rangle} \\ &= \frac{n^2 \langle v, \mu \rangle^2}{\langle u, Cu \rangle} \\ &= \infty \end{aligned} \quad (3.20)$$

For v that $\langle v, \mu \rangle \neq 0$. This admits that $g(w_n)$ also goes to ∞ as $n \rightarrow \infty$.

CHAPTER 3. MINIMIZATION OF THE CLASSIFICATION ERROR

In colcusion, for the case that the covariance matrix is not invertible and μ is not in the image space of C , the maximum of function $g(\cdot)$ is ∞ . Assuming the central limit theorem assumption of the Eq. 3.3 holds, this leads to perfect classification. However, it is more possible that central limit theorem does not hold at all in this case. This will be discussed in the next section.

3.2.1 Central Limit Theorem Assumptions

Intuitively, the central limit theorem assumption for the variable T_d in Eq. 3.8 is reasonable in the case where the number of *important features* a function of d . It grows to infinity when d diverges to infinity, i.e the number of important features is large enough. More rigorously, we prove under some assumptions the central limit theorem is hold. For a feature vector $X = (X_1, \dots, X_d)$, we construct $(X_{B_1}, \dots, X_{B_{n_d}})$ which is a vector decomposition of X into blocks B_i . These blocks are simply group of indices, $B_i \subset \{1, \dots, d\}$ and

$$\sum_{i=1}^{n_d} |B_i| = d$$

where $|B_i|$ is number of elements of B_i .

C.L.T Assumptions

A1. Given the class Y , the X_{B_i} 's are independent.

CHAPTER 3. MINIMIZATION OF THE CLASSIFICATION ERROR

A2. For each i , the lowest-eigenvalue of the covariance matrix of $Z_{B_i} = Y \times X_{B_i}$ is bounded below by a constant $\lambda_0 > 0$ that does not depend on i . Moreover, the covariance matrices of X_{B_i} 's are equal under class 1 and class -1 .

A3. There exist a uniform constant M such that $|B_i| < M$.

A4. One can arrange the indices i such that the largest eigenvalue of $cov(Z_i, Z_j) \leq \frac{1}{j-i}$ for i sufficiently large and $j > i$.

A5. $\exists \delta > 0$, and finite scalar C_δ such that $\max_i E[||X_{B_i}||^{2+\delta}] < C_\delta$

A6. $f(d) = \max_{i \leq n_d} \frac{|w_{B_i}|^2}{\sum_{j=1}^n |w_{B_j}|^2}$ goes to 0 when n goes to $+\infty$.

This assumptions are necessary for the proof of the central limit theorem holds for Td . The first condition is basically saying that we are under a *generalized* Naive Bayes model in which we can replace the individual features by blocks of features. The second condition ensures that the Z_{B_i} is not close to degenerate random variables and that the covariance matrices of the blocks of features do not depend on the class. This assumption is similar to LDA assumption, however, in LDA, there is a general assumption about $P(X|Y)$ to be Gaussian which is not the case here. The third assumption is just ensuring that each of the blocks B_i has finite number of elements. The fourth assumption required the number of correlated blocks is small compared the total number of pairs of blocks. This assumption is immediately satisfied when we perform a PCA on the raw features. Notice also that the last condition is on the

CHAPTER 3. MINIMIZATION OF THE CLASSIFICATION ERROR

classifying vector w , that is not fixed yet. The idea is to find the vector w assuming the C.L.T assumptions hold and then verifying the assumption on w .

Next, we propose the main theorem of this section. In this theorem we assume the mean μ , and the covariance matrix C of the random variable Z is known. Without lack of generality, we assume that Z_i are centered around zero. In reality we can use change of variable $Z'_i = Z_i - \mu_i$ and prove the theorem for Z' .

Theorem 3.2.1 *Assuming the C.L.T assumptions A1 – 6 holds, then*

$$\frac{\sum_i \langle w_{B_i}, Z_{B_i} \rangle}{\left(\text{Var}(\sum_i \langle w_{B_i}, Z_{B_i} \rangle) \right)^{\frac{1}{2}}} \xrightarrow{\mathcal{L}} \mathcal{N}(0, 1). \quad (3.21)$$

where $\xrightarrow{\mathcal{L}}$ denotes convergence in probability.

Before proving this theorem, we need to prove some lemma needed for the main proof.

There are two important theorems used in the proof, namely Lyapunov Central Limit theorem [49] as well as the Slutsky theorem, [50].

Lyapunov Central Limit Theorem: Suppose $U_1, U_2, \dots$ is a sequence of independent random variables, each with finite expected value μ_i and variance σ_i^2 . Define $s_n^2 = \sum_{i=1}^n \sigma_i^2$. If for some $\delta > 0$, the Lyapunov condition:

$$\lim_{n \rightarrow \infty} \frac{1}{s_n^{2+\delta}} \sum_{i=1}^n \mathbb{E} [|U_i - \mu_i|^{2+\delta}] = 0 \quad (3.22)$$

is satisfied, then sum of $(U_i - \mu_i)/s_n$ converges in distribution to a standard normal

CHAPTER 3. MINIMIZATION OF THE CLASSIFICATION ERROR

random variable, as n goes to infinity:

$$\frac{1}{s_n} \sum_{i=1}^n (U_i - \mu_i) \xrightarrow{\mathcal{L}} \mathcal{N}(0, 1). \quad (3.23)$$

For our setting U_i is defined for each block of feature, $U_i = \langle w_{B_i}, Z_{B_i} \rangle$. The following lemma prove that the Central Limit of the Theorem 3.2.1 holds under conditioning on the class Y .

Lemma 3.2.1 *For each Y , we have:*

$$\frac{\sum_{i=1}^{n_d} \langle w_{B_i}, Z_{B_i} \rangle}{(Var(\sum_i \langle w_{B_i}, Z_{B_i} \rangle | Y))^{\frac{1}{2}}} \xrightarrow{\mathcal{L}_{|Y}} \mathcal{N}(0, 1) \quad (3.24)$$

Proof: This lemma is derived easily from the version of Lyapounov theorem stated in [51] (section 1.5). The main outline of the proof is proving the Lyapunov's condition holds for the sequence $U_i = \langle w_{B_i}, Z_{B_i} \rangle$. Here, s_n is bounded from below by:

$$\begin{aligned} s_{n_d} &= \sum_{i=1}^{n_d} Var(\langle w_{B_i}, Z_{B_i} \rangle | Y = y) \\ &= \sum_{i=1}^{n_d} Var(\langle w_{B_i}, X_{B_i} \rangle | Y = y) \\ &= \sum_{i=1}^{n_d} w_{B_i}^T Var(X_{B_i}) w_{B_i} \\ &\geq \lambda_0 \sum_{i=1}^{n_d} \|w_{B_i}\|^2 \end{aligned} \quad (3.25)$$

where the second equality holds because of definition of Z . The fifth C.L.T assumption, A5, were used for the final inequality.

CHAPTER 3. MINIMIZATION OF THE CLASSIFICATION ERROR

The numerator of the Lyapunov's condition can also be bounded from above as follows:

$$\begin{aligned}
 \sum_{i=1}^{n_d} E [|\langle w_{B_i}, Z_{B_i} \rangle|] &\leq \sum_{i=1}^{n_d} E [||w_{B_i}||^{2+\delta}] E [||Z_{B_i}||^{2+\delta}] \\
 &\leq C_\delta \sum_{i=1}^{n_d} ||w_{B_i}||^2 ||w_{B_i}||^\delta \\
 &\leq C_\delta \left(\max_i ||w_{B_i}||^\delta \right) \left(\sum_{i=1}^{n_d} ||w_{B_i}||^2 \right) \quad (3.26)
 \end{aligned}$$

Using the above inequalities, the Lyapunov's condition can be derived as:

$$\begin{aligned}
 \frac{1}{s_{n_d}^{2+\delta}} \sum_{i=1}^{n_d} E [|\langle w_{B_i}, Z_{B_i} \rangle|] &\leq \frac{C_\delta \left(\max_i ||w_{B_i}||^\delta \right) \left(\sum_{i=1}^{n_d} ||w_{B_i}||^2 \right)}{(\lambda_0 \sum_{i=1}^{n_d} ||w_{B_i}||^2)^{1+\delta/2}} \\
 &= K \left(\max_i \frac{||w_{B_i}||^2}{\sum_{j=1}^{n_d} ||w_{B_j}||^2} \right)^\delta \quad (3.27)
 \end{aligned}$$

Where K is a constant function of C_δ and λ_0 . For $n_d \rightarrow \infty$, the right hand side of the inequality goes to 0 and since the left hand side is a positive value, this also goes to 0. Thus the Lyapunov conditions holds which complete the proof.

Slutsky Theorem: Let U_i, V_i be sequences of random elements. If X_i converges in distribution to a random element X and Y_i converges in probability to a constant c , then

CHAPTER 3. MINIMIZATION OF THE CLASSIFICATION ERROR

$$\begin{aligned}
X_i + Y_i &\xrightarrow{\mathcal{L}} X + c \\
X_i Y_i &\xrightarrow{\mathcal{L}} cX \\
X_i / Y_i &\xrightarrow{\mathcal{L}} X/c \quad c \neq 0
\end{aligned} \tag{3.28}$$

Now, we are ready to give the proof of the Theorem ??.

Proof of Theorem ??: First, we prove that:

$$\lim_{n_d \rightarrow \infty} \frac{Var(\sum_{i=1}^{n_d} \langle w_{B_i}, Z_{B_i} \rangle)}{Var(\sum_{i=1}^{n_d} \langle w_{B_i}, Z_{B_i} \rangle | Y)} = 1 \tag{3.29}$$

P_Y almost surely. For this we write:

$$\begin{aligned}
Var\left(\sum_{i=1}^{n_d} \langle w_{B_i}, Z_{B_i} \rangle\right) &= \sum_{i=1}^{n_d} Var(\langle w_{B_i}, Z_{B_i} \rangle) \\
&\quad + \sum_{i \neq j} Cov(\langle w_{B_i}, Z_{B_i} \rangle, \langle w_{B_j}, Z_{B_j} \rangle)
\end{aligned} \tag{3.30}$$

But:

$$\sum_{i=1}^{n_d} Var(\langle w_{B_i}, Z_{B_i} \rangle) = Var\left(\sum_{i=1}^{n_d} \langle w_{B_i}, Z_{B_i} \rangle | Y\right) \tag{3.31}$$

because of the assumption saying that the covariance matrices of the blocks do not depend on the classes, A2, and the conditional independence of the X_{B_i} 's given Y , assumption A1. All we need to prove then is

$$\frac{\sum_{i \neq j} Cov(\langle w_{B_i}, Z_{B_i} \rangle, \langle w_{B_j}, Z_{B_j} \rangle)}{\sum_{i=1}^{n_d} Var(\langle w_{B_i}, Z_{B_i} \rangle)} \rightarrow 0 \tag{3.32}$$

CHAPTER 3. MINIMIZATION OF THE CLASSIFICATION ERROR

Now we have that:

$$\sum_{i \neq j} Cov(\langle w_{B_i}, Z_{B_i} \rangle, \langle w_{B_j}, Z_{B_j} \rangle) \leq \max_i \|w_{B_i}\|^2 \sum_{i \neq j} \lambda_{i,j} \quad (3.33)$$

where $\lambda_{i,j}$ is the largest eigen-value of the covariance matrix $Cov(\langle w_{B_i}, Z_{B_i} \rangle, \langle w_{B_j}, Z_{B_j} \rangle)$.

