Feature Selection Using an Extended Piecewise Linear Orthonormal Floating Search

by

ROHIT RAWAT

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2016

Dedicated to my parents.

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Michael Manry for his valuable guidance and support during the course of my studies and for his constant encouragement to explore new and diverse areas of research. I wish to thank Dr. K. R. Rao, Dr. Ioannis Schizas, Dr. Jean Gao, and Dr. Victoria Chen for providing their valuable time and input as members of my dissertation committee. I would also like to thank the friendly faculty and staff who helped shape my academic experience, and all my friends and colleagues for their company and their diverse input. Finally, I wish to thank my parents and my sisters for their faith and support that allowed me to focus on my work.

December 9, 2016

ABSTRACT

Feature Selection Using an Extended

Piecewise Linear Orthonormal

Floating Search

ROHIT RAWAT, Ph.D.

The University of Texas at Arlington, 2016

Supervising Professor: Michael T. Manry

The piecewise linear orthonormal floating search (PLOFS) is a wrapper method for feature selection that uses a piecewise linear network (PLN) to evaluate candidate subsets. PLOFS has difficulty working on high dimensional data due to overfitting and poor clustering in the PLN subset evaluation function (SEF), and high computational complexity. The presence of noise features aggravates these problems.

In order to improve upon the SEF used by PLOFS we mapped the PLN to a SPLN. Then a second order embedded feature selection was used to generate improved distance measure weights. Next, a second order method for positioning center vectors was developed. The distance measure weights and improved center vectors are mapped back to the PLN, resulting in improved performance.

We analyze the behavior of noise and dependent features in OLS and use the results to develop a reliable method of eliminating these useless features, thereby extending PLOFS to problems with larger numbers of features. We augment the data with artificial random features as probes and use piecewise linear sequential forward search to identify the useless features and remove them from the data. A two-stage feature selection method which builds upon the basic PLOFS algorithm has been developed which removes useless features and then generates subsets of different sizes of the remaining features using floating search. The resulting Extended PLOFS (EPLOFS) algorithm helps eliminate the ill-effects of too many useless features in the final piecewise linear model allowing it to be applicable to larger datasets.

We have evaluated EPLOFS and compared its performance to those of several other feature selection methods. In the presence of a large number of noise features, EPLOFS consistently produced the optimal subset with only the useful features and no noise features. Subsets of various sizes produced by EPLOFS often have smaller testing errors compared to subsets of the same size produced by other methods. The presence of dependent features further deteriorated performance of filter methods while the performance of EPLOFS remained largely unaffected.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

LIST OF ACRONYMS

PLN	Piecewise Linear Network
SPLN	Smoothed PLN
SFS	Sequential Forward Selection
SBS	Sequential Backward Selection
SFFS	Sequential Forward Floating Search
SBFS	Sequential Backward Floating Search
PLOFS	Piecewise Linear Orthonormal Floating Search
EPLOFS	Extended PLOFS
CFS	Correlation based Feature Selection
MBF	Markov Blanket Filters
MARS	Multivariate Adaptive Regression Splines
mRMR	Minimum Redundancy Maximum Relevance
RF-ACE	Random Forest Artifical Contrast with Ensembles
C-SFS	SFS with Contrast
OLS	Orthogonal Least Squares

CHAPTER 1

INTRODUCTION

1.1 Introduction

For applications in classification or approximation, feature selection is the process of selecting subsets of inputs that lead to good performance. Feature selection is an important problem in this era of big data. Advances in fields such as genomics [1, 2], spectral imaging [3], chemical informatics [4], healthcare management [5, 6], weather forecasting [7], and consumer data analytics has led to data sets with hundred to tens of thousands of features. Availability of cheap sensors, Internet of Things, streaming consumer behavior data, and cheap storage have further lead to an exponential increase in the size of datasets. Working with such large datasets is a challenge. A large number of features can lead to problems like 1) the curse of dimensionality [8], 2) poor generalization capability, 3) slow learning process, and 4) poor model interpretability [9]. Data with fewer features is also easier to store and visualize.

Automatic feature selection has not received much attention in recent times with the rapid increase in use of genetic [10] and deep learning [11, 12] methods which are attributed with built-in feature selection capabilities. But these techniques come at a high computation cost where the identity of the truly useful features is abstracted in the complexities of the network. Feature engineering uses human intellect to generate candidate sets of features that may be useful for the task, but it is time consuming and impractical with large datasets [13, 14]. Due to the drawbacks of feature engineering and deep learning, feature selection methods are being investigated as an important modeling step with renewed interest.

1.2 Classification of Feature Selection Methods

Feature selection methods are broadly classified as filter, wrapper, and embedded methods [1, 15]. Filter methods evaluate each feature assigning a score using correlation or mutual information based criteria [16, 17]. The ReliefF [18], FOCUS [19], correlation based feature selection (CFS) [20], Markov blanket filter (MBF) [21], minimum redundancy maximum relevance (mRMR) [22], and local learning based feature selection [23] algorithms are examples of popular filter methods. Filter methods can be fast but they do not model relationships between features very well. Embedded methods integrate feature selection or weighting as a part of model construction. They optimize the error function, and at the same time, penalize the presence of too many variables in the model [24, 25]. This leads to quicker design than wrapper methods for large datasets and lesser chances of overfitting. But embedded methods suffer from same the bias and variance effects as the model in use. Popular embedded feature selection methods are Lasso [26], and MARS [27]. For ensemble embedded methods like random forests [10], the model interpretability is not very good. For high dimensional data with few observations, embedded methods can fail to fit a good model. However, methods like [2] have successfully used an embedded feature selection method to remove a large number of noise features from data.

Wrapper methods use an approximation or classifier as a black box to evaluate feature subsets and choose the best one [15]. With a good subset generation algorithm, wrapper methods eliminate dependent and irrelevant features very well. But even for a small number of features the number of possible subsets can be too large leading to a combinatorial explosion. Non-exhaustive feature subset generation methods are susceptible to nesting [28, 29]. Use of a floating search method alleviates problems related to nesting and combinatorial explosion to a limited extent [30]. Feature selection methods also have problems in properly evaluating the goodness of selected subsets as it can be slow process, and existing methods are plagued with inefficiencies. Piecewise linear networks have the advantage of customizable nonlinearity, monotonicity, fast design, and fast subset evaluations. In [31], a piecewise linear network has been used with the floating search in a method called piecewise linear orthonormal floating search (PLOFS). But PLOFS performance is also adversely affected by the presence of a large number of features which can cause overfitting in the PLN and make subset search painfully slow. Some other two-step algorithms attempt to address this dimensionality problem by reducing the number of features using a filter method in the first stage [32, 33, 34], but such approaches do not employ the optimal filters and wrappers.

1.3 Proposed Work

In this dissertation, we build upon the basic PLOFS algorithm by adding a useless feature removal stage and optimizing the PLN. The resulting Extended PLOFS (EPLOFS) algorithm has increased accuracy and can process datafiles having more features. In chapter II, we describe the general structure of a feature selection system and review various feature selection methods and their drawbacks. In chapter III, we review the PLOFS algorithm. In chapter IV, we propose a smoothed piecewise linear network (SPLN) which allows for directly minimizing the output approximation error with respect to distance measure weights and center vectors and use it to improve the PLN. In chapter V, we present a method where we augment the data with artificial random features as probes and use piecewise linear sequential forward search (PLSFS) to identify the noise features and remove them from the data. We also analyze memorization in linear networks in the presence of noise and redundant features. In chapter VI, we present our two-stage algorithm and improvements to the floating search. In chapter VII, we present our final algorithm and numerical results comparing our feature selection technique to others. Chapter VIII presents our conclusions and future work.

CHAPTER 2

General structure of a feature selection system

2.1 Basic Components

The framework of many feature selection systems is provided by a subset generation algorithm (SGA), which generates potentially useful candidate subsets $\mathbf{x}^{N_1}$ of size N_1. Example SGAs include exhaustive search, branch and bound [35], sequential forward selection (SFS), and sequential backward selection (SBS) [29], Plus-L Minus-R (L-R) [36], and floating search [30]. Both SFS and SBS suffer from nesting effects. The L-R algorithm tries to alleviate this problem, by first using SFS to add L features, then using SBS to remove R features. L and R are constants. Sequential forward floating search (SFFS) and sequential backward floating search (SBFS) methods are highly related to the L-R method. In SFFS, L and R are determined adaptively such that $L > R$, while $L < R$ is used SBFS [30].

The critical second component in the system is the subset evaluation function (SEF) $J(\mathbf{x}^{N_1})$ that processes the candidate subset, along with historical or training data, into a measure of the subset's usefulness. Ideally, the numerical value of the SEF should be an increasing or decreasing function of the probability of classification error P_e. Although this is difficult to achieve, it is possible to find SEFs that have the monotonicity property [35, 28]. An SEF has the monotonicity property if for every subset pair $(\mathbf{x}^{N_1}, \mathbf{x}^{N_2})$ such that $\mathbf{x}^{N_1} \subset \mathbf{x}^{N_2}$, $J(\mathbf{x}^{N_1}) \geq J(\mathbf{x}^{N_2})$.

The arrangement of the SGA, SEF, and $\mathbf{D}$ is shown in Fig. 2.1. An optimal feature selection algorithm would examine every possible subset, in effect, using the exhaustive search or branch and bound SGA. However, even a few tens of features

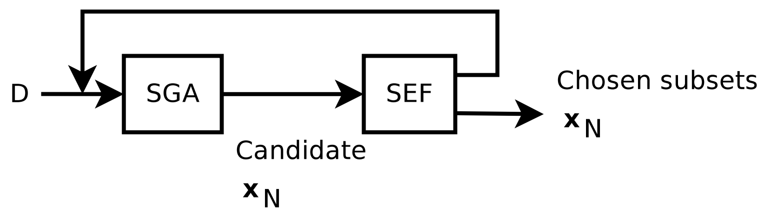


Figure 2.1. Feature Selection System.

can cause combinatorial explosion and these SGAs are not commonly used when the number of features is large. The optimal SEF $J(\mathbf{x}^{N_1})$ would calculate Bayes error e [28] for the subset under consideration. Here e is the minimum probability of error for a classification problem or the minimum mean squared error for a regression problem. This requires that the training data file $\mathbf{D}$ (in Fig. 2.1) be of infinite size, i.e. $N_\mathrm{v} = \infty$. Since an ideal Bayes SEF is impractical, we have several other SEFs. SEFs are usually classified as wrapper methods and filter methods.

2.2 Filter methods

Filter based approaches evaluate features independently of any classification or regression model using criteria like correlation, mutual information, and Fisher scores [37]. These methods do not take into account the biases of the regression or classification model and are not appropriate when removing relevant features may actually improve the performance of the system. Filter methods that rank features employ the Sum of Feature Goodness (SFG) which as the name suggests, is defined as

$$J_{\mathrm{SFG}}(\mathbf{x}^{N_1}) = \sum_{\forall \mathbf{x}_n \in \mathbf{x}^{N_1}} \mathrm{FG}(\mathbf{x}_n)$$

where $\mathrm{FG}(\mathbf{x}_n)$ is a numerical measure of goodness of feature x_n. We describe some filter methods in the following sub-sections.

2.2.1 ReliefF

This method described in [38] is an improvement over the classic Relief algorithm [39]. This is a feature weighting method and can be used to rank features using the distance measure weights. It optimizes the distance measure to maximize nearest neighbors of the same class while minimizing the number of neighbors of other classes. The SGA is SFS, and the SEF is the sum of feature goodnesses J_{SFG}. One of the improvements in ReliefF over Relief is the use of k-nearest neighbors instead of the nearest neighbor and the ability to handle multiclass, noisy, and incomplete problems. Originally intended only for classification problems, a variation of this algorithm exists for regression datasets called RReliefF [40]. The complexity of Relief and ReliefF is $O(N_{\text{v}} \cdot N \cdot N_{\text{it}})$, where N_{it} is the user chosen number of iterations [40].

2.2.2 Correlation based feature selection (CFS)

This is a filter based method that ranks features based on pairwise correlations and entropies. It uses a feature evaluation heuristic that takes into account both the usefulness of individual features in predicting the outputs and also the correlation between them [20]. It gives a score to features that is proportional to their correlation with the output class but inversely proportional to their correlation with other features. The SEF $J_{\text{CFS}}(\mathbf{x}^{N_1})$ for a subset $\mathbf{x}^{N_1}$ with N_1 features is

$$J_{\text{CFS}}(\mathbf{x}^{N_1}) = \frac{N_1 \overline{r_{xt}}}{\sqrt{N_1 + N_1(N_1 - 1)\overline{r_{xx}}}}$$

where $\overline{r_{xt}}$ is the average correlation between the features ($\mathbf{x}_n \in \mathbf{x}^{N_1}$) and the target outcome, and $\overline{r_{xx}}$ is the average correlation among the features in $\mathbf{x}^{N_1}$. The SGA employed can be SFS or SBS. CFS assumes the features are conditionally independent given the class and can fail when there are strong dependencies in the data. CFS complexity is $O(N_{\mathrm{v}} \cdot N^2)$ to find the cross correlations. The SGA has an additional complexity of $O(N^2)$ for SFS or SBS [20].

