5-2010

Reduced hyperBF networks : practical optimization, regularization, and applications in bioinformatics.

Rami Nezar Mahdi 1982-
University of Louisville

REDUCED HyperBF NETWORKS: PRACTICAL OPTIMIZATION, REGULARIZATION, AND APPLICATIONS IN BIOINFORMATICS

By

Rami Nezar Mahdi
B.S., Arab American University at Jenin, 2004
M. S., University of Louisville, 2007

A Dissertation
Submitted to the Faculty of the
Speed School of Engineering of the University of Louisville
in Partial fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

Department of Computer Engineering & Computer Science
University of Louisville
Louisville, Kentucky USA

May 2010

REDUCED HyperBF NETWORKS: PRACTICAL OPTIMIZATION, REGULARIZATION, AND APPLICATIONS IN BIOINFORMATICS

By

Rami Nezar Mahdi
B.S., Arab American University at Jenin, 2004
M.S., University of Louisville, 2007

A Dissertation Approved on

March 23, 2010

By the following Dissertation Committee:

Dissertation Director (Eric Rouchka)

Mehmed Kantardzic

Olfa Nasraoui

Xiang Zhang

Jacek Zurada

DEDICATION

This dissertation is dedicated to my parents for their unconditional love and commitment to education

and

all the unfortunate people of the world with a hope that science advances might make the world a better place to live in.

ACKNOWLEDGEMENT

I have never felt less eloquent than the moment I started writing this acknowledgement section of my dissertation. No words could express my appreciation and gratitude to everyone who helped me accomplish my goal and be in the position I am now.

I am honored to dedicate my dissertation to my family in Gaza who supported me throughout this journey. I would have not been who I am now without their faith and confidence in me.

I would like to express my deepest appreciation to my advisor Dr. Eric Rouchka for his supervision, support, guidance, and patience. Dr. Rouchka has played a vital role in the development and completion of my Ph.D. research. Working with him taught me much about bioinformatics research and I truly feel fortunate to be his student.

I am thankful to Dr. Mehmed Kantardzic, Dr. Olfa Nasraoui, Dr. Xiang Zhang, and Dr. Jacek Zurada for voluntarily accepting to serve on my Ph. D. committee board. Their suggestions and feedback were of great help. I am also grateful for Dr. Nasraoui for giving me the opportunity to work for her Knowledge Discovery & Web Mining research lab during summer 2009 when I was temporarily out of financial support. I would like also to thank Dr. Hichem Frigui for being my advisor at the master's level. Working with Dr Frigui provided me with a rich experience in research. He was a great advisor and still is a good friend.

I would like also to express my appreciation to all my professors at the University of Louisville, at the Arab American University of Jenin, and to all my instructors in all levels of education. I truly feel blessed for having had honest and devoted educators in all levels.

During my stay in Louisville I have known many people in the community who took me in as a part of their family; I am honored to have met them. Special thanks go to Dr. Ibrahim Imam and his family. Dr. Imam was like a father to me who always provided help and guidance. Many thanks go to Dr. Adnan Masri and his family too. And of course I have to thank Dr. Farouq Mosa who helped me come to Louisville in the first place and introduced me to such a wonderful community.

During my life, I have been a student, an acquaintance, a friend, and a colleague to very special and extraordinary people who helped me shape my character, beliefs and outlook on life. Sincere thanks go to Dr. Walid Deeb for his guidance and support. Working with Dr. Deeb at the Young Scientists Club was one of the most mind opening and enriching experiences of my life, and I am indebted to him for the teaching skills that I acquired under his tutelage. Special thanks go to Zuhair Wadi and Naser Al-Sumiri for their friendship, support and faith in me since I was in middle school.

Last and not least, I would like to thank all my dear friends who were always there during good times to share joy and hard times to listen, support, and share concerns: Faris Hindi, Yehya Senousy, Stilios Stelianos, Majdi Odeh, Najeeb Samouh, Walid Missaoui, Anis Hamdi, Aleksey Fadeev, Lujin "Hazin", Rhonda Amer, Pat Geier, Linda George and her family, Omar Ayyash and his family, Dr. Zahi Masri, Abed Al-Hafiz Rabi, Mohammed Al-Samna, Loay Elbasyouni, Asaad Said, Georgios Soteriou, Elom Akabua,

and all others whom I might have unintentionally forgotten. Also, lots of sincere thanks go to my friend Marette Irwin for caring about me and for voluntarily and patiently reviewing some of my written documents.

ABSTRACT

REDUCED HyperBF NETWORKS: PRACTICAL OPTIMIZATION, REGULARIZATION, AND APPLICATIONS IN BIOINFORMATICS

Rami Nezar Mahdi

March 23, 2010

A hyper basis function network (HyperBF) is a generalized radial basis function network (RBF) where the activation function is a radial function of a weighted distance. The local weighting of the distance accounts for the variation in local scaling and discriminative power along each feature. Such generalization makes HyperBF networks capable of interpolating decision functions with high accuracy. However, such complexity makes HyperBF networks susceptible to overfitting. Moreover, training a HyperBF network demands weights, centers and local scaling factors to be optimized simultaneously. In the case of a relatively large dataset with a large network structure, such optimization becomes computationally challenging.

In this work, a new regularization method that performs soft local dimension reduction and weight decay is presented. The regularized HyperBF (Reduced HyperBF) network is shown to provide classification accuracy comparable to a Support Vector Machines (SVM) while requiring a significantly smaller network structure. Furthermore, the soft local dimension reduction is shown to be informative for ranking features based on their localized discriminative power.

In addition, a practical training approach for constructing HyperBF networks is presented. This approach uses hierarchal clustering to initialize neurons followed by a gradient optimization using a scaled Rprop algorithm with a localized partial backtracking step (iSRprop). Experimental results on a number of datasets show a faster and smoother convergence than the regular Rprop algorithm.

The proposed Reduced HyperBF network is applied to two problems in bioinformatics. The first is the detection of transcription start sites (TSS) in human DNA. A novel method for improving the accuracy of TSS recognition for recently published methods is proposed. This method incorporates a new metric feature based on oligonucleotide positional frequencies.

The second application is the accurate classification of microarray samples. A new feature selection algorithm based on a Reduced HyperBF network is proposed. The method is applied to two microarray datasets and is shown to select a minimal subset of features with high discriminative information. The algorithm is compared to two widely used methods and is shown to provide competitive results.

In both applications, the final Reduced HyperBF network is used for higher level analysis. Significant neurons can indicate subpopulations, while local active features provide insight into the characteristics of the subpopulation in specific and the whole class in general.

TABLE OF CONTENTS

PAGE

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The past two decades have witnessed a dramatic increase in the number and quality of applications that utilize machine learning and applied statistical methods. Pattern recognition, modeling, prediction and analysis tools have found their valuable application in various field such engineering, natural science, medicine, marketing and many others. In part, this trend was stimulated by the availability of cheap, reliable and relatively fast computational resources. Furthermore, new supervised learning tools such as Support Vector Machines, learning with kernels, boosting, and regularized models have yielded unprecedented predictive accuracy in various applications. Though machine learning tools still fall short in competing with human intelligence in recognition and analysis, some of these tools have come very close to human performance in specific applications such as the recognition of handwritten digits and characters. Moreover, some of these tools can outperform humans in learning tasks that demand the analysis of large data quantities. For example, a human is incapable of analyzing the expression of tens of thousands of genes for hundreds of patients in a reasonable timeframe. In contrast, supervised or unsupervised learning tools can handle this task efficiently.

Nonetheless, with the exception of sparse linear regression models such as LASSO, most supervised learning approaches do not facilitate higher level analysis. Moreover, sparse linear models provide limited higher level information. With the increasing

availability of data in various fields in addition to increased computation power, it is anticipated that interpretable models will increase in their importance within the field of statistical and machine leaning. Tools such as clustering analysis, regularization, and minimal radial or hyper basis function networks (RBF/HyperBF) possess the potential to provide evidence for causation analysis and to uncover subpopulation information. Such information is of great value in the areas of bioinformatics and artificial intelligence. For example, in bioinformatics, it can provide evidence for biomarkers and uncover information about disease subtypes. In artificial intelligence, such models are needed for knowledge extraction and decision justification.

1. Machine Learning

According to Garbonell, Michalski, and Mitchell [1], "Learning processes include the acquisition of new declarative knowledge, the development of motor and cognitive skills through instruction or practice, the organization of new knowledge into general, effective representations, and the discovery of new facts and theories through observation and experimentation" (p 3). These processes are usually fed with large datasets of high dimensionality, and hence, the learning process can only be performed by a processing machine. In this dissertation, we are only interested in learning processes that utilize observed data for the purpose of knowledge inference.

In machine learning and statistics, phenomena are usually associated with observations $\mathcal{D}$:$\{(x_1, y_1), (x_2, y_2)\ldots. (x_n, y_n)\}$ that are assumed to be generated based on a probability distribution $\mathbb{V}$ called the generator distribution. x_i usually refers to a description of the observation sample while y_i refers to a category or a class to which

sample x_i belongs. For example, in the case of developing a system to recognize the type of trees in a forest, x can be the visual description of every tree while y refers to the type of the tree.

According to Vapnik, who is one of the most influential figures in the development of statistical learning theory, most learning problems can fall into three categories [2]:

1. Classification: The observed data $\mathcal{D}$ is used to learn a machine capable of classifying a new sample x_i into the right category y_i from Y. In the simplest case, Y contains only two categories: (-1, 1).
2. Regression Estimation: Use the observed data $\mathcal{D}$ to approximate a function to map a new sample x_i to a real value $y_i \epsilon \mathbb{R}$.
3. Density Estimation: The observed data $\mathcal{D}$ is used to recover the generator distribution $\mathbb{V}$ or find an approximation of $\mathbb{V}$ or its characteristics.

The above three categories are closely related. Classification is a special case of the regression estimation where a function is approximated to map samples to one of two real values only (i.e. -1 or 1). On the other hand, if the true generator distribution $\mathbb{V}$ can be recovered, a more accurate regression or classification can be approximated as will be explained later.

Although the three types of statistical learning described above are the most prevalent in machine learning literature, one can still define other types that might serve a specific purpose for some applications. Some of these are:

1. Feature Analysis and Ranking: Learn the features that characterize a class and give the most discriminative information. Such analysis is valuable in medical research such as finding the genes that cause a certain disease.
2. Clustering Analysis: The goal is to discover potential subtypes of a certain class or population. For example, Leukemia might be triggered by different subsets of genes and the goal is to discover these subtypes through the analysis of gene expression data for a few hundred patients.

Note that subpopulation analysis and density estimation are related. Density estimation is usually performed with the assumption of a mixture of models or mixtures of Gaussians. Traditionally, clustering analysis algorithms have been used for subtype analysis.

2. Supervised Learning

Classification and regression learning are usually referred to as supervised learning. The goal is to train a model F: $X \rightarrow Y$ that is capable of guessing with arbitrary accuracy the desired output y_i for a new sample x_i. The learning is referred to as supervised or informed because the training process utilizes a set of labeled observations referred to as training data: $\mathcal{D}$:$((x_1, y_1), (x_2, y_2) \ldots . (x_N, y_N))$.

In classification problems, the quality of the trained classifier is measured by the probability of error or misclassifying new samples as follows:

$$L(F) = P(F(x_i) \neq y_i) \tag{1}$$

Therefore, F should assign a new sample x_i to the class with the maximum *a posterior* probability:

$$F(x_i) = arg \max_{y_j} P(Y = y_j | X = x_i) \tag{2}$$

Or equivalently

$$F(x_i) = arg \max_{y_j} P(X = x_i | Y = y_j) \times P(y_j) \tag{3}$$

In (3), $P(X = x_i | Y = y_j)$ is the likelihood of the randomly selected sample X being x_i knowing that the class is y_j, while $P(y_j)$ is the prior probability of class y_j. Recovering these values or distributions is equivalent to recovering the probability density of the generator model $\mathbb{V}$. Therefore, if the density is provided or recovered accurately, the decision rule in (3) becomes Bayes optimal. In other words, it will be the best possible guess someone can make to classify a new sample x_i. However, the density is unlikely to be provided or known *a priori*. Recovering density with high accuracy is not a trivial problem. This is why density based classification methods are not widely used particularly when compared to other methods such as Support Vector Machine (SVM) and neural networks.

3. Risk and Loss Functions

An alternative and more commonly used measure for the quality of the classifier is the risk. An optimal classifier is the classifier that minimizes the risk of misclassification over the space of samples: $X \times Y$ on the generator model $\mathbb{V}$:

$$\mathbb{R} = \int_{X \times Y} L(x, y, F(x)) d(x, y) \tag{4}$$

In (4), the function L is the cost or the loss resulting from misclassifying sample x. Different loss functions have been used in the literature. The most used ones are the square error, the Laplace error and the Hinge error.

$$\text{Square Error: } L(x, y, F(x)) = (y - F(x))^2 \tag{5}$$

$$\text{Laplace Error: } L(x, y, F(x)) = \|y - F(x)\| \tag{6}$$

$$\text{Hinge Error: } L(x, y, F(x)) = \begin{cases} 1 \ if \ y \neq F(x) \\ 0 \ otherwise \end{cases} \tag{7}$$

Of these loss functions, the squared error is the most commonly used, particularly in regression problems [3]. Training to minimize the squared error results in a minimal variance of the error [3]. Nonetheless, the Hinge error had gain popularity after the introduction of Support Vector Machine and maximum margin classifiers.

Finding a machine F^* that minimizes the risk in (4) to the lowest possible risk R^* also demands knowledge about the generator model of $X \times Y$. Even if such knowledge exists, minimizing the continuous integral is a non-trivial problem.

4. Empirical Risk: Learning from Examples

Since finding a model that minimizes the true risk is not feasible, one can learn an approximate model that minimizes the risk over a set of available and limited observations $\mathcal{D}$:$((x_1, y_1), (x_2, y_2) \ldots (x_N, y_N))$. In such a case, the learning task becomes to find a function $\hat{F}$ that minimizes the training error:

$$R(\mathcal{D}, F) = \frac{1}{N}\sum_{i=1}^{N} L\big(x_i, y_i, F(x_i)\big) \tag{8}$$

In the case of using the squared error as a loss function, the objective function becomes the mean squared error (MSE):

$$MSE = \frac{1}{N}\sum_{i=1}^{N} \big(y_i - F(x_i)\big)^2 \tag{9}$$

Minimizing the MSE in (9) is called the minimization of the empirical risk or learning from examples. Learning from examples is motivated by the fact that the observations in $\mathcal{D}$ are very likely to be generated by the true generator model $\mathbb{V}$. Therefore, a function $\hat{F}$ that minimizes the MSE should provide an approximation to the optimal function F^*.

A different justification for learning from examples was provided by Poggio and Girosi in [4;5]. They argued that observations in the real world are characterized by redundancy. This means that unseen observations are very likely to have similar characteristics to the previously seen or available observations. Therefore, a model that fits the training data is also likely to fit yet unseen observations to a certain degree of accuracy.

5. Overfitting vs. Generalization

The goal of learning from examples is to find a model F that generalizes well. In other words, F should be capable of classifying new samples with high accuracy or with a small risk $R(F)$. Note that the best possible F is bounded by the optimal risk R^* which is

not necessarily zero. The optimal risk is not likely to be zero in most real problems because different observations of the same x values might belong to different classes y.

Overfitting is a phenomenon that occurs when the trained model fits the training data to high accuracy but fails to generalize well for testing. Complex models demand more training samples to be approximated accurately. However, in most learning problems, training data is always of limited quantity. Furthermore, for any training data, there is always an unlimited number of solutions that minimize the same empirical loss.

6. Regularization and Complexity Control

A widely used solution to make the learning problem well-posed and make the trained model generalize well for testing is the use of regularization [6-8] or complexity control which tries to find the simplest model that fits the training data. This is accomplished by adding a penalty term to the empirical loss objective resulting in a functional or regularized objective function to be minimized:

$$E_{reg} = \frac{1}{2}\sum_{i=1}^{N} L\big(x_i, y_i, F(x_i)\big) + \frac{1}{2}\lambda\phi[F] \tag{10}$$

In (10), $\phi[F]$ is a measure of the complexity of the function F, and λ is a positive regularization parameter to be determined by cross-validation. Optimal training searches for an optimal tradeoff between fitting the training data and minimizing the complexity of the solution. In Chapter II, different methods for measuring the complexity of a function will be considered.

7. Hypothesis Space & Example Classifiers

The space that contains all possible functions F is usually referred to as the hypothesis space. In principle, the hypothesis F can be any function. However, learning typically involves narrowing the hypothesis space into a subspace of a specific type of functions. Training a model is a search in the hypothesis space for the ideal function that fits predefined conditions such low empirical error and low complexity. The following subsections (7.1-7.4) provide a brief description of example hypothesis spaces or classifiers that can be searched in the training process.

7.1. Linear Regression for Classification

Linear regression [3] is one of the oldest and simplest methods for finding a function that maps the samples from the input space $X \subset \mathbb{R}^Z$ to their desired output $Y \in \mathbb{R}$. A linear regression function has the form:

$$F(x) = Wx + b \tag{11}$$

where $W \in \mathbb{R}^Z$ and $b \in \mathbb{R}$.

Training a linear regression machine involves estimating the matrix W and the constant b that minimize the loss over the training data such as the empirical least square error.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - Wx_i - b)^2 \tag{12}$$

In two classes' classification problems, y_i is either +1 or -1 depending on its true class. Once the model is trained, a new sample x is classified as:

$$y(x) = sign(F(x)) \tag{13}$$

7.2. Radial Basis Function Neural Network (RBF)

Different nonlinear variants have emerged from the simple linear regression. Most of these methods transform the samples from their original representation space X to a new space. Afterward, a linear regression is used to predict the desired output. One of the popular non-linear regression methods is the RBF network where the output function has the form:

$$F(x_i) = \sum_{j=1}^{J} w_j h_j(\| \mu_j - x_i \|) + b \tag{14}$$

In (14), μ_j is the center of neuron j while w_j is the weight associated with the output of neuron j. $h_j(\| \mu_j - x_i \|)$ is the output of the neuron j for a given data sample x_i which is a radial function of the distance between the neuron center μ_j and the sample x_i.

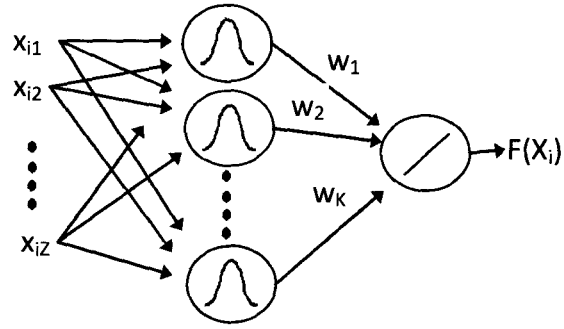

Figure 1: Structure of an RBF Network.

The most widely used radial function is the Gaussian function:

$$h_j(\| \mu_j - x_i \|) = e^{\frac{\|\mu_j - x_i\|^2}{\sigma^2}} \tag{15}$$

In (15), σ is a scaling parameter often referred to as the width of the function.

7.3. Support Vector Machine Networks (SVM)

SVM is probably the most extensively studied, mature and widely used classification tool. Originally, SVM was introduced as a linear classifier with low complexity [9]. Later, it was generalized to the non-linear case using the kernel trick [9;10]. In the general non-linear case, the output of an SVM network for a new sample x is computed as:

$$F(x) = \sum_{i=1}^{N} w_i K(x_i, x) + b \tag{16}$$

In (16), K is a similarity function between the new sample x and a sample from the training data x_i, w_i is a real valued weight associated with the vector x_i, while b is a constant. The function K is typically referred to as the kernel. A commonly used kernel is the radial basis function:

$$K(x_i, x) = e^{\frac{-\|x - x_i\|^2}{\sigma}} \tag{17}$$

where σ is a positive scaling parameter that is selected to minimize the cross-validation error, and in some literature it is referred to as the width of the kernel.

An optimal SVM network is the one that maximizes the separation margin between two classes, while at the same time minimizing the classification error. Such training is accomplished by solving the following optimization problem:

$$\min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i \tag{18}$$

Subject to

$$y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i$$

In (18), ξ_i are slack variables to account for tolerance to non-separable classes and C is a regularization parameter that controls the tradeoff between the maximized margin and the training error. An optimal C can be selected through cross-validation. The generalization of the SVM machine is also dependent on the selection of the kernel.

Existing software tools such as Libsvm [11] implement sophisticated methods for training and parameter selection of SVM networks. In this dissertation, SVM is used as a benchmark. All the reported SVM experiments are performed using Libsvm software.

7.4. HyperBF Networks

A generalization of the Gaussian radial basis function in (15) is the hyper basis function (HyperBF) introduced by Poggio and Girosi in [4;5]. Unlike (15) which uses the Euclidean distance scaled by a single scaling factor σ, a hyper basis function uses a Mahalanobis-like distance as follows:

$$h_j(x_i) = e^{-0.5\times(x_i-\mu_j)^T R_j (x_i-\mu_j)} \tag{19}$$

In (19), R_j is a positive definite squared matrix. In the general case, the role of R_j is to make the similarity to the neuron invariant to local scaling and orientation of the data.

A key advantage of using HyperBF networks is the small network structure compared to SVM networks (number of neurons vs. number of support vectors). This feature makes HyperBF networks competitive solutions for applications where fast classification is needed with limited computation resources.

In spite of being a generalized RBF network, the HyperBF network is still one of the least studied and used methods in machine learning applications, particularly when

compared to the case of a single scaling factor RBF which is used heavily in RBF neural networks and support vector machine networks. The main reasons are:

1. The high degree of freedom of HyperBF networks leads to overfitting and poor generalization.
2. The training objective function is not convex and hence training will tend to find a locally optimal solution instead of the globally optimal solution.
3. Training a HyperBF network is a challenging optimization problem that demands a scalable optimization method to estimate the large number of parameters.

8. Contributions of this Dissertation

In this dissertation, a new regularization method for training HyperBF networks that performs soft local dimension reduction in addition to weight decay is presented. The regularization aims at explicitly minimizing the complexity of the network while fitting the training data. The regularized HyperBF network is shown to provide classification accuracy competitive to SVM while using a significantly smaller network. This smaller network structure is the motivation for its name: *Reduced HyperBF Network.*

Furthermore, a practical training approach to construct HyperBF networks is proposed. A scaled version of the Resilient Propagation algorithm (Rprop) [12] is introduced and used to perform sign based gradient optimization. The proposed Scaled-Rprop is shown to provide faster convergence while causing less oscillation than the regular Rprop.

As a result of its small network structure (fewer neurons and fewer locally active features), the Reduced HyperBF network is shown to be an effective tool for higher level analysis. Active neurons are argued to be indicators of subtypes, while the locally active features provide insight into the unique characteristics of each subtype and the whole class in general.

In addition, a new feature selection based a Reduced HyperBF network is proposed. On microarray datasets, the proposed feature selection algorithm is shown to be an effective tool for feature subset selection that provides very competitive results compared to existing methods.

9. Outline of the Dissertation

In Chapter II, more literature about RBF and HyperBF networks is presented. Existing methods for training RBF and HyperBF networks are described. The chapter also details an overview of regularization theory, its explanation, and some commonly used regularization techniques.

Chapter III provides a detailed description of the proposed regularization and training algorithm for HyperBF networks. Using the Reduced HyperBF network for feature selection is deferred to Chapter V.

Experiments on seven real world datasets are reported in Chapter IV. The proposed optimization algorithm (iSRprop) is compared to the improved Rprop algorithm (iRprop+) and the VSBP algorithm. Furthermore, the classification accuracy of the proposed regularized network (Reduced HyperBF) is compared to the regular HyperBF network and the Support Vector Machine network.

Chapter V presents a case study of using the Reduced HyperBF network in the detection of Transcription Start Sites (TSS) in human DNA using a new metric feature to represent promoter DNA segments. The resulting model (HyperBF-TSS) is compared to other TSS detection tools on the same dataset. The compact network structure is utilized to extract a simple description of the structure of TSS regions.

Chapter VI presents a case study of using Reduced HyperBF networks for feature selection in microarray analysis. A new algorithm (Reduced HyperBF-RFE) is described. The chapter also reviews literature on existing methods for feature selection. Experimental results of the new algorithm on two microarray datasets are reported and compared to two other methods.