Assumption A4 ensures that:

$$\max_i \|w_{B_i}\|^2 \sum_{i \neq j} \lambda_{i,j} \leq c1 \quad (3.34)$$

$$(3.35)$$

$$\max_i \|w_{B_i}\|^2 \quad (3.36)$$

where $c1$ is a positive constant. Now, using assumption A2, we know that:

$$\sum_{i=1}^{n_d} Var(\langle w_{B_i}, Z_{B_i} \rangle) > c \sum_{i \leq n_d} \|w_{B_i}\|^2 \quad (3.37)$$

Finally, assumption A6 ensures the convergence of our ratio to 0. Now using the result:

$$\lim_{n_d \rightarrow \infty} \frac{Var(\sum_{i=1}^{n_d} \langle w_{B_i}, Z_{B_i} \rangle)}{Var(\sum_{i=1}^{n_d} \langle w_{B_i}, Z_{B_i} \rangle | Y)} = 1 \quad (3.38)$$

and using the first Lemma and the Slutsky's theorem, for all Y :

$$\frac{\sum_{i=1}^{n_d} \langle w_{B_i}, Z_{B_i} \rangle}{\left(Var(\sum_{i=1}^{n_d} \langle w_{B_i}, Z_{B_i} \rangle)\right)^{\frac{1}{2}}} \xrightarrow{\mathcal{L}|Y} \mathcal{N}(0, 1) \quad (3.39)$$

If we denote $F_{T_d}(t)$ and $F_{T_d|Y}(t)$ to be the characterisitic function of T_d and $T_d|Y$, respectively, then:

$$F_{T_d}(t) = \mathbb{E}_Y (F_{T_d|Y}(t)) \rightarrow \exp -\frac{t^2}{2} \quad (3.40)$$

which terminates the proof ¹.

3.2.2 Estimation of the Covariance Matrix

The accuracy of the estimation of the covariance matrix of $Z \in \mathbb{R}^{d+1}$ highly depends on the feature dimension, d , as well as the number of training samples, n . For the case that number of samples is significantly lower than the number of dimensions, this estimation is more challenging. Typically for the cases that $\frac{d}{n} \leq 0.1$, estimating the empirical covariance matrix $\hat{C}$ using

$$\hat{C} = \frac{1}{n-1} \sum_{i=1}^n (Z_i - \bar{Z})(Z_i - \bar{Z})^T \quad (3.41)$$

can lead to several issues. First of all, since the rank of $\hat{C}$ is necessarily less than $\min(d+1, n)$, the empirical estimation $\hat{C}$ is not invertible for $n < d+1$. Even when n is greater than d but still close to d , the matrix $\hat{C}$ is ill-conditioned. Moreover, the usual empirical estimator for the covariance matrix results in high variance (overfitting issues) in the case of high dimensional data and relatively small sample size.

It is necessary to propose a regularized estimator fixing the above issues. The estimation method we adopt is presented in [52]. For the true covariance matrix, C , assume the empirical estimator $\hat{C}$ follows a Wishart distribution $\hat{C} \sim \mathcal{W}(C, n-1)$ (it would

¹we don't even need dominated convergence theorem here even if it applies since Y takes a finite number of values

CHAPTER 3. MINIMIZATION OF THE CLASSIFICATION ERROR

follow exactly this distribution if Z was a multivariate Normal vector). Next, put a conjugate prior on C . The most natural conjugate prior in this setting is the inverse Wishart distribution $\mathcal{IW}(S_0^{-1}, \nu_0)$, where S_0 and ν_0 are hyper parameters such that S_0 is a definite positive matrix and ν_0 an integer representing the degrees of freedom of the distribution. The estimator $\hat{C}_{MAP}$ is then derived as follows:

$$\hat{C}_{MAP} = \underset{C}{\operatorname{argmin}} \mathbb{E} \left(\|C - \hat{C}\|_F^2 \right) \quad (3.42)$$

The solution to this problem is:

$$\hat{C}_{MAP} = \lambda \Sigma_0 + (1 - \lambda) \hat{C} \quad (3.43)$$

where $\Sigma_0 = \frac{S_0}{N_0}$, $\lambda = \frac{N_0}{N_0 + N}$ and $N_0 = \nu_0 + 1 + d$. With $S_0 = \operatorname{diag}(\hat{C})$, the estimator $\hat{C}_{MAP}$ can be simplified as, [52]:

$$\hat{C}_{MAP}(i, j) = \begin{cases} \hat{C}(i, j) & \text{if } i = j \\ (1 - \lambda) \hat{C}(i, j) & \text{otherwise} \end{cases} \quad (3.44)$$

The parameter λ will be computed using cross validation. There are several other methods to derive a regularized estimator of the covariance C , see for example [53, 54]. However, so far our best empirical results have been obtained using the cross validation estimation of λ .

3.2.3 Properties

Now that we exposed the steps used to build the $C^{-1}\mu$ classifier, let us underline some of its important properties:

1. It is easy to train: it reduces to an estimation problem, where the only complicated estimation is the covariance matrix estimation which is of dimension $(d+1) \times (d+1)$. In the case where d is very large, the $C^{-1}\mu$ will required to be combined with a *proper* dimension reduction mechanism. This will be discussed in the section 3.3
2. Despite the reduction to an estimation problem, we do not parameterize the joint distribution (X, Y) . Usually, the classifiers obtained by estimating parameters are parametric models (LDA, QDA, Naive Bayes).
3. Finally, the classifier provides an *analytical* estimation of the generalization error,

$$P(\hat{Y} \neq Y) = \phi(-\langle \hat{\mu}, \hat{C}^{-1} \hat{\mu} \rangle^{\frac{1}{2}}) \quad (3.45)$$

This has computational advantages compared to the cross-validation-based estimation of the generalization error.

3.2.4 A Toy Example

In this subsection, we present a toy example which help understanding the intuition behind the $C^{-1}\mu$ classifier in a very simple situation. It is assumed that the covariance matrix C and the mean μ of the vector Z are perfectly known. Denote by π_1 and π_{-1} the probability of each class (1 or -1). We take a feature vector X of dimension d such that the joint distribution (X, Y) is defined as follows. For each i in $1 \leq i \leq d$, $X_i = YU_i$, where the U_i 's are independent Normal random variables with mean $\mu_i = \frac{1}{i}$ and variance σ^2 , that does not depend on i . Assume that μ_i and σ are also known. Here, we designed the joint distribution (X, Y) such that the three first conditions of the Central Limit Theorem discussed above are satisfied. The whole discussion will be on the fourth condition, which concerns the vector w that we choose.

Notice that the LDA classifier in this case (using the known parameters) is the best classifier, since it corresponds to the decision rule :

$$Y = \mathbf{1}_{\eta(X) \geq \frac{1}{2}} - \mathbf{1}_{\eta(X) < \frac{1}{2}} \quad (3.46)$$

But recall also that the *LDA* classifier belongs to the set of linear classifiers. It can be represented by the Fisher discriminant vector that we will denote by w' . On the other hand our proposed vector $w = C^{-1}\mu$ is optimal (asymptotically), with respect to the misclassification problem over the set of w 's that are matching with the central limit theorem stated above. First, we will explain why it is reasonable for this toy example

CHAPTER 3. MINIMIZATION OF THE CLASSIFICATION ERROR

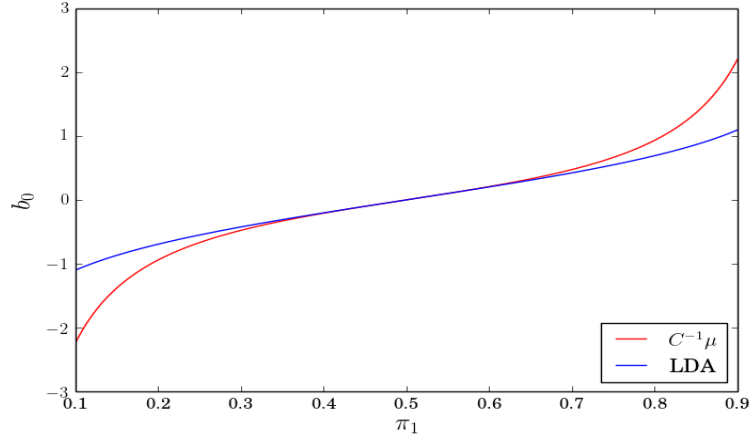


Figure 3.1: The $C^{-1}\mu$ classifier and the LDA classifier agree perfectly in this toy example except for the constant parameter b_0 which is plotted for both estimators. Note that the two estimators are close when the prior for class 1, π_1 is close to 0.5. however, they disagree when the prior is very skewed.

to state that $C^{-1}\mu$ satisfies the central limit theorem assumptions. Then we will examine the cases where the LDA discriminant w' "satisfies" the CLT conditions, and compare the misclassification error of our classifier and the LDA. One should expect that when the LDA discriminant satisfies the C.L.T conditions, our classifier will be very close to the optimal Bayes error, i.e the LDA misclassification error in this case, since it would optimize the misclassification error over a set to which belongs the LDA classifier.