2.2.3 Minimum redundancy maximum relevance (mRMR)

This filter method uses mutual information based metrics for feature ranking. In Max-Relevance (MR), features with the highest mutual information with the target class are selected. But this alone can result in selecting several dependent features. Thus, mRMR uses a heuristic approach to minimize redundancy (mR) and select features with good relevance [22]. These two conditions are combined into a single optimization criteria to be maximized by either the difference or the quotient to form the SEF. The quotient form of the SEF is given below

$$J_{\mathrm{mRMR}}(\mathbf{x}^{N_1} \cup \mathbf{x}_n) = J_{\mathrm{mRMR}}(\mathbf{x}^{N_1}) + \frac{I(\mathbf{x}_n, t)}{(1/N_1) \sum_{\mathbf{x}_j \in \mathbf{x}^{N_1}} I(\mathbf{x}_n, \mathbf{x}_j)}$$

where $I(\cdot)$ is the mutual information of its arguments. The SGA used is SFS [29]. This method is presented as being good for handling gene expression data characterized by a large number of features and few available patterns. Doing an approximate search with mRMR using SFS, the complexity is $O(N^2)$ [22].

2.2.4 Local learning based feature selection

In this filter method [2, 41], a nearest neighbor classifier is trained on the training data using a weighted L_1 distance measure, and a class separation margin is developed that is a function of these distance measure weights. Then, the margin is maximized with respect to the weights using an expectation minimization (EM) algorithm with an L_1 penalty. Features with zero-valued weights are eliminated as being useless. The remaining features are assigned a goodness equal to the magnitude of the distance measure weights and the SEF is J_{SFG}. The algorithm is an extension of RELIEF [39, 42] and I-RELIEF [23], however, the advantage of their method over earlier methods is the probabilistic modelling of the margin which improves its performance in high dimensional feature spaces. Again, the SGA and SEF are formulated the same way as in ReleifF. Since this method uses local learning to linearize the problem, it is often referred to as "fit locally and think globally" (LOGO).

The key advantage of this method is speed and scalability for large datasets. An important disadvantage of methods like this one is that they are unable to handle dependent features well. The weights for all the features in a linearly dependent group get the same treatment, i.e. if any feature in the dependent group is useful, weights for all features in the group increases together, including ones that are dependent.

As in mRMR, this method aims to work effectively on files with fewer observations than the number of features ($N_{\mathrm{v}} \ll N$). This method does not work for regression datasets, thus we have excluded it from the comparisons on those datasets. LOGO has a complexity $O(N_{\mathrm{v}}^2 \cdot N)$ [2].

2.2.5 Boruta

This is an ensemble method that repeatedly ranks features after introducing random contrast variables in the dataset and eliminates features that consistently rank below these artificial features [43, 44]. This method is named Boruta after the mythological forest spirit. Instead of weighting features or predicting a definite subset size, the algorithm makes a high confidence prediction of which features can be eliminated and which features should definitely be kept. This leaves a large number of features in an undecided state, but which can be filtered out to a desired size by manual thresholding. This method is an extension of the work in [45] which originally proposed to determine relevance of real features by comparing them to noise features.

The SEF for this method is the feature score generated by the random forest over several iterations.

$$J_{\text{Boruta}}(\mathbf{x}^{N_1}) = \frac{1}{N_{\text{it}}} \sum_{i=1}^{N_{\text{it}}} \sum_{\mathbf{x}_j \in \mathbf{x}^{N_1}} \text{Imp}(x_j)$$

where $\text{Imp}(\mathbf{x}_j)$ is the random forest importance score of the feature $\mathbf{x}_j$. SFS or SBS can be used as the SGA. Instead of giving a feature ranking, it yields a conservative set of all relevant features and another one of all useless features. It does not strive to produce the minimal *optimal* subset. For Boruta, the time complexity of the procedure in realistic cases is approximately $O(N_{\text{v}} \cdot N)$ [43].

2.3 Wrapper methods

Wrapper methods use an approximation or classifier as a black box to evaluate feature subsets and choose the best one [15]. Several wrapper based methods exist that use SFS, SBS, floating search, or genetic algorithms as their SGA. In order to have monotonicity, a candidate SEF for a wrapper method is the mean squared error of a linear network, functional link network, or piecewise linear network fit to the data [1]. This slows down wrapper approaches since training the SEF is usually slow [46, 25, 47, 31].

SVM-RFE is a popular feature selection algorithm in bioinformatics [25, 24]. The SVM finds support vectors that maximize the margin between the classes. It uses the feature weights assigned by the SVM as a score of feature importance. Features with little importance are discarded and a smaller subset is created. The process is repeated by training another SVM and again analyzing the feature importances, till a desired subset size is reached. The SGA is SBS and the SEF is the sum of squares of the SVM's support vector weights. Since SVM-RFE involves training an SVM (sometimes multiple SVMs) at each elimination step, it can become slow for datasets with large number of patterns. It's performance is also affected by preprocessing applied to the data [24] It has performed well for gene selection and other bioinformatics tasks which have a small number of pattens. SVM training has $O(N \cdot N_{\mathrm{v}}^2 + N^3)$ complexity [25, 48]. If we assume half the features are eliminated at each stage, the overall complexity of the method is $O(\log(N) \cdot N \cdot N_{\mathrm{v}}^2 + \log(N) \cdot N^3)$.

2.4 Noise Feature Removal Methods

Most feature selection methods are capable of ranking features in the order of their usefulness. Given such a ranking, it is difficult to determine where useful features end and useless features begin. Moreover, the presence of useless features by itself introduces an element of variability in the ranking process. The singular value decomposition has been used to remove noise features [49]. The data matrix $\mathbf{D}$ is decomposed into its SVD components, which are sparsified using a thresholding method, as is done in typical data compression procedures.

Researchers have been able to use specially formulated random probe features to determine the cutoff rank for useless features with a reasonable certainty [45, 50]. Random forests and gradient boosted trees are good predictors and also rank features in the order of usefulness [51, 52]. In the method of artificial contrast with ensembles (ACE) [50], the data is augmented with artificial contrast variables and their ranking is used to remove irrelevant features. The key idea is that the probe features have a similar distribution to that of the data, but are independent of the outcomes. Features ranked below these probe features can be eliminated as being useless.

In another feature selection method [43, 44], contrast features ranked using a random forest are used to remove iteratively irrelevant features, but this method does not strive to remove all redundancies like RF-ACE. In earlier work by Stoppiglia [45], the behavior of contrast variables is analyzed in detail for models that are linear in the parameters. If an infinite amount of data is available, the random features are ranked last, or with the other irrelevant features. Since we have a finite amount of data, the rank of random features is itself a random variable. They give an analytical method to determine a cumulative distribution function for this random variable. Then the designer can choose a tolerance threshold on this probability density to determine an appropriate cutoff.

CHAPTER 3

Piecewise Linear Orthonormal Floating Search (PLOFS) Review

In piecewise linear orthonormal floating search (PLOFS) feature selection [31], we use a floating search SGA [30] and a piecewise linear network (PLN) SEF [53, 54]. Wrapper based approaches such as PLOFS incorporate an approximation or classifier into the feature selection process and the SEF is the corresponding mean-squared error [15]. The SEF is run on the dataset with different sets of features removed by the SGA and the feature set returning the highest performance is used to design the final classifier.

3.1 Piecewise Linear Networks (PLN) Review

In piecewise linear orthonormal floating search (PLOFS) feature selection [31], we use a piecewise linear network (PLN) [54] as the SEF. A PLN partitions the input space into K clusters [55, 56] with center vectors $\mathbf{m}_k$, and a different linear network is fitted for each cluster using orthonormal least squares (OLS) [57, 58, 59]. This allows the PLN to model non-linear functions well.

PLNs use the divide and conquer approach in solving a problem. As shown in Fig. 3.1, a PLN contains K N-dimensional cluster center vectors $\mathbf{m}_k$, an N-dimensional distance measure weight vector $\mathbf{b}$, and K weight matrices $\mathbf{W}_k$ of dimensions M by $(N+1)$. The input vector space is divided into several clusters comprising a Voronoi tessellation [60] using the weighted distance measure [61],

$$d\left(\mathbf{x}_p, \mathbf{m}_k\right) = \sum_{n=1}^{N} b(n) \left(x_p\left(n\right) - m_k\left(n\right)\right)^2 \tag{3.1}$$

For the pth input vector $\mathbf{x}_p$ of dimension N, the cluster index k is determined as

$$k = \arg\min_{u} \left(d_p(u)\right)$$

where $d_p(k)$ denotes $d(\mathbf{x}_p, \mathbf{m}_k)$. For each cluster, we generate a linear mapping of an $(N+1)$ dimensional augmented input vector $\mathbf{x}_{ap}$ to an M dimensional output vector $\mathbf{y}_p$ where

$$\mathbf{x}_{ap} = \left(1 : \mathbf{x}_p^T\right)^T$$

The extra input equal to one allows each cluster's linear mapping to have a constant term. For an input vector $\mathbf{x}_p$ belonging to the kth cluster, $\mathbf{y}_p$ is calculated as:

$$\mathbf{y}_p = \mathbf{W}_k \, \mathbf{x}_{ap} \tag{3.2}$$

For an approximation with M outputs, the MSE or empirical error is given by

$$J_{\mathrm{PLN}} = \frac{1}{N_{\mathrm{v}}} \sum_{p=1}^{N_{\mathrm{v}}} \sum_{i=1}^{M} [t_p(i) - y_p(i)]^2 \tag{3.3}$$

Here, N_{v} denotes the number of training patterns available, $t_p(i)$ is the ith desired output, and $y_p(i)$ is the approximated output. For the classification case, since Bayes classifiers do not exist, a conventional classifier can be used in the SEF. However, a classifier's P_{e} lacks guaranteed monotonicity when $N_{\mathrm{v}} < \infty$. In order to make monotonicity attainable, we can use a mean-squared error (MSE) measure in place of the P_{e}.

For each cluster, a linear network is trained for the patterns belonging to that cluster by minimizing the following error equation:

$$E_k = \sum_{p \in S_k} \left\| \mathbf{t}_p - \mathbf{W}_k \cdot \mathbf{x}_{ap} \right\|^2 + \lambda_k \left\| \mathbf{W}_k \right\|^2 \tag{3.4}$$

where S_k is the set of pattern numbers for the kth cluster, $\mathbf{W}_k$ is an $M \times (N+1)$ weights matrix for the kth cluster, and λ_k is a regularization parameter for the cluster.

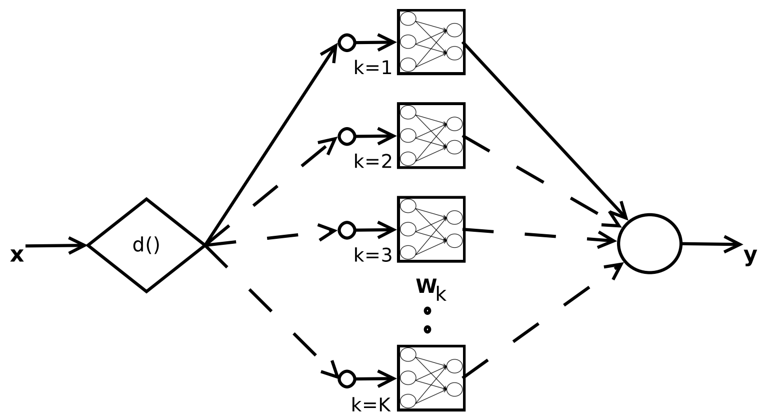


Figure 3.1. Structure of a piecewise linear network.

For each cluster, a validation set is taken out from the training data and the value of λ_k that gives the smallest error on the validation data is used for the cluster. This L_2 regularization helps reduce overfitting in the PLN clusters [62].

3.2 Basic PLN Training

Given N_v patterns $(\mathbf{x}_{ap}, \mathbf{t}_p)$, where $\mathbf{t}_p$ denotes the p^{th} M-dimensional target vector, PLN training occurs in three stages:

1) Clustering: Piecewise linear networks divide the N-dimensional input space into K volumes or clusters. The value of K is so chosen such that the resulting network is capable of representing the nonlinearity of the mapping problem. A clustering algorithm such as SOM [55] or k-means [63] is used to obtain K cluster center vectors $\mathbf{m}_k$. For each cluster, a linear network is independently trained, producing K weight matrices. The output vector is obtained using equations (3.1) and (3.2).

2) Solving for weights: For the k^{th} cluster's linear mapping given in (3.2), the cumulative squared error between the actual output vector $\mathbf{y}_p$ and the desired output vector $\mathbf{t}_p$ is given by:

$$E_k = \sum_{p \in S_k} \|\mathbf{t}_p - \mathbf{W}_k \, \mathbf{x}_{ap}\|^2 \tag{3.5}$$

where S_k is the set of pattern numbers for the k^{th} cluster. The total network mean squared error (MSE) is the sum of the cumulative errors from each cluster divided by the total number of patterns.

$$E = \frac{1}{N_{\text{v}}} \sum_{k=1}^{K} E\left(k\right) \tag{3.6}$$

The autocorrelation matrix $\mathbf{R}_k$ and cross correlation matrix $\mathbf{C}_k$ for the k^{th} cluster have elements defined as:

$$r_k\left(n, l\right) = \frac{1}{N_{\text{v}}\left(k\right)} \sum_{p \in S_k} x_{ap}\left(n\right) x_{ap}(l) \tag{3.7}$$

$$c_k\left(n, i\right) = \frac{1}{N_{\text{v}}\left(k\right)} \sum_{p \in S_k} x_{ap}\left(n\right) t_p(i) \tag{3.8}$$

where $N_{\text{v}}(k)$ is the count of patterns for the k^{th} cluster. For each cluster, a linear network is trained for the patterns belonging to that cluster by minimizing equation (3.6) with respect to the cluster weights $\mathbf{W}_k$ as:

$$\frac{\partial E_k}{\partial w_k\left(m, l\right)} = -2\left[c_k\left(l, m\right) - \sum_{n=1}^{N+1} w_k\left(m, n\right) r_k\left(n, l\right)\right] = 0 \tag{3.9}$$

For each cluster, this yields M sets of $N+1$ linear equations in $N+1$ variables. Using orthogonal least squares [59], the weight matrix $\mathbf{W}_k$ for the k^{th} linear network is found by solving

$$\mathbf{R}_k\, \mathbf{W}_k^T = \mathbf{C}_k$$

3) Pruning: PLN pruning is the process of deleting less useful PLN modules from a network designed with a large number of modules. Too many clusters can lead to memorization due to small cluster size, while too few clusters prevent the formation of a good mapping [63, 64]. A PLN can be pruned down to the required size to suit the processing and accuracy needs of a particular application [54]. The usefulness of a module is measured in terms of the reduction in the global MSE by

the presence of that module. Pruning those modules whose removal leads to the least increase in the MSE produces more compact PLN structures. The pruning algorithm is summarized here, and also described in [54].