Finally, Chapter VII outlines a brief summary and discussion of the contributions presented in addition to remaining challenges and future research directions that can expand on this work.

CHAPTER II

BACKGROUND

1. Radial Basis Function Networks

Radial Basis Function (RBF) networks are a special type of feed-forward artificial neural networks introduced by Broomhead and Lowe (1988) [13]. An RBF network uses a single hidden layer of neurons with RBFs as activation functions (Figure 1). The RBF model is motivated by the locally tuned response observed in biological neurons of the central nervous system. Such a network topology has been shown to be capable of interpolating complex functions. RBF networks have been successfully used in applications of function approximation [14-16], classification and pattern recognition [16;17], and dynamical modeling and control [18]. Figure 1 shows a typical two class RBF network. The output function of such a network is computed as:

$$F(x_i) = \sum_{j=1}^{J} w_j h_j(\| \mu_j - x_i \|) + b \tag{20}$$

In **(20)**, b is a constant, μ_j is the center of neuron j while w_j is the weight associated with the output of neuron j. $h_j(\| \mu_j - x_i \|)$ is the output of the neuron for a given data sample x_i which is a radial function of the distance between the neuron center μ_j and the sample x_i.

Different types of radial functions have been successfully used in literature [19-21] as activation functions. Some of these are:

$$h(r) = e^{\frac{-r^2}{\sigma^2}} \tag{21}$$

$$h(r) = (r/\sigma^2)^2 \log(r/\sigma^2) \tag{22}$$

$$h(r) = (r^2 + \sigma^2)^{\frac{1}{2}} \tag{23}$$

$$h(r) = (r^2 + \sigma^2)^{-\frac{1}{2}} \tag{24}$$

In equations (21), (22), (23) and (24), σ is a scaling parameter often referred to as the width of the function. The most commonly used radial basis function is the Gaussian function (equation (21)). In this case, the output of the neuron peaks to one when the given sample is at a zero distance from the center of the neuron and keeps decreasing toward zero as the given sample goes further from the neuron, thus acting as a similarity measure or a membership function.

2. Hyper Basis Function Networks (HyperBF)

A generalization from the Gaussian radial basis function in (21) is the hyper basis function introduced by Poggio and Girosi in [4;5]. Unlike (15) which uses the Euclidean distance scaled by a single scaling factor σ, the hyper basis function uses a Mahalanobis-like distance as follows:

$$h_j(x_i) = e^{-0.5\times(x_i-\mu_j)^T R_j (x_i-\mu_j)} \tag{25}$$

In (25), μ_j is the center of the neuron, while R_j is a positive definite squared matrix. In the general case, the role of R_j is to make the similarity to the neuron independent of any local scaling or orientation. However, as described in [22] and depending on the application and the dataset, R_j can be constrained to one of the following:

1. All neurons have spherical shape of the same size: $R_j = \left(\frac{1}{\sigma^2}\right) Id \quad \forall j$ where Id is the identity matrix.
2. All neurons have a spherical shape but different size: $R_j = \left(\frac{1}{\sigma_j{}^2}\right) Id \quad \forall j.$
3. Every neuron has an elliptical shape with a varying size but with restricted orientation aligned with the original input coordinates: $R_j = diag\left(\frac{1}{\sigma_{j1}^2}, \frac{1}{\sigma_{j2}^2} \ldots \ldots \frac{1}{\sigma_{jz}^2}\right).$
4. Every neuron has elliptical shape with a varying size and orientation: R_j is a positive definite square matrix that is not diagonal.

Though case 4 is the most general and can account for the local correlation between the different dimensions, it is computationally expensive to estimate R_j for high dimensionality and can lead to severe overfitting due to the high degrees of freedom of the model. On the other hand, case 3 makes a good tradeoff between extreme generality given by case 4 and the over simplification given by cases 1 and 2. In case 3, the coefficients can be interpreted as a local weighting method of the dimensions or as scaling factors that ensure the solution to be invariant to the local scaling of the dimensions. Note that the scaling of the local distance can also be applied to radial basis functions other than the Gaussian function.

3. RBF/HyperBF NN Training

Usually training RBF networks for a two class classification problem involves estimating the hidden layer neurons' parameters and optimal weights so that the network gives samples from two classes distinctively different scores (i.e. -1 and 1). In the case of the squared error loss function, the problem becomes to find all the parameters that result in the least squared error over the training data:

$$E = \frac{1}{2}\sum_{i=1}^{N}\left(t_i - f(x_i)\right)^2 \tag{26}$$

In (26), $f(x_i)$ is the output score of the network for the given sample x_i while t_i is the desired output which would be -1 or 1 depending on x_i.

An extension for the multiclass classification case (K classes) can be achieved by adding a separate output node for every class with its own set of connecting weights to the same hidden layer (Figure 2).

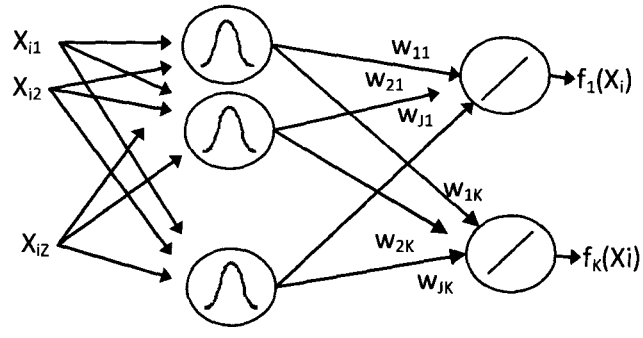

Figure 2: Structure of an RBF Network for a K-Class Network.

In the ideal case, for a given sample x_i, each output node k should give a score f_{ik} that is as close as possible to a desired output t_{ik} (i.e. t_{ik}=1 if $x_i \in k^{th}$ class and 0 otherwise). Thus the objective function becomes:

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{K}(t_{ik} - f_{ik})^2 \quad (27)$$

This approach is similar to the one vs. all classification scheme. A one vs. one topology could alternatively be implemented. In the later case, a separate network is needed for every pair of classes and at the end a probability modeling of the outputs of these networks is needed to compute the posterior of every class for a given sample [23;24].

A large number of methods have already been developed to train RBF networks. Most training methods can be categorized as dynamic or static. In static learning, the structure of the network is determined *a priori* followed by parameter estimation. In contrast, dynamic learning uses an incremental approach where neurons are added or deleted as training and parameter estimation proceeds [15;25].

In static training, most training algorithms perform optimization in two phases where the neurons centers are chosen first using data summarization methods. Once neurons are selected, they are fixed and the weights are estimated in a second phase using gradient descent or pseudo-inverse methods [26-30].

Different data summarization and clustering methods have been used to initialize the centers of the neurons. These methods include: k-means clustering [28;31], fuzzy partitioning of the input space [32], fuzzy c-means [27;29], one-pass clustering algorithm APC-III [30], decision trees [26] and vector quantization [22].

Nonetheless, there is no consensus in the literature on the optimal clustering algorithm to initialize RBF networks for optimal classification results. Furthermore, one obvious problem with the two phase learning of RBF networks is that neuron selection is

not performed to maximize the distinction between the two classes. While the weights are chosen to minimize the error after neurons are fixed, neurons are selected without any knowledge about the weights' values. Therefore, the resulting networks perform badly in classification tasks [33].

Training with moving centers and adaptable shapes of the neurons was first described by Poggio and Girosi [4;5]. A network that uses neurons with variable shapes and sizes was termed a HyperBF network. All parameters in a HyperBF network are optimized to minimize training error. Though a HyperBF network still needs to be initialized by a clustering algorithm, the final solution is much less sensitive to the choice of clustering algorithm.

A three learning phase approach for constructing a HyperBF network was described and evaluated in [22]. Regular two phase methods were used to initialize the network. Then, in the third phase, centers, scaling factors and weights are estimated simultaneously by gradient descent and back propagation so that the network results in the least squared error:

$$\Delta w_{jk} = \eta \sum_{i=1}^{N} h_j(x_i)\left(t_{ik} - f_k(x_i)\right) \tag{28}$$

$$\Delta \mu_{jz} = \eta \sum_{i=1}^{N} h_j(x_i) \frac{x_{iz} - \mu_{jz}}{\sigma_{jz}^2} \sum_{k=1}^{K} w_{jk}\left(t_{ik} - f_k(x_i)\right) \tag{29}$$

$$\Delta \sigma_{jz} = \eta \sum_{i=1}^{N} h_j(x_i) \frac{\left(x_{iz} - \mu_{jz}\right)^2}{\sigma_{jz}^3} \sum_{k=1}^{K} w_{jk}\left(t_{ik} - f_k(x_i)\right) \tag{30}$$

The authors used a back propagation training algorithm called BPVS [34] that uses a single variable learning factor for all parameters which is estimated adaptively. However,

such a single learning factor is not expected to be appropriate in the case of a large number of parameters especially when those parameters belong to three different categories: means, scaling factors and weights. Those different parameters are scaled differently and hence may demand separate learning rates.

In this work, a multiple step size gradient algorithm based on the Resilient Propagation algorithm (Rprop) is presented. The Rprop is further improved by taking into consideration the effect of the changes made to the variables.

Rprop is a first order method for gradient optimization. It uses a separate step size for every variable. Rprop is a sign based method that only uses the sign of the derivative to decide the direction of the change while ignoring the magnitude. Rprop has become very popular due to its efficiency and ease of implementation. Furthermore, different variants have emerged such as iRprop+ [35], JRprop[36], and GRprop[37].

4. Regularization

The generalized formulation of the HyperBF network gives it a high capacity to fit training data with high precision. However, from statistical learning theory, it is known that such complexity comes with a higher risk of overfitting and poor generalization [38-41]. Estimating a large number of parameters demands a large training dataset which is not the case in most real world applications. As a result, the trained model overfits the training data and fails to classify new data samples correctly [6]. A widely used solution to make the problem well-posed and make the network generalize well for testing is the use of regularization [6-8]. To regularize a trained model, a penalty term is added to the objective function that penalizes the complexity of the trained model. In the case of

squared error loss function, the resulting objective is referred to as the regularized least square and it has the following form:

$$E_{reg} = \frac{1}{2}\sum_{i=1}^{N}\left(t_i - f(x_i)\right)^2 + \frac{1}{2}\lambda\phi[f] \tag{31}$$

In (30), $\phi[f]$ is a measure of the complexity of the function f, and λ is a positive regularization parameter to be determined by cross-validation. In some literature, $\phi[f]$ is referred to as a stabilizer [4;5].

4.1. Bayesian Interpretation

The most popular interpretation of minimizing the regularized objective function in (30) is that it results in a model with a maximum *a posteriori* probability (MPA). The *a posteriori* probability of a model F is proportional to the probability of observing the data given this model multiplied by the probability of the model itself.

$$P(F|\mathcal{D}) = P(\mathcal{D}|F) \times P(F) \tag{32}$$

The optimal model F^* is the one that has the maximum *a posteriori* probability:

$$F^* = arg\ \max_F\ P(\mathcal{D}|F) \times P(F)\,, \tag{33}$$

or alternatively

$$F^* = arg\ \max_F log\big(P(\mathcal{D}|F) \times P(F)\big) \tag{34}$$

It is intuitive to assume that the probability of observing the data $\mathcal{D}$ given the model F is proportional to how well the model F fits the data and hence $P(\mathcal{D}|F)$ can be approximated as:

$$P(\mathcal{D}|F) \approx e^{\frac{-\Psi}{2}\sum_{i=1}^{N}(t_i - f(x_i))^2} \tag{35}$$

In (35), Ψ is constant. In order to estimate the prior $P(F)$ over all possible functions, *a priori* knowledge about the problem and the likely and unlikely solutions is needed. If such knowledge is not available, a best effort guess can be made. According to Occam's Razor learning principle [40], simple solutions are the most likely solutions. Furthermore, from our experience as human observers, decision rules to distinguish between classes are typically simple [4;5;42]. Therefore, a likely modeling of the *a priori* $P(F)$ is a one that is inversely proportional to the complexity of F and hence $P(F)$ can be approximated as:

$$P(F) \approx e^{\frac{-\Omega}{2}\phi[F]} \tag{36}$$

In (36), Ω is a constant and $\phi[F]$ is a measure of the complexity of the model F. By substituting (35) and (36) in (34), the result is equivalent to the regularized minimization objective function in (10):

$$\begin{aligned} \mathrm{F}^* &= \arg\max_F -\frac{\Psi}{2}\sum_{i=1}^{N}\big(t_i - f(x_i)\big)^2 - \frac{\Omega}{2}\phi[F] \\ &= arg\ \min_F \frac{1}{2}\sum_{i=1}^{N}\big(t_i - f(x_i)\big)^2 + \frac{\Omega}{2\Psi}\phi[F] \end{aligned} \tag{37}$$

4.2. Complexity and Error Bounds

One of the earliest attempts to measure the effect of the complexity of a trained model on its generalization was the work of Vapnik and Chervonenkis [39;41]. They studied the relationship between empirical error and the generalization error and under which

conditions the empirical misclassification rate $R_{emp}(F)$ of the trained model F uniformly converges to the true misclassification rate as the number of training examples increases:

$$\lim_{N\to\infty} R_{emp}(F) = R(F) \tag{38}$$

A model whose empirical classification error converges to its true error as the number of training samples goes to infinity is called consistent. The rate at which the difference between those two errors decreases as the number of training samples increases is called the convergence rate. A significant outcome of Vapnik and Chervonenkis' work was the formulation of the VC dimension which is a measure of the complexity of a function that enables an estimation of a probabilistic upper bound on the true error of the trained model.

Definition: The VC dimension of a set of indicator functions: $\{f \in \mathcal{F}\}$ is the maximum number h of vectors x_1, x_2, x_3 .... x_h that can be separated into two classes in all 2^h possible ways using functions from the same set. If for any number N, it is possible to find N points x_1, x_2, x_3 .... X_N that can be separated in all the 2^N possible ways using functions from the same set, then the VC-dimension of the set is infinite.

Knowing the VC dimension h of the space of functions $\{f \in \mathcal{F}\}$ being searched to fit the training data, Vapnik and Chervonenkis showed that with a probability $1-\delta$, the following upper bound on the generalization misclassification error of the trained model holds to be true:

$$R(f) \leq R_{emp}(f) + \sqrt{\left(\frac{h\left(log\left(2N/h\right)\right) - log\left(\delta/4\right)}{N}\right)} \tag{39}$$

As a result, a minimal classification risk $R(f)$ on future samples can be achieved by a function that fits the training data while having a very small VC dimension. The search or the optimization for such a function is called Structural Risk Minimization (SRM) and it is very similar to regularized training.

The VC error bound is a pessimistic loose upper bound and usually estimates an error larger than the true one. Since the introduction of the VC dimension and the VC bound, many other dimensions and error bounds have been introduced [38;43;44]. Some of these bounds were shown to provide tighter error estimation than the VC dimension. Nonetheless, estimating the VC dimension or any other complexity measure of a trained model is a non trivial task. Moreover, error estimation through leave-one-out or K-fold cross-validation is still used to validate these error bounds.

4.3. Regularization Methods: Examples

Optimal training should search for an optimal tradeoff between fitting the training data and minimizing the complexity of the solution. Different methods have been developed to realize this principle. Recent surveys about such methods are available in [42;45]. Two of the most popular approaches are explored in the next two sections.

a) Bridge Regression

Bridge regression is one of the simplest techniques to penalize a regression model [45;46] . A penalty term is added to penalize the size of the regression coefficients. In the case of linear regressions, the objective function to be minimized becomes:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - Wx_i - b)^2 + \lambda\sum_{z=1}^{Z}|w_z|^{\gamma} \quad (40)$$

In (40), λ and γ are two positive constants to be estimated by cross-validation or some prior knowledge. As $\lambda \to \infty$, all coefficients w_z go to zero and the model has a zero complexity but is incapable of accurate classifications whereas when $\lambda \to 0$, the solution becomes the unregularized trained model.

In the case of $\gamma = 2$, bridge regression becomes the well known Ridge regression and the final model contains small coefficients w_z with low variation. When $\gamma = 1$, bridge regression becomes LASSO regression and the trained model contains small coefficients w_z with higher variation. Moreover, in the LASSO case or other cases where $\gamma \leq 1$, more coefficients w_z will end up with a zero value. Therefore, the resulting solution is invariant to many dimensions and hence is easier to interpret.

Similar techniques to bridge regression were applied to the multi-layer neural network where the penalty is applied to the sizes of all weights in all layers [47]. In neural network literature, this penalty is usually referred to as a weight decay term and has been shown to improve the generalization of the network [44;47].

b) Regularization by Smoothness

A commonly used regularization technique in nonlinear models is to increase the smoothness of the approximation function or the separating hyper-plane. Smoothness is motivated by the known prior that decision boundaries in the real world are smooth [21]. Moreover, smooth functions demand significantly fewer examples to be learned [38;48].

There are a number of different heuristics to measure the smoothness of a function. For example, in Hilbert space, the smoothness of a function can be measured by an L_2 norm operator such as the following [42]:

$$\|f\|_{\mathbb{H}} = \left[\int_X \left(|f|^2 + \left|\frac{df}{dx}\right|^2 + .. + \left|\frac{d^m f}{dx^m}\right|^2\right) dx\right]^{1/2} \tag{41}$$

A special case of the generalized smoothness norm is the smoothing splines [42;49]. Smoothing splines measure the complexity of a function using the integral over the second derivative only and it is estimated as:

$$\|f\|_S = \int_X \left|\frac{d^2 f}{dx^2}\right|^2 dx \cong \sum_{i=1}^{N}\sum_{z=1}^{Z}\left|\frac{d^2 f}{dx_z{}^2}(x_i)\right|^2 \tag{42}$$

In (42), x_z refers to the dimension z while x_i refers to the training example i. In the case of a HyperBF network, minimizing functionals with either of the above smoothing measures is computationally expensive. Moreover, as the regularization parameter λ goes to infinity, the resulting model becomes a linear classifier and is still able to classify training data to some degree. The appendix gives details about the derivatives of the resulting functional of the smoothing splines for HyperBF networks as well as the time and memory complexity needed.

In this work, a new functional objective function is presented that combines training error, weight decay and local dimension reduction. The proposed regularization yields a smaller network structure that takes significantly less time for classification, making it a more attractive solution for applications where fast classification is needed.

CHAPTER III

METHODS

1. HyperBF Training

Training a HyperBF network demands the weights, centers and local scaling factors to be optimized simultaneously to minimize a certain objective function such as the regularized least squared error. In the case of a relatively large dataset with a large network structure, such optimization becomes computationally challenging. For example, training a 10-class network of 100 neurons to perform handwritten digits recognition of the MNIST dataset [50] where every sample is represented by 748 feature values, requires estimating 157,800 variables.

In this work, a two stage training approach is proposed. In the first stage, neurons are initialized using hierarchal clustering and weights are initialized to an initial value. In the second phase, a new gradient optimization based on the Resilient Propagation algorithm (Rprop) is used to estimate all parameters simultaneously in a direction to minimize the objective function.

1.1. Simplified Notation

Though the notation of the scaling matrix R_j in HyperBF networks as a covariance matrix is common, it is not unique and sometimes misleading. Many authors erroneously

refer to R_j as a covariance while it is not computed like a covariance. The elements of R_j can be best described as scaling factors that are estimated to minimize the classification error or any other training criteria. Therefore, for simplicity of notation R_j in the diagonal case will be represented as:

$$R_j = diag(v_{j1}, v_{j2} \ldots, v_{jz} .., v_{jZ}), \qquad v_{jz} \geq 0 \ \forall z. \tag{43}$$

For the rest of this dissertation, the elements of R_j will be referred to as local coefficients since the membership is computed as:

$$h_j(x_i) = exp\left(-0.5 \times \sum_{z=1}^{Z} v_{jz} \times (x_{iz} - \mu_{jz})^2\right) \tag{44}$$

1.2. Network Initialization

Clustering algorithms are typically used to initialize RBF or HyperBF networks. The goal of clustering is to initialize neurons in regions that are rich with samples. Although there is scarce literature about HyperBF networks, existing literature on RBF networks suggests a number of methods that are efficient initialization tools (see Chapter II). However, there is no consensus that a particular algorithm provides better classification results.

In the proposed training, agglomerative hierarchal clustering is used in the first phase to locate potential centers within every class independently. Though hierarchal clustering is not fast compared to other algorithms such as K-means, it is less sensitive to the random initialization of the clustering itself. In addition, the time needed to perform hierarchal clustering is usually significantly less than the time needed for the whole

network optimization in the second phase and hence does not significantly affect the overall complexity of training.

As will be discussed later in this chapter and illustrated in the experiments in the next chapter, the classification accuracy of HyperBF networks is heavily dependent on regularization. Generating a regularized HyperBF network is computationally demanding. To perform a fair comparison between different initialization methods, a regularized network needs to be trained each time with a new search for optimal regularization parameters through cross-validation. Due to limited computational resources and limited time, such a comparison is not possible. As an alternative, only the proposed initialization is used and the classification of the resulting regularized networks is compared to unregularized HyperBF networks and to SVM networks. SVM is considered to be the state of the art in classification.

Furthermore, since the second phase of training HyperBF network optimizes all network parameters simultaneously including neuron centers and scaling matrices, it is anticipated that the final network is less sensitive to the initialization.

a) Agglomerative Hierarchal Clustering

Agglomerative hierarchal clustering is typically initialized with a large number of clusters. Afterward, the algorithm iteratively merges pairs of similar clusters. As the algorithm merges clusters, it functions as a regular k-means algorithm by reassigning samples to their closest clusters and updating the corresponding cluster parameters including centers and covariances. Different distance measures have been used in literature to measure a distance between two clusters (i.e. Single Linkage, Average

Linkage, Complete Linkage, Ward's Linkage ... etc) [48]. In all experiments reported in this work, Ward's linkage method is used to calculate the distance between two clusters. Ward's linkage is based on the analysis of variance (ANOVA) to minimize the sum of variances within clusters. The distance between two clusters (θ_g, θ_j) is computed as the increase in the error sum of squares (ESS) caused by the merge:

$$d(\theta_g, \theta_j) = ESS(\theta_{g,j}) - ESS(\theta_g) - ESS(\theta_j) \tag{45}$$

Where

$$ESS(\theta_j) = \sum_{x_i \in \{\theta_j\}} (x_i - \mu_j)^2 \tag{46}$$

b) Clustering with Elliptical Clusters

Since membership to a HyperBF neuron is computed as a function of a weighted distance (Mahalanobis distance), the usage of a clustering algorithm with similar characteristics should provide a better initialization. Therefore, in order to assign samples to clusters, the normalized Mahalanobis distance is used:

$$d_{ij}^2 = |C_j|^{\frac{1}{Z}} (x_i - \mu_j)^T C_j^{-1} (x_i - \mu_j) \tag{47}$$

In (47), d_{ij} is the distance between sample i and cluster j, Z is the number of dimensions, while μ_j and C_j are the center and the covariance matrix of cluster j respectively. The normalization of the Mahalanobis distance aims at preventing one cluster from becoming very large. The complete clustering algorithm is detailed in the appendix of this dissertation.

c) Optimal Number of Clusters / Neurons

There is no theoretical foundation on the optimal number of neurons in an RBF network or a HyperBF network. A best effort practice would be to try a different number of neurons and select the setting that results in the best cross-validation or leave one out classification accuracy. A more practical solution that will be discussed later in this chapter is to start the network with a relatively large number of neurons and rely on a weight penalty for pruning. Nonetheless, since HyperBF neuron has a rich membership function (a Gaussian with a weighted distance), it is anticipated that the number of neurons would be much less than what would be needed in a regular RBF network or a regular SVM network. Due to the limited computational resources, in all experiments reported in this work the initial number of neurons from a class was typically less than $\sqrt{N}$ where N is the number of samples in the same class.

d) Post Clustering Initialization

Once clustering is completed, neuron centers are initialized with cluster centers while the local coefficient matrices are initialized by the clusters' diagonal covariance after being scaled:

$$R_j = \varphi_l C_j^{-1} \tag{48}$$

In (48), φ_l is a positive number used to scale all neurons initialized by clusters from class l to satisfy:

$$J(l|\varphi_l) = \frac{1}{N}\sum_{i\in\{l\}}\sum_{j\in\{l\}} e^{-0.5(x_i-\mu_j)^T R_j (x_i-\mu_j)} \cong 1 \tag{49}$$

In (49), $i \in \{l\}$ refers to samples from class l while $j \in \{l\}$ refers to neurons initialized from class l. The covariance scaling step is necessary to ensure the weighted distance is scaled appropriately. To find φ_l effeciently we use the following search algorithm.