First of all let us compute explicitly w and w' . Computing $C^{-1}\mu$ in this case gives:

$$w = \left(\frac{2\pi_1 - 1}{1 - (2\pi_1 - 1)^2}, \frac{\mu_1}{\sigma^2}, \dots, \frac{\mu_i}{\sigma^2}, \dots, \frac{\mu_d}{\sigma^2} \right)^T \quad (3.47)$$

CHAPTER 3. MINIMIZATION OF THE CLASSIFICATION ERROR

The vector b' corresponding to LDA is:

$$w' = \left(\frac{1}{2} \log\left(\frac{\pi_1}{1 - \pi_1}\right), \frac{\mu_1}{\sigma^2}, \dots, \frac{\mu_i}{\sigma^2}, \dots, \frac{\mu_d}{\sigma^2} \right)^T \quad (3.48)$$

Notice that the two vectors w and w' have the same components from component 1 to d . The difference between the two vectors lies in the first components w_0 and w'_0 . It is reasonable (since the sum $\sum_{i=1}^d \frac{1}{i}$ diverges) to assume that condition 5 in the stated CLT is satisfied for w and w' when the prior π_1 is close to $\frac{1}{2}$ since w_0 and w'_0 stay close to zero around $\pi_1 = \frac{1}{2}$ and diverges to infinity at $\pi_1 = 0$ and $\pi_1 = 1$, see Figure 3.1. This is perfectly consistent with the $C^{-1}\mu$ classifier optimality property that we expect: The $C^{-1}\mu$ classifier should be (at least approximately) optimal over all the classifiers for which the C.L.T holds, which is the case when the prior is not too skewed. LDA classifier belongs also to the set of these classifiers when the prior is not too skewed also, and at the same time it is optimal over all the classifier. This implies that when the prior is not too skewed, the misclassification error of LDA and $C^{-1}\mu$ classifiers should be (approximately) the same. This is exactly what we observe since $w \approx w'$ when the class prior is close to $\frac{1}{2}$. Note that in practice, in case the prior is skewed, an alternative version of the $C^{-1}\mu$ classifier consists in estimating w_1 as described and then estimating w_0 using cross validation.

3.3 Non-linear Extension

Designing a recognition system with $C^{-1}\mu$ classification constitutes two pahsess: 1. Feature extraction 2. Parameter Estimation. While the feature extraction parameters can be estimated independently of the classification parameters, there is always a benefit of training the feature extraction *jointly* with the classification module. Such joint training has been addressed in the context of *Discriminative Feature Engineering* [55].

The feature extraction module maps the raw measured pattern x_i into the feature vector $f_\theta(x_i)$ for which θ is the feature extraction parameters. The feature map belongs to a feature space $\mathbb{F}_\theta$ which can be parameteric or non-parameteric. Notating the linear classifier parameters by w , the objective loss function for the $C^{-1}\mu$ classifier is as follows:

$$L(\theta, w) = \frac{-\langle w, \mu_\theta \rangle}{\sqrt{\langle w, C_\theta w \rangle}} \quad (3.49)$$

where, μ_θ and C_θ are the first and the second order statistics of the $z_i = f_\theta(x_i)y_i$ while y_i is the i 'th pattern label.

The objective of the joint recognition system with $C^{-1}\mu$ classifier, is to find pair (w^*, θ^*) that minimizes the above loss function:

$$(\theta^*, w^*) = \underset{\theta, w}{\operatorname{argmin}} L(\theta, w) \quad (3.50)$$

CHAPTER 3. MINIMIZATION OF THE CLASSIFICATION ERROR

over all possible feature map parameters θ and the coefficients w . In Section 3.2, the classifier coefficient that minimizes the loss for fixed feature mapping f_θ was shown to be $w^* = C_\theta^{-1}\mu_\theta$. This induce the follwing two step minimization scheme to find the optimum pair (θ^*, w^*) :

step 1: Let $w = C_\theta^{-1}\mu_\theta$ in the loss function and find the optimum θ^* :

$$\begin{aligned}\theta^* &= \underset{\theta}{\operatorname{argmin}} L(\theta, C_\theta^{-1}\mu_\theta) \\ &= \underset{\theta}{\operatorname{argmin}} -\langle \mu_\theta, C_\theta \mu_\theta \rangle\end{aligned}\tag{3.51}$$

step 2: The optimum w^* is then simply equal to $C_\theta^{-1}\mu_\theta$ for $\theta = \theta^*$.

The optimum θ^* can be derived using gradient descent. At each step i , the θ_i can be updated as follows:

$$\theta_i = \theta_{i-1} - \alpha \frac{\partial L(\theta_i, w_{\theta_i})}{\partial \theta_i}$$

where,

$$\frac{\partial L(\theta, w_\theta)}{\partial \theta} = \sum_{j=1}^n \frac{\partial L(\theta, w_\theta)}{\partial f_\theta(x_j)} \frac{\partial f_\theta(x_j)}{\partial \theta}\tag{3.52}$$

here, w_θ is $C_\theta^{-1}\mu_\theta$ and α is the learning rate.

CHAPTER 3. MINIMIZATION OF THE CLASSIFICATION ERROR

The graident derivation is pretty straightforward and is summarized as follows:

$$\begin{aligned}
\frac{\partial L(\theta, b_\theta)}{\partial f_\theta(x_j(k))} &= \mu_\theta^T \frac{C_\theta^{-1}}{\partial f_\theta(x_j(k))} \mu_\theta^T \\
&+ 2\mu_\theta^T C_\theta^{-1} \frac{\partial \mu_\theta}{\partial f_\theta(x_j(k))} \\
\frac{C_\theta^{-1}}{\partial f_\theta(x_j(k))} &= -C_\theta^{-1} \frac{C_\theta}{\partial f_\theta(x_j(k))} C_\theta^{-1}
\end{aligned} \tag{3.53}$$

for $s \neq t$,

$$\frac{\partial C_\theta(s, t)}{\partial f_\theta(x_j(k))} = \begin{cases} \frac{1}{n-1} y_j(z_j(t) - \mu_z(t)) & \text{if } i \in \{s, t\} \\ 0 & \text{Otherwise} \end{cases} \tag{3.54}$$

and

$$\frac{\partial C_\theta(i, i)}{\partial f_\theta(x_j(k))} = \frac{2}{n-1} y_j(z_j(i) - \mu_z(i)) \tag{3.55}$$

The convergence of the above iterating process highly depends on the choice of the feature function space, $\mathbb{F}_\theta$. In this paper, we use graident descent to minmize above loss for two class of feature spaces: *Neural Network*, and *Reproducing Kernel Hilbert Space* (RKHS).

3.3.1 Neural Networks

One can view the many successful Deep Neural Network results as special cases of joint optimization of features and classification. From this perspective, the primary layers right before the classification layer provides the feature space with parameters corresponding to the layers weights. The primary layers can be chosen from a pool of possibilities among linear and non-linear structures while the classification layer is usually set to be the softmax.

Here, we propose to use the $C^{-1}\mu$ classifier as the final layer of a deep structure. The only change which need to be made is the derivation of the loss gradients which was given in the previous section. The same back-propagation algorithm [56] is then used for learning the feature space parameters. For the neural network, the feature space $\mathbb{F}_\theta$ is parameteric since it is the weights of the layers before the classification layer. This faciitate the minimization process of step 1. Next, a non-parametric feature space is addressed.

3.3.2 Reproducing Kernel Hilbert Space (RKHS)

Lets the feature space $\mathbb{F}_\theta$ be RKHS of functions taking values in $\mathbb{R}^l$ produced by a multi-dimensional kernel matrix $K(x_i, x_j)$ which associates the matrix for any sample

CHAPTER 3. MINIMIZATION OF THE CLASSIFICATION ERROR

pairs x_i and x_j .

The minimum over the feature space is achieved within a subspace $\mathbb{F}'_\theta$ of functions $f_\theta(x)$ which

$$f_\theta(x) = \sum_{k=1}^n K(x, X_k) \theta_k \quad (3.56)$$

The minimization problem is then aim to minimize the following loss function over the subspace $\mathbb{F}'_\theta$:

$$L'(\theta, w) = L(f_\theta(x_1), \dots, f_\theta(x_n)) + \lambda \|f\|_{RKHS}^2 \quad (3.57)$$

here we introduce the regularization term $\lambda \|f\|_{RKHS}^2$ to prohibit the possibility of overfitting due to the large size of feature space. The graidents are then computed as follows:

$$\frac{\partial L'(\theta, w_\theta)}{\partial f_\theta(x_j)} = \frac{\partial L(\theta, w_\theta)}{\partial f_\theta(x_j)} + 2\lambda f_\theta(x_j) \quad (3.58)$$

3.4 Experiments

The experiments are designed with special attentions to the following issues: (1) Head-to-head comparison of the $C^{-1}\mu$ performance against the other classifiers, LDA, SVM, and Logistic Regression; (2) The extent the proposed classifier scaled with data set size; (3) The extent to which joint optimization of feature and classification improves

over separate optimization for the $C^{-1}\mu$ classifier.

3.4.1 Data

Dopamine transporter (DAT) imaging is increasingly utilized for diagnostic purposes in suspected Parkinson's diseases (PD). The mean regional uptake in the Striatum, which is a subcortical part of the forebrain is the measurement of reference. However, it is hypothesised that finer measurement, at the voxel level, would allow for a finer diagnosis and the tracking of the progression of PD. The raw data was acquired as part of the Parkinson's Progressive Marker Initiative, see [57]. The processed data was obtained using the technique described in [58]. It contains 195 images of the Caudate (left and right hemisphere) which is a substructure of the Striatum for 77 healthy control (HC) subjects and 118 subjects diagnosed with PD. The images are registered using a sophisticated registration algorithm, see [59], onto a common coordinate system allowing to compare voxel per voxel the DAT SPECT uptake for each subject. Each image is made of 3969 voxels, with values ranging from 5.00 to 540.34. We will consider the classification problem of HC versus PD for each image. Due to the relatively small data size ($n=195$) and the absence of a separate test set, we assess the performance of the classification algorithms using leave-one-out cross-validation.

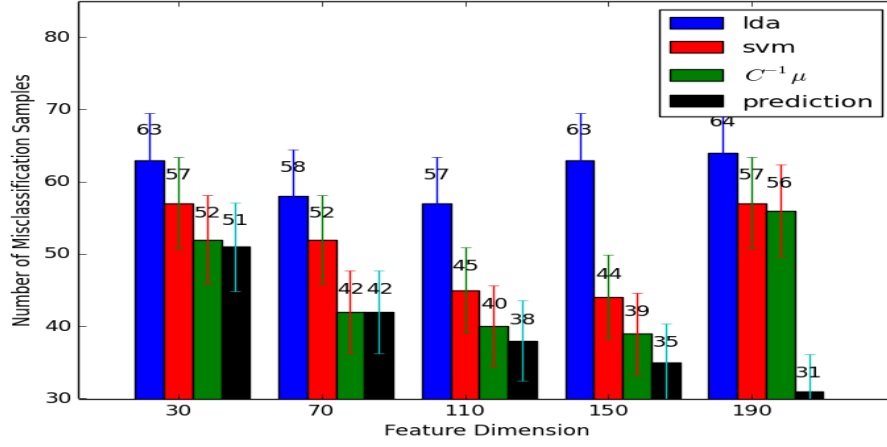


Figure 3.2: Misclassification rate versus number of principle components.