To remove one least useful cluster from the existing set of clusters:

1. Let k be the index of the module to be potentially eliminated and E_k the error of the network after module k has been pruned. Set $E_k = 0$, for $1 \leq k \leq K$.
2. For every input vector $\mathbf{x}_p$, $p = 1$ to N_v:
    a. Find the closest cluster k_1 (the cluster it currently belongs to) and the second closest cluster k_2 (the cluster it will end up in if the closest cluster k_1 is deleted) for the pattern.

    b. Compute the error for the pattern if it belonged to the first cluster as e_1, and if it belonged to the second cluster as e_2.

    c. Supposing cluster k_1 was deleted, the pattern will move from cluster k_1 to cluster k_2 since it is the second closest cluster. This causes the pattern to cease contributing to the error for cluster k_1, and instead start contributing to the error in cluster k_2. For $k = 1$ to K, accumulate errors as:

    $$E_k \leftarrow E_k + e_1, \quad k \neq k_1$$

    $$E_k \leftarrow E_k + e_2, \quad k = k_1$$

3. Delete the cluster k_min with the smallest pruning error $E_{k\text{min}}$, distributing its patterns among the remaining clusters. Recompute the linear mapping for the modified clusters.
4. Run the validation data through the current network and save the validation error $E_{\text{v-}k}$.

This process can be repeated multiple times to remove multiple clusters. The configuration that produces the least validation error $E_{\text{v-}k}$ is chosen as the final one. This is a way to implement structural risk minimization [65].

3.3 Theoretical Justification

Since an ideal Bayes classifier or approximation is not available, we use the PLN's mean squared error as our SEF since it approximates the Bayes error [27]. First, for the classification case

$$J_{\text{PLN}}(\mathbf{x}^{N_1}) \approx E\left[\left|\left|\mathbf{d}_{\text{Bayes}}(\mathbf{x}^{N_1}) - \mathbf{y}_{\text{PLN}}(\mathbf{x}^{N_1})\right|\right|^2\right]$$

where $\mathbf{d}_{\text{Bayes}}(\mathbf{x}^{N_1})$ is the Bayes discriminant vector for $\mathbf{x}^{N_1}$ [66, 67].

For an approximation application, our SEF satisfies

$$J_{\text{PLN}}(\mathbf{x}^{N_1}) \approx E\left[\left|\left|\mathbf{t}(\mathbf{x}^{N_1}) - \mathbf{y}_{PLN}(\mathbf{x}^{N_1})\right|\right|^2\right]$$

This is equivalent to the J_{PLN} of (3.3) and has the monotonicity property.

As the number of training patterns N_{v} tends to infinity, partition based estimates of a regression function converge to the true function and the mapping converges to the corresponding Bayes estimate [51, 27]. PLNs are thus consistent non-parametric estimators. PLNs therefore have the universal approximation property [68, 69], so the J_{PLN} SEF can approximate Bayes error.

3.4 PLOFS advantages

Combining J_{PLN} with the SFFS SGA, we've developed a very effective feature selection algorithm [31, 70] known as piecewise linear orthonormal floating search (PLOFS). The algorithm is essentially a piecewise Orthogonal Matching Pursuit (OMP) method, where an OMP procedure is performed in each cell and the feature subset evaluation is averaged across all cells. PLOFS has been shown to reduce the effects of nesting because of the floating search SGA. The piecewise linear SEF is good at approximating both linear and highly non-linear problems very well. The method has been shown to outperform many others, and has successfully been used in bioinformatics applications [71, 72].

3.5 PLOFS application

In the sleep apnea detection problem [71], which has 90 ECG spectrogram texture features, PLOFS has almost eliminated whole classes of nonlinearly dependent features including entropy, dissimilarity, inverse difference and inverse recursivity. In detection of cancer in breast tissue, N = 989 light reflectance spectroscopy (LRS) features (columns) at wavelengths from 500 nm to 850 nm were analyzed. The number of available samples was only 45 (rows), so the data file is very wide. Feature extraction/compression was performed by averaging and sub-sampling down to N = 39 features. Then PLOFS was used to obtain 7 most useful features leading to non-uniform sampling of the spectrum, shown in Fig. 2(b).

The expensive LRS system may be replaced by a much cheaper system having seven optical filters and an MLP [73] classifier with one hidden unit. Then, the classifier produced a cross-validation classification accuracy of 89.5% and sensitivity and specificity of 0.88 and 0.97, respectively. It is envisioned that similar results may occur in speech recognition, and other applications of spectrograms. In the search for colon cancers [72], PLOFS gives as good a performance as ES, and a better performance than stepwise search (SFS or SBS), genetic algorithms [74] and progressive search, with far less computation.

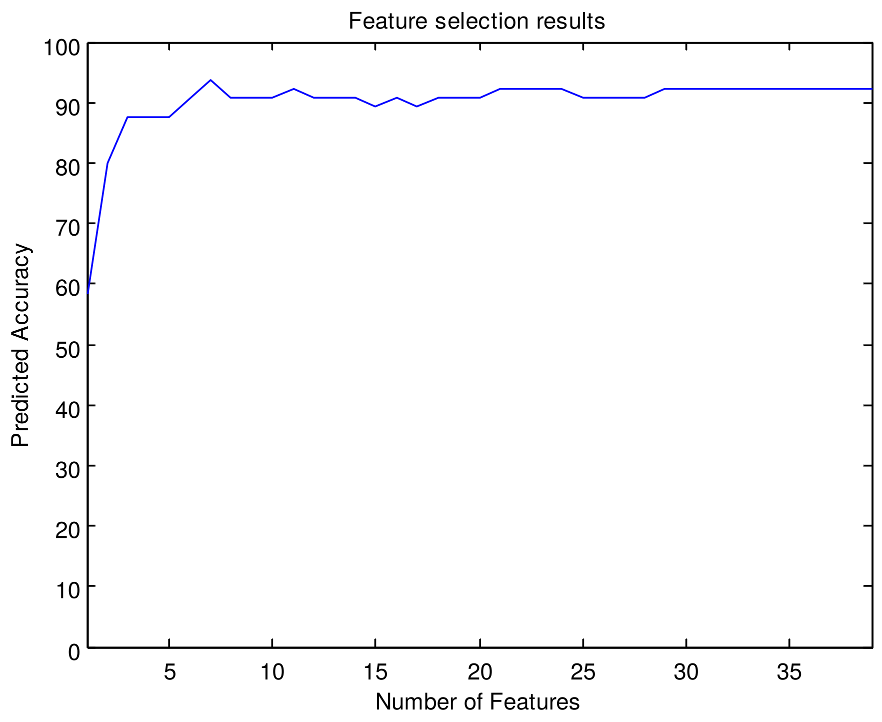


Figure 3.2. Choosing the best subset size.

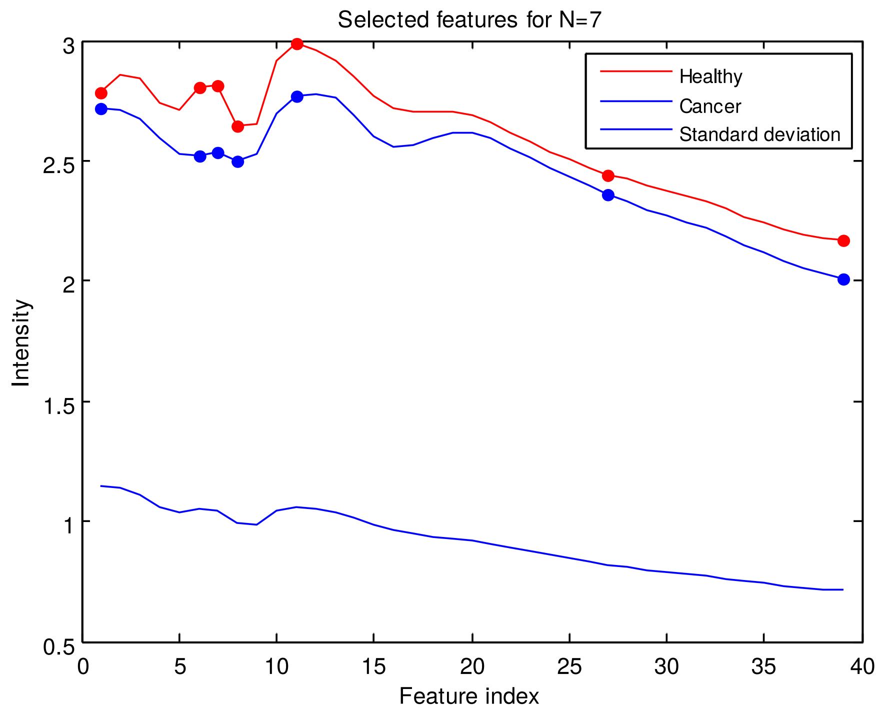


Figure 3.3. Selected features on the LRS curve.

3.6 Complexity of PLOFS

The complexity of PLOFS lies between that of SFS/SBS ($O(N^2)$) and BB ($O(2^N)$). Since it is adaptive, we cannot estimate the number of steps it will take to traverse all subset sizes. To make the complexity of SFFS more deterministic, we limit the number of consecutive backward steps to $B = 3$ which allows SFFS to have a complexity of $O(N^2)$.

3.7 PLOFS inefficiencies

There are still several problems associated with PLOFS which are exacerbated in the presence of excessive noise features.

P1. **Floating search is time consuming**: A less thoughtfully implemented floating search can be very inefficient. The floating search method comprises of SFS and SBS steps. SFS adds one more feature to the current feature subset and evaluates its performance. This is done efficiently in OLS. In the SBS steps, if the feature to be removed is not the last one added, the entire OLS needs to be recomputed.

P2. **Overfitting in SEF**: The PLN can memorize when the number of patterns in a cluster is $\leq (N+1)$. The variability in the distribution of patterns means that each cluster has a different amount of overfitting.

P3. **Poor clustering and Rigid Voronoi cells in the SEF**: Since center vectors $\mathbf{m}_k$ are not changed after initialization, and are not chosen to minimize E, they are not optimal.

P4. **PLN clustering dependent on initialization**: The initialization of the clustering algorithm has a major effect on the quality of the resultant clusters [56]. A slightly different initialization can result in vastly different results.

P5. **Poor clustering due to noise**: The presence of excessive noise features can negatively affect the PLN design due to poor clustering in high dimensions. Useless features contaminate the distance measure which leads to sub-optimal clusters. Nearest neighbor classifier distance measures, and plausibly those in the PLN, may work poorly if the number of inputs is greater than 20 [75].

P6. **Combinatorial explosion in SEF**: A large number of features can make floating search very slow. With too many noise features, the floating search take a long time to converge.

CHAPTER 4

PLN Improvements

4.1 Problems

Although PLN training is fast, several problems prevent this network from being a viable competitor with the multilayer perceptron (MLP). These are listed below:

Q1. Gradient based algorithms for improving $\mathbf{b}$ and $\mathbf{m}_k$ are inapplicable - since $\partial E/\partial b(n)$ and $\partial E/\partial \mathbf{m}_k(n)$ are zero.

Q2. Discontinuous output - For a small change in input, the PLN output can change by a lot if the cluster membership of the input changes. This can be a problem in systems which expect a smooth, continuous output.

Q3. It has been shown that for distance based classifiers like the nearest neighbor classifier, a large number of noise features can lead to poor performance[76, 77]. Similar problems exist with approximations utilizing a distance measure. Although the number of clusters and computed distances is significantly smaller in a PLN network, a large number of features in the distance measure is still a problem.

Q4. Rigid Voronoi cells - Three step training of Sect. 3.2 generates center vectors using clustering. Since center vectors $\mathbf{m}_k$ are not changed after initialization, and are not chosen to minimize E, they are not optimal.

Q5. The set of center vectors generated by clustering, and the resulting network, are not unique. This occurs because clustering is sensitive to the choice of initial clusters, the number of clustering iterations, clustering parameters etc. [63, 78].

Q6. Memorization or overfitting when $N_{\text{v}}(k)$ too small - PLNs have several problems related to generalization including the following: E_k can be zero when $N_{\text{v}}(k) \leq (N+1)$, since $\text{rank}(\mathbf{R}_k) \leq N_{\text{v}}(k)$. This results in an infinite number of solutions for $\mathbf{W}_k$. The variability of $N_{\text{v}}(k)$ means that each cluster has a different amount of overfitting.

PLOFS problems P3 and P4, and to some extent P5, are a direct consequence of these PLN drawbacks. We present our approach to solve these problems in the following sections.

4.2 Structure of Smoothed PLN (SPLN) and Embedded Feature Selection

In this section, we develop a new network that solves problems Q1, Q2 and Q3.