Algorithm 1:

Init
$\varphi_l = 1$
Repeat
  Compute $J(l|\varphi_l)$ *as given in (49)*.
  If $J(l|\varphi_l) < 1$
    $\varphi_l = \varphi_l \times 1.1$
  Else
    $\varphi_l = \varphi_l \times 0.95$
Till $\| J(l|\varphi_l) - 1 \| < 0.01$

Furthermore, in the initialization phase, weights connecting neurons to the output nodes are initialized to an initial value of appropriate scale. An output node k associated with class l is initialized to connect to neurons initialized from the same class with a common value $\bar{w}_l$ and to zero to all other neurons. $\bar{w}_l$ is chosen so that it minimizes:

$$E(k) = \frac{1}{2}\sum_{i=1}^{N}\left(t_{ik} - f_k(x_i)\right)^2 \tag{50}$$

where

$$f_k(x_i) = \sum_{j \in \{l\}} \bar{w}_l h_j(x_i) \tag{51}$$

The optimal $\bar{w}_l$ that minimizes (50) should satisfy:

$$\frac{dE(k|l)}{d\bar{w}_l} = 0 \tag{52}$$

$$\bar{w}_l = \frac{\sum_{i=1}^{n} \sum_{j \in \{l\}} t_{ik} h_j(x_i)}{\sum_{i=1}^{n} \sum_{j \in \{l\}} h_j(x_i)} \tag{53}$$

After initializing all network parameters, all weights, means and scaling factors were optimized simultaneously by the Improved Scaled Rprop (iSRprop).

1.3. Scaled Rprop (SRprop)

In the regular Rprop algorithm, all step sizes are initialized to appropriate values η_{init} that are neither too small to cause a slow learning start nor too large to cause a jump away from the closest local minimum. η_{max} serves to limit the size of the step to avoid divergence or jumps. Nonetheless, a very small η_{max} will be counter- effective leading to slow optimization.

In the proposed Scaled Rprop method, a sufficient tradeoff between safe and fast learning is achieved by choosing separate η_{init} and η_{max} for every variable in the means and local coefficients. The values are chosen to be relative to their corresponding variable scale and the expected change to the output of the neuron so that the resulting change to the network is bounded. In contrast, all weights are treated similar to the regular Rprop with $\eta_{max}(w) = 2$ and $\eta_{init}(w) = 0.01$ for all weights. In addition, $\eta_{min} = 10^{-10}$ was found to work well for all variables in all experiments.

For a mean variable μ_{jz}, both η_{init} and η_{max} are computed to be relative to the weighted average distance $\overline{d_{zj}}$ to the neuron j along the dimension z:

$$\eta_{init}(\mu_{jz}) = 0.001 \times \overline{d_{jz}} + 10^{-6} \tag{54}$$

$$\eta_{max}(\mu_{jz}) = 0.1 \times \overline{d_{jz}} + 10^{-6} \tag{55}$$

where

$$\overline{d_{jz}} = \left(\frac{\sum_{i=1}^{N} h_j(x_i)(\mu_{jz} - x_{iz})^2}{\sum_{i=1}^{N} h_j(x_i)} \right)^{1/2} \tag{56}$$

In all experiments, $\overline{d_{zj}}$ is recomputed every five iterations and is not found to change rapidly.

In the case of the local coefficient factors, both η_{init} and η_{max} are computed so that the change to the membership of samples is bounded. The membership function in (44) can be factorized as:

$$h_j(x_i) = \prod_{z=1}^{Z} e^{-0.5 \times v_{jz} \times (x_{iz} - \mu_{jz})^2} \tag{57}$$

Figure 3 shows the changes of the individual exponential factor $e^{-0.5 \times v_{jz} \times d}$ as a function of the coefficient v_{jz}.

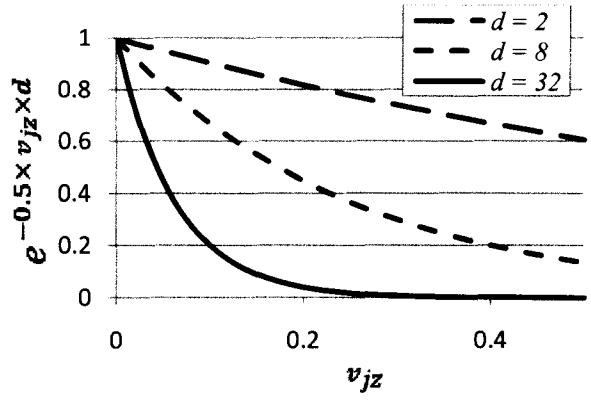

Figure 3: The Exponential Cofactor as a Function of the Local Coefficient.

It is evident from Figure 3 that the resulting change to the exponential factor does not depend solely on the amount of change to v_{jz} but also on the current value of v_{jz} and the

Euclidean distance along that dimension. Computing the expected change over all samples is computationally expensive. As a rough approximation of the change, the expected change is computed for a single virtual sample for which the distance from the neuron center along the same dimension z is the weighted average distance $\overline{d_{jz}}$ (equation (56)). To find η_{max} or η_{init} that will result a specific amount of change β to the exponential term, the following equation has to be solved for Δv_{zj}:

$$e^{\frac{(v_{jz}+\Delta v_{jz})\times\overline{d_{jz}}^2}{-2}} - e^{\frac{v_{jz}\times\overline{d_{jz}}^2}{-2}} = \begin{cases} min\left(\beta, 1 - e^{\frac{v_{jz}\times\overline{d_{jz}}^2}{-2}}\right) if\ \Delta v_{jz} < 0 \\ -min\left(\beta, \quad e^{\frac{v_{jz}\times\overline{d_{jz}}^2}{-2}}\right) if\ \Delta v_{jz} > 0 \end{cases} \tag{58}$$

Note that when $\Delta v_{jz} < 0$, the exponential term will increase towards one while when $\Delta v_{jz} > 0$, it will decrease toward zero. The exponential factor cannot go below zero nor above one. Thus a minimum term is needed. For all experiments in this dissertation, β was set to 10^{-2} and 10^{-4} to compute Δ_{max} and Δ_{init} respectively for every local coefficient factor separately. As a result, the exponential term will increase or decrease at most by 0.01 until it reaches one or zero respectively. Since this method is approximate and the distribution of the samples around the neuron center is not known, the change to the local coefficients is further constrained as:

$$\eta_{max}^{`} = \begin{cases} min\left(\eta_{max}, \gamma(v_{zj} - v_{min})\right) if\ \Delta v_{zj} < 0 \\ min\left(\eta_{max}, \gamma(v_{max} - v_{zj})\right)\ if\ \Delta v_{zj} > 0 \end{cases} \tag{59}$$

Applying (59) guarantees that v_{jz} will not go beyond the range $[v_{min} - v_{max}]$ in order to avoid numerical errors. This same processing is also applied to the initialization: $\eta_{init}^{`}(\sigma_{jz})$. In all experiments reported, $v_{min} = 0$, $v_{max} = 10^{20}$ and in the case of $\eta_{max}^{`}$,

$\gamma = 0.1$ while in the case of η_{init}, $\gamma = 0.01$. Those values provide a mild restriction to η_{max} and η_{min} but are protective from worst case scenarios that would lead to numerical representation problems.

The proposed method of selecting η_{max} and η_{init} for both the means and coefficient factors is better than using specific constant values for all variables like the regular Rprop. This approach is scale independent and ensures the steps are small enough to not cause jumps but large enough to cause significant learning.

The proposed modification to Rprop can be implemented with any variant of Rprop. Due to the proposed changes, oscillation of the network error became so minimal that using variants such as iRprop or JProp are of less benefit. All experiments in this work are implemented using the variant iRprop+ which adds a backtracking step whenever the error of training increases. Furthermore, iRprop+ was shown to outperform the original Rprop algorithm [35].

1.4. Iterative Training

Our implementation of both the Rprop and the proposed scaled Rprop updates the model in a semi-coordinate descent approach. In every iteration, a single neuron is considered at a time. The gradient directions of the considered neuron mean, local coefficients, and connecting weights are computed at once and the corresponding variables are updated after. Afterward, the output of the neuron and the output of the whole network are updated before the next neuron is considered. This approach guarantees that the gradient directions of the next neuron take into consideration the updates that are already made to previous neurons in the same iteration.

Furthermore, during training, the variable b in (20) was set to zero in the first hundred iterations. Afterward, it was updated using a coordinate descent with an exact line search (see Appendix). This is because the value of b is very influential on the output of the network. As a result, slightly larger than needed learning steps by the Rprop can result in a significant oscillation of the network error as it is being trained.

1.5. Partial Local Backtracking Step: Improved SRprop (iSRprop)

Though changes to the neuron parameters are locally in the right gradient direction, there is no guarantee that it is with the right magnitude. As a result, the network error might increase after a neuron is updated and hence the derivatives and the changes to the next neuron are not optimal. To ease this problem, once a neuron is updated, the error of the network is evaluated. If the error has increased, a partial backtracking step is performed along every variable ω in the mean and the local coefficients of the same neuron whose step size was increased as:

$$if \ \frac{dE^{(t-1)}}{d\omega} \times \frac{dE^{(t)}}{d\omega} > 0$$

$$\widehat{\omega}^{(t)} = \omega^{(t)} + \rho \times \eta_{\omega}^{(t)} \times sign\left(\frac{dE^{(t)}}{d\omega}\right) \tag{60}$$

$$\hat{\eta}_{\omega}^{(t)} = (1-\rho) \times \eta_{\omega}^{(t)}$$

The above partial backtrack step is performed only one time whenever needed for a neuron. In all experiments reported, ρ is set to 0.25 which means 25% of the last changes to a neuron are backtracked if the network error increases. Note that the signs of the derivatives are not recomputed to perform partial backtracking. Once the partial

backtracking is performed, memberships of the samples to the neuron are recomputed again. Since computing the derivatives and re-computing the memberships of a single neuron are of the same time complexity, the partial backtrack step adds 50% more computation whenever it is performed. However, the partial backtracking increases the probability that the derivatives are in the right global direction and decreases the probability of divergence. Scaled Rprop with the local partial backtracking will be referred to as iSRprop (improved Scaled Rprop).

2. Reduced HyperBF Networks (RHyperBF): Bridge Regression and Soft Local Dimension Reduction

HyperBF networks are rich complex models and by using efficient training approach they can approximate complex functions with high accuracy. To avoid overfitting and ensure a good generalization of the network as explained earlier (Chapter II), two regularization terms are added to the objective function:

$$E_{reg} = \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{K}(t_{ik} - f_{ik})^2 + \lambda_w \sum_{j=1}^{J}\sum_{k=1}^{K}\left\|w_{jk}\right\|^{\alpha_w} + \lambda_v \sum_{j=1}^{J}\sum_{z}^{Z} {v_{jz}}^{\alpha_v} \qquad (61)$$

The first term in (61) is the regular least square training error in the multi-class/outputs case, the second term is the bridge regression penalty (sometimes called weight decay) while the third term is the local dimension reduction penalty. λ_w and λ_v are two positive regularization parameters estimated by cross-validation. All derivative equations needed to optimize (61) using iSRprop are listed in the appendix.

The bridge regression term in (61) serves to minimize the values of the weights and hence eliminate or reduce the effect of neurons to the corresponding output. The choice of α_w affects the solution significantly [45;46]. For example in the case of α_w=2 which is known as ridge regression, the resulting network will have weights with small values and low variation. In contrast, in the case of α_w=1 which is known as LASSO, the resulting network will have small weights but with higher variation. Moreover, smaller values of α_w (i.e. $\alpha_w \in (0,1]$), will be more effective in driving more weights to zero. On the other hand, the third term in (61) serves to minimize the local coefficients of the distances along the different dimensions for the purpose of reducing or eliminating the effect of as many dimensions as possible on the membership to the neuron. The choice of the power α_v should also have a similar effect as the choice of α_w.

3. Model Selection

In the ideal case, the four variables λ_w, λ_v, α_w, and α_v need to be estimated through cross-validation to obtain optimal generalization. However, such training and parameter selection is unpractical. In this work, the search is constrained by setting both α_w, and α_v to one. This choice is motivated by the desire to obtain a small network structure.

The name Reduced HyperBF is motivated by the fact that the final network will be smaller than the initial network in terms of the number of active neurons and number of locally active dimensions. For the rest of this paper, the names Reduced HyperBF and Regularized HyperBF will be used interchangeably.

To avoid a greedy grid search to find λ_w and λ_v, we first search for optimal λ_v while setting λ_w to zero. Once λ_v is found, it is fixed and a new search starts for λ_w. This search

technique is practical though not optimal. Furthermore, the second regularization term is found to be more important for generalization. This could be due to the curse of dimensionality. The second term makes the network locally invariant to as many input features as possible and is expected to become more important as the dimensionality of the data increases. Moreover, when optimizing for λ_w or λ_v we start with a very small value that increases with small steps. This technique is used to avoid starting a different run for each parameter value.

4. Feature Ranking in Reduced HyperBF Based on Saliency

The proposed objective function in (61) performs an embedded localized feature selection. In this way, it drains every neuron from the effect of as many features as possible. As a result, features that are not informative for classification will inevitably have a zero or near-zero local weight in most neurons. Using the local weights of the feature, easy to compute ranking heuristics can be defined such as:

$$Score_1(z) = \sum_{j=1}^{J} v_{jz} \tag{62}$$

Though a very small value of $Score_1$ for feature z indicates that feature z is not important for classification, this measure is not accurate in comparing the importance of two features unless one of them have a near-zero $Score_1$ (i.e. $Score_1(z) < 10^{-6}$ for a standardized data). This is mainly because the value v_{jz} also depends on the localized scaling and distribution of samples along feature z around the center of neuron j.

A more accurate ranking of features is the saliency measure. The saliency of a feature is the magnitude of the error increase in the trained network as the underlined feature is being eliminated from the model:

$$Score_2\ (z) = \Delta E(z) \tag{63}$$

In (63), $\Delta E(z)$ is the change of the training error when feature z is discarded. A higher value of $\Delta E(z)$ indicates a higher important of feature z.

The Optimal Brain Damage algorithm (OBD) [51] is an efficient algorithm to approximate the effect of dropping a variable out of a model. OBD approximates the importance of a variable using the second derivative of the error surface with respect to the variable. OBD has been successfully used for feature selection and variable pruning in neural networks [52] and SVM networks [53;54].

The Optimal Brain Surgeon (OBS) [55] is a generalized version of OBD that approximates the increase in the network error in addition to an approximation of the network adjustment as the variable is being discarded. OBS demands the estimation of the inverse of the complete Hessian matrix and hence is a more computationally demanding tool. Due to the high number of variables in a HyperBF network, computing the Hessian matrix is not practical.

Fortunately, the saliency of features in HyperBF networks can be computed directly and efficiently without a need for approximation. In the two class case, the saliency of a feature z can be computed as:

$$\Delta E_v(z) = \Delta E\big(v_{jz} = 0, \quad \forall j\big) = \sum_{i=1}^{N} (t_i - f_i^z)^2 - E \tag{64}$$

where

$$f_i^z = \sum_{j=1}^{J} w_{jk} \times h_{ij} \times exp\left(\frac{1}{2} v_{jz} \times \left(x_{iz} - \mu_{jz}\right)^2\right) \tag{65}$$

In (64) and (65), the terms E and $w_{jk} \times h_{ij}$ are computed one time only for all features making the calculation for scoring a single feature efficient and of order $O(N \times J)$.

Furthermore, knowing that a particular feature is to be dropped out of the model, some adjustment to the network can be predicted and hence even a more accurate prediction of the saliency can be computed. For example, if dropping feature z out of neuron j causes a higher error than discarding the whole neuron, then it is intuitive to predict that training after discarding the feature will drop the whole neuron out of the network. Therefore, a better saliency of feature z can be approximated as:

$$Score_3\ (z) = \sum_{j=1}^{J} min\left(\Delta E\left(v_{jz} = 0\right), \quad \Delta E_w(j)\right) \tag{66}$$

In (66), $\Delta E_w(j)$ is the error increase in the trained network as neuron j is being discarded. Dropping a neuron out of a network can be achieved by either two ways: 1) set $w_j = 0$ or 2) set $v_{jz} = 0, \forall z$ or alternatively and more efficiently to compute: set $h_{ij} = 0, \forall i$. It is anticipated that, after discarding feature z, training will converge to the case with the least error and hence:

$$\Delta E_w(j) = min\left(\Delta E\left(h_{ij} = 0, \forall i\right), \quad \Delta E\left(w_j = 0\right)\right) \tag{67}$$

Computing $\Delta E_w(j)$ for all neurons is also very efficient (of order: $O(N \times J)$) and needs to be computed only one time for all neurons.

In Chapter VI, both scores: $Score_1$ and $Score_3$ are used to iteratively perform feature selection in microarray data. In problems with very large initial dimensionality, $Score_1$ can be used to filter and discard all features with very small $Score_1$ values. In all experiments performed in this work, discarding features with $Score_1 < 10^{-6}$, had no significant effect on the error of the trained model (zero effect in most cases). Once the number of features is small enough, $Score_3$ can be used for a better approximation of the saliency.

In Chapter IV, the proposed training and regularization are evaluated on seven real datasets and compared to other existing methods. Networks to classify handwritten digits are used to visualize the network structure and the effect of the local dimension reduction penalty. Chapters V and VI are dedicated to two case studies for utilizing Reduced HyperBF networks in bioinformatics applications. The networks are used for both accurate classification and higher level analysis.

CHAPTER IV

EXPERIMENTS AND RESULTS

To evaluate the proposed regularization of the HyperBF network, a number of experiments are performed using seven real datasets. First, to visually demonstrate the local dimension reduction, a network is trained to perform two-class classification using the MNIST handwritten digit dataset (digit 3 versus digit 5). Digits 3 and 5 are very similar in the lower part but different in the upper part. Thus, it is anticipated that neurons will select the upper pixels of the images as informative. Second, different networks will be trained to perform classification on seven different datasets. Cross-validation is used to select the regularization parameters. A comparison of classification accuracy between Reduced HyperBF, Regular HyperBF and SVM is reported. For every dataset, an SVM network is trained with similar settings of cross-validation using Libsvm software [11]. The comparison of classification accuracy is made using both cross-validation error and testing error. Afterward, a comparison of network size and interpretability is exposed in the discussion section. Finally, the proposed training algorithm iSRprop is evaluated and compared to the regular Rprop and the VSBP algorithms.

1. Datasets

1) USPS: USPS dataset is a handwritten digits dataset. It was collected from mail envelopes in Buffalo, NY [56]. It is composed of 9,298 handwritten digits of the 10 digit classes (7,291 for training, 2,007 for testing). As in [57], the images were smoothed with a Gaussian kernel of width= 0.75. Every image is of resolution 16 × 16 with pixel intensity values ranging from -1 to 1.

2) MNIST: MINST is an additional handwritten digits dataset [50] available as images of 28 × 28 resolution with intensity values at each pixel ranging from 0 to 255. 6,000 images for every class are available for training and another 1,000 for testing.

In all experiments reported in this work using USPS or MNSIT dataset, the raw intensities of pixels are used as a feature vector.

3) Human TSS: Transcription start sites (TSS) are specific locations in the DNA where the transcription of the DNA starts as an initial step to produce proteins. The dataset is composed of 8,508 positive and 85,042 negatives examples. The data was divided into 50% for training and 50% validation. Every sample is represented by 1,024 features in Euclidean space. Furthermore, since the data is unbalanced, the objective function in (61) was weighted to give more weight to the minority class (see Chapter VI for more details).

4) ISOLET: This phonetic dataset is available at the UCI repository (http://archive.ics.uci.edu/ml/). It consists of the pronunciation of the 26 English letters by 150 different persons. Every sample is represented by 617 attributes of audio features that include spectral coefficients, contour features, sonorant features, pre-sonorant features, and post-sonorant features [58]. One fifth of the data is left out for testing (see Table 1).

5) Wisconsin Breast Cancer: This dataset is also available at the UCI repository (http://archive.ics.uci.edu/ml/). The original samples are digitized images of a fine needle aspirate (FNA) of a breast mass of 569 different people labeled into two diagnostic classes: (M = malignant, B = benign). Every sample is represented by 32 visual features of the cell nuclei [59] including: radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension.

6) Protein: This dataset set is available at [60] and it is composed of 24,387 protein samples. For every protein sample, given the amino acid sequence, the goal is to predict the secondary structure of the protein: (helix, sheet, or coil). For very data point, a feature vector of 357 attributes is provided that represents the local amino acids frequencies in the given sequence in addition to other related and well aligned proteins [61;62].

7) Satimage: This dataset is part of the StarLog database and it is available in [60]. It is generated from 6,435 satellite images of land. The learning task is to classify these images into one of six classes based on soil type. Every data point is represented by multi-spectral values of pixels in 3x3 neighborhoods in the image.

Table 1

Details of the Datasets Used for Evaluation and Comparisons.

Dataset	# of Samples	# of Classes	# of Test Samples	# of Features
MNIST	60,000	10	10,000	784
USPS	7,291	10	2,007	256
TSS	93,550	2	N/A	1,024
ISOLET	6,238	26	1559	617
W. Breast Cancer	569	2	N/A	32
Protein	17,766	3	6,621	357
SatImage	4,435	6	2,000	36

2. Experimental Results

2.1. Local Dimension Reduction in Handwritten Digits Classification

The first set of experiments was performed using the handwritten digits datasets only (USPS and MNIST). All data is normalized so that all attributes are bounded in the range [0-1]. For visualization, a neuron is visually represented by four images:

1. Center Image: This image represents the actual center of the neuron(μ_j). In some cases, centers can move out of the samples' volume and hence some pixels values at the neuron center might become negative. For visualization, negative values in μ are set to zero while values larger than one are set to one.
2. Local Coefficients Image: Intensity of pixel (z) in this image of neuron (j) is made proportional to the log of the magnitude of (v_{jz}: local coefficient). Therefore, dark pixels correspond to pixels that are not important for classification.
3. Medoid Image: Image of the sample with the highest membership in the neuron (closest sample in space).
4. Generative Mean Image: In the two classes case with one output, the generative mean for a neuron with a positive weight is the weighted average of samples from the first class (t = +1) while the generative mean with a negative weight is the weighted average of samples from the second class (t = -1):

$$G\mu_j = \begin{cases} \dfrac{\sum_{x_i \in \{t=+1\}} h_{ji} \times x_i}{\sum_{x_i \in \{t=+1\}} h_{ji}} & if\ w_j > 0 \\ \dfrac{\sum_{x_i \in \{t=-1\}} h_{ji} \times x_i}{\sum_{x_i \in \{t=-1\}} h_{ji}} & if\ w_j < 0 \end{cases} \tag{68}$$

In the multiclass case, the generative mean is the weighted average of samples whose class corresponding output node is connected to the neuron with a positive weight.

In the first experiment a single network was trained to discriminate digit 3 from digit 5 in the MNIST dataset. The network is initialized with 20 neurons (10 clusters from each class). Only local dimension reduction is used for regularization ($\lambda_w = 0$). With $\lambda_v = 0.4$, the resulting training and testing error was 0.8% and 0.7% respectively. Figure 4 shows the details of the first 8 neurons initialized from each class. Every column shows the four pictures of a neuron: the center, the local coefficients image, the generative mean and the medoid respectively.

The second row of Figure 4 shows the local coefficients of the corresponding dimension (pixels). As can be seen in this row, neurons chose the upper pixels as the most informative (as indicated by white pixels). On average, 87% of the pixels ended up with a zero local coefficient leading to a greatly reduced network. In addition to having a small size, the resulting network has meaning by revealing the informative areas of the images that need to be examined in order to discriminate between digit 3 and digit 5.