MNIST: The standard MNIST handwritten character recognition data set was used. From the total 60k training samples, 59k was used for training and the rest was used for validation. The standard 10k test sample size was used for testing. Every instance is a 28 28 pixel grayscale image which used as raw features for all experiments [60]. The binary recognition tasks were defined to discriminate the classes below 5 from classes above it.

3.4.2 Parkinson Recognition

Figure 3.2 compares the misclassification rate of the $C^{-1}\mu$ classifier against two other classifiers, svm and LDA on the left caudate data. The features are the principle components after performing PCA. This simulates the *separate* recognition scheme discussed earlier. Same validation data was used for tuning the parameters of all three

CHAPTER 3. MINIMIZATION OF THE CLASSIFICATION ERROR

classifiers. We also have examined the logistic regression on this data set, however the performance was far from the other three classifiers shown in Figure 3.2. The predicted misclassification rate for the $C^{-1}\mu$ classifier is also depicted in this figure. The predicted values are very accurate for lower dimensional features. The difference between the predicted error and classification error for large dimensional features can be explained by overfitting due to the small size of the training data.

Due to small size of the Parkinson data, we did not observe significant improvement on training joint recognition systems using $C^{-1}\mu$ classifier. However the best misclassification rate achieved was 30 for which the both orientation of the caudate data was used to train a separate recognition system with PCA components as feature and $C^{-1}\mu$ as classifier.

3.4.3 Character Recognition

The efficiency of joint recognition systems over the separate ones are more visible on the character recognition task. Table 3.1 compares the recognition performance of three systems. In all three systems, the original high dimensional feature is embedded to a lower dimension subspace. This has been done using PCA for the first system. For the other two systems, the projection was done through the joint training of a hidden layer and loss function. As this tables shows, the joint training outperforms

CHAPTER 3. MINIMIZATION OF THE CLASSIFICATION ERROR

the separate training for all feature dimensions shown in the table. However, there is not a significant difference between the joint system trained with $C^{-1}\mu$ loss and the one trained with the cross entropy loss.

feat-dim	PCA + $C^{-1}\mu$	NNet + $C^{-1}\mu$	NNet + Softmax
10	27.5 ± 0.0018	5.3 ± 0.001	5.6 ± 0.0009
30	19.5 ± 0.0016	3.8 ± 0.0008	4.2 ± 0.0008
50	17.5 ± 0.0016	3.4 ± 0.0007	3.6 ± 0.0008

Table 3.1: Comparison of the misclassification rate for the separate and joint recognition system.

In addition to the neural network architecture, the kernelized version of the proposed classifier is examined in Table 3.2. Due to the scalability of the kernel methods for large data sets, for this experiment we only used the first 10k of the original training data to train the kernel parameters. For the kernel systems we used a simple linear kernel. The kernel systems compared to the separate system trained on the same dataset. Again, the kernelized $C^{-1}\mu$ is in par with the SVM classifier while both kernel systems outperform the separate system.

feat-dim	PCA + SVM	PCA + $C^{-1}\mu$	Kernel + $C^{-1}\mu$
10	27.8 ± 0.0045	27.8 ± 0.0045	16.2 ± 0.0037
30	18.0 ± 0.0038	19.9 ± 0.0040	15.4 ± 0.0036
50	16.8 ± 0.0037	17.6 ± 0.0038	14.7 ± 0.0035

Table 3.2: Comparison of the misclassification rate for the separate and joint recognition system using kernel method.

3.4.4 Error Prediction

Table 3.3 presents the prediction error for three recognition systems used the $C^{-1}\mu$ classifier. The prediction error and the actual error are pretty close for the separate system and the joint kernel system. However, the error predicted for the joint system with embedded hidden layer, is far smaller than the observed error.

Front-End	Classification	Prediction
PCA	19.5 ± 0.0016	19.9 ± 0.0016
NNet	3.8 ± 0.0008	0.01 ± 0.0004
Kernel	15.4 ± 0.0036	15.1 ± 0.0036

Table 3.3: Comparison of the misclassification rate and the prediction rate for three systems.

3.5 Conclusion

We have presented a novel family of classifiers named $C^{-1}\mu$ which applies to the situation where the number of features is large enough. We have demonstrated using real data that it provides competitive results compared to other binary classifiers, as SVM, Logistic regression, LDA and Neural Networks. However, many open questions remain open. In particular, this classifier does not apply to all classification situation as it relies on the Normal approximation of the classification error. Would it be possible before hand to know if this classifier is applicable to a given situation? Also,

CHAPTER 3. MINIMIZATION OF THE CLASSIFICATION ERROR

most classification problems are not binary. Is there then a method to generalize the

Normal approximation to the multiclass setting?

Chapter 4

Multiclass Classification with C^{-1}_μ

In Chapter 2, we described the main concept of C^{-1}_μ classification method for case of binary classification. The natural idea of minimizing the probability of error is easily extendable to the multi class classification which is of important interest for many practical machine learning tasks. This chapter is devoted to develop the multiclass C^{-1}_μ classifier. Starting from linear classification, the classifier is extended to the non linear classification by using it as classification layer of deep neural network structure.

4.1 Introduction

Most classification algorithms were devised first for binary classification problems. However, most large scale practical problems involve classifying the feature vector X into a class Y among more than two choices. This defines the essence of the multiclass classification problem.

There are two ways of deriving a multi class classification based on the original binary classifier. The Support vector machines (SVMs) is first introduced for binary classification [61] They are accurate, robust and quick to apply to test instances. However, the elegant theory of using large-margin hyperplanes cannot be easily extended to multi-class classification problems [62]. For such algorithms, usually a *reduction* algorithm is used to reduce the multiclass problem into set of binary classification.

There exist a number of reduction approaches; One Versus the Rest (OVR) method [63], Pairwise Comparison [64], Direct Graph Traversal [65], Error Correcting Output Coding (ECOC) [66, 67] and specifically for SVM, multiclass objective functions has been proposed [68]. In practice choice of the reduction method is not easy. Each of these methods works under certain circumstances of the problem [67]. In addition, reduction of large scale multiclass problem into large number of binary classification is not currently feasible in large scale domain for all classification techniques mainly due to the training time and computation efficiency [62].

CHAPTER 4. MULTICLASS CLASSIFICATION WITH $C^{-1}\mu$

For some classification algorithms, the multiclass extension is very straight forward because of the elegant idea behind the classification design at the first place. Linear Discriminant Analysis was first described as two classes problem [7]. The multiclass extension of LDA could be easily derived via multiple discriminant analysis [69]. The multiclass LDA has been widely used in practice [70, 71]. Logistic Regression is another type of classification technique that inherit multiclass definition from its original idea. This with the feature extraction capability of the Deep Neural Networks [24] has led to superior recognition performance in many tasks [19, 20, 21, 22].

The idea behind developing the $C^{-1}\mu$ classifier is easily extendable to the multiclass classification. Similar to the binary case, we develop an algorithm that aim to minimize the *true probability of error*. Combining with deep feature extraction ability of Deep Neural Networks, we proposed a classification technique which is deep and at the same time minimizing the probability of misclassification error explicitly.

4.2 $C^{-1}\mu$: Multi-Class Extension

For given set of n training pairs $\{x_i, y_i\}_{i=1}^n$, goal of a multiclass classification algorithm is to derive a map between feature space $\mathbb{X}$ to class space $\mathbb{Y}$. It is assumed the feature space lie down in $\mathbb{R}^d$ where d is the feature dimension and there is $|\mathbb{Y}| = C$ number of classes. For a feature vector $X \in \mathbb{X}$, the multiclass of interest decides class $\hat{Y} = f(X)$

CHAPTER 4. MULTICLASS CLASSIFICATION WITH $C^{-1}\mu$

with the following decision rule:

$$\begin{aligned} f(X) &= \operatorname{argmax}_{i \in \{1, \dots, C\}} W_{i0} + \sum_{j=1}^d W_{ij} X_j \\ &= \operatorname{argmax}_{i \in \{1, \dots, C\}} \langle W_i, X^e \rangle \end{aligned} \quad (4.1)$$

where W_i is the i 'th row of the $C \times (d+1)$ weight matrix W . Here, X^e is the *extended* X which is derived simply by adding a single 1 as the first component of X .

Introducing this notation make the error notations easier to describe.

Similar to the binary case, the goal is to find the weight matrix W which minimizes the classification error. The classification error occurs if there exist a class $i \in \{1, \dots, C\}$ which $i \neq y$ and:

$$W_{y0} + \sum_{j=1}^d W_{yj} X_j < W_{i0} + \sum_{j=1}^d W_{ij} X_j \quad (4.2)$$

In the occasion of error, there should be at least one such i while there is at most $C - 1$ of them, since there is only C classes.

4.2.1 Notations

Definition: For each class $y \in \{1, \dots, C\}$, we define C dimensional vectors $U_{i|y}$, $1 \leq i \leq C - 1$ constructed as follows:

- $U_{i|y}(j = y) = 1$

CHAPTER 4. MULTICLASS CLASSIFICATION WITH $C^{-1}\mu$

- If $i \leq y$, $U_{i|y}(j = i) = -1$
- If $i > y$, $U_{i|y}(j = i + 1) = -1$
- Otherwise $U_{i|y}(j) = 0$

As an example, the following vectors represent $U_{i|y}$ for different possible values of i and y :

	$U_{1 y}$	$U_{2 y}$
$y = 1$	$(+1, -1, 0)^T$	$(+1, 0, -1)^T$
$y = 2$	$(-1, +1, 0)^T$	$(0, +1, -1)^T$
$y = 3$	$(-1, 0, +1)^T$	$(0, -1, +1)^T$

Remark: The $U_{\cdot|y}$ is a random variable with the representation $U_{i|y}$, for $1 \leq i \leq C-1$.

Using the above definition, we can mathematically describe the occasion of the misclassification error of Eq. 4.2:

$$\begin{aligned}
 \langle W_y, X^e \rangle - \langle W_j, X^e \rangle &< 0 \\
 \langle U_{j|y}, W X^e \rangle &< 0
 \end{aligned} \tag{4.3}$$

CHAPTER 4. MULTICLASS CLASSIFICATION WITH $C^{-1}\mu$

The random variable Z is then defined as follows:

$$Z_{j|y} = \tilde{X}U_{j|y} \quad (4.4)$$

where $\tilde{X}$ is $(C(d+1)) \times C$ dimensional vector:

$$\tilde{X} = (I|X_1I|X_2I|\dots|X_dI)^T \quad (4.5)$$

here, I is the $C \times C$ dimensionanl identity matrix. The X_i denote the value i of the original feature vector X . In other words, $\tilde{X}$ contains d blocks of X_iI and one block of I , where blocks are separated by the notation $|$ in the above vector. The single block I is in fact simulate the addition of 1 to the vector X in definition of X^e .