4.2.1 SPLN Structure and Operation

We propose a smoothed PLN, in this subsection, where the output is a weighted sum of the individual outputs from all K clusters. This smoothing solves problem Q1, since $\partial E/\partial b(n)$ and $\partial E/\partial m_k(n)$ become nonzero. Also, the outputs become continuous, relieving problem Q2.

The structure of the smoothed PLN (SPLN) is shown in Figure 4.1. Given an augmented input vector $\mathbf{x}_{ap}$, the SPLN calculates the k^{th} cluster's output vector $\mathbf{y}_{pk}$ as

$$\mathbf{y}_{pk} = \mathbf{W}_k \, \mathbf{x}_{ap} \tag{4.1}$$

The SPLN output $\mathbf{y}_p$ is computed as a weighted sum of $\mathbf{y}_{pk}$ as

$$\mathbf{y}_p = \sum_{k=1}^{K} \theta_p \left(k\right) \mathbf{y}_{pk} \tag{4.2}$$

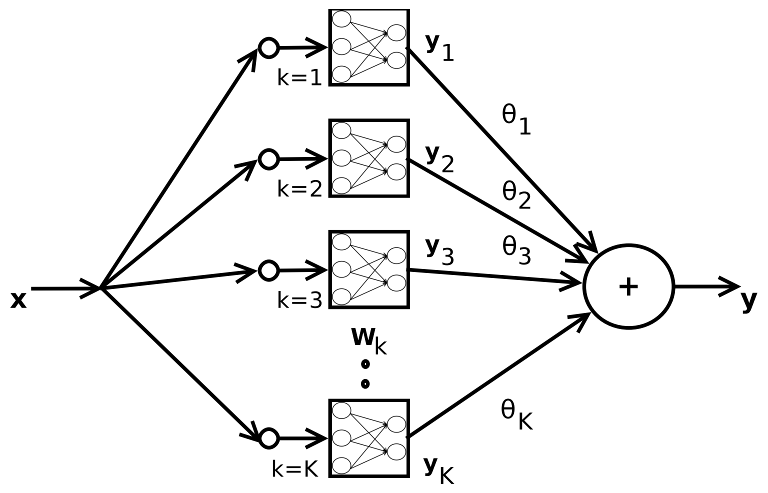


Figure 4.1. Structure of a SPLN.

where $\theta_p(k)$ is the weight for the k^{th} cluster obtained as:

$$\theta_p\left(k\right) = \frac{d_p^{-a}\left(k\right)}{D_p}, \tag{4.3}$$

$$D_p = \sum_{l=1}^{K} d_p^{-a}(l) \tag{4.4}$$

where a is a positive parameter that controls the cluster weighting. For the remainder of this paper, we drop the subscript p from $\theta(k)$, $d(k)$, and D for simplicity, since it is implied that they are computed fresh for every value of p.

The SPLN is initialized by training and pruning a discrete PLN, yielding the cluster centers $\mathbf{m}_k$, and weight matrices $\mathbf{W}_k$. The mean squared error (MSE) for the SPLN is given by:

$$E = \frac{1}{N_{\text{v}}} \sum_{p=1}^{N_{\text{v}}} \sum_{i=1}^{M} \left[t_p\left(i\right) - y_p\left(i\right)\right]^2 \tag{4.5}$$

where $y_p(i)$ denotes the i^{th} element of $\mathbf{y}_p$ in (4.2).

4.2.2 Relationship to conventional PLN

The parameter a determines the similarity between the SPLN, where the output is a weighted sum of the outputs from all clusters, and a conventional PLN where the output comes only from one cluster. By establishing this relationship, we show that the SPLN can be used to solve problem Q2 described in Sect. 4.1. When $a = 0$, all clusters are given equal weights $\theta(k) = 1/K$. For $a > 0$, the clusters nearer to the input vector $\mathbf{x}_p$ are weighted higher than ones farther from it, providing a controllable degree of locality. Consider the limiting case of infinite a,

$$\theta(k) = \lim_{a\to\infty} \frac{d_p^{-a}(k)}{\sum_{l=1}^{K} d_p^{-a}(l)} = \begin{cases} 1 \ \text{ if } k = \arg\min\limits_u (d_p(u)) \\ 0 \ \text{ otherwise} \end{cases} \tag{4.6}$$

For the case in (4.6), $\mathbf{y}_p$ is simply $\mathbf{y}_{pk}$ which is the output from a conventional PLN that emits the output of the cluster closest to the input pattern. When m clusters with indices $\mathbf{k} = \{k_1, k_2, ..., k_m\}$ are equidistant to $\mathbf{x}_p$ where $m > 1$, equations (4.6) and (4.2) reduce to

$$\theta(k) = \frac{1}{m}$$
$$\mathbf{y}_p = \frac{1}{m}\sum_{u\in\mathbf{k}} \mathbf{y}_{pu} \tag{4.7}$$

which again, is the same as result we obtain from a PLN. Thus, with larger finite values of the parameter a, the SPLN's output $\mathbf{y}_{\text{SPLN}}(a)$ approaches that of the PLN ($\mathbf{y}_{\text{PLN}}$), while still having a smooth and differentiable error function. This is summarized in the following lemma.

Lemma: $\lim\limits_{a\to\infty} \mathbf{y}_{\text{SPLN}}(a) = \mathbf{y}_{\text{PLN}}$

4.2.3 Distance Measure Improvement

Since the derivatives $\partial E/\partial b(n)$ and $\partial E/\partial m_k(n)$ are nonzero in the SPLN, as pointed out in Sect. 4.2.1, a gradient-based embedded feature selection approach can be used to de-emphasize less useful inputs. This approach solves problem Q3.

To ensure that our distance measure only takes relevant features into account, we propose to optimize the distance measure weight vector $\mathbf{b}$. This should also lead to better clustering of the data. Distance measure optimization is an integral part of the generalized relevance learning vector quantization (GRLVQ) algorithm [79], and it is very useful in SPLN training as well. We minimize the SPLN error (4.5) with respect to the weight vector $\mathbf{b}$ in (3.1) using Newton's method because of its quadratic convergence [80].

The error function E in (4.5) is already a function of $\mathbf{b}$. Expressing the improved $\mathbf{b}$ vector as $\mathbf{b} + \mathbf{e}_b$ where $\mathbf{e}_b$ is the optimal change in $\mathbf{b}$ we have

$$\mathbf{H}_b \, \mathbf{e}_b = \mathbf{g}_{eb} \tag{4.8}$$

where

$$g_{eb}(v) = -\frac{\partial E}{\partial e_b(v)} \tag{4.9}$$

and

$$h_b\,(u,v) = \frac{\partial^2 E}{\partial e_b\,(u)\,\partial e_b(v)} \tag{4.10}$$

Equation (4.8) is solved for $\mathbf{e}_b$ using orthogonal least squares (OLS) [59] and $\mathbf{b}$ is updated as

$$\mathbf{b} \leftarrow \mathbf{b} + \mathbf{e}_b \tag{4.11}$$

We continue to update $\mathbf{b}$ in this fashion till the change in the training error from the previous iteration is less than $\epsilon = 10^{-6}$. We evaluate the SPLN performance on a validation data set split off from the training set in each iteration and save the network if it has the smallest observed validation error. Assuming that the SPLN has been initialized, the distance measure improvement algorithm is summarized in Algorithm 1.

4.2.4 Examples and Simulations

We augmented some real world data sets with synthetic noise and dependent features to see if the distance measure weights corresponding to the bad features are successfully suppressed by the distance measure optimization. The original data sets are described in Appendix C.

4.2.4.1 Noise Feature Removal

Random noise features drawn from the standard normal distribution were added to data files and the larger dataset was put through the distance measure optimization process with $a = 2$. The noisy datasets generated are described in Table 4.1.

Table 4.1. Test datasets used to show removal of noise features.

Data Set	Original features	Added noise features
Twod-10RAND	8	10
Concrete-8RAND	8	8
Spiral-2RAND	2	2

Figure 4.2 shows the distance measure weights before and after optimization. The distance measure weights corresponding to the noise features are completely eliminated.

To measure how the distance measure optimization improves the overall SPLN performance, two identically initialized SPLNs were trained both with and without distance measure optimization and the final 10-fold testing errors are compared in Table 4.2. $a = 2$ was used for these experiments.

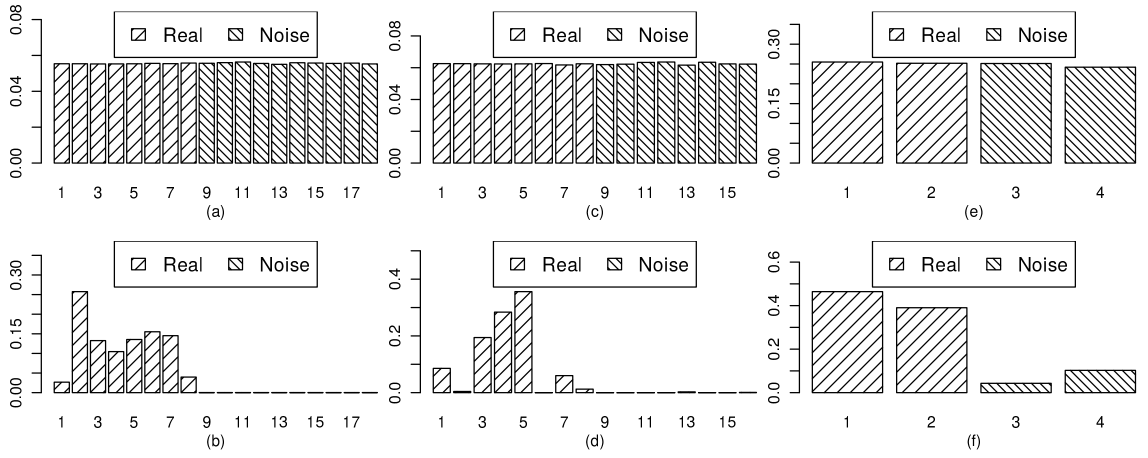


Figure 4.2. Effect of distance measure optimization on random noise features (using $a = 2$). (a), (c), (e) show the initial distance measure for Twod-10RAND, Concrete-8RAND, and Spiral-2RAND respectively, and (b), (d), (f) show the optimized distance measure.

4.2.4.2 Dependent Feature Removal

Additional dependent features were generated by adding together pairs of original features. This allows for easy identification of the dependent features and assesses their removal. Given N features in the original dataset, the k^{th} dependent feature $\mathbf{x}_{N+k}$ was generated as $\mathbf{x}_{N+k} = \mathbf{x}_{2k-1} + \mathbf{x}_{2k}$, for $k = 1 \ldots [N/2]$. Figure 4.3 shows that for every triplet of dependent features, distance measure optimization success-

Table 4.2. 10-fold SPLN testing errors for noisy features with distance measure optimization.

Dataset	Average 10-Fold MSE_{test}	
	Without distance measure optimization	With distance measure optimization
Twod-10RAND	2.4258	2.2611
Concrete-8RAND	0.6794	0.3882
Spiral-2RAND	2.7012	2.4662

Algorithm 1 SPLN distance measure improvement algorithm.

```
1. Initialize $b(n) = 1/N$ for $1 \leq n \leq N$.
2. Let MaxIter = 100, $\epsilon = 10^{-6}$.
3. For $i_t = 1$ to MaxIter
    a. Initialize $\mathbf{H}_b$ and $\mathbf{g}_{eb}$ to zeros.
    b. For $p = 1$ to $N_v$
        i. Compute $\mathbf{y}_{pk}$, $\mathbf{d}$, and $\boldsymbol{\theta}$ using equations (4.1) to (4.4).
        ii. Accumulate $\mathbf{H}_b$ and $\mathbf{g}_{eb}$ using equations (A.1) and (A.2) in Appendix A.1.
       End
    c. Solve equation (4.8) for $\mathbf{e}_b$.
    d. Update $\mathbf{b}$ using equation (4.11).
    e. Compute the SPLN training and validation MSEs.
    f. If the validation MSE ($E_v$) is smaller than smallest validation MSE ($E_{v\text{-min}}$)
       seen during training, save $\mathbf{b}$ as $\mathbf{b}_{best}$.
    g. If the reduction in training MSE in this iteration is smaller than $\epsilon$, stop
       the training iterations.
   End
4. Set $\mathbf{b} = \mathbf{b}_{best}$ as the final distance measure weight vector.
```

Table 4.3. Test datasets used to show removal of dependent features.

Data Set	Original features	Added dependent features
Twod-4DEP	8	4
Concrete-4DEP	8	4
Spiral-1DEP	2	1

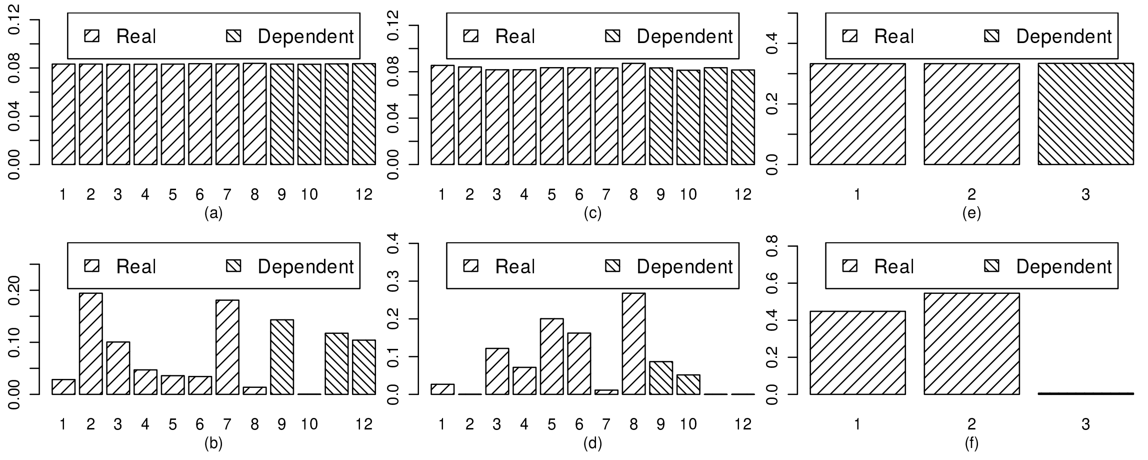


Figure 4.3. Effect of distance measure optimization on dependent features (using a = 2). (a), (c), (e) show the initial distance measure for Twod-4DEP, Concrete-4DEP, and Spiral-1DEP respectively, and (b), (d), (f) show the optimized distance measure.

fully eliminates or significantly reduces the distance measure weights corresponding to one of the three features, breaking the dependency. It is not guaranteed that the method will preserve only the original features, but since these are linear networks, any non-dependent linear combination of features should perform just as good as the original features.