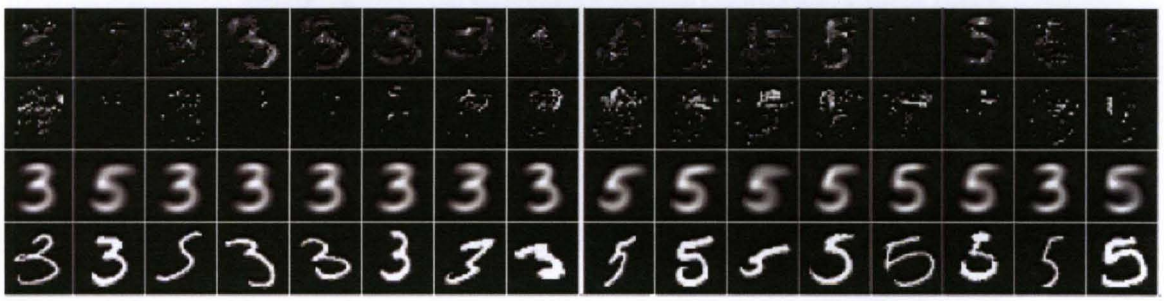

Figure 4: Sample Neurons from a Network Trained to Discriminate Digit 3 from 5. The first eight neurons were initialized from class 3 and the remaining eight from class 5.

Initially, it was thought that the resulting neuron centers would be subprototypes and should be similar to centers of clusters. Surprisingly that was not the case. It is evident from Figure 4, the resulting centers did not look like any of the digits. This could be due to the fact that the resulting centers are discriminative centers and not summarization centers. In other words, they move based on what is common between a class samples that make them different from other classes. This is unlike clustering centers which group samples mainly based on what is similar among them.

In the second experiment, two different regularized networks of size 200 were initialized to classify the ten digits of MNIST and USPS separately. In each case, neurons were initialized by 20 clusters from each class. Figures 5 and 6 show subsets of the neurons from the MNIST network and USPS network respectively. In each case, the first four neurons initialized from the same class are displayed.

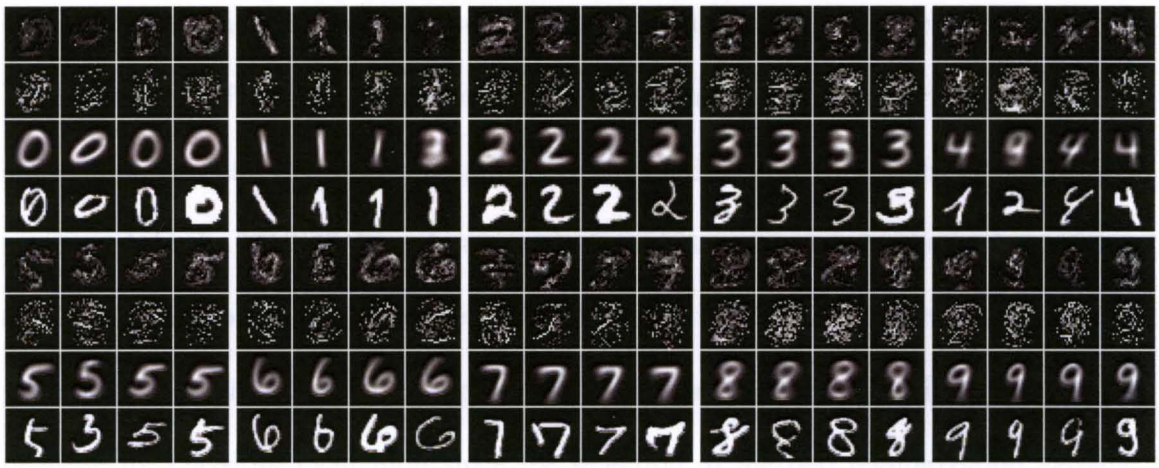

Figure 5: Sample Neurons from Network Trained to Recognize the 10 Digits of MNIST. From every class, the first four initialized neurons are listed. Every neuron is represented in a column by four images.

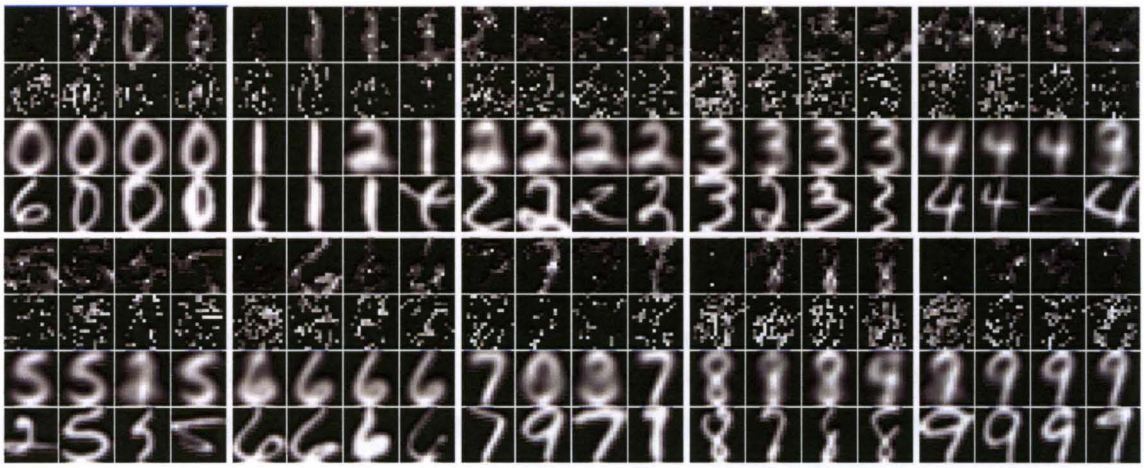

Figure 6: Sample Neurons from Network Trained to Recognize the 10 Digits of USPS. From every class, the first four initialized neurons are listed. Every neuron represented in a column by four images.

2.2. Classification Accuracy and Comparison

To evaluate the generalization of the proposed regularization, a Reduced HyperBF network is trained for every dataset. In addition, regularization parameters are selected through cross-validation (see methods). Table 2 lists the classification error of HyperBF, Reduced HyperBF and SVM networks. The classification error is reported over both the cross-validation and the testing data. In five of the datasets, 10-fold cross-validation is performed. In the MNIST dataset case, only 5-fold is performed due to the huge size of data. In the Wisconsin Breast dataset case, only cross-validation is reported since the data is not available with standard testing set. A 10-fold cross-validation by itself is a credible measure for classifier evaluation since the selection of the two regularization variables (λ_w and λ_v) is unlikely to cause overfitting over the 10 left out folds. Similarly, all SVM networks are trained with the same cross-validation and parameter selection settings using Libsvm software and according to the best practices guide offered by the authors of the tool [11;63]. In addition, all SVM networks are trained with a parameterized

Gaussian RBF kernel (equation (17)), where both, the width of the kernel (σ in equation (17)) and the regularization parameter (C in equation (18)) are selected through a grid search to minimize the cross-validation error [63].

Table 2

Comparison of Classification Error between HyperBF, Reduced HyperBF, and SVM networks on Six Datasets. Best results for every dataset are face-bolded.

Data Set	K-Folds	CV Error%			Test Error%		
		HBF	R-HBF	SVM	HBF	R-HBF	SVM
USPS	10	2.47	**1.37**	1.74	5.83	**4.38**	4.78
MNIST	5	3.33	2.29	**1.52**	3.23	2	**1.4**
ISOLET	10	4.44	3.03	**2.45**	6.54	3.78	**3.21**
Breast	10	4.04	**1.67**	1.93	N/A	N/A	NA
Protein	10	38.61	31.27	**29.56**	38.07	**30.04**	**29.9**
SATIMAGE	10	9.8	8.71	**7.86**	10.7	9.5	**8.8**

Table 3 lists the auROC in TSS classification using HyperBF, Reduced HyperBF, and SVM networks. The TSS data is significantly imbalanced (~10 times more negative samples). Therefore, classification accuracy is not well suited for evaluation. In this experiment, regularization parameters in both cases of Reduced HyperBF and SVM networks were selected so that area under the Receiver Operating Characteristic curve (auROC) is maximized in the classification of the validation data.

Table 3

Comparison of auROC Between HyperBF, Reduced HyperBF, and SVM Networks in the Classification of (TSS) Sites in Human DNA. Best result is face-bolded.

Dataset	Validation auROC%		
	HBF	R-HBF	SVM
TSS	88.5	94.06	**94.42**

Tables 2 and 3 show that Reduced HyperBF networks consistently and significantly outperformed the regular HyperBF networks. Furthermore, Reduced HyperBF networks have resulted in competitive classification accuracy when compared to SVM networks. In terms of cross-validation accuracy, Reduced HyperBF outperformed SVM in two datasets out of six. In the other four datasets, with the exception of Protein dataset, SVM outperformed Reduced HyperBF networks with a small difference: (0.77%, 0.58%, 1.7%, and 0.85%). In terms of testing, the average difference of classification accuracy was 0.32% for the sake of SVM. Furthermore, in the case of TSS detection, SVM outperformed Reduced HyperBF in terms of auROC by 0.36% only. Nonetheless, as will be shown and discussed later in section 2.4, the resulting Reduced HyperBF networks had 1-3 orders of magnitude smaller network structure than their SVM counterparts.

2.3. Sensitivity to Regularization Parameters

To select the optimal regularization parameters, an exhaustive grid search was performed, and for every pair of values of λ_w and λ_v a network was trained for every training partition of the data. Figure 7 shows the classification error for each grid search for six of the datasets in Table 1.

It is evident from Figure 7 that the regularization parameters have improved the classification accuracy of HyperBF networks. Furthermore, the local dimensionality reduction penalty term is shown to be more important than the weight decay term. Searching for λ_v alone improves classification better than searching for λ_w alone.

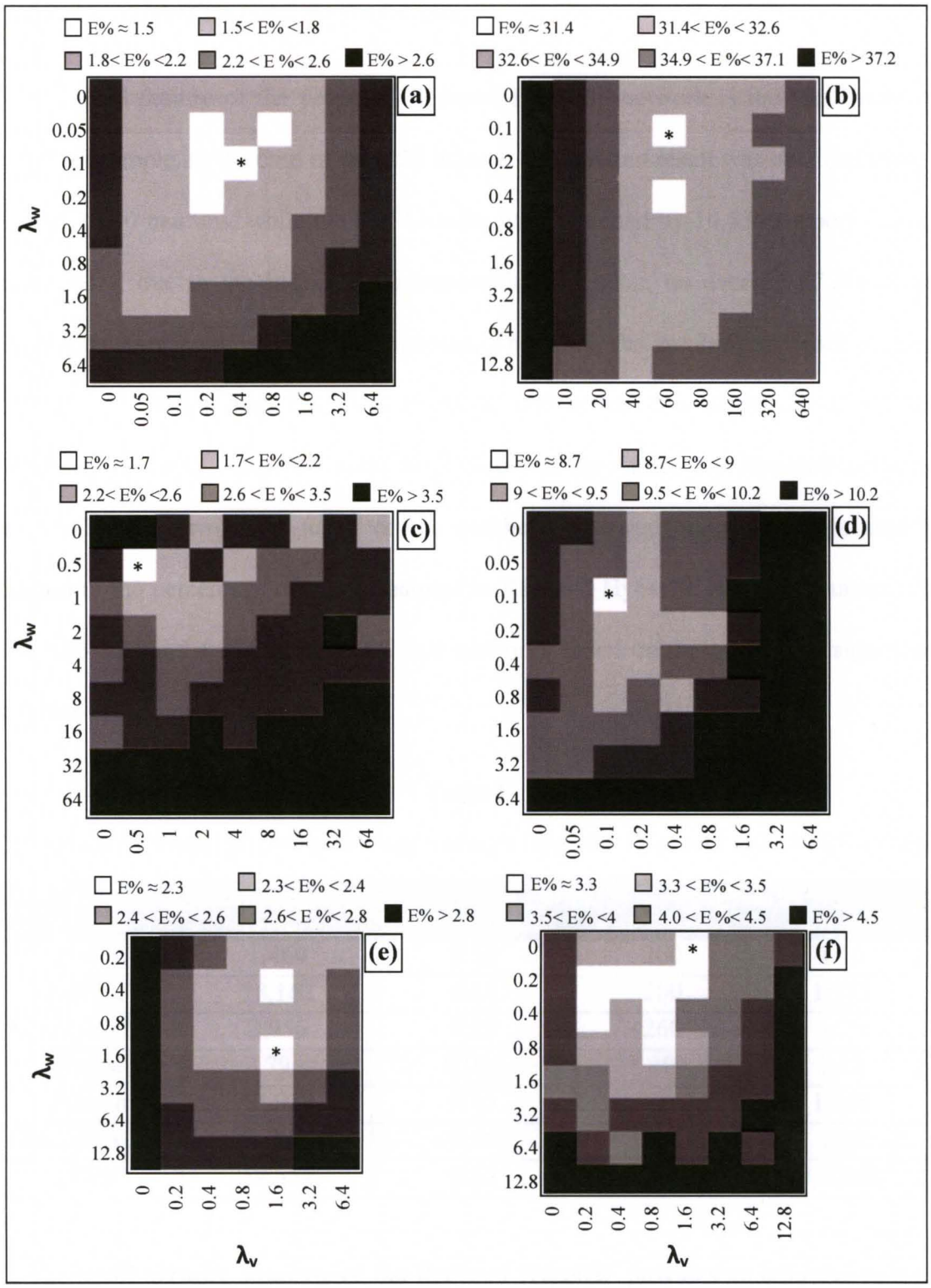

Figure 7: Effect of Regularization Parameters (λ_w and λ_v) on the Cross-Validation Classification Error of Each Dataset: a) USPS, b) Protein, c) Wisconsin Breast Cancer, d) Satimage, e) MNIST, and f) ISOLET. Starred boxes are the ones with the highest CV accuracy.

2.4. Network Size and Interpretability

A critical feature of the proposed Reduced HyperBF network is the small network size. For example, in the case of the TSS dataset, the reported result was obtained using a network of 30 neurons, while the SVM results were obtained by 14,554 support vectors. Furthermore, due to the localized dimensionality reduction, on average 87.3% of the dimensions were completely locally ignored. Therefore, the resulting network is about 1,900 times smaller than the SVM network, and is hence a better choice for fast classification. Table 4 shows a comparison between the regularized HyperBF networks and the SVM networks in terms of the number of neurons/support vectors used in addition to the percentage of active features in Reduced HyperBF network neurons. The table also shows the size ratio of the two networks based on the number of significant variables in each network.

Table 4

Comparison of Model Structure between Reduced HyperBF Networks and SVM Networks

Dataset	# of Support Vectors	Active Dims %	# of Neurons	~Size Ratio
USPS	1,464	0.36	200	1:10
MNIST	18,162	0.24	200	1:172
ISOLET	3,956	0.29	260	1:26
Breast	79	0.084	40	1:12
Protein	12,019	0.15	60	1:668
SATIMAGE	1,322	0.46	60	1:24
TSS	14,554	0.13	30	1:1,900

The small network structure of the Reduced HyperBF provides an effective tool for higher level analysis. For example, in the case of initializing the network with 30 neurons, due to the weight decay term, only three neurons end up with a positive weight

greater than 0.001. Since those neurons are the reason why the network gives positive samples a score of one, it is intuitive to argue that those are centers of subpopulations of TSS samples. Furthermore, the active local dimensions (dimensions with non-zero local coefficients) provide insightful information about the unique characteristics of a subpopulation in specific and the whole class in general. The case of (digit 3 versus digit 5) network is an example of this type of information. Chapters V and VI of this dissertation are dedicated to two case studies for utilizing this interpretability of the Reduced HyperBF networks in bioinformatics applications.

2.5. Evaluation of iSRprop

Figure 8 shows the network error as a function of the training iterations on six different unregularized networks using iSRprop, iRprop+ and BPVS algorithms separately. The Y-axis is scaled to the $\log_2$ because the initial error is very high.

Both iRprop+ and iSRprop are almost of the same time and memory complexity (see appendix). However, the localized partial backtracking adds extra computation when executed. On average, it was found that iSRprop demands 5-10% extra computation time per iteration which is not a significant increase. On the other hand, BPSV demands more computation because it tries multiple learning rate values in the same iteration.

From Figure 8, it is evident that both iRprop+ and the proposed iSRprop outperform the BPVS. Furthermore, the proposed iSRprop in all experiments converges faster than iRprop+ and suffers less oscillation. Oscillation in the training might propagate into divergence. The behavior of the iRprop+ in the Protein dataset case (Figure 8e) is an example of such a possible divergence of regular Rprop training algorithms. A proof of

whether or not the proposed iSRprop guarantees convergence is beyond the scope of this dissertation. Nonetheless, as explained analytically in Chapter III, and demonstrated experimentally in Figure 8, the proposed iSRprop is less likely to cause significant oscillation, and hence is more likely to converge.

Note that the modifications to the iSRprop are even more significant than it appears in Figure 8. In fact, the used implementation of the regular iRprop+ is semi-scaled. We found the default values of iRprop+ as given in [35] to result in unstable training that diverged in many cases. Therefore, in our implementation of the regular iRprop+, η_{max} and η_{init} for all means' variables were set to be the average of those in the iSRprop while η_{max} and η_{init} for all local coefficients were set to be proportional for the same values as for the mean variables.

Table 5 lists the time in minutes needed to finish a hundred training iterations on a single core (Xeon processor: 2.4 GHz/2MB cache memory). The total time depends on the stopping point which varies between datasets. In most cases, iSRprop converged in less than 400 iterations. All experiments reported in this chapter demanded approximately two weeks of computation on a machine of 24 processing cores running in parallel (Xeon processor 2.4 GHz/2MB cache memory). Most of the computations were spent on the search for the regularization parameters. The MNIST dataset had the highest share of computation time though the search for parameters was limited. We should mention that training and parameter selection of SVM on MNIST dataset using Libsvm software was also time consuming, and it took about two weeks of computation on six cores running in parallel.

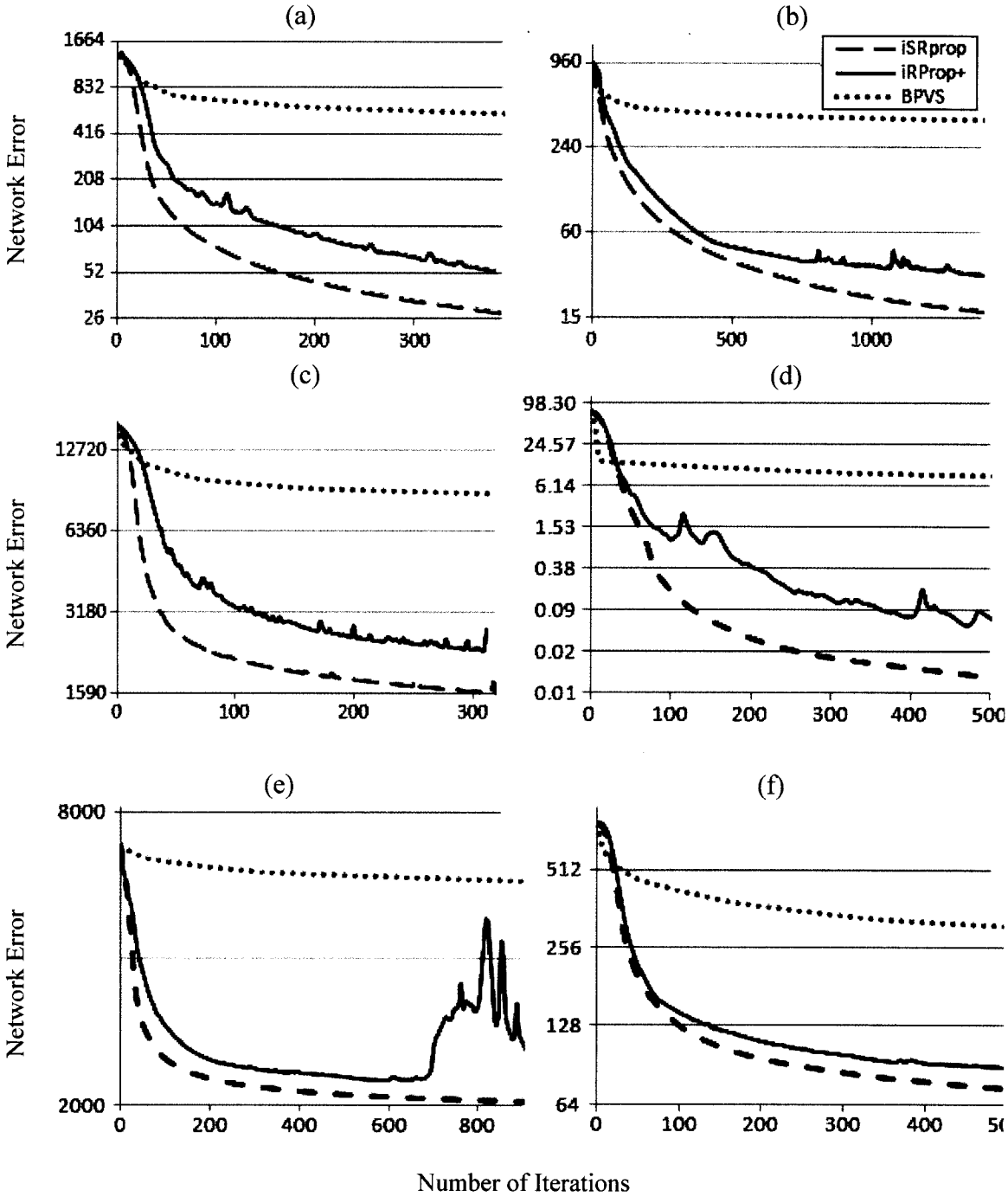

Figure 8: Training with both iRprop+ (solid line), **iSRprop** (dashed line), **and BPVS** (dotted line) **of Six Networks** (a) USPS network with 100 neurons, (b) TSS network with 30 neurons, (c) MNIST network with 100 neurons, (d) Breast Cancer network with 40 neurons, (e) Protein network with 30 neurons, and (f) Satimage of 60 neurons. The X-axis is the number of iterations and the Y-axis is network error scaled to $\log_2$.

Table 5

Training Time in Minutes for a Hundred iSRprop Iterations for Every Network

Dataset	# of Neurons	Time in Minutes / 100 iterations
USPS	200	14
MNIST	200	190
ISOLET	260	70
Breast	40	1
Protein	60	50
SATIMAGE	60	2
TSS	30	21

3. Conclusions

Reduced HyperBF network was shown to provide a classification accuracy competitive to SVM networks while requiring a significantly smaller network structure. Furthermore, the resulting compact network representation gives a powerful tool for higher level analysis. Significant centers can be argued to be centers of subpopulations of the class, while the locally significant dimensions provide information about the unique properties of the subpopulation.

On the other hand, the proposed iSRprop training algorithm (scaled Rprop with localized partial backtracking) results in a smoother training that is less likely to diverge. In all experiments presented in this dissertation, the proposed training algorithm did not diverge a single time. Furthermore, the proposed scaling and the local partial backtracking in Rprop training are general concepts and can be applied with necessary modifications to different optimization problems.

CHAPTER V

CASE STUDY: IDENTIFICATION OF TRANSCRIPTION START SITES IN HUMAN DNA USING OLIGONULEOTIDE POSITIONAL FREQUENCIES

1. Motivation

The accurate identification of promoter regions and transcription start sites (TSSs) is an important step for in-silico gene discovery and understanding of the transcription regulation mechanisms. Every eukaryotic gene has a core promoter region in the 5' untranslated region (UTR) that contains, at a minimum, one TSS signal. Most eukaryotic genes are transcribed by RNA Polymerase 2 (Pol-II) which binds at the TSS segment. Promoter regions are found to share common subtle patterns or models known as motifs that act as binding sites where other transcription factors (TFs) attach to facilitate or regulate transcription. For example, up to 80% of human promoters contain an initiator element (Inr) located at the transcription start site with a consensus sequence of YCAYYYYY, where Y represents a pyrimidine base C or T [64]. Roughly 30% of human core promoters are found to contain a TATA box at position of -20 to -30 from the TSS with the consensus TATAAA [64]. The TATA box tends to be surrounded by GC rich sequences. Promoter signals with greater variation are found in the promoter region proximal to the TSS, where motifs such as the CAAT, GC, E, and GATA boxes

are located [65]. More details about compositional characterization of known human promoter motifs can be found in [65].