For each class y there is $C - 1$ vector $Z_{i|y}$. The random vector Z_i is then defined as:

$$Z_i = \{Z_{i|y}|y \in \{1, \dots, C\}\} \quad (4.6)$$

We also use μ_i and C_i to present the mean and covariance of random variable Z_i , respectively.

The w is ued to present the vectorized version of the weight matrix W

$$w = (W'_1, W'_2, \dots, W'_{d+1})^T \quad (4.7)$$

Here W' is the transpose of W . The $C(d+1)$ dimensional vector w is built using concatenation of the columns of the weight matrix W .

4.2.2 Probability of the Misclassification

The misclassification even in case of multiclass classification can be formulated as follows:

$$f(X) \neq Y \iff \{\exists i \in \{1, \dots, C\} | w^T Z_i < 0\} \quad (4.8)$$

In other words, classification error occurs if and only if there exist a $1 \leq i \leq C - 1$ that $w^T Z_i < 0$. The probability of error is then written as:

$$\begin{aligned} P(f(X) \neq Y) &= P(\{\exists i \in \{1, \dots, C\} | w^T Z_i < 0\}) \\ &= P(\min_i w^T Z_i < 0) \\ &= 1 - P(\forall i : w^T Z_i > 0) \end{aligned} \quad (4.9)$$

In the last equality above, the subtracted term is in fact the classification accuracy. Assuming the density of the above probability is known, one can easily minimize the probability of error. Assuming central limit theorem holds, this density can be estimated with a multivariate normal. Minimizing the probability of error is then straightforward.

4.2.3 Bounds on Probability of Error

This section is devoted to introduce the upper bounds for the probability of error. We first describe a loose bound derived using the Bonferroni inequality [72]. Furthermore

CHAPTER 4. MULTICLASS CLASSIFICATION WITH $C^{-1}\mu$

a tighter upper bound is introduced using the Slepian inequality [72].

Theorem 4.2.1 *The minimum classification error for multi class classification is bounded from above as follows:*

$$\min P(f(X) \neq Y) \leq (C - 1) \max_i \min_w P(\langle w, Z_i \rangle < 0) \quad (4.10)$$

Proof: The classification error happens if

$$f(X) \neq Y \longleftrightarrow \{\exists i \in \{1, \dots, C\} | w^T Z_i < 0\} \quad (4.11)$$

Then,

$$\begin{aligned} P(f(X) \neq Y) &\leq \sum_{i=1}^{C-1} P(w^T Z_i < 0) \\ &\leq (C - 1) \max_{1 \leq i \leq C-1} P(w^T Z_i < 0) \end{aligned} \quad (4.12)$$

where the last inequality holds using Bonferroni inequality, [72]. Minimizing both side of the above inequality and using minmax equality, is then led to the desired inequality mentioned in the Theorem:

$$\begin{aligned} \min_w P(f(X) \neq Y) &\leq (C - 1) \min_w \max_{1 \leq i \leq C-1} P(w^T Z_i < 0) \\ &= (C - 1) \max_{1 \leq j \leq C-1} \min_w P(w^T Z_j < 0) \end{aligned} \quad (4.13)$$

which completes the proof.

CHAPTER 4. MULTICLASS CLASSIFICATION WITH $C^{-1}\mu$

The above theorem gives an upper bound for the misclassification error. By minimizing this upper bound, the misclassification error will be minimized too. For each term i in the upper bound

$$P(w^T Z_i < 0) = P\left(\frac{\langle w, Z_i - \mu_i \rangle}{\langle w, C_i w \rangle^{\frac{1}{2}}} < \frac{-\langle w, \mu_i \rangle}{\langle w, C_i w \rangle^{\frac{1}{2}}}\right) \quad (4.14)$$

Notating:

$$T_d^i = \frac{\langle w, Z_i - \mu_i \rangle}{\langle w, C_i w \rangle^{\frac{1}{2}}} \quad (4.15)$$

and assuming central limit theorem conditions hold, we can approximate the contribution of the i 'th term in the error sum as follows:

$$P(T_d^i < \frac{-\langle w, \mu_i \rangle}{\langle w, C_i w \rangle^{\frac{1}{2}}}) \sim \phi\left(\frac{-\langle w, \mu_i \rangle}{\langle w, C_i w \rangle^{\frac{1}{2}}}\right) \quad (4.16)$$

Next, note

$$\begin{aligned} \min_w P(w^T Z_j) &= \phi\left(\frac{-w^T \mu_j}{\langle w, C_j w \rangle^{\frac{1}{2}}}\right) \\ &= \phi(-\langle \mu_j, C_j^{-1} \mu_j \rangle^{\frac{1}{2}}) \end{aligned} \quad (4.17)$$

Using the above equality we can find the upper bound for the classification error:

$$\min_w P(f(X) \neq Y) \leq (C - 1) \max_j \phi(-\langle \mu_j, C_j^{-1} \mu_j \rangle^{\frac{1}{2}}) \quad (4.18)$$

So, this can lead to the first multi class classification algorithm for $C^{-1}\mu$ multi class classification.

CHAPTER 4. MULTICLASS CLASSIFICATION WITH $C^{-1}\mu$

Algorithm 1: Two step approximation of the $C^{-1}\mu$ parameters:

step 1: Find j^*

$$j^* = \operatorname{argmax}_j \langle \mu_j, C_j^{-1} \mu_j \rangle^{\frac{1}{2}}$$

step 2: Set w^*

$$w^* = C_{j^*}^{-1} \mu_{j^*}$$

The first multi class algorithm derived using the approximation of the upper bound of the error with the Bonferroni inequality. However, this is not the most tight error bound since it does not consider a lot of common event errors. A tighter upper bound is desired particularly for more complicated problems. Using the Slepian theorem [73], we derive a tighter upper bound for the missclassification rate.

Slepian Lemma: for Gaussian random variables $X = (X_1, \dots, X_n)$ and $Y = (Y_1, \dots, Y_n)$ in $\mathbb{R}^n$ satisfying the following:

$$E[X] = E[Y] = 0 \tag{4.19}$$

$$E[X_i^2] = E[Y_i^2], i = 1 \dots, n, \tag{4.20}$$

$$E[X_i X_j] \leq E[Y_i Y_j] \text{ for } i \neq j \tag{4.21}$$

the following inequality holds for all real numbers $r_1, \dots, r_n$:

$$P[X_1 \geq r_1, \dots, X_n \geq r_n] \leq P[Y_1 \geq r_1, \dots, Y_n \geq r_n] \tag{4.22}$$

CHAPTER 4. MULTICLASS CLASSIFICATION WITH $C^{-1}\mu$

Theorem 4.2.2 *Under the following assumption*

$$E[T_d^i T_d^j] \geq 0 \quad \forall 1 \leq i, j \leq C-1 \quad (4.23)$$

then minimizing the upper bound of the classification error is equivalent to the maximization of the following loss function:

$$L(Z, \mu, C) = \prod_{j=1}^{C-1} \phi\left(\frac{w^T \mu_j}{\langle w, C_j w \rangle^{\frac{1}{2}}}\right) \quad (4.24)$$

where ϕ is the multivariate normal distribution with mean 0 and covaraince matrix I , which is the $C \times C$ dimensional identity matrix.

Proof:

$$\begin{aligned} P(f(X) \neq Y) &= P(\exists i | w^T Z_i < 0) \\ &= 1 - P(\forall i | w^T Z_i > 0) \\ &= 1 - P(T_d^1 > r_1, \dots, T_d^{C-1} > r_{C-1}) \end{aligned} \quad (4.25)$$

where

$$r_i = \frac{-\langle w, \mu_i \rangle}{\langle w, C_i w \rangle^{\frac{1}{2}}} \quad (4.26)$$

and T_d^i was defined in Eq. 4.15. Using the Slepian lemma:

$$P(T_d^1 > r_1, \dots, T_d^{C-1} > r_{C-1}) > P(V_1 > r_1, \dots, V_C > r_{C-1}) \quad (4.27)$$

CHAPTER 4. MULTICLASS CLASSIFICATION WITH $C^{-1}\mu$

where V is a multivariate gaussian random variable with independent components and

$$E[V] = E[T_d] = 0 \quad (4.28)$$

$$E[V_i^2] = E[T_d^{i^2}], i = 1 \dots, n, \quad (4.29)$$

$$E[V_i V_j] = 0 \text{ for } i \neq j \quad (4.30)$$

under the assumption that $E[T_d^i T_d^j] \geq 0$ for any pairs of $1 \leq i, j \leq C-1$, we can apply the Slepian lemma to derive the following:

$$\begin{aligned} P(T_d^1 > r_1, \dots, T_d^{C-1} > r_{C-1}) &= \prod_{i=1}^{C-1} P(V_i > r_i) \\ &= \prod_{j=1}^{C-1} \phi\left(\frac{w^T \mu_j}{\langle w, C_j w \rangle^{\frac{1}{2}}}\right) \end{aligned} \quad (4.31)$$

The left side of the above equality is the correct classification event. Minimizing the error is equivalent to the maximization of the accuracy, and the right side of the above inequality is the proposed loss in the theorem.

The possibility of deriving a closed form equation for the w^* than minimize the above loss is not clear. Instead, we use Gradient Descent to find the weight vector that maximize the loss. The following equation provides the gradient of the above loss

CHAPTER 4. MULTICLASS CLASSIFICATION WITH $C^{-1}\mu$

with respect to the weight vector w :

$$\begin{aligned}
 \frac{\partial L}{\partial w} &= \sum_{j=1}^{C-1} \frac{\partial}{\partial w} \phi\left(\underbrace{\frac{\langle w, \mu_j \rangle}{\langle w, C_j w \rangle^{1/2}}}_x\right) \\
 &= \sum_{j=1}^{C-1} f(x, \mu_j, C_j) \frac{\partial x}{\partial w} \\
 &= \sum_{j=1}^{C-1} f(x, \mu_j, C_j) \langle w, C w \rangle^{-1/2} (\mu_j - \langle w, C w \rangle^{-1} C_j w w^T \mu_j) \quad (4.32)
 \end{aligned}$$

The second multi class algorithm is shown in the Algorithm 2. Here, we used Slepian bound derived above to minimize the classification error. The algorithm starts with the random initialization of the weight vector w_0 . At each iteration, the first and second order statistics are computed. These parameters are then used to derive the loss which itself is the accuracy of the model at the current iteration. Finally the partial derivative of the loss with respect to the weight vector w_{i-1} is computed and together with the learning rate α is used to update the weight w_i for next iteration. The algorithm continues until the desired number of iterations N .