Table 4.4. 10-fold SPLN testing errors for dependent features with distance measure optimization.

Dataset	Average 10-Fold MSE_{test}	
	Without distance measure optimization	With distance measure optimization
TWOD-4DEP	2.0611	2.0301
CONCRETE-4DEP	0.3337	0.2967
SPIRAL-1DEP	1.4939	1.4732

Table 4.4 shows that optimizing the distance measure in an SPLN reduces the testing error. The results of Table 4.4 look better than those of Table 4.2 because the dependent features caused less damage to the mapping than did the random features.

4.3 Second Order Center Vector Optimization

To solve problem Q4 and partially solve Q5, it is necessary to find a way to move clusters in a way that minimizes the MSE, E. LVQ is a technique that does something similar with probability of error, but it is applicable only to classifiers. Also, LVQ is not guaranteed to minimize the P_{e} of a classifier [55]. In this section, we develop methods to optimize the center vectors using Newton's method. Newton's method has already been successfully used to reduce the error by adjusting RBF network center vectors [81].

4.3.1 Approach

Here, we minimize the SPLN error from equation (4.5) with respect to the k^{th} cluster's center vector $\mathbf{m}_k$. First, we calculate the negative gradient of the error with respect to $\mathbf{m}_k$ as,

$$\mathbf{g}_m(k) = -\frac{\partial E}{\partial \mathbf{m}_k} \tag{4.12}$$

The cluster center vectors $\mathbf{m}_k$ are to be updated as:

$$\mathbf{m}_k \leftarrow \mathbf{m}_k + z_m(k)\,\mathbf{g}_m(k) \tag{4.13}$$

where the vector $\mathbf{z}_m$ of learning factors is to be found using Newton's method by solving

$$\mathbf{H}_m\,\mathbf{z}_m = \mathbf{g}_{zm} \tag{4.14}$$

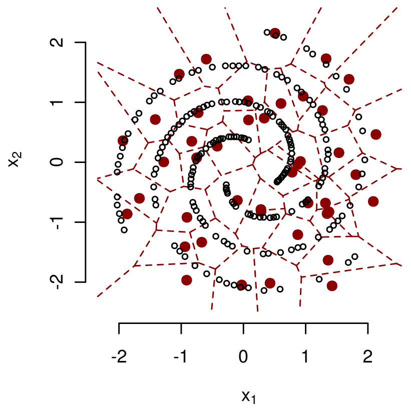
Here $\mathbf{H}_m$ and $\mathbf{g}_{zm}$ are the Hessian and negative Jacobian of the error with respect to the learning factor vector $\mathbf{z}_m$, described in Appendix A.2.

We continue to update the center vector elements in this fashion till the change in the training error from the previous iteration is less than $\epsilon = 10^{-6}$. In each iteration, we store a copy of the center vectors along with the SPLN validation error. The stored center vectors which yield the smallest validation error are kept as the final center vectors $\mathbf{m}_k$. The center vector improvement algorithm is summarized in Algorithm 2.

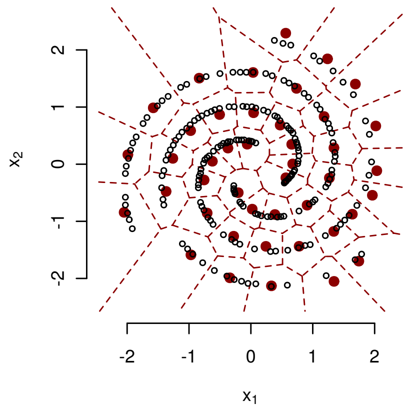
4.3.2 Cluster Optimization Example

To visually assess the effectiveness of the center vector optimization algorithm, we initialized a SPLN on the three spirals dataset from the UCI repository [82]. The input vectors in the dataset are represented by the smaller hollow circles, which form three spiralling lines, each belonging to a different class. This classification problem was converted to a regression problem with three one-hot encoded outputs. The cluster centers are represented by the dark red filled circles and the dashed red lines are the cluster boundaries visualized using [83]. The initial clusters in the SPLN were displaced by adding a small amount of noise. The cluster centers before and after optimization are shown in Figure 7.16.

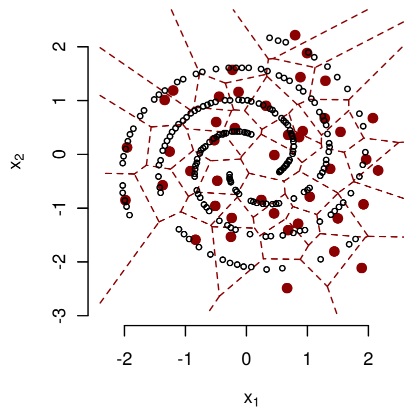
The initial locations of the cluster centers in Figure 7.16(a,c,e) can be seen to have several problems such as non-uniform distribution and the existence of cluster centers near class boundaries which are midway between adjacent spiral arms. It is observed that the center vector optimization method corrects the displacements in the cluster centers, bringing them back to the original spirals away from the class boundaries as shown in Figure 7.16(b,d,f). The clusters are also seen to be converging to a similar final configuration, alleviating the effects of problem Q5. We trained a PLN on the Spiral data using the noisy and optimized cluster centers depicted in Figure 7.16. A small amount of noise was added to deform the spirals and in-



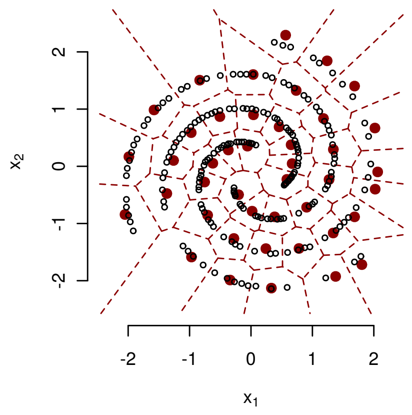
(a) $\mathbf{m}_k$ with noise 1



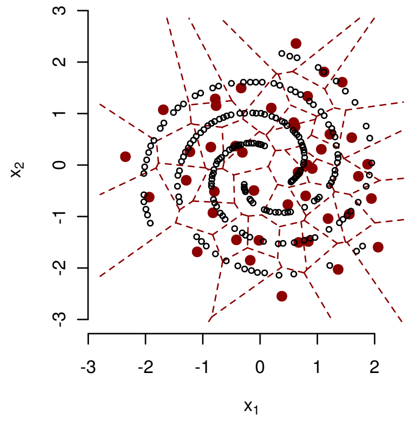
(b) $\mathbf{m}_k$ of (a) after optimization



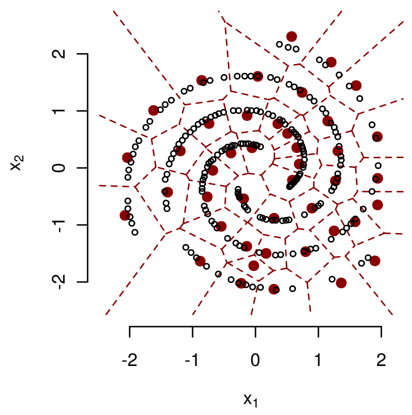
(c) $\mathbf{m}_k$ with noise 2



(d) $\mathbf{m}_k$ of (c) after optimization



(e) $\mathbf{m}_k$ with noise 3



(f) $\mathbf{m}_k$ of (e) after optimization

Figure 4.4. SPLN cluster centers for the three spirals data before and after optimization.

Table 4.5. 10-fold PLN testing error for Spiral data with noisy clusters after $\mathbf{m}_k$ optimization.

PLN's $\mathbf{m}_k$ initialization	PLN's average MSE_{test}	
	Without $\mathbf{m}_k$ optimization	With $\mathbf{m}_k$ optimization
$\mathbf{m}_k$ + noise 1	0.0750	0.0012
$\mathbf{m}_k$ + noise 2	0.5228	0.0007
$\mathbf{m}_k$ + noise 3	0.0151	0.0008

crease the difficulty of the problem. A PLN was trained using the noisy clusters from Figure 7.16(a) and another one was trained using the optimized clusters from Figure 7.16(b). The average testing errors for the two PLNs are compared in Table 4.5 along with the two remaining cases. The testing errors agree with the clustering improvement seen in Figure 7.16. Further results are presented in Sect. 4.6.

4.4 Optimizing SPLN output weights

Since the SPLN is initialized from a PLN, the SPLN weights are not optimized for the mapping described in (4.1) and (4.2). Here, we use gradient descent to optimize all the $\mathbf{W}_k$ matrices in the SPLN.

4.4.1 Derivation

To promote generalization and solve problem Q6 in Sect. 4.1, the error function of (4.5) is modified using a regularization parameter λ [62], as

$$E = \frac{1}{N_{\text{v}}} \sum_{p=1}^{N_{\text{v}}} \sum_{i=1}^{M} \left[t_p\left(i\right) - y_p\left(i\right)\right]^2 + \lambda \sum_{k=1}^{K} \|\mathbf{W_k}\|^2 \tag{4.15}$$

The negative gradient of the error function of (4.15) with respect to weight element $w_u(j,v)$ is given by,

$$g_w^u\left(j,v\right) = \frac{-\partial E}{\partial w_u\left(j,v\right)} = \frac{2}{N_{\mathrm{v}}}\sum_{p=1}^{N_{\mathrm{v}}}\sum_{i=1}^{M}\left[t_p\left(i\right) - y_p\left(i\right)\right]\frac{\partial y_p\left(i\right)}{\partial w_u\left(j,v\right)} - 2\;\lambda\;w_u(j,v) \tag{4.16}$$

where

$$\frac{\partial y_p\left(i\right)}{\partial w_u(j,v)} = \sum_{k=1}^{K}\theta\left(k\right)\frac{\partial y_{pk}(i)}{\partial w_u\left(j,v\right)}$$

Since $y_{pk}(i)$ is only a function of $w_k(i,n)$, $1 \leq n \leq N+1$,

$$\frac{\partial y_{pk}(i)}{\partial w_u\left(j,v\right)} = \begin{cases} x_{ap}\left(v\right), & u = k \; and \; i = j \\ 0, & \text{otherwise} \end{cases}$$

On obtaining the elements $g_w^k\left(j,v\right)$ of $\mathbf{G}_w^k$, the cluster weights $\mathbf{W}_k$ can be updated as:

$$\mathbf{W}_k \leftarrow \mathbf{W}_k + z\,\mathbf{G}_w^k \tag{4.17}$$

where z is the learning factor. Initially, z is set to 0.1. We use a simple heuristic to adjust z in every iteration as

$$z \leftarrow 1.1\,z \quad ifE_i < E_{i-1},$$
$$z \leftarrow 0.5\,z \quad otherwise$$

where E_i is the error for the i^{th} iteration. This is approach is similar to the damping strategy for λ used in the Levenberg-Marquardt (LM) algorithm [84, 85]. The weight optimization algorithm is summarized in Algorithm 3.

4.5 Theory and Assembly of the Final Algorithm

So far, we've motivated and briefly demonstrated component algorithms for separately optimizing coefficients in the **b**, **M**, and **W** arrays of the SPLN, where **M** and $\mathbf{W}$ store the $\mathbf{m}_k$ and $\mathbf{W}_k$ arrays respectively. As an alternative, we could form a giant weight vector as $\mathbf{w} = \text{vec}(\mathbf{b}, \mathbf{M}, \mathbf{W})$ and solve Newton's equation,

$$\mathbf{H}\,\mathbf{w} = -\mathbf{g}$$

for all the SPLN's weights simultaneously, where $\mathbf{H}$ denotes the network Hessian. In general, Newton-like approaches are justified by their quadratic convergence [86]. However, they're lacking in scalability since $\mathbf{H}$ grows too quickly and becomes rank deficient as $\dim(\mathbf{w})$ increases.

In this section we justify the use of Newton's method in general, specifically justify the use of separate Newton's algorithms in two of the component algorithms, and describe how the component algorithms are assembled into the final SPLN training method.

4.5.1 Problems with the SPLN Hessian

For fast convergence we would like to use Newton's method to train all the weights of our SPLN, but the Hessian $\mathbf{H}$ for all the weights of a multilayer perceptron (MLP) is singular [87, 88] and this may also be true for the SPLN. In this subsection we investigate the properties of Newton's algorithm as applied to the SPLN. Using these properties, we can decide whether or not to (1) implement a multi-step method assembled from our component algorithms or (2) combine our component algorithms into one giant Newton's method.