Many recently published methods have achieved high identification accuracy of TSS. However, models providing more accurate modeling of promoters and TSS are needed. In this chapter, a novel identification method for identifying transcription start sites that improves the accuracy of TSS recognition for recently published methods is proposed. This method incorporates a metric feature based on oligonucleotide positional frequencies, taking into account the nature of promoters. A Reduced HyperBF is trained and employed as a classification algorithm. Using non-overlapping chunks (windows) of size 50 and 500 on the human genome, the proposed method achieves an area under the Receiver Operator Characteristic Curve (auROC) of 94.75% and 95.08% respectively, providing increased performance over existing TSS prediction methods.

2. TSS Detection Algorithms

A number of algorithms for promoter and TSS recognition are currently available. Each attempts to model promoter pattern(s) using features such as CpG islands and known transcription factor binding sites (TFBS) to distinguish promoters from non-promoters. Some methods such as Autogene [66] and Promoter Scan [67] use position weight matrices (PWM) to signal the presence of a high density of binding sites indicating potential promoters. However, it has been shown that both the location and combination of different binding sites are important for promoter recognition [68;69]. Eponine [70] improves recognition by associating every PWM with a probability distribution based on its position relative to the TSS. A more recent tool that tries to

model the oligonucleotide positional densities is described in [71]. However, this particular design employs a naïve Bayes classifier that assumes every oligonucleotide's positional distribution is independent, and is therefore unable to capture the co-occurrence of a specific combination of binding sites.

In a recent study, Bajic and colleagues conducted a large scale comparison study of eight known TSFs [72]. They demonstrate that a number of these tools perform well, yet leave a lot of room for improving detection accuracy. Among the most successful tools identified were Eponine [70], McPromoter [73], FirstEF [74] and DragonGSF[75].

A more successful approach is the ARTS tool developed by Sonnenburg and colleagues [76] which uses a support vector machine (SVM) with multiple advanced sequence kernels. ARTS is able to achieve a high accuracy with the area under the ROC curve of 92.77% and 93.44% for genomic DNA chunk sizes of 50 and 500 respectively, demonstrating a superiority to Eponine [70], McPromoter [73] and FirstEF [74]. As part of the ARTS system, a large training and testing dataset was constructed along with measures for testing and evaluating promoter detection approaches in a consistent fashion. This dataset and methodologies are used to compare the results of our approach, RBF-TSS, to ARTS, which has been shown to be the best performing approach previously available. In the comparison section, the performance measures of ARTS, Eponine, McPromoter and FirstEF are listed as they were reported in [76].

3. HyperBF-TSS

In this dissertation, a new method is presented to model the positional frequency of oligonucleotides to form a single feature to represent the given sequences for promoter

detection. Unlike [77], which measures the frequency at every single base pair position from the TSS, our approach takes the sequence around the TSS and divides it into overlapping windows for which the frequency of oligonucleotides of specific length are measured. A number of different combinations of window sizes, varying overlapping lengths and oligonucleotides length were examined. The combination resulting in the largest area under the ROC curve in classifying the validation data was chosen for the testing phase. The extracted positional frequency feature is used as an input into a HyperBF network for training.

The same experimental setting published to test the ARTS method and others in [76] is used to evaluate HyperBF-TSS. The proposed method showed to be superior to the ARTS in terms of the area under the ROC curve but not in terms of the area under precision recall curve (PRC). However, the PRC might not be a suitable measure of the performance of promoter identification tools since some samples labeled as true negatives might indeed be novel promoter regions that are not discovered yet. For example, the removal of 100 negative samples out a million causes the area under the PRC to increase by 6.36% and 10.86% with chunk sizes of 50 and 500, respectively, while the area under the ROC remains nearly identical.

4. Methods

4.1. Feature Prototype (Local Oligonucleotides Frequencies)

Promoter regions function as such due to the co-occurrence of a specific set of motifs at specific yet flexible distances from the TSS [68;69]. However, none of the published

studies or tools has found a single common pattern that can explain all promoters, indicating the likelihood of multiple promoter patterns.

In order to capture the characteristics of the given promoter sequences, training sequences with known TSS are divided into overlapping regions (Figure 9). Either 4-mer or 3-mer oligonucleotide frequencies are measured in every sub-region. All of these sub-frequencies are combined to form a feature vector to describe and represent the given sequence sample. This approach is a compromise between methods that use the frequencies of all oligonucleotides around the TSS regardless of their positions, and those that measure positional densities at every single base relative to the TSS. Knowing the region in which each oligonucleotide occurs yields approximate positional information about the motifs.

Eight combinations of region lengths and overlap sizes are tested to extract separate features, including seven with oligonucleotide of length four and one with oligonucleotide of length three. The overlapping regions considered for each of these combinations are listed in Table 6, with the position relative to the known TSS. These regions are further illustrated in Figure 9 for combination 7. In general, regions and overlap areas close to the TSS are short and increase in length as they go farther from the TSS. This is due to our knowledge that common motifs in the core promoter region (close to the TSS) are found to have more strict positions than common motifs found in the promoter proximal region area (farther from the TSS) [68;69]. After each combination is considered, the one resulting in a classifier with the highest area under the ROC for the validation data is selected for testing.

Table 6

Sub-Regions and Oligonucleotide Lengths Considered for Feature Extraction.

Feature	Sub-Region ranges (Relative to the TSS)	Oligonucleotide
1	(-500,-230),(-300,-50),(-100,20),(-20,99)	4 mer
2	(-500,-220),(-310,-40),(-110,30),(-30,99)	
3	(-500,-240),(-290,-60),(-90,10),(-10,99)	
4	(-600,-230),(-280,-40),(-70,70),(40,199)	
5	(-600,-240),(-270,-50),(-60,60),(50,199)	
6	(-600,-280),(-330,-110),(-150,20),(-20,149)	
7 (Figure 9)	(-600,-230),(-280,-40),(-70,70),(40,249)	
8	(-650,-490),(-550,-400),(-450,-310),(-350,-220),	3 mer

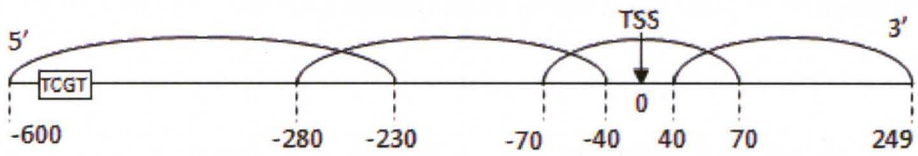

Figure 9: Training Sequences Are Divided Around the TSS with Overlapping Regions. This specific subdivision shows feature 7 settings, as described in Table 6.

4.2. Imbalanced Training

Since the training data is imbalanced (nearly 10 times more negative samples), the minority class might have a minimal effect on the resulting network leading to unsatisfactory results. A well known and used approach to counter-effect this property [78] is to weight the training error in the objective function in (61) to give higher weight to the minority class as:

$$E_{reg} = \frac{1}{2}\sum_{i=1}^{N} u(t_i) \times (t_i - f_i)^2 + \lambda_w \sum_{j=1}^{J} \|w_j\| + \lambda_v \sum_{j=1}^{J}\sum_{z}^{Z} v_{jz} \qquad (69)$$

In (61), $u(t_i)$ is a weight dependent on the class. In this case of training TSS network, $u(t_i)$ will be 1 for positive samples and 0.1 for negative samples. Two other networks are trained with different weighting (1, 0.33) and (1, 1).

5. Experiments and Results

5.1. Dataset

The dataset used for evaluating ARTS [76] was downloaded from (http://www.fml.tuebingen.mpg.de/raetsch/projects/arts) and used to evaluate RBF-TSS. This dataset is divided into three parts: training, validation and testing. As a summary of the ARTS paper, the training and validation were extracted from the dbTSS version 4 (dbTSSv4) [79] which is based on the UCSC human genome sequence assembly and annotation version 16 ("hg16") [80]. RefSeq [81] identifiers from dbTSSv4 were used to extract the corresponding mRNA using NCBI nucleotide batch retrieval. Afterward, they aligned all the retrieved mRNA from NCBI to hg16 genome using BLAT [82]. The best alignment position at the genome was compared to the putative TSS positions as stated in dbTSSv4. Sequences whose positions did not meet the following checks were discarded:

1. Chromosome and strand of the TSS position and of the best BLAT hit match.
2. The TSS position is within 100 base pairs from the gene start as found by the BLAT alignment.
3. There is not any processed putative TSS is within 100bp of the current one.

As a result, 8,508 genes were accepted and positive examples were extracted as a window of size [-1200, +1200] around the TSS.

For this dataset, 85,042 negative samples were created by randomly extracting 10 subsequences of window length [-1200, +1200] from the interior of every gene between 100bp downstream of the known TSS and the end of the gene [72]. This method is arguable since it cannot be guaranteed these negative samples do not contain promoters. However, it is near certain most of the extracted negative samples are true negatives since TSS are found to be rare compared to the size of the genome. Furthermore, there is not any other natural method of recognizing true negatives in the genome.

The 8,508 positive and 85,042 negatives examples were both divided into 50% for training and 50% for validation. The testing dataset was extracted as the set of all new genes from dbTSSv5 [83] which is based on hg17 and did not appear in dbTSSv4. Genes that have more than a 30% mRNA overlap are removed from consideration.

5.2. Training and Model Selection

Eight different features were extracted as described in the "Feature Prototype" section. Clustering to initialize the HyperBF network was performed using hierarchal clustering within the positive class only and all weights were initialized to be positive. Furthermore, since the data is unbalanced, the objective function in (61) was weighted as described in [78] to give more weight to the minority class.

Initially, for every feature, a separate HyperBF network of 30 neurons was constructed without regularization. The two best performing features in classifying validation data in terms of the auROC were chosen for further training. Those two features were four and seven (Table 6). Both features were extracted by measuring the frequency 4-mers in four overlapping sub-regions of the given sequences as described in

Table 6. Afterward, a parameter selection through cross-validation was performed with both features 7 and 4 separately (see methods in Chapter 3). Feature 7 was found to make the best performance in terms of area under the ROC.

Finally, two different networks of two different sizes (30 and 60 neurons) were trained with cross-validation. Table 10 lists the auROC for the training and validation datasets for both networks in addition to the auROC from an SVM network. The table also lists the number of neurons/support vectors used in every network. The SVM network was trained for the same problem using Libsvm [11]. In both cases of the regularized HyperBF and the SVM network, regularization parameters were selected to maximize the area under the ROC for the validation data. Furthermore, in both cases the objective function was balanced to give more weight for the minority class (weight of 1 for positive samples and 0.1 for negative samples).

Table 7

auROC of TSS Detection in the Validation Data Using Reduced HyperBF and SVM Networks.

Network	auROC		Network Size: # of Neurons/Support Vectors
	Training	Validation	
RHBF-30	96.84%	94.06%	30
RHBF-60	96.1%	94.1%	60
SVM	99.79%	94.42%	14554

Though, as shown in Table 10, SVM slightly outperformed the best regularized HyperBF network, HyperBF network is about three orders of magnitude smaller than the SVM network. Furthermore, a network of 60 neurons did not provide any substantial improvement over a network of 30 neurons.

To investigate the effect of imbalance in the training data, two more networks were trained with cross-validation with different weightings where the positive examples had a

weight of 1 in both of them whereas negative samples had a weight of 0.33 and 1. Table 8 lists the auROC and auPRC of the validation data in each case. Table 8 also lists the number of active positive and active negative neurons (neurons with significant weight value (larger than 0.001)).

Table 8

Effect of Training with Weighted Error on auROC and auPRC of the Validation Data and the Number of Resulting Active Neurons (with positive and negative weights (+/-)).

	Negative class weight	# of Active Neurons (+/-)	Training		Validation	
			auROC	auPRC	auROC	auPRC
Net-1	0.1	5/3	97.08	83.7	94.06	77.18
Net-2	0.33	2/3	96.53	83.82	93.86	78
Net-3	1.0	3/4	96.18	85.04	93.43	78.28

It is evident from Table 8 that a small weight for the negative samples increased the auROC while a larger weight of the negative samples decreased the auROC and increased the auPRC. The results on the validations are also consistent with the testing results (Table 10). Furthermore, due to the LASSO penalty on the weights, most neurons became insignificant and only a fraction of them end with non-zero weight and hence making the classification faster and the whole model easier to interpret.

Figure 10 shows the average single base validation data score of the network (Net-2) in the range [-600 to 600] around the known TSS position compared to the average score for negative examples. At every base, the feature vector was extracted using sub-regions as if that base was the TSS. It is clear from the curve that the classifier is able to produce output scores capable of distinguishing positive from negative examples. These scores get significantly higher the closer we get to the true TSS.

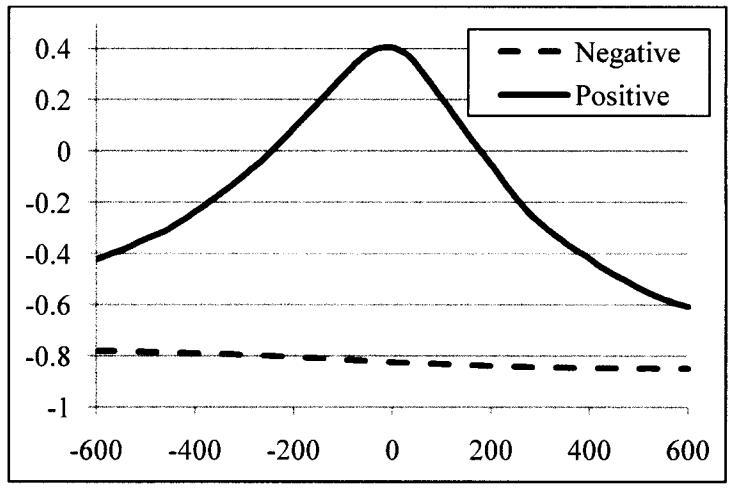

Figure 10: Average Scores at Positions Around the True TSS vs. Average Scores of Negative Examples in Validation Data. The x-axis represents the relative position to the true TSS within the positive examples.

5.3. Testing Procedure

We performed the same testing procedure as described in [76]. Every chromosome strand was divided into non-overlapping chunks of size 50 and 500 bases (Figure 11). Any chunk that falls within 20bp from any known TSS position of any of the testing genes was considered as a positive sample (Figure 11). Any chunk that falls between +20bp downstream of the start of any of these genes to the end of the same gene and was not labeled positive was considered a negative sample (Figure 11). On the other hand, non- ACGT bases (i.e. long N-sleds) were randomly substituted by A, T, C or G. Table 10 lists the number of true positive and true negative samples as a result of the employed chunking and labeling approach.

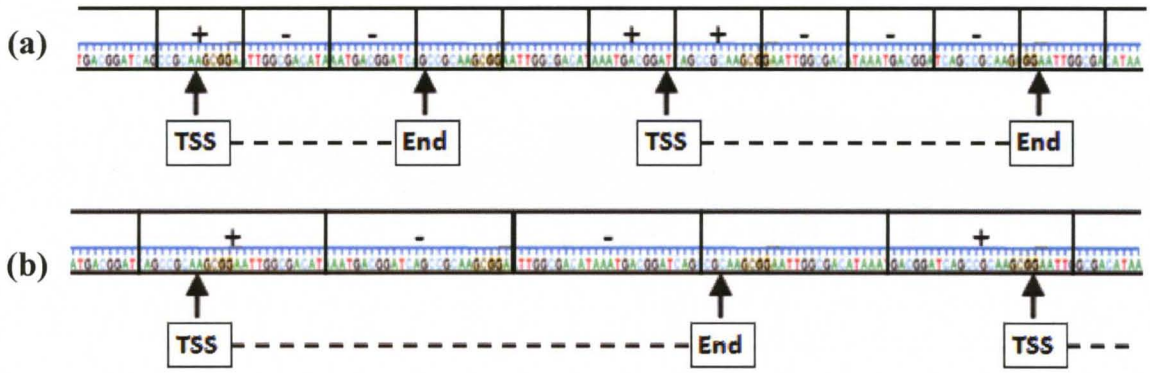

Figure 11: Schematic Diagram of the Chunking and Labeling Approach in the Testing Phase. (a) Chunking the DNA sequence into non-overlapping Sequences of Length 50. (b) Chunking the DNA sequence into non-overlapping sequences of length 500.

Table 9

Number of Positive and Negative Samples as a Result of the Chunking and Labeling Approach Used in Testing.

	True Positive (TP)	True Negative
50	1588	1,087,666
500	943	108,782

For every chunk, a feature vector was extracted at every single base as if that base was the TSS position. A network score is computed at every base and each chunk is assigned the maximum value found for any of the bases contained within it. This may result in chunking and labeling of positive samples despite being up to 20bp away from the true TSS. This design acknowledges the flexibility of POL-II which does not always bind to a specific single base but rather anywhere in the range [-20, +20] from the start of the TSS. Table 10 shows the auROC and auPRC in the testing phase for the three networks.

Table 10

Effect of Training with Weighted Error on the auROC and auPRC of Validation Data.

	Negative weight	Chunk 50			Chunk 500		
		auROC	auPRC	auPRC*	auROC	auPRC	auPRC*
Net-1	0.1	95.09	19.09	43.43	94.95	44.7	57.67
Net-2	0.33	94.83	24.11	49.23	95.01	53.96	66.24
Net-3	1.0	93.61	26.16	50.77	94.66	55.81	66.99

A comparison of these rates is shown in Figures 12 and 13. The true positive rate (TPR) for TSS identification was calculated as the percentage of positive samples identified as such by HyperBF-TSS while the false positive rate was calculated as the percentage of true negative samples mistakenly labeled as positive. The positive predictive value (PPV) is calculated as the ratio of the positive samples whose true label is positive to the total number of samples classified as positive. As illustrated in Figure 5, the area under the precision recall curve is relatively low due to the fact that the ratio of negative to positive samples is very high, and varies widely between the two cases of chunk size of 50 and 500.

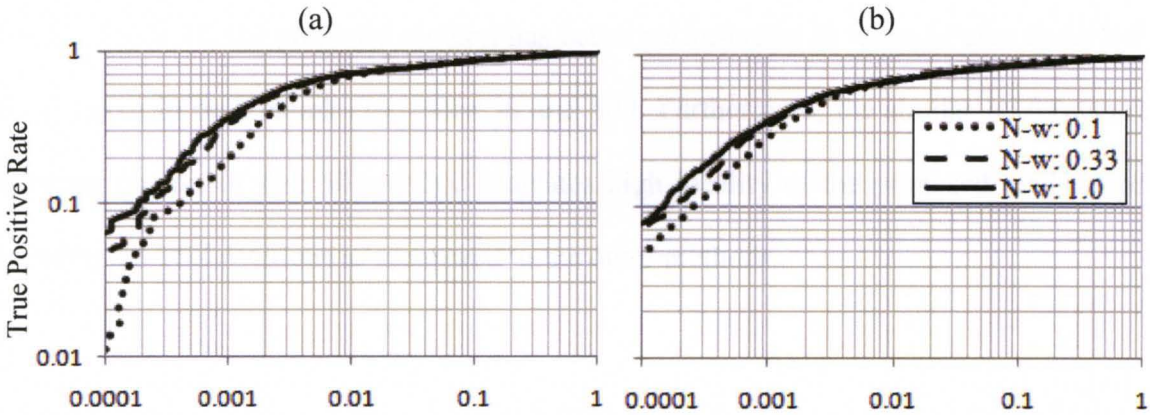

Figure 12: Effect of Weighted Training on Testing auROC in both Chunking Cases: a) Chunk Length = 500. b) Chunk Length = 50. N-w refers to the weight assigned to negative samples (majority class).

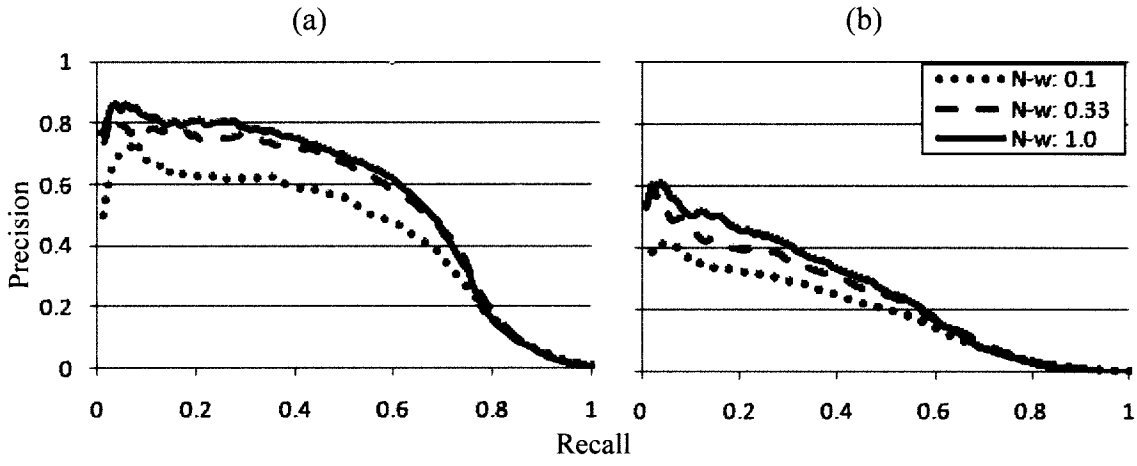

Figure 13: Effect of Weighted Training on Testing auPRC in both Chunking Cases: a) Chunk Length = 500. b) Chunk Length = 50. N-w refers to the weight assigned to negative samples (majority class).

5.4. Comparison to Other Methods

For comparison with other methods, Net-2 is used since it provides a trade-off between auROC and auPRC as compared to the other two networks. The performance of HyperBF-TSS was compared to other methods using both auROC and auPRC measures (Table 11). Note that results for the ARTS, Eponine, McPromoter and FirstEF methods are taken as reported in [76] which employed the same testing procedure used here. As seen in Table 11, the proposed method has better performance in terms of area under the ROC curve in both chunk size cases 50 and 500. Furthermore, the similar performance between chunks of size 50 and 500 indicates high locality of the proposed method for locating the TSS positions as compared to the other methods.

Table 11

auROC and auPRC for HyperBF-TSS (Net-2), ARTS and Others.

Curve	auROC %		auPRC %	
Chunk Size	50	500	50	500
HyperBF-TSS	95.01	94.83	24.11	53.96
ARTS	92.77	93.44	26.18	57.19
Eponine	88.48	91.51	11.79	40.80
McPromoter	92.55	93.59	6.32	24.23
FirstEF	71.29	90.25	6.54	40.89

On the other hand, the proposed method fails to exceed the ARTS method when using area under precision recall curve. This should be of no surprise since it has been analytically shown in [84] that optimizing the area under the ROC curve is not guaranteed to optimize the area under the PRC curve.

The precision recall curve is found to be very sensitive to having few negative samples with high scores with HyperBF-TSS. For example, the removal of 0.001 of the negative samples with the highest network scores results in a big change in the auPRC from 24.44% to 49.23% and 53.96% to 66.24% for chunk sizes of 50 and 500 respectively. In contrast, the change in the auROC was minimal, increasing from 95.01 to 95.05and 94.83% to 94.9% for chunk sizes of 50 and 500 respectively.

Table 12

Effect of Removing The Thousandth Highest Scoring Negative Samples on Both auROC and auPRC in Both Chunking Cases.

	Chunk 50		Chunk 500	
	auROC	auPRC	auROC	auPRC
All Data	95.01	24.11	94.83	53.96
Filtered Data	95.05	49.23	94.9	66.24

The removal of 0.001 of the negative samples illustrates the sensitivity of the auPRC. The use of the auPRC should be considered with caution as an evaluation measure of

TSS finders since the PRC has a demonstrated sensitivity. Labeled negative samples could be unknown TSS, which is shown to potentially have a significant effect on the auPRC.

6. Higher Level Analysis of HyperBF-TSS

A key advantage of the proposed method is that once training the HyperBF network is finished, the set of resulting neurons with positive weights can be perceived as a mixture of Gaussians providing an approximation of the TSS samples probability distribution in the new Euclidean space. Such knowledge can pave the way for a higher level analysis in the time space. For example, having many promoter sequences with high membership to one neuron indicates that they belong to one cluster and hence share many of their of oligonucleotides' frequencies in the same sub-regions. Figure 14 shows the details of the final network structure of Net-2.