CHAPTER 4. MULTICLASS CLASSIFICATION WITH $C^{-1}\mu$

Algorithm 2: Error minimization using Slepian bound:

step 1: Parameter Initialization

$$w_0 \in \mathbb{R}^{C(d+1)} \quad (4.33)$$

step 2: Mean and Covariance Computation

$$\begin{aligned} \mu_j &= E[Z_j] \\ C_j &= E[(Z_j - \mu_j)^T(Z_j - \mu_j)] \end{aligned} \quad (4.34)$$

while $i \leq N$ **do**

step 3: Loss Evaluation

$$L(Z, \mu, C) = \prod_{j=1}^{C-1} \phi\left(\frac{w^T \mu_j}{\langle w, C_j w \rangle^{\frac{1}{2}}}\right) \quad (4.35)$$

step 4: Parameter Update

$$w_i = w_{i-1} + \alpha \frac{\partial L}{\partial w_{i-1}} \quad (4.36)$$

end

4.3 Experiments

The multiclass $C^{-1}\mu$ experimented with two perspectives. The first set of experiments is designed to evaluate the performance of this classification technique on separate recognition module where the features are extracted separately from the classification parameters. To simulate this situation we used PCA as the mean of feature extraction.

CHAPTER 4. MULTICLASS CLASSIFICATION WITH $C^{-1}\mu$

The multiclass $C^{-1}\mu$ has been also evaluated on joint scheme. By removing the traditional softmax and layer on the top of the neural network, we used the proposed loss at the top of the deep neural network. This provides the possibility of having any number of linear or nonlinear layer before the loss.

The experiments presented here is using the MNIST dataset. The standard 59k training samples were used for the training. There is 1k samples used for validation of the model parameters, like the covariance shrinkage parameter λ . The standard 10k test set used for evaluation of the multiclass problem. Since the number of classes in MNIST dataset is 10, we only limited ourselves to model with fewer number of parameters. One important fact about the experiments designed here is, we do not like to evaluate our model on tuning scenarios. Thus, we believe the experiments are reproducible as it is mentioned here.

4.3.1 PCA + $C^{-1}\mu$

In the first experiment the original 784 dimensional features of the images were used to derive the PCA matrix. Among these 784 features, we only selected few number of features, 10, 20, 40, 80, 160.

The main parameters of our model are μ_j , C_j and the weight matrix W . After extraction of the PCA matrix, we directly compute the first and second order statistic

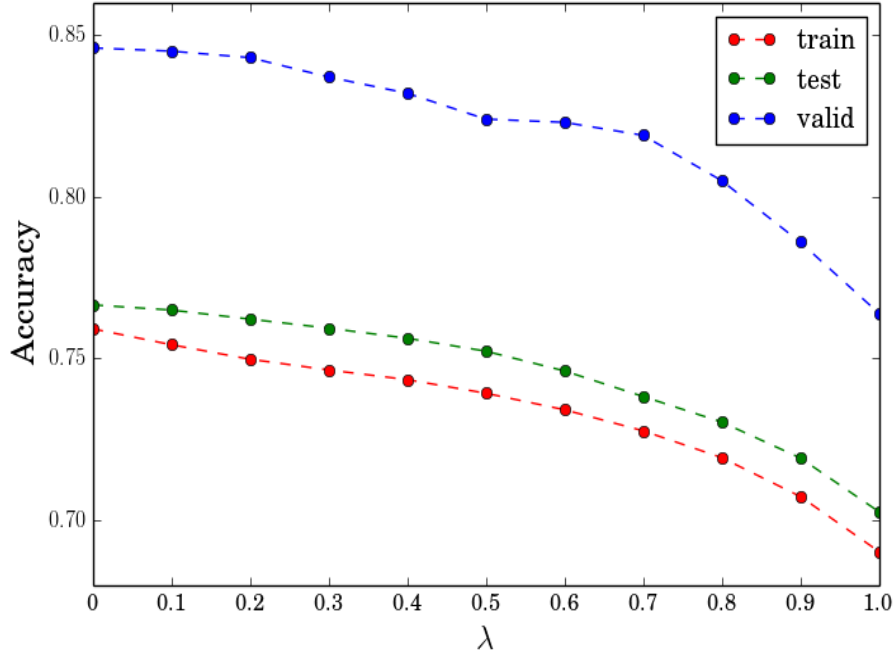


Figure 4.1: Effect of regularization term for shrinking the covariance matrix. PCA with 10 dimension is used for feature extraction.

parameters μ_j , C_j for $1 \leq j \leq C - 1$.

Efect of λ :

Like to the binary case, we use a grid search over the shrinkage parameter $0 \leq \lambda \leq 1$ with the incremental step 0.1 to find the λ to be used for shrinking the covariance matrix. Figure 4.1 presents how different values of λ affects the model accuracy on validation, training, as well as test data. We selecting the λ that maximize the accuracy on validation data. The author observed that for this dataset, usually $\lambda = 0$ is the optimum shrinkage parameter. Figure 4.1 also supports this claim for the 10 dimensional feature extracted by PCA.

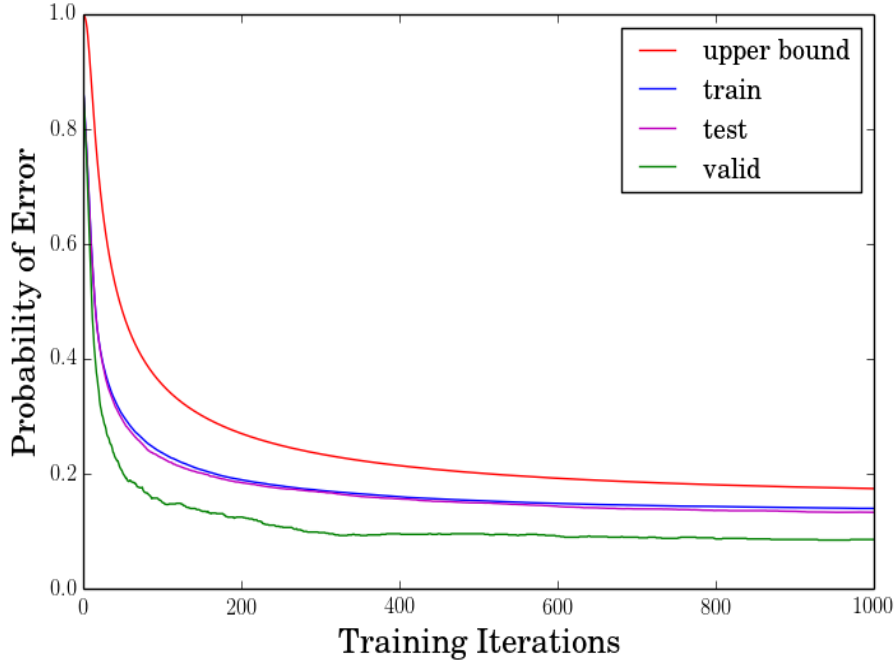


Figure 4.2: The proposed upper bound is minimized over number of iterations. The PCA dimension of 40 is used as feature with $\lambda = 0$.

Minimizing the Probability of Error:

The main idea behind the $C^{-1}\mu$ classifier is minimization of the probability of error by assuming the normal density for the probability of error. At each step of the Algorithm 2, we update the weight parameters W such that it minimizes the upper bound of the probability of error.

The second experiment is designed to monitor the training procedure of the $C^{-1}\mu$ using Algorithm 2. The 40 dimensional features were extracted separately using the *PCA*. As this figure shows, the true empirical probability of error derived on training, test and validation data is always below the upper bound proposed by the Slepian

CHAPTER 4. MULTICLASS CLASSIFICATION WITH $C^{-1}\mu$

lemma. Carefully observing the Theorem assumption, led to this finding that for all values of W , the covariances $E[T_d^i T_d^j]$ are positive which means the assumptions of theorem holds.

Effect of Feature Dimension

The feature dimension is of important interest in classification comparison. From one perspective, one can say increasing the feature dimension lead to larger models with more number of free parameters, thus higher accuracy is expected. However, if the number of samples is not large enough training large models might not necessarily result in huge improvement gap. These facts make it interesting to observe how the multiclass $C^{-1}\mu$ performs in case of large dimension model size. Similar to the previous cases, PCA was used to extract features in 5 different sizes, 10, 20, 40, 80, 160.

Figure 4.3 plots the upper bound of the misclassification error in Algorithm 2 as well as the true empirical error for training, validation and test data over the five different feature dimension of interest. The error is decreased by increasing the feature dimension. However, the improvement is not huge at the larger dimensional features. This could be explained by the number of parameters our model required. There is $d + 1$ free parameters for estimating each μ_j and $\frac{(d+1) \times (d+1)}{2}$ parameters in total for estimation of each covariance matrix C_j . In total, the $C^{-1}\mu$ classifier required $\frac{(C-1)(d+1)(d+3)}{2}$ parameters to be estimated. The ratio of the total number of parameter for the 160 dimensional feature to the total number of training samples is 4 which is relatively

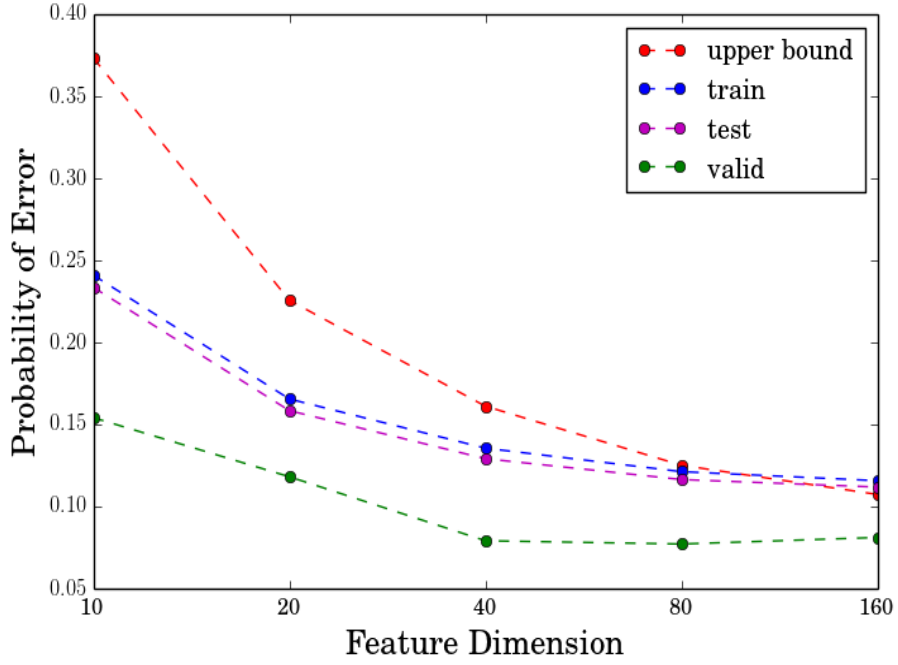


Figure 4.3: The probability of error versus number of PCA dimension.

smaller than typical number of samples required per parameter.