Let's start by rewriting the MSE in (4.5) as a quadratic function

$$E_H(\mathbf{w}) = E_o - (\mathbf{w} - \mathbf{w}')^T\mathbf{g} + 0.5(\mathbf{w} - \mathbf{w}')^T\mathbf{H}(\mathbf{w} - \mathbf{w}') \tag{4.18}$$

where $\mathbf{w}$ stores all elements of $\mathbf{b}$, $\mathbf{M}$, and $\mathbf{W}$ and $\mathbf{M}$ stores all of the center vectors $\mathbf{m}_k$, and $\mathbf{W}$ stores all of the weight matrices $\mathbf{W}_k$. Also, $\mathbf{w}'$ denotes $\mathbf{w}$ from the previous iteration, with $\mathbf{w}'$ being fixed. When applied to an MSE as in equation (4.18), Newton's algorithm assumes that

(A1) E is approximately quadratic in $\mathbf{w}$ for small weight changes

(A2) $\mathbf{y}_p$ is well approximated as a first degree function of $\mathbf{w}$.

Note that (A2) follows immediately from (A1). We investigate whether (A2) is a valid assumption by constructing a simplified model for $y_p(i)$. A modified error function model that yields the same Hessian and gradient as E is

$$E'(\mathbf{w}) = \frac{1}{N_{\text{v}}} \sum_{p=1}^{N_{\text{v}}} (t_p(i) - y'_p(i))^2 \tag{4.19}$$

where an approximation for $y_p(i)$ is developed as

$$y_p(i) = \sum_{k=1}^{K} \theta_p(k) y_{pk}(i) = \sum_{k=1}^{K} \theta_p(k) \sum_{n=1}^{N+1} w_k(i,n) x_{ap}(n)$$

A Taylor series for $\theta_p(k)$ in the variable $d_p(k)$ is

$$\theta_p(k) \approx a_p(k) + b_p(k)[d_p(k) - d'_p(k)]$$

where

$$a_p(k) = \theta_p(k)|_{\mathbf{w}=\mathbf{w}'}$$
$$b_p(k) = \left.\frac{\partial \theta_p(k)}{\partial d_p(k)}\right|_{\mathbf{w}=\mathbf{w}'}$$

and $d'(k)$ is $d(k)$ from the previous iteration. This yields the piecewise nonlinear model

$$y'_p(i) = \sum_{n=1}^{N+1} \sum_{k=1}^{K} \left[a_p(k) + b_p(k) \left(\sum_{m=1}^{N} b(m)(x_p(m) - m_k(m))^2 - d'_p(k) \right) \right] w_k(i,n) x_{ap}(n) \tag{4.20}$$

We have used a first order Taylor series for each coefficient $\theta_p(k)$ in each pattern and cluster in the training file. The validity of this model is demonstrated by,

$$E(\mathbf{w})|_{\mathbf{w}=\mathbf{w}'} = E'(\mathbf{w})|_{\mathbf{w}=\mathbf{w}'} \tag{4.21}$$

$$\left.\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}'} = \left.\frac{\partial E'(\mathbf{w})}{\partial \mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}'} \tag{4.22}$$

$$\left.\frac{\partial^2 E(\mathbf{w})}{\partial \mathbf{w}^2}\right|_{\mathbf{w}=\mathbf{w}'} = \left.\frac{\partial^2 E'(\mathbf{w})}{\partial \mathbf{w}^2}\right|_{\mathbf{w}=\mathbf{w}'} \tag{4.23}$$

When the vector $\mathbf{w}$ includes all elements of $\mathbf{b}$, $\mathbf{M}$, and $\mathbf{W}$, $y'_p(i)$ is not a first degree function of the weights in $\mathbf{w}$, even though the nonlinear component $\theta_p(k)$ was replaced by its first order Taylor series. Examining equation (4.20), $y'_p(i)$ includes terms such as $b(m)(m_k(m))^2 w_k(i,n)$ so its degree in $\mathbf{w}$ is 4. Similarly, $E'(\mathbf{w})$ is a degree 8 function of $\mathbf{w}$ so assumptions (A1) and (A2) are violated.

The following comments can be made

(C1) The full network Hessian can be expressed in block form as:

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_b & \mathbf{H}_{mb} & \mathbf{H}_{wb} \\ \mathbf{H}_{bm} & \mathbf{H}_m & \mathbf{H}_{mw} \\ \mathbf{H}_{bw} & \mathbf{H}_{wm} & \mathbf{H}_w \end{bmatrix} \tag{4.24}$$

where the subscripts b, m, and w signify $\mathbf{b}$, $\mathbf{M}$, and $\mathbf{W}$ respectively and where $\mathbf{H}_{bm} = \mathbf{H}_{mb}^T$ etc.

(C2) There is a large discrepancy between the degrees of $\mathbf{w}$ in $E_H(\mathbf{w})$ in equation (4.18) and $E'(\mathbf{w})$ in (4.19). Since the products $b(m)(m_k(m))^2 w_k(i,n)$ cause this discrepancy, the corresponding cross terms in the network Hessian $\mathbf{H}$ are sources of error in training an SPLN using Newton's method.

(C3) The high degree of $y'_p(i)$ in $\mathbf{w}$ means that multiple solutions of $\mathbf{b}$, $\mathbf{M}$, and $\mathbf{W}$ can occur, so $\mathbf{H}$ can be singular.

(C4) In contrast, the diagonal blocks $\mathbf{H}_b$, $\mathbf{H}_m$, and $\mathbf{H}_w$ are each calculated such that only one of the arrays $\mathbf{b}$, $\mathbf{M}$, or $\mathbf{W}$ is variable at a time. So the problems in (C2) and (C3) are alleviated.

4.5.2 Implications for SPLN training

The implications of our degree analyses are:

1. There are at least two approaches for improving on Newton's algorithm for $\mathbf{w}$. First, we can add positive numbers to the diagonal of $\mathbf{H}$, as in LM [84, 85]. However, this leaves the off-diagonal sub-matrices in place, where they continue to cause problems. Also, considerable computational complexity remains because of the large dimensions of $\mathbf{H}$.
2. If we could replace the off-diagonal sub-matrices of $\mathbf{H}$ by matrices of zeroes, the result would be the modified Hessian

$$\mathbf{H}' = \begin{bmatrix} \mathbf{H}_b & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_m & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{H}_w \end{bmatrix} \tag{4.25}$$

   Although this modified Hessian $\mathbf{H}'$ is not valid, we can use the ideas of comment (C4) to minimize $E(\mathbf{w})$ with respect to $\mathbf{b}$ while holding $\mathbf{M}$, and $\mathbf{W}$ constant, $\mathbf{M}$ while holding $\mathbf{b}$ and $\mathbf{W}$ constant, and $\mathbf{W}$ while holding $\mathbf{b}$ and $\mathbf{M}$ constant. This approach generates $\mathbf{H}_b$, $\mathbf{H}_m$, and $\mathbf{H}_w$, but not the off-diagonal blocks $\mathbf{H}_{bm}$, $\mathbf{H}_{bw}$, and $\mathbf{H}_{wm}$. This approach, where different portions of $\mathbf{w}$ are optimized one at a time, is called block coordinate descent (BCD) [89, 90].

4.5.3 Computational Complexity Considerations

Here, we discuss our final choices for the three component algorithms described earlier. Following the reasoning in Sect. 4.5.2:

1. The number of rows in $\mathbf{H}_b$ is N and the solution of the linear equations in (4.8) for Newton's algorithm requires $\mathrm{O}(N^3)$ operations [91]. For all but the largest networks, Newton's algorithm can be applied for finding $\mathbf{b}$.

2. The number of rows in $\mathbf{H}_m$ is $K{\cdot}N$ so the Newton's algorithm requires $\mathrm{O}(K^3N^3)$ operations. Using Newton's method to optimize a separate learning factor for each cluster requires that K unknowns be solved for, requiring $\mathrm{O}(K^3)$ operations. This is a reasonable compromise between steepest descent and the use of Newton's algorithm to optimize all $K{\cdot}N$ unknowns in $\mathbf{M}$.
3. The number of rows in $\mathbf{H}_w$ is $K{\cdot}M{\cdot}N$ so the Newton's algorithm requires $\mathrm{O}(K^3M^3N^3)$ operations. Instead, in order to avoid excessive computational complexity, we choose to apply steepest descent in the component algorithm for estimating $\mathbf{W}$.
4. It is possible to replace Newton's method with LM for optimizing $\mathbf{b}$ and $\mathbf{M}$. However, because of the reduced discrepancies seen in our component algorithms, and experimental evidence, this is unnecessary.

4.5.4 Final SPLN Training Algorithm

Although we have justified the use of separate component algorithms in 4.5.2 and 4.5.3, there are many ways they can be assembled into a final SPLN training algorithm. After extensive experiments we have generated algorithm 4. Note that one can use more than one iteration through all of the component algorithms.

We use the pruned PLN to initialize the SPLN clusters. After the center vector optimization step, we eliminate clusters whose membership approaches zero, reducing the size of the SPLN.

4.6 Results

In this section, we present results on several data sets to show that the new methods are successful in producing an improved SPLN. These data sets are described in detail in Appendix C. Curves showing the training and validation errors during

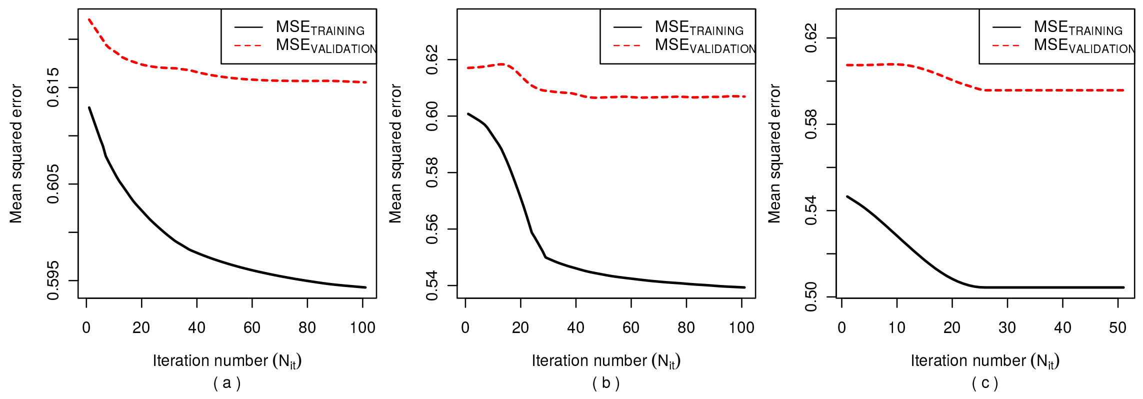


Figure 4.5. Optimization of a) distance measure b) center vectors c) weights for Red wine dataset.

distance measure optimization, center vector optimization, and weight optimization are presented. Several values of the parameter a are tried. The maximum number of training iterations was set to 100 for all three optimization algorithms.

4.6.1 Training and Validation Results

Here, we show some representative training and validation curves for the three optimization algorithms on various data sets described in Appendix C. These plots were generated for $a = 2$.

4.6.1.1 Red Wine Quality data set

This dataset is related to red variants of the Portuguese wine. The data has 1599 patterns with 11 inputs and one output. In Figure 4.5, the training and validation errors reduce during all three stages of optimization. The center vector and weight optimizations both reduce the MSE by a significant amount, the latter converging very early.

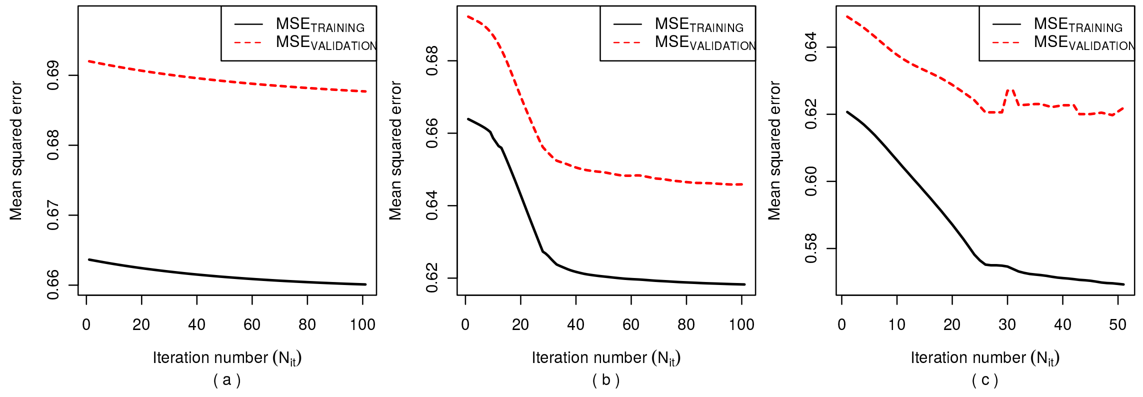


Figure 4.6. Optimization of a) distance measure b) center vectors c) weights for White wine dataset.

4.6.1.2 White Wine Quality data set

This dataset is related to white variants of the Portuguese wine. The data has 4898 patterns with 11 inputs and one output. Again, we see in Figure 4.6 that the training and validation error reducing for all three stages of optimization with the distance measure and weight optimizations responsible for significantly reducing the MSE.

4.6.1.3 Twod data set

This data is used in the task of inverting scattering parameters [92, 93], and has 8 inputs, 7 outputs, and 2768 patterns. For the distance measure optimization curves in Figure 4.7(a), we see the validation curve starting to oscillate around iteration 8 but the algorithm retains the distance measure for the minimum validation error.