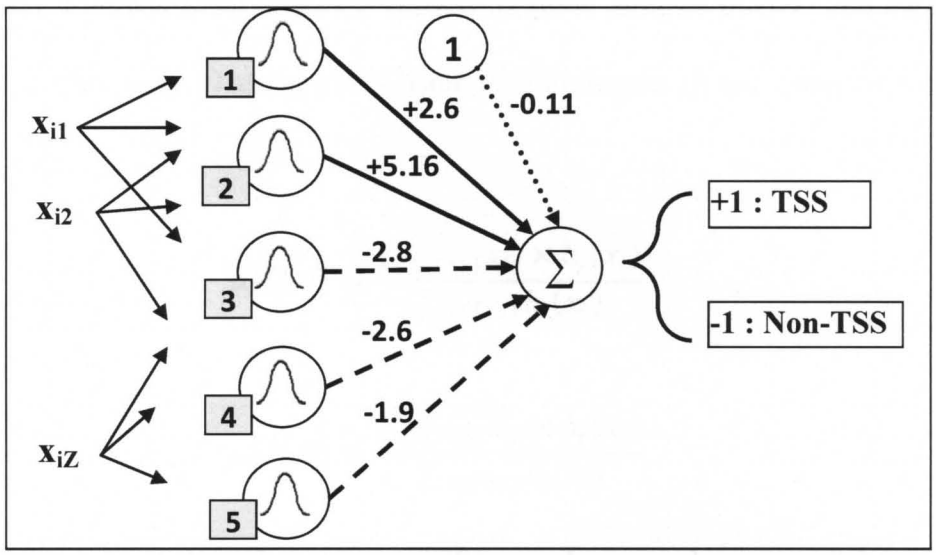

Figure 14: Final Network Structure of Reduced HyperBF-TSS Network (Net-2).

As shown in Figure 14, due to the wieght decay penalty, 25 or the initial 30 neurons ended up with a near zero weight value and hence their removal would have no significant effect on the classification accuracy. Two of these neurons are connected with positive weights to the output. Positive neurons give insight into what are the characteristics of TSS samples while negative neurons give insight into the characters of none-TSS samples. Since the classification accuracy of this network is relatively very high, we can use it as a supportive evidence of the existence of two types of TSS.

6.1. TSS Subtypes Characteristics

By analyzing the active features in each positive neuron, we can deduce the characteristics of the corresponding subtype of TSS samples that makes them different from non-TSS samples. Active features indication which oligonucleotides' positional frequencies are important in discrimination. First, in every neuron, active features are ranked base on how much their removal would increase the training error (see methods in Chapter III). Afterward, the weighted mean of positive samples $\mu_P(j,z)$ and the mean of negative samples $\mu_N(j,z)$ along every feature (z) in neuron (j) are computed separately as:

$$\mu_P(j,z) = \frac{\sum_{i\in\{TSS\}} x_{iz} \times h_j(x_i)}{\sum_{i\in\{TSS\}} h_j(x_i)} \tag{70}$$

$$\mu_N(j,z) = \frac{\sum_{i\notin\{TSS\}} x_{iz} \times h_j(x_i)}{\sum_{i\notin\{TSS\}} h_j(x_i)} \tag{71}$$

If the weighted mean along feature (z) of TSS samples around neuron (j) is higher than that of the negative samples, then it is the relative over-representation of the feature (z)

oligonucleotide in the corresponding region in TSS that make them different from non-TSS. Note that the over/under-representation is a simplified analysis. It is possible that a neuron might fall in the middle of negative samples and hence the over/under-representation labeling is inappropriate. In the later case, the value of the weighted mean of positive samples itself can be used to describe the characteristics of the subtype. However, for simplicity of description in this dissertation, the over/under-representation of oligonucleotides' regional frequencies is used to describe the characteristics of the two subtypes of TSS. Table 13 shows the most important 35 features in positive neurons (N-1 and N-2) from Net-2 and whether they are over-represented or under-represented. Figures 15 and 16 show a simplified description of the characteristics of subtype 1 and 2 respectively. Each figure shows the most important 12 oligonucleotides which are regionally over-represented and the most important 12 oligonucleotides which are regionally under-represented in TSS samples as compared to negative samples.

Table 13

List of the top 40 4-mer characteristics of TSS Sequences for Subtypes N-1 and N-2. Over represented oligonucleotides in TSS samples are denoted by +1 while underrepresented oligonucleotides are denoted by -1.

	N-1 Features Region:	Representation Type	N-2 Features Region:	Representation Type
1	4:GGTA	+1	4:AAAA	-1
2	3:AAAA	-1	3:TTTT	-1
3	4:TAAG	-1	2:TTTT	-1
4	4:AAAA	-1	3:ATGT	-1
5	4:GTAA	-1	3:TAAT	-1
6	4:AAGA	-1	4:TGTA	-1
7	2:ATCC	-1	3:AAAT	-1
8	3:TATT	-1	4:TTTT	-1
9	2:TGAC	-1	4:ATAA	-1
10	3:ATGT	-1	4:ATTA	-1
11	2:GATC	-1	4:CACT	-1
12	3:GAGC	+1	4:CATA	-1
13	4:AGGC	+1	3:TGTT	-1
14	2:CGAC	+1	3:TTTA	-1
15	4:TAGG	-1	3:GCGG	+1
16	3:CGAC	+1	3:ATCT	-1
17	4:ATAA	-1	3:TATT	-1
18	4:AGGT	+1	3:ATAT	-1
19	3:TAGC	-1	4:CGCG	+1
20	3:GTAG	-1	3:CTAC	-1
21	3:ATTA	-1	2:CATG	-1
22	4:GTCC	+1	2:GCTG	-1
23	2:GTGT	-1	3:TACA	-1
24	3:TAGT	-1	3:AATG	-1
25	1:AATA	-1	2:TGCT	-1
26	2:CACC	+1	3:AATT	-1
27	4:TGAG	-1	3:TTGT	-1
28	4:GTGG	+1	3:GTGT	-1
29	1:ATTC	-1	4:TAAT	-1
30	1:AGCA	-1	3:TATC	-1
31	2:ATAT	-1	3:CAAG	-1
32	3:TCCT	-1	4:ATGT	-1
33	3:GGAA	+1	2:TTCC	+1
34	4:CACG	+1	3:GAAC	-1
35	3:GAAG	+1	4:TTTA	-1

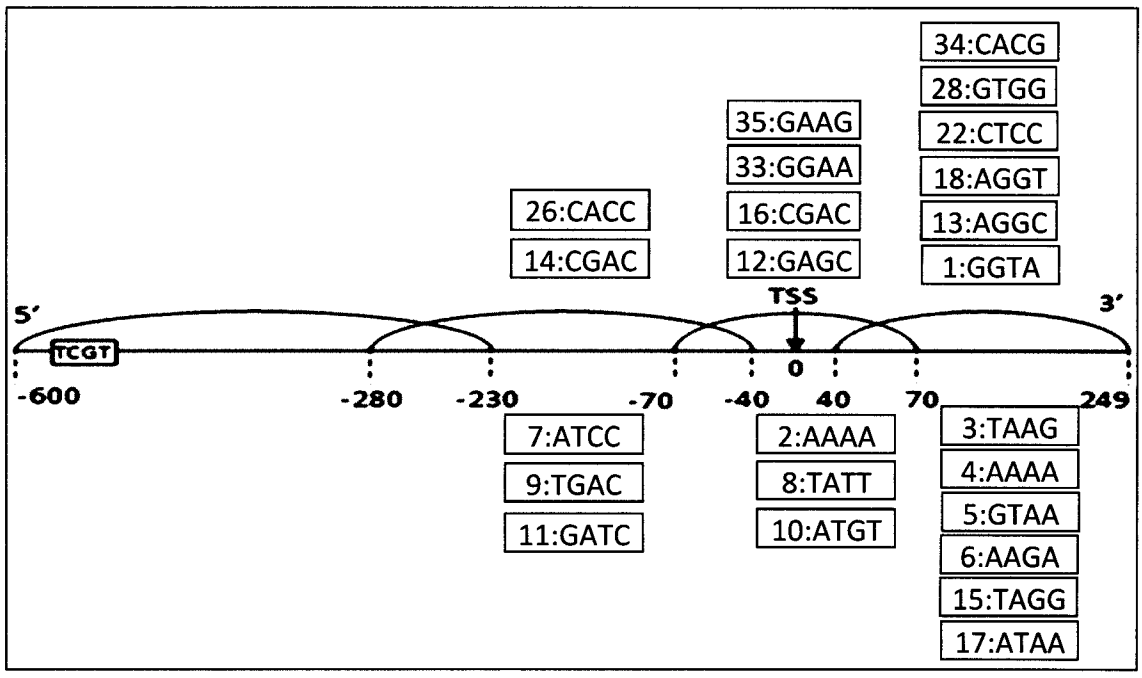

Figure 15: TSS Subtype-1 Characteristics. An oligonucleotide above a region indicates a relative over- representation in that region while an oligonucleotide below a region indicates a relative under-representation in that region. Oligonucleotides are numbered based on their rank of importance.

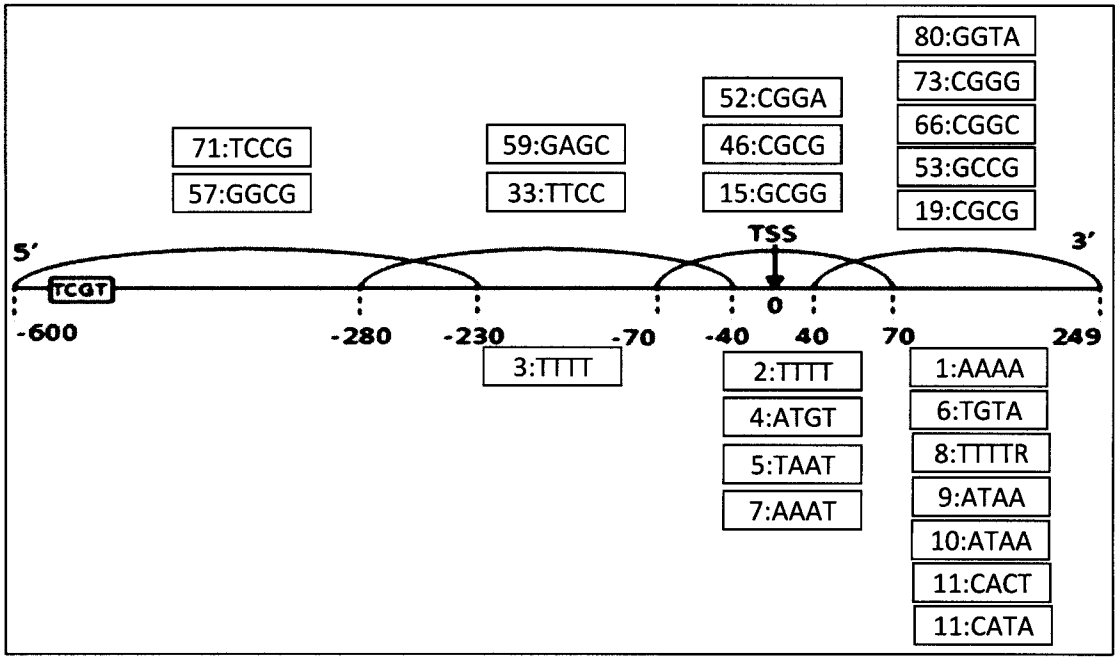

Figure 16: TSS Subtype-2 Characteristics. An oligonucleotide above a region indicates a relative over- representation in that region while an oligonucleotide below a region indicates a relative under-representation in that region. Oligonucleotides are numbered based on their rank of importance.

7. Conclusion

A new novel feature is presented that transforms the problem from sequences and temporal space to Euclidean space. Such a feature makes it possible to cluster promoter sequences and build a HyperBF neural network.

The proposed HyperBF-TSS method has demonstrated high accuracy performance in detecting transcription start sites and proven to be very competitive to the high performing ARTS tool and others. The proposed method achieved an area under the ROC of 95.01% and 94.83% for chunks of size 50 and 500 as compared to 92.77% and 93.44% achieved by the ARTS using the same dataset and testing procedure. The high performance of the proposed method with chunk size of 50 proves that HyperBF-TSS has increased the classification accuracy over previously described TSS prediction algorithms, and performs well with high locality precision.

Finally Reduced HyperBF is shown to be an informative tool for higher level analysis. In that significant neurons are indicators of potential subpopulations that group sample based on discriminative information only. In addition, active local features provide insight into the characteristics of each subpopulation in specific and the whole class in general.

CHAPTER VI

CASE STUDY: FEATURE SELECTION AND SUBTYPE DISCOVERY IN MICROARRAY DATA ANALYSIS

1. Motivation

Microarray technology is used for measuring the expression level of genes under a particular condition by looking at the presence of mRNA tags. The expression of the mRNA is an indicator of the level of abundance of individual proteins in a cell and hence is a determinant of the functionality of the cell. Every healthy functional tissue in an organism is known to occupy a certain stable expression levels of genes with some tolerated variation. Analysis of microarray data provides insightful information about the functionality of genes and their interactions as it relates to conditions such as development, disease, or response to stimuli.

Disturbances in gene expression levels can factor in disease development such as cancer. A comparison of gene expression levels between a group of sick people (condition) and a group of healthy people (control) has the potential to pinpoint genes that may play a role in the development of the disease. Furthermore, clustering methods have the potential to pinpoint different subtypes of the same disease in that, the same phenotype or symptoms might be triggered by completely different mechanisms. Recovering this type of information about a disease may help develop effective drugs in

addition to developing accurate tools for diagnosis and prediction of the susceptibility to future occurrence of the disease.

Due largely to the experimental expenses and in some cases scarcity of volunteers, microarray samples are usually available in small quantities (tens to a few hundred samples) where every sample is represented by a profile of tens of thousands genes expressions. As a result, regular classification and clustering tools are likely to suffer from overfitting and discover patterns that have nothing to do with the disease. With such high dimensionality, feature selection as a preprocessing step is a necessity.

Considering the small number of samples, the high dimensionality and the complexity of interactions and correlations between genes expressions, feature selection in microarray data analysis is one of most challenging variable and feature selection problems [85]. In this chapter, a new feature selection algorithm based on a Reduced HyperBF network is proposed. The proposed algorithm is applied to two microarray datasets and is shown to select a minimal subset of features with high discriminative information. The proposed algorithm is compared to two other algorithms (ReliefF and SVM-RFE) and is shown to have very competitive results. Moreover, since a HyperBF network uses a mixture of Gaussians to represent each class, significant neurons in the final penalized networks can be used as an evidence of multiple subtypes of the disease.

2. Overview of Feature Selection Methods

Feature selection in classification problems is a learning task that aims at finding the smallest possible subset of features that enables accurate classification (preferably the most possible accurate classification). Feature elimination does not necessarily mean a

loss of interesting information. For example in gene expression problems, feature elimination can be justified under two main arguments:

- Irrelevant features: For example a certain gene is not related to the disease and hence information about this gene only adds noise to the learning task.
- Redundant Information: For example, the expression of two genes might be correlated or dependent on each other. As a result, discarding one of them does not involve a loss of information.

Simple feature selection approaches measure the goodness of every feature separately, and select the features with the most discriminative information [54]. More sophisticated techniques aim at evaluating subsets of features and hence measuring the goodness of every feature as it is being used in combination with others. The later approach is more general and is less likely to select features with redundant information. However, a search for an optimal subset of discriminative and non redundant features is computationally expensive. Most feature selection approaches can be classified into three categories [54;85]: filters, wrapper and embedded.

2.1. Filters

Filters are generic feature ranking and selection methods that are typically used in the preprocessing phase before applying a classification algorithm. These methods usually employ a heuristic or a certain information criterion to rank features in an effort to estimate their discrimination power. Examples of such methods are the F-Test measure [86], the Chi-Squared measure [87], the Relief algorithm [88] and its improved version ReliefF [89]. Feature subset ranking and selection filters methods have also been

developed. An example is the CFS algorithm [90]. CFS starts by evaluating features individually. Afterward, features with the highest discrimination power and the least redundancy (correlation) with the already selected features are incrementally selected.

Although filters have not proven to outperform more sophisticated techniques such as wrappers [91], their ease of use and fast computation still make them a practical and efficient option. Furthermore, in high dimensionality problems, using filters as a preprocessing step to applying wrappers or embedded feature selection is a common practice to make computation practical [54].

2.2. Embedded Feature Selection

The common theme between methods in this category is the definition of an object function that combines fitting the training data while penalizing the usage of more features [54]. Fitting training data can be achieved by any loss function such as the squared error while the penalty of using features depends on the structure of the model being trained. Such a function would have to be practical to minimize and not be sensitive to local minimum issues. An example of such methods is the penalized linear regression such as bridge regression [45]:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - Wx_i - b)^2 + \lambda\sum_{z=1}^{Z}|w_z|^{\gamma} \tag{72}$$

In (72), W is a vector of the linear regression coefficients, b is a constant, y_i is the desired output for sample x_i, while λ and γ are two positive constants to be estimated by cross-validation. Optimizing (72) with $0 < \gamma \leq 1$ will drive many coefficients of the

regression to zero making the classification decision invariant to the corresponding features.

2.3. Wrappers

Wrappers are simple but computationally expensive feature selection tools [91]. A wrapper typically performs feature subset selection for a specific classification tool in a direction to either fit the training data with a minimal set of features or to maximize the prediction accuracy of the classification algorithm in a leave-one-out or a cross-validation setting. Wrappers select features through either a forward selection or a backward elimination. In forward selection, an initial set of one or few features is created and the algorithm incrementally adds new features. In a backward elimination, the algorithm starts with all features and iteratively discards those that will not hurt classification. The linear SVM with recursive feature elimination (SVM-RFE) algorithm proposed in [53] is one such example.

3. Methods

3.1. Recursive Feature Elimination in Reduced HyperBF Network (Reduced HyperBF-RFE)

Reduced HyperBF on its own is an embedded feature selection method and once training is finished, all features that do not help in classification are discarded. The localized feature penalty in a HyperBF network is motivated by the real world observation that many patterns are recognized to belong to one class due to the co-

occurrence of specific values along a specific set of features. Furthermore, one class is very likely to be composed of multiple sub- models that are different in their characteristics.

Nonetheless, Reduced HyperBF classification performance and the elimination of features is sensitive to the regularization parameters: λ_w and λ_v. At a certain stage of training and parameter selection, specific values of λ_w and λ_v might be optimal for cross-validation classification accuracy. Once a significant number of features are discarded from the model, it is likely that a different pair of values of λ_w and λ_v are needed for optimal classification.

Furthermore, as discussed in Chapter III, the saliency measure ($score_3$) is a more accurate ranking measure than the magnitude of the local coefficients. Therefore, simultaneous recursive feature elimination based on saliency ranking and a continued re-selection of λ_w and λ_v is anticipated to provide better feature selection.

3.2. Seeded Reduced HyperBF–RFE

Due to the very high dimensionality of the microarray data, an initial feature selection using a filter method is a necessity. Seeding Reduced HyperBF–RFE by a filtered subset is unlikely to downgrade the overall selection as long as the filtered subset of features is relatively large.

Furthermore, the first round of training a Reduced HyperBF network results in an elimination of a large number of features. For example, in the Leukemia dataset, ReliefF was used to rank 7,129 features and only 512 of them were seeded to training a Reduced HyperBF network. The resulting network only used 35 features and the elimination of

477 features caused no increase in the training regression error ($\Delta E < 10^{-7}$). There is no guarantee that such a subset is optimal or unique. And more importantly, there is no guarantee that from this subset we can find a smaller optimal subset that we cannot find in the 477 features set. To ease the sub-optimality problem, if a significant number of features is eliminated by applying Reduced HyperBF–RFE, the next iteration can be seeded with the selected subset from the current Reduced HyperBF–RFE in addition to a limited set of features from the top of the filter ranked list that were not selected by the current Reduced HyperBF. This re-seeding from the filter can only improve the subset selection of Reduced HyperBF–RFE since it provides a bigger pool of features for selection. In addition, the filter ranking of features is only computed one time at the first initialization.

In all experiments reported in this chapter, the re-seeding was only applied after the first round of training Reduced HyperBF mainly because the resulting network eliminated a large percentage of features (more than 90% of the features had 0 local coefficient in every neuron). Algorithm 2 shows the details of the Reduced HyperBF–RFE.

Algorithm 2: Reduced HyperBF–RFE Algorithm

Init

Rank features using a filter method (i.e. ReliefF alg.)

Select a subset of features {S} with the highest rank.

(i.e. 1000 features)

Search for optimal λ_v and λ_w that minimize the CV error.

{S} ← {S} – {all discarded features from resulting network}

If size(S) is very small

{S} ← {S} U {Small set from top of the Filter-Ranked List}

Loop

Set q and Q based on |S| = size(S):

(i.e. Q= |S|/2, q = |S|/10)

Loop

Train a Reduced HyperBF network with {S}, λ_v and λ_w

Rank features based on saliency ($score_3$)

{S} ← {S} – { The q Features with the smallest saliency ($score_3$)}

Until *size(S)* $\leq Q$

Re-Search for optimal λ_v, λ_w that minimize the CV error.

Until *1 feature is remaining*

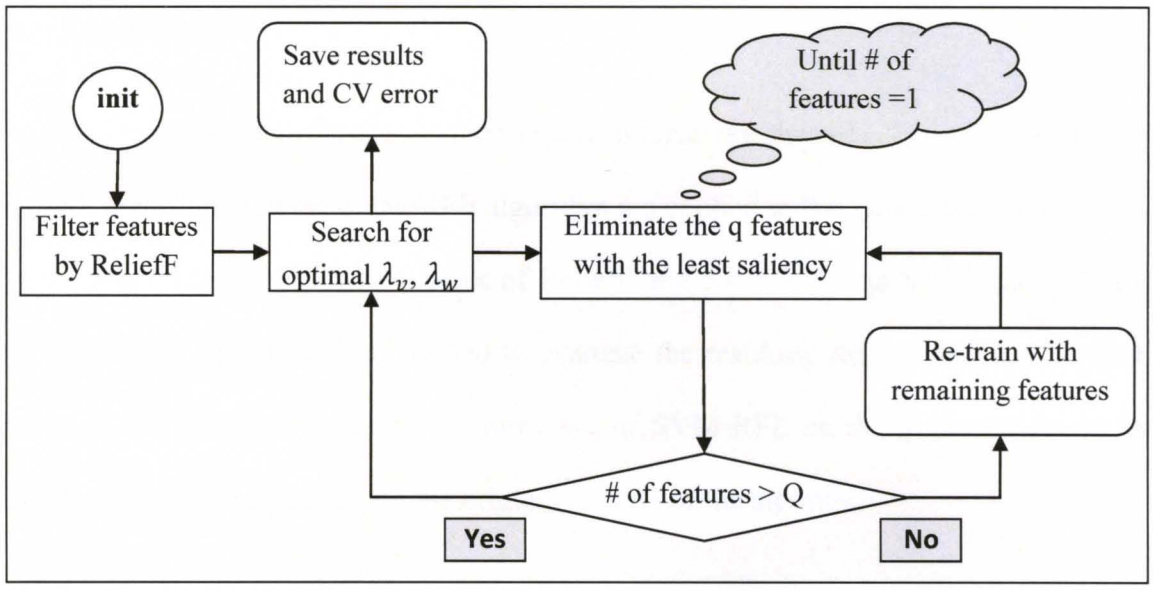

Figure 17: Flowchart of Reduced HyperBF–RFE Algorithm. The re-seeding step is omitted for simplicity of visualization.

3.3. Reduced HyperBF Network for Functional Clustering

There is a striking similarity between RBF/HyperBF networks and density estimation approaches in the case of mixture of Gaussians. Nonetheless, RBF/HyperBF network is not a density estimation method in its classical sense. Density estimation methods or their alternative clustering algorithms aim at grouping samples from one population based on their similarity in order to recover an estimation of the generator models. A significant cluster is thought to be a sub-model or subpopulation of the whole class.

Clustering algorithms are unlikely to be useful for microarray analysis of diseases. This is mainly because samples of the sick group share a high similarity along so many features that are not related to the disease. In contrast, HyperBF networks only use the characteristics that make the two classes different and therefore results in clusters of interest in studying the disease.