4.3.2 Neural Network + $C^{-1}\mu$

Replacing the traditional optimization module of the neural network with the $C^{-1}\mu$ loss is of sufficient interest for the author. The cross entropy loss only minimize the entropy between the true label probability vector and the estimated vector derived using the softmax layer. Here, we replace this last two layers with the multiclass $C^{-1}\mu$ loss function of the Algorithm 2. By doing normal back propagation, we update all parameters of the model jointly together.

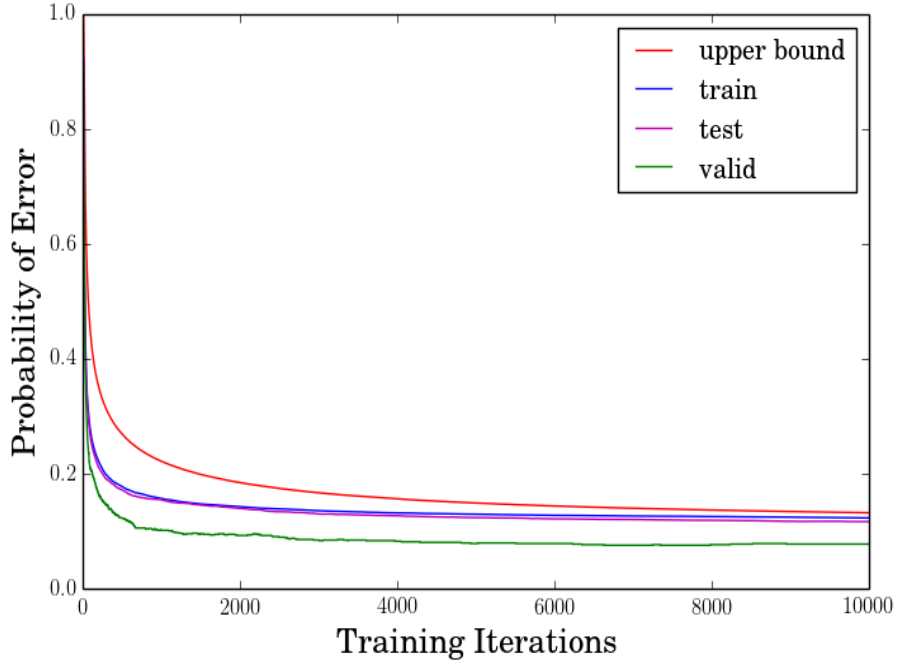


Figure 4.4: The probability of error versus number of iterations for single linear hidden layer and the $C^{-1}\mu$ loss.

Here, we start with raw 784 dimensional features. There is no extra feature engineering. The first layer is a linear layer. The experiments were done on the same feature dimension used for the PCA experiments. We designed two experiments, one is the linear module and the other one is the non linear system.

Linear Model

Figure 4.4 presents the proposed theoretical upper bound of the misclassification error and the true empirical error for training, validation and test condition. The dimension of the hidden layer is 10.

The final error rate of this model is 11.61% which is significantly lower than the

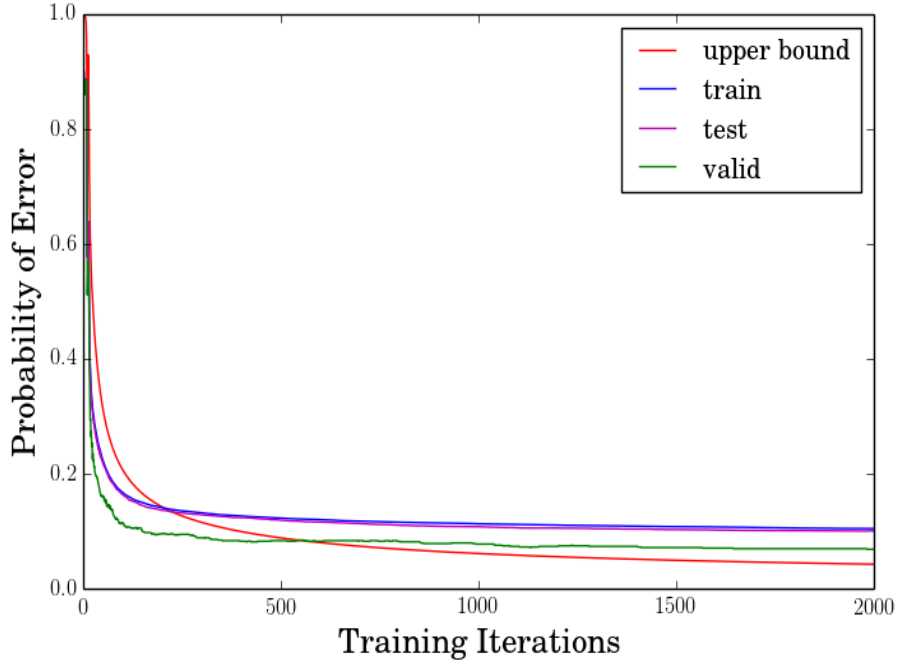


Figure 4.5: The probability of error versus number of iterations for single sigmoid hidden layer and the $C^{-1}\mu$ loss.

error rate of the PCA setting in Figure 4.3 which is 27.6%. Both models have same number of parameters and same classification module. This boost in improvement clearly comes from the joint optimization discussed in previous chapters.

Effect of Non-Linearity

One of the main benefit of the neural network structure over other structures is the non-linearities used in different layer. This has been examined in details in Chapter 2. Here, we used a sigmoid non-linearity in front of the hidden layer.

Figure 4.5 shows the training curve of the model with 10 dimensional hidden layer and a sigmoid non-linearity. In terms of improvement, the non-linear structure provides

additional 4% improvement over the linear structure after convergence. However, we observe that for the first time the theoretical upper bound does not hold after few hundred training epochs. Looking at the weights at this stage, we noticed that the positivity of the off-diagonal component of the covariance matrix does not hold for this case. This is one major direction of the further research.

4.4 Conclusion

This chapter extended the binary classification technique $C^{-1}\mu$ to the general multiclass classification. By approximating the classification error probability with the multivariate Gaussian distribution, we proposed a classification technique that minimized the probability of the error. The linear and non-linear classification scheme of the proposed classifier was examined and discussed. The main further effort is in the direction of the better upper bound for the misclassification error.

Chapter 5

Conclusion

This thesis provides multiple novel techniques in the machine learning, deep learning and speech recognition.

5.0.1 Deep Gaussian Mixture Model

The concept of Deep Gaussian Mixture Model is proposed and evaluated. This model combines the advantages of deep feature extraction and GMM representation. Joint training of this model with ASGD led to a superior model, strongly competitive with DNNs on two Voice Search tasks. Extension of this model to Deep Subspace GMMs resulted in additional improvements.

5.0.2 Distributional Approximation of the Classification Accuracy

Maybe the most natural idea of classification is behind this technique. It is shown how to model the accuracy event. Then it is shown how to estimate the density of this event and using that, we find the approximation of the classification error through the training process.

Bibliography

- [1] P. Mermelstein, “Distance measures for speech recognition, psychological and instrumental,” *Pattern recognition and artificial intelligence*, vol. 116, pp. 374–388, 1976.
- [2] K. Pearson, “Liii. on lines and planes of closest fit to systems of points in space,” *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.
- [3] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.
- [4] T. M. Mitchell, “Machine learning. 1997,” *Burr Ridge, IL: McGraw Hill*, vol. 45, 1997.
- [5] A. Jordan, “On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes,” *Advances in neural information processing systems*, vol. 14, p. 841, 2002.

BIBLIOGRAPHY

- [6] S. Russell, P. Norvig, and A. Intelligence, “A modern approach,” *Artificial Intelligence. Prentice-Hall, Englewood Cliffs*, vol. 25, 1995.
- [7] R. A. Fisher, “The use of multiple measurements in taxonomic problems,” *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.
- [8] F. Rosenblatt, “The perceptron: a probabilistic model for information storage and organization in the brain,” *Psychological review*, vol. 65, no. 6, p. 386, 1958.
- [9] A. Deev, “Representation of statistics of discriminant analysis and asymptotic expansions in dimensionalities comparable with sample size,” *Reports of Academy of Sciences of the USSR*, vol. 195, no. 4, pp. 756–762, 1970.
- [10] S. Raudys and V. Pikelis, “On dimensionality, sample size, classification error, and complexity of classification algorithm in pattern recognition,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 3, pp. 242–252, 1980.
- [11] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [12] H. A. Bourlard and N. Morgan, *Connectionist speech recognition: a hybrid approach*. Springer Science & Business Media, 1994, vol. 247.
- [13] L. Rabiner, “A tutorial on hidden markov models and selected applications in speech recognition,” *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.
- [14] F. Jelinek, *Statistical methods for speech recognition*. MIT press, 1997.

BIBLIOGRAPHY

- [15] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the em algorithm,” *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.
- [16] D. Reynolds, T. Quatieri, and R. Dunn, “Speaker verification using adapted Gaussian mixture models,” *Digital Signal Processing*, vol. 10, no. 1, pp. 19–41, 2000.
- [17] A. Biem, S. Katagiri, E. McDermott, and B.-H. Juang, “An Application of Discriminative Feature Extraction to Filter-Bank Based Speech Recognition,” *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 2, pp. 96–110, 2001.
- [18] A. Biem, S. Katagiri, and B.-H. Juang, “Pattern recognition using discriminative feature extraction,” *IEEE Transactions on Signal Processing*, vol. 45, no. 2, pp. 500–504, 1997.
- [19] Y. Bengio, “Learning deep architectures for ai,” *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.
- [20] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.
- [21] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and

BIBLIOGRAPHY

- new perspectives,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 8, pp. 1798–1828, 2013.
- [22] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85–117, 2015.
- [23] F. Seide, G. Li, X. Chen, and D. Yu, “Feature engineering in context-dependent deep neural networks for conversational speech transcription,” in *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*. IEEE, 2011, pp. 24–29.
- [24] E. Variani, E. McDermott, and G. Heigold, “A gaussian mixture model layer jointly optimized with discriminative features within a deep neural network architecture,” in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015.
- [25] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean *et al.*, “On rectified linear units for speech processing,” in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3517–3521.
- [26] C. Gulcehre, K. Cho, R. Pascanu, and Y. Bengio, “Learned-norm pooling for deep feedforward and recurrent neural networks,” in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2014, pp. 530–546.