4.6.1.4 Three Spirals data

This data set is originally a classification problem, involving three spirals in two dimensions belonging to three different classes. It was converted to a regression

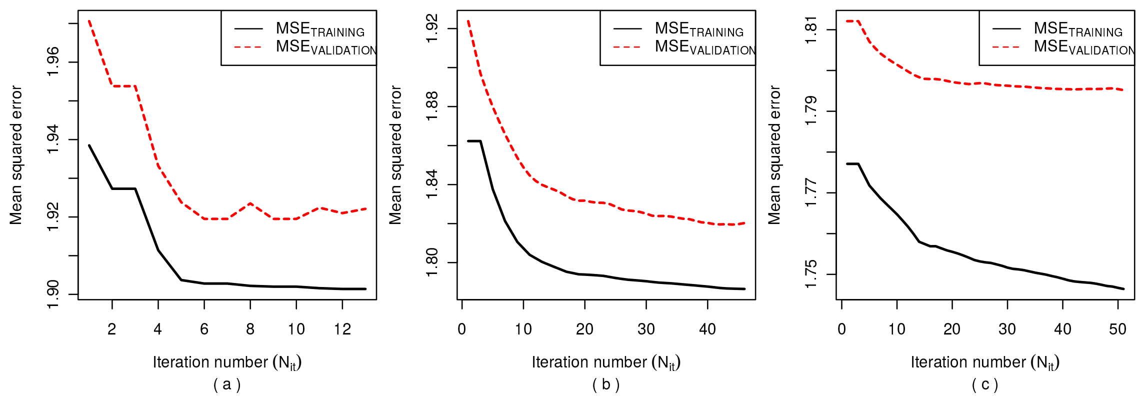


Figure 4.7. Optimization of a) distance measure b) center vectors c) weights for Twod dataset.

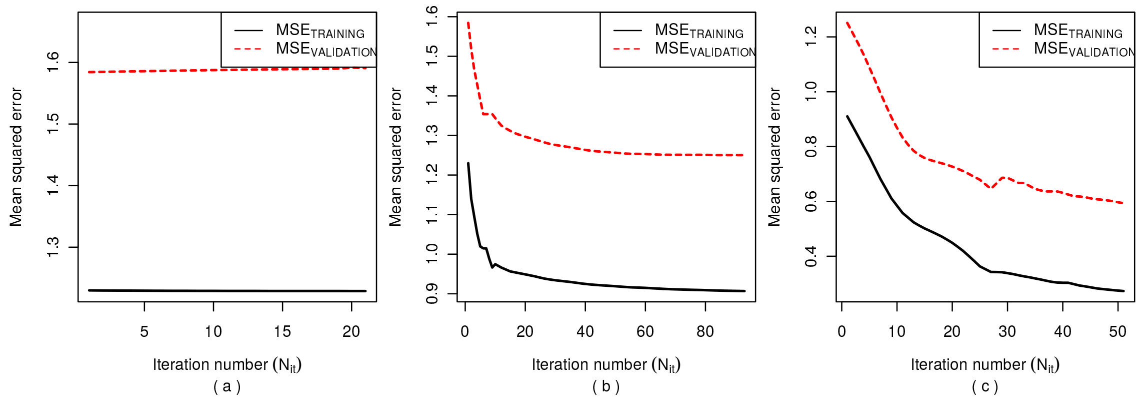


Figure 4.8. Optimization of a) distance measure b) center vectors c) weights for Three Spirals dataset.

problem by decoding the classes as binary outputs. This data has 312 patterns with two inputs and three binary one-hot coded outputs. With only two inputs (the x and y dimensions, both of which are equally important), we do not see major improvements in the distance measure weight optimization stage and the curves in Figure 4.8(a) are flat. But the center vector optimization helps move the cluster centers closer to the real data points as was illustrated in section 4.3.2 and also seen in Figure 4.8(b). This helps reduce the error quickly in the weight optimization step. The promising results on this dataset make the case for implementing SPLN classifiers in future work.

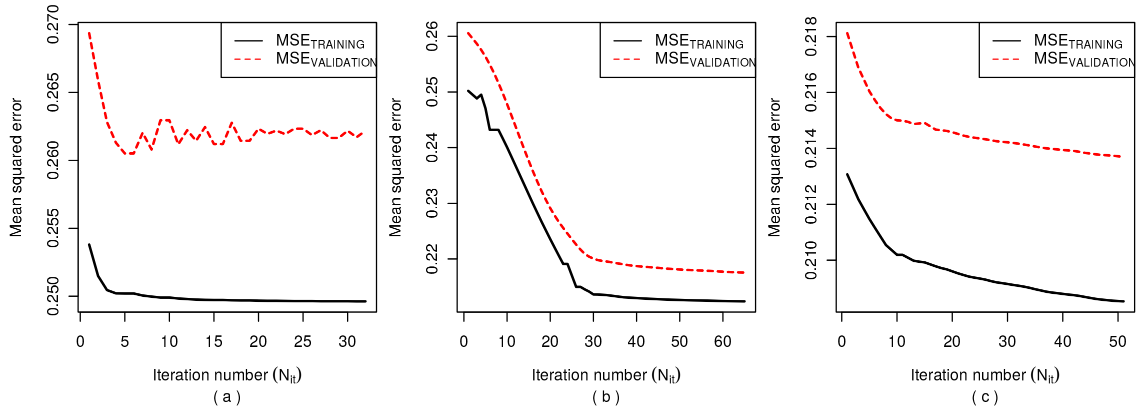


Figure 4.9. Optimization of a) distance measure b) center vectors c) weights for Oh7 dataset.

4.6.1.5 Oh7 data set

This data set relates to radar scattering from soil [94], and has 20 inputs, three outputs, and 15000 patterns. As seen in Figure 4.9, we again find the training and validation error decreasing during all three optimization stages. The largest improvement comes from the combined effort of optimizing the distance measure and center vectors which reveals the deficiencies in the original clustering for this data.

4.6.1.6 Melting Point data set

This data set predicts melting points from structural properties of molecules [95]. It has 202 inputs, one output, and 4401 patterns. Figure 4.10 shows improvement due to all three optimizations. This is a difficult dataset due to the large number of inputs which means a big, noisy and difficult to optimize distance measure vector. We see oscillations in the validation error for the distance measure optimization plot but the algorithm is able to settle around a minima. Overfitting in the output weights is also very likely but it is controlled well by the regularization.

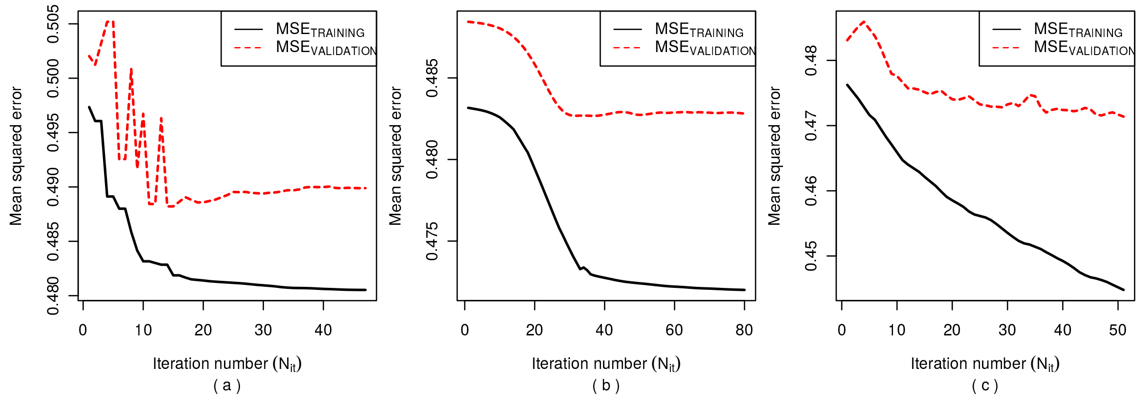


Figure 4.10. Optimization of a) distance measure b) center vectors c) weights for Melting Point dataset.

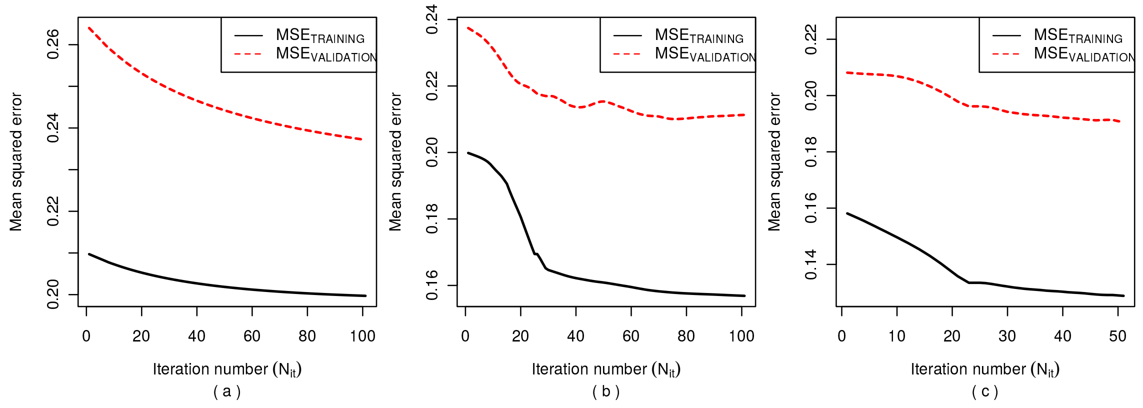


Figure 4.11. Optimization of a) distance measure b) center vectors c) weights for Concrete dataset.

4.6.1.7 Concrete data set

This data set predicts compressive strength of concrete from its components and age [96]. It has 8 inputs, one output, and 100 patterns. As seen in Figure 4.11, the algorithm works for all three optimization steps.

4.6.2 Testing Results

Ten-fold testing was performed on each dataset. The data file was split into ten equal size disjoint partitions. Ten training-testing data set pairs were created by sequentially using one partition as the test set and the other nine combined as the training set. For each fold, the training set was further split into disjoint training and validation sets keeping 70% of the data for training and 30% for validation. For each fold, the SPLN design algorithm was executed using the training and validation data. After each optimization stage, the SPLN was evaluated on the testing data from that fold to record the testing error. The training and testing MSEs from all ten folds were averaged and presented in Table 4.6. These results are obtained with $a = 2$ and $N_{\text{repeat}} = 1$.

Table 4.6. 10-Fold training and testing errors after optimizing the distance measure, center vectors, and the weights of an SPLN with $a = 2$.

		SPLN Testing MSE				
Dataset	K	SPLN as initial-ized	First Wts. Opt.	Dist. Meas. Opt.	Center Vec. Opt.	Second Wts. Opt.
RedWine	18	0.7074	0.6515	0.6445	0.6374	0.6346
WhiteWine	21	0.7071	0.6674	0.6647	0.6419	0.6212
Twod	16	2.0588	1.9938	1.9755	1.9132	1.9029
Spiral	17	1.4727	0.6692	0.6698	0.4930	0.2754
Oh7	36	0.3090	0.2898	0.2864	0.2385	0.2321
MeltingPt	4	0.5329	0.5092	0.5052	0.5045	0.4995
Concrete	20	0.3637	0.2513	0.2417	0.1975	0.1841

We see that the distance measure optimization step reduces the testing error for all the files, except for the Spiral data. This is expected, since the Spiral data is two dimensional and its initial distance measure weights can't be significantly improved. The center vector and weight optimization steps are also seen to reduce the testing error for all files. Repeating the three optimizations in a loop further reduces the 10-fold testing error for some files. These results are summarized in table 4.7.

Table 4.7. 10-fold testing errors with different values of N_{repeat}. ($a = 2$)

Dataset	K	$N_{\text{repeat}} = 1$	$N_{\text{repeat}} = 2$	$N_{\text{repeat}} = 3$
RedWine	18	0.6346	0.6337	0.6371
WhiteWine	21	0.6212	0.6187	0.6238
Twod	16	1.9029	1.9001	1.9020
Spiral	17	0.2754	0.1939	0.0837
Oh7	36	0.2321	0.2256	0.2230
MeltingPt	4	0.4995	0.4986	0.4972
Concrete	20	0.1841	0.1685	0.1497

Table 4.8 shows the initial PLN 10-fold testing error, and the PLN 10-fold testing error after redesigning it using the improved distance measure and center vectors obtained from the SPLN for different values of the parameter a.

Table 4.8. Effect of "a" on final 10-fold testing errors for a PLN using SPLN parameters.

Dataset	PLN Testing MSE					
	Initial	Final PLN using SPLN parameters				
	PLN	$a = 1$	$a = 2$	$a = 3$	$a = 4$	$a = 5$
RedWine	0.7007	0.6448	0.6425	0.6474	0.6601	0.6643
WhiteWine	0.6639	0.6476	0.6333	0.6387	0.6397	0.6394
Twod	1.9751	2.0136	1.9389	1.9351	1.9456	1.9424
Spiral	1.4912	0.5654	0.3596	0.2794	0.2637	0.2638
Oh7	0.2496	0.2810	0.2330	0.2247	0.2254	0.2263
MeltingPt	0.5125	0.5027	0.5035	0.5092	0.5055	0.5037
Concrete	0.3044	0.2332	0.2206	0.2114	0.2203	0.2153

We see that using the SPLN's parameters improves the PLN's testing error more for larger values of a, as expected from the lemma in section 4.2.2. The best results are seen for $a = 2$ and $a = 3$ because they strike a balance between having a smooth optimizable error surface, and being similar to a conventional PLN.

4.6.3 Comparison with other networks

We compare the performance of the SPLN to the multi-layer perceptron (MLP) from MATLAB to show its practicality. The SPLN was trained using the method of section 4.5.4 for $a = 2$. We used the Levenberg-Marquardt (LM) algorithm from MATLAB's Neural network toolbox for training the MLPs.