4. Experiments

The proposed algorithm is applied to two microarray datasets. For comparison the ReliefF algorithm and the SVM-RFE algorithm are applied to the same datasets using the Weka data mining tool. In both cases of ReliefF and SVM-RFE, an SVM classification network using Libsvm [11] is trained to evaluate the resulting feature selection of both methods. On the other hand, the performance of SVM-RFE on the leukemia dataset is listed as it is reported in [53] by the original authors of the algorithm.

4.1. Datasets

a) ICMLA 2009 Cancer Dataset

This dataset contains microarray scans taken from samples belonging to three different types of cancer: breast, colon, and lung. That dataset is available at (http://www.icmla-conference.org/icmla09/) and is divided into 400 training samples and 250 testing samples (Table 14). Every sample is represented by 54,613 probes (gene and other DNA transcript expressions).

Table 14

Number of Samples in Every Class in the ICMLA 2009 Cancer Dataset

	Breast	Colon	Lung
# of Training Samples	200	130	70
# of Testing Samples	100	100	50

b) Leukemia Dataset

This dataset contains microarray scans of 47 patients with acute lymphoblastic leukemia (ALL) and 25 patients with acute myeloid leukemia (AML). Every sample is represented by 7,129 probes (gene expressions and other DNA transcript expressions). The samples are divided into training dataset of 38 samples and a testing dataset of 34 samples (Table 15).

Table 15

Number of Sample in Every Class in the Leukemia Dataset

	ALL	AML
# of Training Samples	27	11
# of Testing Samples	20	14

4.2. Results

a) ICMLA 2009 Cancer Dataset

Table 16 and Figure 18 list the classification accuracy of the three algorithms (Reduced HyperBF-RFE, SVM-RFE, and ReliefF) on the ICMLA-09 cancer dataset. Classification accuracy is reported on the testing dataset and the cross-validation in the training dataset. For efficiency of computation, SVM-RFE was seeded by 5,000 features selected by ReliefF while Reduced HyperBF-RFE was initially seeded by 800 features selected by ReliefF. At 800 features, 5-fold cross-validation was performed for regularization parameters selection for both Reduced HyperBF and SVM. In the remainder of the cases, 20-fold cross-validation was used.

Table 16

Cross-validation and Testing Classification Accuracy of the Three Methods at Different Levels of Feature Selection on ICMLA-09 Cancer Dataset. Best rates in each iteration are face-bolded.

# of Features	Reduced HyperBF-RFE		SVM - SVM-RFE		SVM - Relief	
	CV	Test	CV	Test	CV	Test
800	92.75	**93.6**	**97.75**	**93.6**	93	90
100	93.75	**93.6**	**98.75**	92	91.5	89.6
60	93.75	94	**97.75**	**95.6**	91.25	90
40	94.75	**94.8**	**96.5**	93.2	90	91
30	94.25	**95.6**	**96.25**	93.2	90	91
20	94.75	**94.8**	**96.75**	**94.8**	89.75	90.75
10	**94.25**	**94.4**	94	94	87.75	84
5	**94.25**	**95.6**	93.5	93.6	85.25	86.4
3	92.75	**94.4**	**93.5**	93.75	78.75	76
1	78	76	77.75	75.2	**79.75**	**78.8**

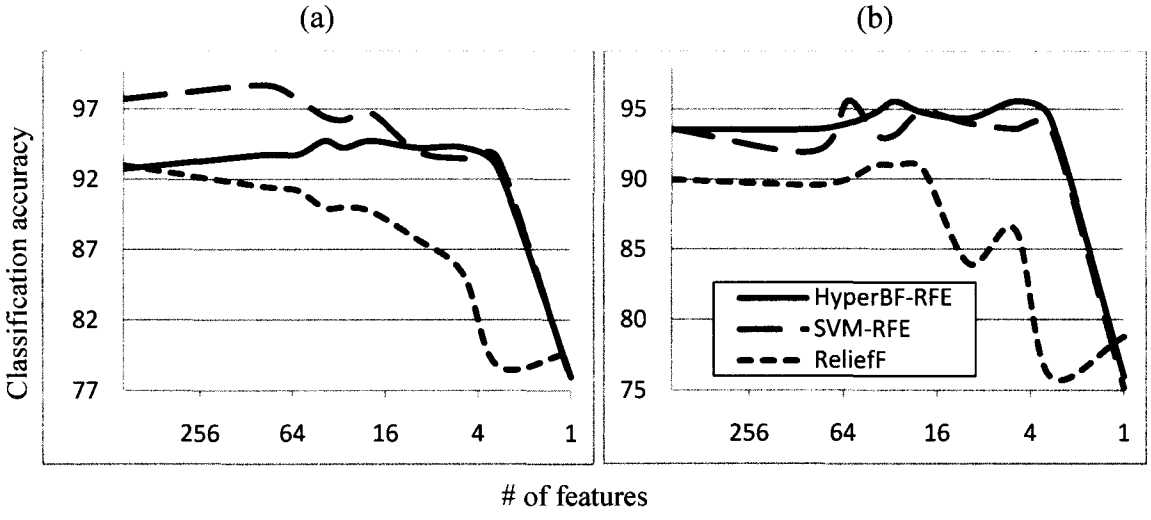

Figure 18: Classification Accuracy on the ICMLA 2009 Cancer Dataset of the Three Methods. (a) Leave-one-out cross-validation and (b) Test data.

From Table 16 and Figure 18, both algorithms Reduced HyperBF-RFE and SVM-RFE consistently outperformed the ReliefF filter algorithm in terms of the cross-validation and the testing classification error. On the other hand, SVM-RFE resulted in the best cross-validation accuracy in general. Nonetheless, Reduced HyperBF-RFE resulted in slightly better results with the testing dataset. Moreover, at a very low number of features (10, 5, 3), Reduced HyperBF-RFE has a slight upper edge over SVM-RFE with both the cross-validation and testing dataset.

b) Leukemia Dataset

Table 17 and Figure 19 list the classification accuracy of the three algorithms (Reduced HyperBF-RFE, SVM-RFE, and ReliefF) on the Leukemia dataset. Due to the very small number of training samples (38), a leave-one-out is performed for parameter selection for both Reduced HyperBF and SVM (38-fold cross-validation). Classification accuracy is reported on the testing dataset and the cross-validation in the training dataset.

For efficiency of computation, features were initially filtered to 512 features by ReliefF. Classification results for SVM-RFE are listed as reported by the original authors of the algorithm.

Table 17

Cross-Validation and Testing Classification Accuracy of the Three Methods at Different Levels of Feature Selection on the Leukemia Dataset. Best rates in each iteration are face-bolded.

# of Features	Reduced HyperBF-		SVM - SVM-RFE		SVM - Relief	
	CV	Test	CV	Test	CV	Test
512	**100**	92.10526	97	88	**100**	85.29
64	97.36842	**97.36842**	**100**	94	94.73	91.176
16	**100**	97.36842	**100**	**100**	97.36	97.06
8	**100**	97.36842	**100**	**100**	**100**	94.11
4	**100**	**97.36842**	97	91	**100**	97.06
2	97.36842	94.73684	97	88	94.73	**97.05**
1	92.10526	**92.10526**	92	79	**94.73**	79.411

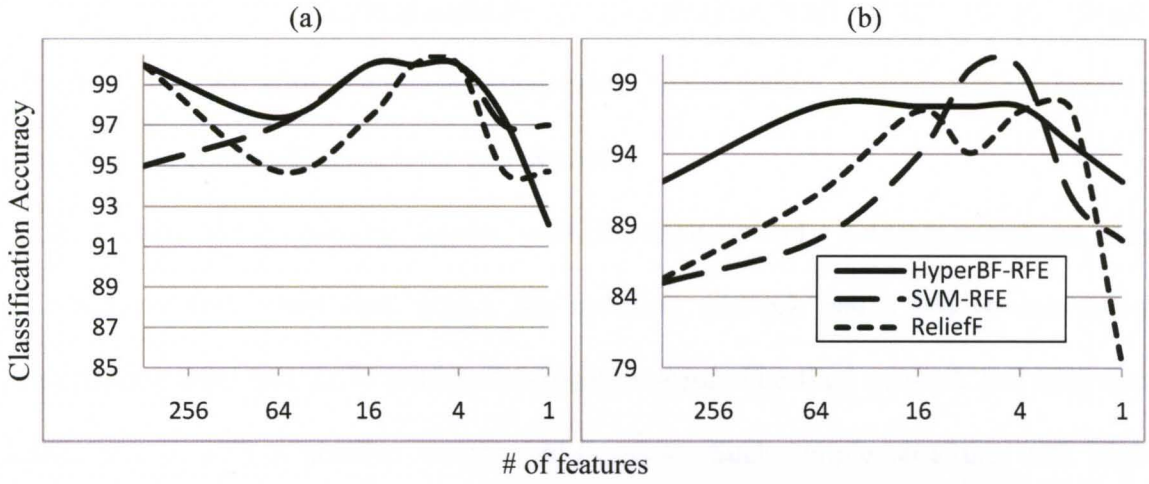

Figure 19: Classification Accuracy on the Leukemia Dataset of the Three Methods. (a) leave-one-out cross-validation and (b) test data.

The result on the leukemia dataset shows Reduced HyperBF-RFE to produce by a slight margin the best results on the leave-one-out validation data. In the testing set, Reduced HyperBF always failed to classify one of the samples correctly which made it

appear less accurate than SVM-RFE. Nonetheless, the classification results of Reduced HyperBF-RFE are more stable and oscillate less than the other two methods.

5. Higher Level Analysis and Discussion

Once a minimal set of genes are indentified and proven to be sufficient for satisfactory classification, a biologist can use this list of genes for further analysis to recover the true causes of the disease. Furthermore, by using the Reduced HyperBF network the significant neurons can be studied for potential subtypes of the disease.

Unfortunately, in the two datasets studied in this chapter, there is no control group. Thus, all classes contain only patients with some form of cancer. Therefore, the role of an important gene in classification is ambiguous. For example in the leukemia case an important gene for classification could be involved in either AML or ALL and we cannot tell which one for sure. Moreover, supervised discriminative analysis cannot recover genes that are common between the two diseases.

In the case of the leukemia dataset, with four features and a network initialized with four neurons (two from each class), the resulting network had a 0% leave-one-out classification error and 2.6% testing classification error. The final network had only one active neuron with a positive weight (ALL class). Such simple structure with high accuracy is an indicator that there are no potential ALL subtypes. Table 18 lists the four features used by the final network for classification.

Table 18

The Four Probes Sufficient for Accurate Classification in Leukemia Dataset.

Probe	Gene Symbol	Chromosomal Position	Description
M29932_s_at	ADRB3	8p12	Adrenergic, Beta 3-, Receptor
M23114_at	ATP2A2	12q23-q24.1	ATPAse, CA++, Transporting, Cardiac Muscle, Slow Twitch 2
Y12478_at	NRL	14q11.2	Neural Retina Leucine Zipper
M95925_at	WRB	21q22.3	Tryptophan Rich Basic Protein

For further analysis, each of the resulting probe identifiers were converted to gene symbols using the DAVID gene ID conversion tool (http://david.abcc.ncifcrf.gov/) [92;93]. Subsequently, we used the gene symbols as inputs into the NCBI's Online Mendelian Inheritance in Man (OMIM) database [94] to search for known relationships to leukemia, specifically acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL). At first glance, none of the four features show a relationship to ALL or AML. However, both of these diseases are known to be affected by chromosomal translocations [95-97]. As it turns out, three of these features occur in regions affected by these translocations, including the ADRB gene which is close to WHSC1L1 (NSD3) [98], known to be involved in gene fusion events associated with AML; NRL, which is close to the CEBPE gene associated with ALL [97]; and WRB that lies in an AML familial leukemia region close to the AML1 oncogene [95]. The remaining factor, ATP2A2 was shown by Golub et al [99] to be one of the 50 genes most highly correlated with the ALL-AML class distinction with high expression levels in ALL and low expression in AML. As a result, the four features selected fit into the story of AML and ALL leukemia, albeit by an indirect mechanism.

In the case of the ICMLA-09 cancer dataset, with five features and a network initialized with fifteen neurons (five from each class), the resulting network had a 5.75% 20-fold cross-validation classification error and 4.4% testing classification error. The final network contains five active neurons only (saliency > 2). Table 20 lists the significant weights (|w|>0.02) connecting important neurons to the output layer while Table 19 lists the importance of every neuron in terms of saliency (increase of error at the output if the neuron is removed).

Table 19

Significant Neurons of the Final ICMLA-09 Cancer Reduced HyperBF Network: Saliency of every neuron is listed for every output (only high saliencies are listed: S > 1)

Neruron	breast	colon	lung
2	85.8	14.9	-
6	2.3	-	-
7	1.1	1.8	-
8	-	4.9	14.2
12	-	16.7	-

Table 20

Final Structure of the ICMLA-09 Cancer Reduced HyperBF Network: Only significant weights connecting significant neurons to every output are listed (|w|>0.02).

Neurons	breast	colon	lung
2	1.0	-0.4	-
6	-0.6	0.2	0.1
7	-0.3	0.4	-
8	-	0.4	-0.7
12	-	-0.6	-

Neurons that connect to the output of breast cancer with a positive weight are representative of what distinctively describe breast cancer samples whereas negative neurons describe what is not a breast cancer sample. Therefore, Tables 20 and 19 indicate one type of breast cancer, two types of lung cancer and three types of colon cancer.

The previous analysis though can be considered by a biologist, it is not very accurate and the network is somewhat ambiguous. For example, neuron 6 is shared between colon and lung cancer output. Furthermore, the resulting network does not take in consideration all the genes that are active in the development of the three diseases. Table 21 lists the five genes active in the final network and some of the known information about them.

Table 21

The Four Probes Sufficient for Accurate Classification in ICMLA-09 Cancer Dataset.

Probe	Gene Symbol	Description
210302_s_at	MAB21L2	MAB-21-Like 2
230772_at	EST Sequence	N/A
209810_at	SFTPB	Surfactant, Pulmonary-Associated Protein B
209604_s_at	GATA3	GATA Binding Protein 3
209708_at	MOXD1	Monooxygenase, DBH-Like 1

For further analysis, each of the probe identifiers were converted to gene symbols using the David gene ID conversion tool as with the AML-ALL data. Gene symbols were used as inputs into OMIM as a first pass. One of the features, 209810_at, fits perfectly into this dataset as it is associated with coating the lungs as a pulmonary surfactant protein. Further analysis of 209810_at shows that this protein is often associated with respitory distress and/or injury [100], indicating it can be affected by lung cancer. Feature 209604_s_at has been shown to be involved in invasive breast carcinomas as well as hormone-dependent breast cancer [101;102]. The remaining three probes were examined for EST expression profiles using NCBI's Unigene [81]. Of these, 210302_s_at shows a high expression in neuroectodermal tumor tissues, but its relationship to lung, breast and colon cancer is unclear. Feature 209708_at shows a high expression in lung tissue. Thus, it may be associated with lung cancer. An alternative is that since there are not any

controls, this simply differentiates lung tissue from breast and colon tissue. The remaining sequence, 230772_at does not belong to a known gene, but is rather an expressed sequence tag (EST) that comes from a protein's precursor. This EST was originally sequenced from a colon tumor, and thus is likely to have a relationship with that group since its expression profile is otherwise low.

6. CONCLUSION

Compared to the other two algorithms (ReliefF and SVM-RFE), Reduced HyperBF-RFE results in a consistent and stable classification accuracy. On two microarray datasets, the Reduced HyperBF-RFE significantly outperforms the ReliefF algorithm while being very competitive with SVM-RFE. Furthermore, the Reduced HyperBF-RFE yields in more stable results than SVM-RFE. For example, the classification accuracy for Reduced HyperBF oscillates less than SVM between the different iterations of feature elimination. This can be mainly due to the fact that Reduced HyperBF has embedded soft feature selection and always makes the solution invariant to many of the features it uses for training.

A small subset of features generally facilitates the work of a biologist or a medical investigator to discover the factors of a certain disease. However, it can be more informative to have a bigger set of genes that are correlated with the disease for novel discovery of subtle mechanisms at work. One way to obtain a bigger set of genes is to stop the feature elimination in the early iterations of Reduced HyperBF-RFE. An alternative approach is to eliminate the first minimal set of features and restart the search for another set of genes and so forth.

In both of the datasets, Reduced HyperBF-RFE selected a small subset of features with high discriminative power. This minimum feature set can be useful in assembling an accurate diagnostic tool. Based on further research in public databases, these features show meaningful associations at least as far as the presented data is concerned. One caveat is that the dataset does not include any normal controls, making the model somewhat ambiguous to interpret. In addition, it is difficult to rule out other associations and detection of features shared between types of cancers.

CHAPTER VII

SUMMARY AND CONCLUSIONS

1. Dissertation Summary

By employing a multi-scaled Gaussian activation function, HyperBF networks possess a great capacity to learn complex decision boundaries using a small network structure. The localized weighted distance in a HyperBF network can uncover information about the local variation in scaling and the discriminative power of every feature. Nonetheless, a HyperBF network model suffers multiple problems. First, estimating such a complex model is a computationally challenging optimization problem. Second, the high capacity of the network is coupled with a tendency to overfit training data. Third, the optimization problem is non-convex and usually the solution converges to a local minimum.

In this work, two of these problems were addressed. A new regularization method that performs soft local dimensionality reduction in addition to weight decay is proposed and evaluated. In all experiments reported, Reduced HyperBF networks are shown to provide classification accuracy that is competitive to SVM networks while requiring a significantly smaller network structure. Furthermore, the resulting compact network representation gives a tool for higher level analysis. Significant centers can be argued to

be centers of subpopulations of the class while the locally significant dimensions provide information about the unique properties of the subpopulation.

On the other hand, the proposed iSRprop optimization algorithm (scaled Rprop with localized partial backtracking) results in a smooth training that is less likely to diverge than regular Rprop algorithms. In all experiments presented in this dissertation, the proposed training algorithm did not diverge a single time. Furthermore, the proposed scaling and the local partial backtracking in Rprop training are general concepts and can be applied to Rprop with necessary modifications to different optimization problems.

Another contribution in this dissertation is the development of a new feature selection algorithm: Reduced HyperBF-RFE. Reduced HyperBF-RFE iteratively eliminates features that have the least effect on the training error of the Reduced HyperBF network. It is motivated by the localized dimensionality reduction in Reduced HyperBF networks which takes in consideration the localized discriminative power of each feature in addition to the co-occurrence of specific values along a specific set of features. Based on experimental results, the proposed Reduced HyperBF-RFE algorithm is shown to be an effective tool in selecting a minimal subset of discriminative features in microarray analysis. Reduced HyperBF-RF is also anticipated to have similar performance on other datasets of high dimensionality such as text document categorization.

In two different case studies of bioinformatics applications: Transcriptions Start Site (TSS) detection in human DNA and microarray analysis, Reduced HyperBF is shown to be useful for both accurate classification and higher level analysis. The resulting recognition tools were shown to either outperform other existing methods or result in very competitive results.

2. Future Research Directions

In spite of the significance of the contributions presented in this dissertation, many questions remain unsolved and need to be handled in future research. Methods to ease or avoid the local minimum problem such as simulated annealing [103] need to be investigated. Also, determining the optimal number of neurons remains an unsolved question. One possible solution is to start the network with a large number of neurons and use the weight decay to eliminate most of them so that the cross-validation error is minimal.

Moreover, training a Reduced HyperBF network demands a search for regularization parameters which is computationally demanding and time consuming. Future research should investigate the development of regularization path finding algorithms [45] which should cut the time for such a search significantly.

Furthermore, all methods and experiments in this dissertation ignore the importance of the non-diagonal elements of the scaling matrices. As a result, the localized correlation between features is not taken in consideration. This is mainly due to the prohibitive memory and computation time needed to handle such an optimization problem. Nonetheless, full scaling matrices for low dimensionality problems can be practical. In addition, a selective approach to non-diagonal elements of scaling matrices can also be practical. Future research should investigate the two lateral cases.

REFERENCES

[1] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, *Machine learning: an artificial intelligence approach.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc, 1986.

[2] V. N. Vapnik, "An overview of statistical learning theory," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 988-999, Sept.1999.

[3] D. Freedman, *Statistical Models: Theory and Practice*. New York, NY : Cambridge University Press, 2005.

[4] T. Poggio and F. Girosi, "Networks for approximation and learning," in *Proceedings of the IEEE* 1990, pp. 1481-1497.

[5] T. Poggio and F. Girosi, "A theory of networks for approximation and learning," *A. I. Memo 1140, MIT*, July1989.

[6] A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill Posed Problems*. New York: Wiley, 1977.

[7] M. Bertero, "Regularization methods for linear inverse problems," in *Inverse Problems* New York: Springer Berlin / Heidelberg, 1986, pp. 52-112.

[8] V. V. Ivanov, *The Theory of Approximate Methods and Their Application to the Numerical Solution of Singular Integral Equations*. Leyden, The Netherlands: Noordhoff International Publishing, 1976.

[9] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *5th Annual ACM Workshop on COLT* ACM Press, 1992, pp. 144-152.

[10] C. Cortes and V. N. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273-297, Sept.1995.

[11] C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines," Technical report,2001.

[12] M. Riedmiller and H. Braun, "Rprop - description and implementation details," in *Technical report* 1994.

[13] D. S. Broomhead and D. Lowe, "Radial basis functions, multi-variable functional interpolation and adaptive networks," Technical report, Royal Signals and Radar Establishment Malvern, England,Mar.1988.

[14] J.Park and I.W.Sandberg, "Universal approximation using radial-basis-function networks," *Neural Computation*, vol. 3, no. 2, pp. 246-257, 1991.

[15] H. Guang-Bin, P. Saratchandran, and N. Sundararajan, "A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation," *IEEE Transactions on Neural Networks*, vol. 16, no. 1, pp. 57-67, 2005.

[16] M. J. L. Orr, "Introduction to radial basis function networks," Technical report, Centre for Cognitive Science, University of Edinburg,Apr.1996.

[17] J. E. Meng, W. Shiqian, and L. Juwei, "Face recognition using radial basis function (RBF) neural networks," *IEEE Transactions on Neural Networks*, vol. 13, no. 3, pp. 697-710, May2002.

[18] X. Hong and C. J. Harris, "Experimental design and model construction algorithms for radial basis function networks," *International Journal of Systems Science*, vol. 34, no. 14-15, pp. 733-745, Nov.2003.

[19] J. D. Powell, *Radial basis function approximations to polynomials* Longman Publishing Group, 1988, pp. 223-241.

[20] A. R. Webb and S. Shannon, "Shape-adaptive radial basis functions," *IEEE Transactions on Neural Networks*, vol. 9, no. 6, pp. 1155-1166, Nov.1998.

[21] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural Computation*, vol. 7, no. 2, pp. 219-269, Mar.1995.

[22] F. Schwenker, H. A. Kestler, and D. S. Prestridge, "Three learning phases for radial-basis-function networks," *Neural Networks*, vol. 14, no. 4-5, pp. 439-458, May2001.

[23] T.-F. Wu, CJ. Lin, and R.-C. Weng, "Probability estimates for multi-class classification by pairwise coupling," *Journal of Machine Learning Research*, vol. 5, pp. 975-1005, Dec.2004.

[24] D Price, S Kneer, L Personnaz, and G Dreyfus, "Pairwise neural network classifiers with probabilistic outputs," in *Neural Information Proceedings Systems* 1994.

[25] N. B. Karayiannis and G. W. Q. Mi, "Growing radial basis neural networks: Merging supervised and unsupervised learning with network growth techniques," *IEEE Transactions on Neural Networks*, vol. 8, no. 6, pp. 1492-1506, Nov.1997.

[38] T. Evgeniou, M. Pontil, and T. Poggio, "Regularization networks and support vector machines," *Advances in Computational Mathematics*, vol. 13, no. 1, pp. 1-50, 2000.

[39] V. N. Vapnik and A. Y. Chervone, "Uniform convergence of relative frequencies of events to their probabilities," *Theory of Probility and Its Applications,Ussr*, vol. 16, no. 2, pp. 264-280, Jan.1971.

[40] V. N. Vapnik, *The Nature of Statistical Learning Theory*, 2nd ed. New York: Springer, 2000.