BIBLIOGRAPHY

- [27] J. S. Bridle, “Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition,” in *Neurocomputing*. Springer, 1990, pp. 227–236.
- [28] S.-i. Amari and M. A. Arbib, “Competition and cooperation in neural nets,” 1982.
- [29] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, “Deep neural networks for acoustic modeling in speech recognition,” *IEEE Signal Processing Magazine*, vol. 29, pp. 82–97, November 2012.
- [30] A. Mohamed, G. Hinton, and G. Penn, “Understanding how deep belief networks perform acoustic modelling,” in *Proc. IEEE ICASSP*, March 2012, pp. 4273–4276.
- [31] G. Heigold, “A log-linear discriminative modeling framework for speech recognition,” Ph.D. dissertation, RWTH Aachen University, 2010.
- [32] G. Heigold, H. Ney, P. Lehnen, T. Gass, and R. Schluter, “Equivalence of generative and log-linear models,” *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 5, pp. 1138–1148, 2011.
- [33] D. Yu and M. L. Seltzer, “Improved bottleneck features using pretrained deep neural networks,” in *INTERSPEECH 2011, 12th Annual Conference of the In-*

BIBLIOGRAPHY

- ternational Speech Communication Association, Florence, Italy, August 27-31, 2011*, 2011, pp. 237–240.
- [34] T. Sainath, B. Kingsbury, and B. Ramabhadran, “Auto-encoder bottleneck features using deep belief networks,” in *Proc. IEEE ICASSP*, 2012, pp. 4153–4156.
- [35] L. Deng and J. Chen, “Sequence classification using the high-level features extracted from deep neural networks,” in *Proc. IEEE ICASSP*, 2014.
- [36] K. Demuynck and F. Triefenbach, “Porting concepts from DNNs back to GMMs,” in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 356–361.
- [37] M. Paulik, “Lattice-based training of bottleneck feature extraction neural networks.” in *INTERSPEECH*, 2013, pp. 89–93.
- [38] A. van den Oord and B. Schrauwen, “Factoring variations in natural images with deep gaussian mixture models,” in *Advances in Neural Information Processing Systems*, 2014, pp. 3518–3526.
- [39] E. McDermott, “Discriminative Training for Speech Recognition,” Ph.D. dissertation, Waseda University, School of Engineering, March 1997.
- [40] M. Schuster, “On supervised learning from sequential data with applications for speech recognition,” *Doktoro disertacija, Nara Institute of Science and Technology*, 1999.

BIBLIOGRAPHY

- [41] E. McDermott, “Discriminative prototype-based methods for speech recognition,” in *Handbook for Neural Networks in Speech Processing*. Morwood, MA, USA: Artech House, 2000, pp. 159–216.
- [42] E. McDermott, T. Hazen, J. Le Roux, A. Nakamura, and S. Katagiri, “Discriminative training for large vocabulary speech recognition using Minimum Classification Error,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, no. 1, pp. 203–223, January 2007.
- [43] C. Bishop, *Neural networks for pattern recognition*. Oxford University Press, 1995.
- [44] B. Kingsbury, “Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling,” in *Proc. IEEE ICASSP*, 2009, pp. 3761–3764.
- [45] E. McDermott, G. Heigold, P. Moreno, A. Senior, and M. Bacchiani, “Asynchronous stochastic optimization for sequence training of deep neural networks: Towards big data,” in *Proc. Interspeech*, 2014.
- [46] Y. Bengio and Y. LeCun, “Scaling learning algorithms towards AI,” *Large-scale kernel machines*, vol. 34, pp. 1–41, 2007.
- [47] D. Povey, L. Burget, M. Agarwal, P. Akyazi, K. Feng, A. Ghoshal, O. Glembek, N. Goel, M. Karafiát, A. Rastrow, R. C. Rose, P. Schwarz, and S. Thomas, “Sub-

BIBLIOGRAPHY

- space gaussian mixture models for speech recognition,” in *Proc. IEEE ICASSP*. IEEE, 2010, pp. 4330–4333.
- [48] V. N. Vapnik, *The nature of statistical learning theory*, ser. Statistics for engineering and information science. New York: Springer, 2000. [Online]. Available: <http://opac.inria.fr/record=b1097848>
- [49] A. Lyapunov, *Collected works of academician AM Lyapunov*. Defense Technical Information Center, 1967.
- [50] E. Slutsky, *Über stochastische asymptoten und grenzwerte*. Amministrazione del” Metron”, 1925.
- [51] J. Shao, *Mathematical Statistics*. Springer, 2008.
- [52] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.
- [53] J. Bai and S. Shi, “Estimating high dimensional covariance matrices and its applications,” 2011.
- [54] Y. Chen, A. Wiesel, Y. C. Eldar, and A. O. Hero, “Shrinkage algorithms for mmse covariance estimation,” *Signal Processing, IEEE Transactions on*, vol. 58, no. 10, pp. 5016–5029, 2010.
- [55] A. Biem, S. Katagiri, E. McDermott, and B.-H. Juang, “An application of discriminative feature extraction to filter-bank-based speech recognition,” *Speech and Audio Processing, IEEE Transactions on*, vol. 9, no. 2, pp. 96–110, 2001.

BIBLIOGRAPHY

- [56] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, 1988.
- [57] K. Marek, D. Jennings, S. Lasch, A. Siderowf, C. Tanner, T. Simuni, C. Coffey, K. Kieburtz, E. Flagg, S. Chowdhury, W. Poewe, B. Mollenhauer, P.-E. Klinik, T. Sherer, M. Frasier, C. Meunier, A. Rudolph, C. Casaceli, J. Seibyl, S. Mendick, N. Schuff, Y. Zhang, A. Toga, K. Crawford, A. Ansbach, P. D. Blasio, M. Piovella, J. Trojanowski, L. Shaw, A. Singleton, K. Hawkins, J. Eberling, D. Brooks, D. Russell, L. Leary, S. Factor, B. Sommerfeld, P. Hogarth, E. Pighetti, K. Williams, D. Standaert, S. Guthrie, R. Hauser, H. Delgado, J. Jankovic, C. Hunter, M. Stern, B. Tran, J. Leverenz, M. Baca, S. Frank, C.-A. Thomas, I. Richard, C. Deeley, L. Rees, F. Sprenger, E. Lang, H. Shill, S. Obradov, H. Fernandez, A. Winters, D. Berg, K. Gauss, D. Galasko, D. Fontaine, Z. Mari, M. Gerstenhaber, D. Brooks, S. Malloy, P. Barone, K. Longo, T. Comery, B. Ravina, I. Grachev, K. Gallagher, M. Collins, K. L. Widnell, S. Ostrowizki, P. Fontoura, T. Ho, J. Luthman, M. van der Brug, A. D. Reith, and P. Taylor, "The parkinson progression marker initiative (ppmi)," *Progress in Neurobiology*, vol. 95, no. 4, pp. 629 – 635, 2011, biological Markers for Neurodegenerative Diseases. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0301008211001651>
- [58] S. Jain, Y. Salimpour, L. Younes, G. Smith, Z. Mari, V. Sossi, and A. Rahmim,

BIBLIOGRAPHY

- “Application of pattern recognition framework for quantification of parkinson’s disease in dat spect imaging,” in *IEEE Nucl. Sci. Symp.*, 2014.
- [59] M. Vaillant and J. Glaunes, “Surface matching via currents,” *Proceedings of Information Processing in Medical Imaging*, pp. 381–392, 2005.
- [60] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [61] V. N. Vapnik and V. Vapnik, *Statistical learning theory*. Wiley New York, 1998, vol. 1.
- [62] T. Li, S. Zhu, and M. Ogihara, “Using discriminant analysis for multi-class classification: an experimental investigation,” *Knowledge and information systems*, vol. 10, no. 4, pp. 453–472, 2006.
- [63] L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, L. D. Jackel, Y. LeCun, U. A. Muller, E. Sackinger, P. Simard *et al.*, “Comparison of classifier methods: a case study in handwritten digit recognition,” in *International Conference on Pattern Recognition*. IEEE Computer Society Press, 1994, pp. 77–77.
- [64] T. Hastie, R. Tibshirani *et al.*, “Classification by pairwise coupling,” *The annals of statistics*, vol. 26, no. 2, pp. 451–471, 1998.

BIBLIOGRAPHY

- [65] J. C. Platt, N. Cristianini, and J. Shawe-Taylor, “Large margin dags for multi-class classification.” in *nips*, vol. 12, 1999, pp. 547–553.
- [66] T. G. Dietterich and G. Bakiri, “Solving multiclass learning problems via error-correcting output codes,” *arXiv preprint cs/9501101*, 1995.
- [67] E. L. Allwein, R. E. Schapire, and Y. Singer, “Reducing multiclass to binary: A unifying approach for margin classifiers,” *The Journal of Machine Learning Research*, vol. 1, pp. 113–141, 2001.
- [68] J. Weston and C. Watkins, “Multi-class support vector machines,” Citeseer, Tech. Rep., 1998.
- [69] R. A. Johnson, D. W. Wichern *et al.*, *Applied multivariate statistical analysis*. Prentice hall Englewood Cliffs, NJ, 1992, vol. 4.
- [70] K. Fukunaga, *Introduction to statistical pattern recognition*. Academic press, 2013.
- [71] A. Stuhlsatz, J. Lippel, and T. Zielke, “Feature extraction with deep neural networks by a generalized discriminant analysis,” *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 23, no. 4, pp. 596–608, 2012.
- [72] C. E. Bonferroni, *Teoria statistica delle classi e calcolo delle probabilita*. Libreria internazionale Seeber, 1936.

BIBLIOGRAPHY

- [73] D. Slepian, “The one-sided barrier problem for gaussian noise,” *Bell System Technical Journal*, vol. 41, no. 2, pp. 463–501, 1962.

Vita



Ehsan Variani started his PhD in Johns Hopkins University, department of Electrical and Computer Engineering (ECE), in August 2009. He received Master of Science in ECE in 2011. Before Phd study, Ehsan got his B.Sc in Electrical Engineering and Pure Mathematics from Shairf University of Technology, Tehran, Iran.

Ehsan's research interest are theory and applications of Machine Learning and Information Theorey in Speech and Image recognition. During his PhD study, He has spent three summers in Google (two summers) and M*Modal working on state of the art speech recognition. He also has several publications in IEEE conferences like ICASSP, Interspeech and ISIT. He has been awarded the best paper award of ICASSP 2015. He is also the receipt of the Best paper award in the SLTC (Speech

VITA

and Language Technology community) track of IEEE ICASSP 2015 and the SLT travel grant.