[41] V. N. Vapnik and A. Y. Chernove, "The necessary and sufficient conditions for consistency in the empirical risk minimization method," *Pattern Recognition and Image Analysis*, pp. 283-305, 1991.

[42] Z. Chen and S. Haykin, "On different facets of regularization theory," *Neural Computation*, vol. 14, no. 12, pp. 2791-2846, Dec.2002.

[43] D. McAllester, "Simplified PAC-Bayesian margin bounds," *Learning Theory and Kernel Machines*, vol. 2777, pp. 203-215, 2003.

[44] P. L. Bartlett, "For valid generalization, the size of the weights is more important than the size of the network," in *Neural Information Processing Systems 9* MIT Press, 1997.

[45] J. H. Friedman, "Fast sparse regression and classification," Technical report, Department of Statistics, Stanford University,2008.

[46] J. H. Friedman, "An overview of predictive learning and function approximation," in *From Statistics to Neural Networks. Theory and Pattern Recognition Applications* Berlin: Springer, 1994, pp. 1-61.

[47] J. E. Moody, S. J. Hanson, A. Krogh, and J. A. Hertz, "A simple weight decay can improve generalization," *Advances in Neural Information Processing Systems*, pp. 950-957, 1995.

[48] O. D. Richard, P. E. Hart, and G. S. David, *Pattern Classification*, 2 ed Wiley-Interscience, 2000.

[49] G. Wahba, *Spline Models for Observational Data*. Philadelphia: SIAM, 1990.

[50] LeChun, Y. and Cortes, C., "MNIST dataset,", http://yann.lecun.com/exdb/mnist/

[51] Y. LeCun, J. Denker, S. Solla, R. E. Howard, and L. D. Jackel, "Optimal brain damage," in *Advances in Neural Information Processing Systems* Morgan Kauffman, 1990, pp. 598-605.

[52] A. Krogh, "A quantitative study of pruning by optimal brain damage," *International Journal of Neural Systems*, vol. 2, no. 4, pp. 159-171, 1993.

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

[54] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, pp. 1157-1182, Mar.2003.

[55] B. Hassibi and D. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Advances in Neural Information Processing Systems 5* Morgan Kaufmann, 1993, pp. 164-171.

[56] Le, Y. n, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, Hubbard W., and L. W. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541-551, 1989.

[57] B. Scholkopf, K. K. Sung, C. J. C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. N. Vapnik, "Comparing support vector machines with Gaussian kernels to radial basis function classifiers," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2758-2765, Nov.1997.

[58] F. Mark and C. Ronald, "Spoken letter recognition," in *Proceedings of the 1990 conference on Advances in neural information processing systems 3* Denver, Colorado, United States: Morgan Kaufmann Publishers Inc., 1990, pp. 220-226.

[59] W. N. Street, W. H. Wolberg, and O. L. Mangasarian, "Nuclear feature extraction for breast tumor diagnosis," in *Biomedical Image Processing and Biomedical Visualization* 1993, pp. 861-870.

[60] "Libsvm Datasets,", http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

[61] J. Y. Wang, "Application of support vector machines in bioinformatics." Dissertation, National Taiwan University, 2002.

[62] B. Rost and C. Sander, "Prediction of protein secondary structure at better than 70% accuracy," *Journal of Molecular Biology*, pp. 232-584, 1993.

[63] C. W. Hsu, C. C. Chang, and C. J. Lin, "A practical guide to support vector classification," Technical report, Walden University,2010.

[64] Y. Suzuki, T. Tsunoda, J. Sese, H. Taira, J. Mizushima-Sugano, H. Hata, T. Ota, T. Isogai, T. Tanaka, Y. Nakamura, A. Suyama, Y. Sakaki, S. Morishita, K. Okubo, and S. Sugano, "Identification and characterization of the potential promoter regions of 1031 kinds of human genes," *Genome Res*, vol. 11, no. 5, pp. 677-684, May2001.

[65] V. B. Bajic, V. Choudhary, and C. K. Hock, "Content analysis of the core promoter region of human genes," *In Silico Biology*, vol. 4, no. 2, pp. 109-125, Nov.2004.

[66] Y. V. Kondrakhin, A. E. Kel, N. A. Kolchanov, A. G. Romashchenko, and L. Milanesi, "Eukaryotic promoter recognition by binding sites for transcription factors," *Comput Appl Biosci.*, vol. 11, no. 5, pp. 477-488, Oct.1995.

[67] D. S. Prestridge, "Predicting Pol II promoter sequences using transcription factor binding sites," *J Mol Biol*, vol. 249, no. 5, pp. 923-932, June1995.

[68] T. Werner, "Models for prediction and recognition of eukaryotic promoters," *Mammalian Genome*, vol. 10, no. 2, pp. 168-175, Feb.1999.

[69] O. V. Kel-Margoulis, A. E. Kel, I. Reuter, I. V. Deineko, and E. Wingender, "TRANSCompel: A database on composite regulatory elements in eukaryotic genes," *Nucleic Acids Research*, vol. 30, no. 1, pp. 332-334, Jan.2002.

[70] T. A. Down and T. J. Hubbard, "Computational detection and location of transcription start sites in mammalian genomic DNA," *Genome Research*, vol. 12, no. 3, pp. 458-461, Mar.2002.

[71] V. Narang, W. K. Sung, and A. Mittal, "Computational modeling of oligonucleotide positional densities for human promoter prediction," *Artificial Intelligence in Medicine*, vol. 35, no. 1-2, pp. 107-119, Sept.2005.

[72] V. B. Bajic, S. L. Tan, Y. Suzuki, and Sugano S., "Promoter prediction analysis on the whole human genome," *Nature biotechnology*, vol. 22, no. 11, pp. 1467-1473, Nov.2004.

[73] U. Ohler, "Promoter prediction on a genomic scale--the Adh experience," *Genome Res*, vol. 10, no. 4, pp. 539-542, Apr.2000.

[74] R. V. Davuluri, I. Grosse, and M. Q. Zhang, "Computational identification of promoters and first exons in the human genome," *Nature Genetics*, vol. 29, no. 4, pp. 412-417, Dec.2001.

[75] V. B. Bajic and S. H. Seah, "Dragon gene start finder: an advanced system for finding approximate locations of the start of gene transcriptional units," *Genome Res*, vol. 13, no. 8, pp. 1923-1929, Aug.2003.

[76] S. Sonnenburg, A. Zien, and G. Ratsch, "ARTS: Accurate recognition of transcription starts in human," *Bioinformatics*, vol. 22, no. 14, p. e472-e480, July2006.

[77] V. Narang, W. K. Sung, and A. Mittal, "Computational modeling of oligonucleotide positional densities for human promoter prediction," *Artif Intell Med*, vol. 35, no. 1-2, pp. 107-119, Sept.2005.

[78] X. Fu, L. Wang, K. S. Chua, and F. Chu, "Training RBF neural networks on unbalanced data," in *Proceedings of the 9th International Conference on Neural Information Processing, ICONIP '02,* 2 ed 2002, pp. 1016-1020.

[79] Y. Suzuki, R. Yamashita, K. Nakai, and S. Sugano, "DBTSS: DataBase of human Transcriptional Start Sites and full-length cDNAs," *Nucleic Acids Res*, vol. 30, no. 1, pp. 328-331, Jan.2002.

[80] D. Karolchik, R. M. Kuhn, R. Baertsch, G. P. Barber, H. Clawson, M. Diekhans, B. Giardine, R. A. Harte, A. S. Hinrichs, F. Hsu, K. M. Kober, W. Miller, J. S. Pedersen, A. Pohl, B. J. Raney, B. Rhead, K. R. Rosenbloom, K. E. Smith, M. Stanke, A. Thakkapallayil, H. Trumbower, T. Wang, A. S. Zweig, D. Haussler, and W. J. Kent, "The UCSC Genome Browser Database: 2008 update," *Nucleic Acids Res*, vol. 36, no. Database issue, p. D773-D779, Jan.2008.

[81] D. L. Wheeler, T. Barrett, D. A. Benson, S. H. Bryant, K. Canese, V. Chetvernin, D. M. Church, M. Dicuccio, R. Edgar, S. Federhen, M. Feolo, L. Y. Geer, W. Helmberg, Y. Kapustin, O. Khovayko, D. Landsman, D. J. Lipman, T. L. Madden, D. R. Maglott, V. Miller, J. Ostell, K. D. Pruitt, G. D. Schuler, M. Shumway, E. Sequeira, S. T. Sherry, K. Sirotkin, A. Souvorov, G. Starchenko, R. L. Tatusov, T. A. Tatusova, L. Wagner, and E. Yaschenko, "Database resources of the national center for biotechnology information," *Nucleic Acids Research*, vol. 36, no. Database issue, p. D13-D21, Jan.2008.

[82] W. J. Kent, "BLAT--the BLAST-like alignment tool," *Genome Research*, vol. 12, no. 4, pp. 656-664, Apr.2002.

[83] R. Yamashita, Y. Suzuki, H. Wakaguri, K. Tsuritani, K. Nakai, and S. Sugano, "DBTSS: DataBase of Human Transcription Start Sites, progress report 2006," *Nucleic Acids Res*, vol. 34, no. Database issue, p. D86-D89, Jan.2006.

[84] J. Davis and M. Goadrich, "The relationship between Precision-Recall and ROC curves," in *The 23rd International Conference on Machine learning,* 148 ed 2006, pp. 233-240.

[85] Y. Saeys, I. Inza, and P. Larranaga, "A review of feature selection techniques in bioinformatics," *Bioinformatics*, vol. 23, no. 19, pp. 2507-2517, Oct.2007.

[86] Y. W. Chen, Lin, and C.-J.Lin, "Combining SVMs with various feature selection strategies," in *Feature Extraction Foundations and Applications* Springer, 2006, pp. 315-324.

[87] H. Liu and R. Setiono, "Chi2: Feature selection and discretization of numeric attributes," in *Proceedings of the 7th International Conference on Tools with Artificial Intelligence, Washington D.C* 1995, pp. 388-391.

[88] K. Kenji and A. R. Larry, "A practical approach to feature selection," in *Proceedings of the ninth international workshop on Machine learning* Aberdeen, Scotland, United Kingdom: Morgan Kaufmann Publishers Inc., 1992, pp. 249-256.

[89] M. Robnik-Sikonja, "An adaption of Relief for attribute estimation in regression," in *Proc. Int. Conf. on Machine Learning, ICML-97* Morgan Kaufmann, 1997, pp. 296-304.

[90] M. A. Hall, "Correlation-based Feature Selection for Machine Learning," in *PhD thesis* 1998.

[91] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial Intelligence*, pp. 273-324, 1997.

[92] D. W. Huang, B. T. Sherman, and R. A. Lempicki, "Systematic and integrative analysis of large gene lists using DAVID bioinformatics resources," *Nature Protocols*, vol. 4, no. 1, pp. 44-57, 2009.

[93] G. Dennis, B. T. Sherman, D. A. Hosack, J. Yang, W. Gao, H. C. Lane, and R. A. Lempicki, "DAVID: Database for annotation, visualization, and integrated discovery," *Genome Biology*, vol. 4, no. 9 Aug.2003.

[94] McKusick-Nathans Institute of Genetic Medicine, Johns Hopkins University (Baltimore, MD Bethesda MD, National Center for Biotechnology Information, and National Library of Medicine, "Online Mendelian Inheritance in Man, OMIM (TM),", http://www.ncbi.nlm.nih.gov/omim/

[95] M. Horwitz, E. L. Goode, and G. P. Jarvik, "Anticipation in familial leukemia," *American Journal of Human Genetics*, vol. 59, no. 5, pp. 990-998, Nov.1996.

[96] T. R. Golub, G. F. Barker, S. K. Bohlander, S. W. Hiebert, D. C. Ward, P. Brayward, E. Morgan, S. C. Raimondi, J. D. Rowley, and D. G. Gilliland, "Fusion of the Tel Gene on 12P13 to the Aml1 Gene on 21Q22 in Acute Lymphoblastic-Leukemia," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 92, no. 11, pp. 4917-4921, May1995.

[97] E. Papaemmanuil, F. J. Hosking, J. Vijayakrishnan, A. Price, B. Olver, E. Sheridan, S. E. Kinsey, T. Lightfoot, E. Roman, J. A. E. Irving, J. M. Allan, I. P. Tomlinson, M. Taylor, M. Greaves, and R. S. Houlston, "Loci on 7p12.2, 10q21.2 and 14q11.2 are associated with risk of childhood acute lymphoblastic leukemia," *Nature Genetics*, vol. 41, no. 9, pp. 1006-1U73, Sept.2009.

[98] R. Rosati, R. L. Starza, A. Veronese, A. Aventin, C. Schwienbacher, T. Vallespi, M. Negrini, M. F. Martelli, and C. Mecucci, "NUP98 is fused to the NSD3 gene in acute myeloid leukemia associated with t(8;11)(p11.2;p15)," *Blood*, vol. 99, no. 10, pp. 3857-3860, Aug.2002.

[99] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, and M. A. Caligiuri, "Molecular classification of cancer: class discovery and class prediction by gene expression monitoring," *Science*, vol. 286, pp. 531-537, Oct.1999.

[100] J. A. Whitsett and T. E. Weaver, "Hydrophobic surfactant proteins in lung function and disease," *New England Journal of Medicine*, vol. 347, pp. 2141-2148, Oct.2002.

[101] A. Albergaria, J. Paredes, B. Sousa, F. Milanezi, V. Carneiro, J. Bastos, S. Costa, D. Vieira, N. Lopes, E. W. Lam, N. Lunet, and F. Schmitt, "Expression of FOXA1 and GATA-3 in breast cancer: the prognostic significance in hormone receptor-negative tumours," *Breast Cancer Research*, vol. 11, no. 3 June2009.

[102] V. Ciocca, C. Daskalakis, R. M. Ciocca, A. Ruiz-Orrico, and J. P. Palazzo, "The significance of GATA3 expression in breast cancer: a 10-year follow-up study," *Human Pathology*, vol. 40, no. 4, pp. 489-495, Apr.2009.

[103] W. L. Goffe, G. D. Ferrier, and J. Rogers, "Global optimization of statistical functions with simulated annealing," *Journal of Econometrics*, vol. 60, no. 1-2, pp. 65-99, Feb.1994.

APPENDIX

1. Agglomerative Hierarchal Clustering

Agglomerative hierarchal clustering is used to initialize all HyperBF networks reported in this dissertation. Wards' distance is used to measure similarity between clusters. The algorithm iteratively merges similar clusters while simultaneously performing k-means clustering to assign samples to clusters and update clusters parameters. The employed k-means clustering algorithm uses the normalized Mahalanobis distance to assign samples to clusters. After every iteration of assigning samples to their closest clusters, the center and the covariance of each cluster are updated as follows:

$$\mu_j = \frac{\sum_{i=1}^{N} u_{ij} \times x_i}{\sum_{i=1}^{N} u_{ij}} \tag{73}$$

$$C_j = \frac{\sum_{i=1}^{N} u_{ij} \times (x_i - \mu_j)^2}{\sum_{i=1}^{N} u_{ij}} \tag{74}$$

Where

$$u_{ij} = \begin{cases} 1 & if \ x_i \in cluster\ j \\ 0 & \text{otherwise} \end{cases}$$

In (73) and (74), u_{ij} is the membership of sample i in cluster j.

Init

Set number of desired clusters: csCount.
Set initial number of clusters: initCsCount (i.e. initCsCount = 5 × csCount)
Initialize initCsCount clusters randomly.

Loop

Start k-Means clustering
Loop:
Assign samples to their closest clusters
Update clusters parameters (mean and covariance)
Until convergence or fixed number of iterations
End k-Means.

Merge the closest two clusters based on Wards distance.

2. Equations and Gradient Derivatives

In the multiclass case (N samples, J hidden neurons and K outputs/classes), the output of the HyperBF network at the k^{th} output node is computed as:

$$f_{ik} = f_k(x_i) = \sum_{j=1}^{J} w_{jk} \times h_{ij} + b_k \tag{75}$$

where w_{jk} is the connection weight from hidden neuron j to output node k while h_{ij} is the output of neuron j for the given sample x_i and it is computed as:

$$h_{ij} = h_j(x_i) = exp\left(-0.5 \times \sum_{z=1}^{Z} v_{jz} \times d_{ijz}{}^2\right) \tag{76}$$

where $d_{ijz} = (x_{iz} - \mu_{jz})$ and $v_{jz} > 0 \ \forall z$

The training error of the network in the multiclass case is:

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{K}(t_{ik} - f_{ik})^2 \tag{77}$$

The gradient direction of the network error with respect to weights (w_{jk}), scaling factors (v_{jz}), and centers' variables (μ_{jz}) is computed as:

$$\frac{dE}{dw_{jk}} = -\sum_{i=1}^{N}(t_{ik} - f_{ik}) \times h_{ij} \tag{78}$$

$$\frac{dE}{dv_{jz}} = \frac{1}{2}\sum_{i=1}^{N} h_{ij} \times {d_{ijz}}^2 \times \sum_{k=1}^{K} \mathrm{w_{jk}}(t_{ik} - f_{ik}) \tag{79}$$

$$\frac{dE}{d\mu_{jz}} = -\sum_{i=1}^{N} h_{ij} \times v_{jz} \times d_{ijz} \times \sum_{k=1}^{K} \mathrm{w_{jk}}(t_{ik} - f_{ik}) \tag{80}$$

The reduced HyperBF objective function is as follows:

$$E_{reg} = E + \lambda_w \sum_{j=1}^{J}\sum_{k=1}^{K} \left\|w_{jk}\right\|^{\alpha_w} + \lambda_v \sum_{j=1}^{J}\sum_{z}^{Z} {v_{jz}}^{\alpha_v} \tag{81}$$

Computing the gradient direction of the regularized network error with respect to weights (w_{jk}), scaling factors (v_{jz}), and centers' variables (μ_{jz}) is as follows:

$$\frac{dE_{reg}}{dw_{jk}} = \frac{dE}{dw_{jk}} + \lambda_w \times \alpha_w \times \left\|w_{jk}\right\|^{\alpha_w - 1} \times sign(w_{jk}) \tag{82}$$

$$\frac{dE_{reg}}{dv_{jz}} = \frac{dE}{dv_{jz}} + \lambda_v \times \alpha_v \times {v_{jz}}^{\alpha_v - 1} \tag{83}$$

$$\frac{dE_{reg}}{d\mu_{jz}} = \frac{dE}{d\mu_{jz}} \tag{84}$$

The HyperBF objective function with smoothing splines regularization is:

$$E_{ss} = E + \frac{\lambda}{2}\sum_{i=1}^{N}\sum_{z}^{Z}\sum_{k=1}^{K} {S_{ikz}}^2 \tag{85}$$

where

$$S_{ikz} = \frac{d^2 f_k}{d{x_z}^2}(x_i) = \sum_{j=1}^{J} w_{jk} \times h_{ij} \times \left[{v_{jz}}^2 \times {d_{ijz}}^2 - v_{jz}\right] \tag{86}$$

where d_{ijz} is computed as in (76).

$$\frac{dE_{ss}}{dw_{jk}} = \frac{dE}{dw_{jk}} + \lambda\sum_{i=1}^{N}\sum_{z}^{Z} S_{ikz} \times \frac{dS_{ikz}}{dw_{jk}} \tag{87}$$

$$\frac{dE_{ss}}{dv_{zj}} = \frac{dE}{dv_{zj}} + \lambda\sum_{i=1}^{N}\sum_{k=1}^{K} S_{kzi} \times \frac{dS_{kzi}}{dv_{jz}} \tag{88}$$

$$\frac{dE_{ss}}{d\mu_{jz}} = \frac{dE}{d\mu_{jz}} + \lambda\sum_{i=1}^{N}\sum_{k=1}^{K} S_{ikz} \times \frac{dS_{ikz}}{d\mu_{jz}} \tag{89}$$

$$\frac{dS_{ikz}}{dw_{jk}} = h_{ij} \times \left[{v_{jz}}^2 \times {d_{ijz}}^2 - v_{jz}\right] \tag{90}$$

$$\frac{dS_{ikz}}{dv_{jz}} = w_{jz} \times h_{ij} \times \left[\frac{{v_{jz}}^2 \times {d_{ijz}}^4}{-2} + \frac{3 \times v_{jz} \times {d_{ijz}}^2}{2} - 1\right] \tag{91}$$

$$\frac{dS_{ikz}}{d\mu_{jz}} = w_{jk} \times {v_{jz}}^2 \times h_{ij} \times d_{ijz} \times \left[v_{jz} \times {d_{ijz}}^2 - 3\right] \tag{92}$$

In all trained HyperBF networks reported in this dissertation, the variable b_k was set to zero for the first hundred iterations. Afterward, it was updated by solving the minimum:

$$\frac{dE}{db_k} = -\sum_{i=1}^{N}(t_{ik} - f_{ik}) = 0 \tag{93}$$

However, in the experiments, where iSRprop was compared to iRprop+, b_k was set to zero in all networks for both algorithms.

3. Time and Memory Complexity

Both the proposed regularized and the unregularized networks have the same time and memory complexity for each training iteration. In the multi-class case, the time needed to compute all the derivatives and to re-compute all memberships in one iteration is of order $O(N \times J \times Z \times K)$ where N, J, K and Z are the number of samples, the number of neurons, the number of outputs and the number of dimensions respectively. On the other hand, for efficient implementation, the output from the hidden neurons and output neurons for every sample needs to be stored in memory and hence the training demands an extra memory of order $O(N \times [J + K])$.

If smoothing splines is used for regularization, the time needed to compute the derivatives, re-compute the membership matrix and re-compute the matrix S_{kzi} is also of order $O(N \times J \times Z \times K)$ but with more multiplication operations in each iteration. To store the three dimensional matrix S_{kzi} in memory, an extra memory of order $O(N \times Z \times K)$ is needed. Iterating through such a large matrix repetitively is so inefficient due main memory being much slower than the processor and the cache memory is small in most processors.

CURRICULUM VITAE

NAME: Rami Nezar Mahdi

ADDRESS: J.B. Speed School of Engineering
University of Louisville, Louisville KY 40292

EMAIL: ramimahdi@yahoo.com

DOB: Khanyounis, Palestine – July 18, 1982

EDUCATION:

B.S., Computer Science
Arab American University of Jenin
2000-2004

MS., Computer Engineering and Computer Science
University of Louisville
2005-2007

Ph. D., Computer Science and Engineering
University of Louisville
2007-2010

AWARDS: First Position in Palestine National Math Olympics Competition, 1998

PUBLICATIONS:

- Rami N. Mahdi and Eric C. Rouchka, "Reduced HyperBF Networks: Regularization by Explicit Complexity Reduction and Scaled Rprop Based Training", Under revision for IEEE Transactions on Neural Networks.

- Rami N. Mahdi and Eric C. Rouchka, "Feature selection in cancer classification from mRNA data based on localized dimension reduction", Proceedings of the Seventh International Conference on Machine Learning and Applications (ICMLA 2009), pp. 443-448, December 12-15, 2009, Miami, Florida.

- Rami N. Mahdi and Eric C. Rouchka, "Model based unsupervised learning guided by abundant background samples", Proceedings of the Seventh International Conference on Machine Learning and Applications (ICMLA 2008), pp. 203-210, December 11-13, 2008, San Diego, California.

- Rami N. Mahdi and Eric C. Rouchka, "RBF-TSS: Identification of Transcription Start Site in Human Using Radial Basis Functions Network and Oligonucleotide Positional Frequencies", PloS One, Vol. 4, No. 3., e4878, 2009.

- Rami N. Mahdi and Eric C. Rouchka, "Evidence of bias towards buffered codons in human transcripts ", Proceedings of the Eighth IEEE Symposium on Signal Processing and Information Technology (ISSPIT 2008), pp. 29-34, December 16-19, 2008, Sarajevo, Bosnia & Herzegovina.

- Hichem Frigui, Rami N. Mahdi, Meredith, J., "Combining feedback and image database categorization in CBIR", Proceedings of the 2008 IEEE International Conference on Multimedia and Expo, pp. 1277-1280, June 23-28, 2008, Hanover, Germany.

- Hichem Frigui, Rami N. Mahdi, "Semi-supervised clustering and feature discrimination with instance-level constraints", In Proceedings of the IEEE International Fuzzy Systems Conference, pp. 1-6, July 23-26, 2007.