To find a suitable MLP configuration for comparison: we determined the number of free parameters available to the SPLN as

$$N_{\mathrm{w-SPLN}} = N + K\,N + K\,M\,(N+1) \tag{4.26}$$

Since the number of free parameters in an MLP with N_{h} hidden units is

$$N_{\mathrm{w-MLP}} = M\,(N + 1 + N_{\mathrm{h}}) + N_{\mathrm{h}}\,(N+1) \tag{4.27}$$

we determine the number of hidden units that make the number of free parameters equal in both the networks by equating (4.26) and (4.27)

$$N_{\mathrm{h-equivalent}} = \left\lceil \frac{N + K\,N + K\,M\,(N+1) - M\,(1+N)}{M + N + 1} \right\rceil \tag{4.28}$$

Since an MLP will typically require fewer hidden units to approximate a non-linear function than the number suggested in (4.28), we use $N_{\mathrm{h-equivalent}}$ as the upper limit for the MLP size.

The comparison method is summarized as follows. For each K-fold training-validation-testing pair,

1. We train an SPLN using the method of section 4.5.4 for $a = 2$ and $N_{\text{repeat}} = 2$ and observe the number of clusters K used in the SPLN network.
2. We compute the $N_{\text{h}-\text{equivalent}}$ from (4.28). We choose 10 uniformly spaced values for N_{h} between 1 and $N_{\text{h}-\text{equivalent}}$ and train and validate an MLP using LM from MATLAB's NN toolbox and save the network with the smallest validation error. The maximum number of iterations is set to 300 but early stopping is enabled which allows training to stop at the optimal number of iterations.
3. We apply the SPLN and MLP networks of steps 1 and 2 to the testing data and record the network testing performance.

Table 4.9. 10-Fold testing error of an MLP compared to an SPLN with $a = 2$.

Dataset	SPLN		MLP			
	K	MSE_{test}	$N_{\text{h}-\text{equiv.}}$	N_{h}	N_{it}	MSE_{test}
RedWine	18	0.6371	32	20	16	0.6671
WhiteWine	21	0.6238	37	27	18	0.6643
Twod	16	1.9004	68	48	125	1.2300
Spiral	17	0.0837	30	26	40	0.0649
Oh7	36	0.2230	119	78	16	0.1897
MeltPt	4	0.4972	8	5	14	0.5019
Concrete	20	0.1497	31	23	33	0.1059

Table 4.9 shows that the SPLN performs better than the MLP on three of the seven datasets and that the two methods perform similarly for the remaining four files according to ten-fold testing results. Thus SPLNs are not only useful for designing improved PLNs, but are also accurate and efficient approximations by themselves.

4.7 Conclusions

We have developed a smoothed piecewise linear network and algorithms for optimizing its distance measure weights, cluster centers, and output weights. The distance measure optimization is an embedded second order feature selection that successfully eliminates useless and dependent features leading to better clustering. Newton's method is used in an algorithm that improves center vector locations. We have shown that our method produces smaller testing errors than the discrete PLN. The SPLN training algorithm is also more robust than standard PLN training for datasets with many useless features. The performance of a discrete PLN was found to have improved when the optimized distance measure and center vectors from an SPLN were used to retrain it. We also show the SPLN to be a useful network comparable to the MLP. In future work, we will implement SPLN pruning, improved output weight training, and SPLN classifiers.

Algorithm 2 SPLN center vectors improvement algorithm.

```
1. Given initial center vectors $\mathbf{m}_k$ for $1 \leq k \leq K$
2. Let MaxIter = 100, $\epsilon = 10^{-6}$
3. For $i_{\mathrm{t}} = 1$ to MaxIter
    a. Initialize $\mathbf{g}_m(k)$ for $1 \leq k \leq K$, $\mathbf{H}_m$, and $\mathbf{g}_{zm}$ to zeros.
    b. For $p = 1$ to $N_{\mathrm{v}}$
        i. Compute values for $\mathbf{y}_{pk}$, $\mathbf{d}$, and $\boldsymbol{\theta}$ using equations (4.1) to (4.4),
        ii. Accumulate the gradient $\mathbf{g}_m(k)$ for $1 \leq k \leq K$ using equation (A.3) in Appendix A.2.
        iii. Accumulate $\mathbf{H}_m$ and $\mathbf{g}_{zm}$ using equations (A.4) and (A.5) in the Appendix A.2.
       End
    c. Solve equation (4.14) for $\mathbf{z}_m$
    d. Update $\mathbf{m}_k$ using equation (4.13) for $1 \leq k \leq K$
    e. Compute the SPLN training and validation MSEs
    f. If the validation MSE is smaller than smallest validation MSE seen during training, save $\mathbf{m}_k$ as $\mathbf{m}_{k-\mathrm{best}}$.
    g. If the reduction in training MSE in this iteration is smaller than $\epsilon$, stop the training iterations.
   End
4. Set $\mathbf{m}_k = \mathbf{m}_{k-\mathrm{best}}$ as the final cluster centers.
```

Algorithm 3 SPLN weights improvement algorithm.

```
1. Given an SPLN with initial weights $\mathbf{W}_{k-\text{initial}}$ for $1 \le k \le K$, and $\epsilon = 10^{-6}$
2. Let MaxIter = 50
3. For $\lambda = 0, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^{2}, 10^{3}$
    a. Set $\mathbf{W}_k = \mathbf{W}_{k-\text{initial}}$ to begin.
    b. For $i_{\text{t}} = 1$ to MaxIter
        i. Initialize $\mathbf{G}_w^k = \mathbf{0}$ for $1 \le k \le K$.
        ii. For $p = 1$ to $N_{\text{v}}$
            1. Compute $\mathbf{y}_{pk}$, $\mathbf{d}$, and $\boldsymbol{\theta}$ using equations (4.1) to (4.4).
            2. Accumulate the gradient $\mathbf{G}_w^k$ for $1 \le k \le K$, using equation (4.16).
           End
        iii. Update $\mathbf{W}_k$ using equation (4.17) for $1 \le k \le K$
        iv. Compute the SPLN training MSE
        v. Compute the SPLN validation MSE by passing through the validation
           data
        vi. If the validation MSE ($E_{\text{v}}$) is smaller than smallest validation MSE
           ($E_{\text{v-min}}$) seen during training, save $\mathbf{W}_k$ as $\mathbf{W}_{k-\text{best}}$, $\lambda$ as $\lambda_{\text{best}}$, and
           iteration number $i_{\text{t}}$ as $N_{\text{it-best}}$
        vii. If the reduction in training MSE in this iteration is smaller than $\epsilon$
           times the MSE, stop the training iterations
       End
   End
4. The validation data is put back with the training data, and for the fixed values
   of $\lambda = \lambda_{\text{best}}$ and MaxIter = $N_{\text{it-best}}$, repeat steps 3-a) and 3-b-i) to 3-b-iv). Set
   $\mathbf{W}_k$ as the final SPLN weights.
```

Algorithm 4 Final SPLN training algorithm.

```
1. Randomly split the training data file into disjoint training (90%) and validation
   (10%) data.
2. A PLN network is designed using the training and validation data. The latter is
   used to select the regularization parameters and for determining the optimum
   number of clusters in the network. The validation data is then combined with
   the training data and used to train the final network.
3. An SPLN is initialized using the PLN's center vectors, distance measure, and
   weights. To improve the initialization, optimize the $\mathbf{W}_k$ matrices as in Sect.
   4.4.
4. For $i_\mathrm{t} = 1$ to $N_\mathrm{repeat}$
    (a) The SPLN's distance measure is optimized using the method described in
        Sect. 4.2.3 using the training and validation data.
    (b) The SPLN's center vectors are optimized using the method described in
        Sect. 4.3 using the training and validation data.
    (c) Regularization factor $\lambda$ for output weight optimization is estimated using
        the validation data.
    (d) The validation data is combined with the training data.
    (e) The SPLN's weights are optimized using the method described in Sect. 4.4.
5. If an improved PLN network is desired, the distance measure and center vectors
   from the SPLN are used to design a new PLN.
```

CHAPTER 5

Useless Feature Removal

Useless features lead to overfitting in the SEF (P2), poor clustering (P5), and slow feature subset search (P6). In this section, we analyze the behavior of noise and dependent features in OLS and use the results to develop a reliable method of eliminating these useless features.

5.1 Feature Model

Here, we model the effects of noise and dependent features. We assume that the features in $\mathbf{x}$ are a vector function of a set of useful features $\mathbf{u}$ of dimension N_u and a useless noise feature vector $\mathbf{n}$ of dimension N as

$$\mathbf{x} = \mathbf{f}(\mathbf{u}, \mathbf{n})$$

In the k^{th} cluster, however, where the variation of each feature x_n is smaller, we assume that $\mathbf{x}$ can be modeled as:

$$\mathbf{x} = \mathbf{B}\mathbf{u} + \mathbf{n}$$

where $\mathbf{n}$ is a vector of additive noise of length N, and $\mathbf{B}$ is a rectangular $N \times N_u$ matrix that can generate linearly dependent features from the useful set of feature $\mathbf{u}$. In this model, elements of $\mathbf{x}$ can consist of:

i. Pure useful features $x_k = u_j$ when $b(k,i) = \delta(i-j)$, and $n_k = 0$.

ii. Useful features with additive noise, as $x_k = u_j + n_k$, when $b(k,i) = \delta(i-j)$, and $n_k \neq 0$.

iii. Linear combinations of features in $\mathbf{u}$, as

$$x_k = \sum_{i=1}^{N_u} b(k,i)u_i; \quad n_k = 0$$

iv. There can be linear combinations with additive noise:

$$x_k = \sum_{i=1}^{N_u} b(k,i)u_i + n_k$$

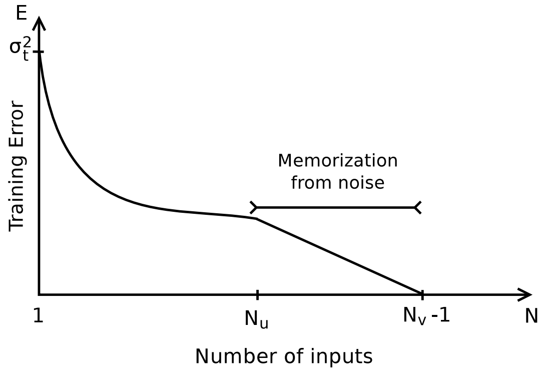
v. There are some pure noise features $x_k = n_k$, when $b(k,i) = 0$ for all i.

When orthogonal least squares (OLS) is used to find the linear mapping matrix $\mathbf{W}_k$ for the k^{th} cluster, elements of $\mathbf{x}$ are ordered as

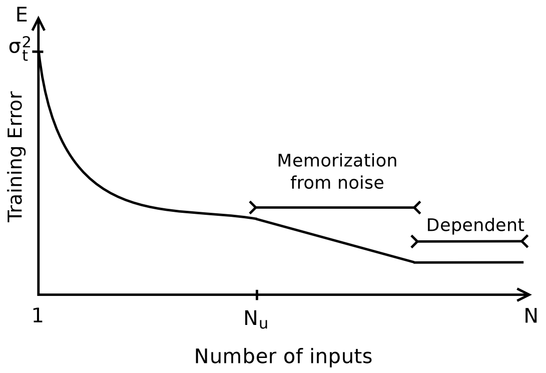
$$x_{\text{o}}(n) = x(o(n))$$

$x_o(n)$ denotes the n^{th} most useful feature where OLS builds $o(n)$ such that $x(o(n))$ yields the largest drop in E_k when $x(o(n))$ is grouped with features $x(o(1))$, $x(o(2))$, $\ldots$ $x(o(n-1))$ [57, 58, 59]. Then orthonormal features are generated as $\mathbf{x}' = \mathbf{A}\cdot\mathbf{x}_o$ where $\mathbf{A}$ is lower traingular as in the QR decomposition [57, 58]. Given that $x'(1)$ through $x'(n-1)$ have already been found, the n^{th} row of $\mathbf{A}$, and $o(n)$ are chosen so that $x'(n)$ causes the maximum possible decrease in E.

Consider a simple case where the first N_{u} features are linearly independent non-zero type (i) features and the remaining $N-N_{\text{u}}$ features are type (v), and $N+1 = N_{\text{v}}$. Following the results of appendix A, this results in the curve of Fig. 5.1(a). In a more typical case, let the first $N_{\text{u}} + N_{\text{d}}$ features be type (iii) features, resulting in N_{u} linearly dependent features followed by N_{d} dependent ones. Assume that the remaining features are type (iv). OLS then groups the features in a sequence as $\{\mathbf{x}_u^T, \mathbf{x}_n^T, \mathbf{x}_d^T\}$ where column vectors $\mathbf{x}_u^T, \mathbf{x}_n^T, \mathbf{x}_d^T$ consist of N_{u} useful features, $N - N_{\text{u}} - N_{\text{d}}$ noise features, and N_{d} dependent features. Note that the noise features are created when OLS removes the useful components of the type (iv) features. This is illustrated



(a) Only noise features



(b) Noise and dependent features present

Figure 5.1. Training error curves for data with noise and dependent features.

in Fig. 5.1(b) below. Dependent features are rejected by OLS and do not contribute to memorization or lowering the MSE. Thus they appear as the flat portion at the end of the curve. When there are not enough noise features for complete memorization, the flat portion is raised a little as seen in Fig. 5.1(b).

Clearly, we can improve PLOFS if we can find the value of N_u or at best eliminate most features in the linear or flat regions seen in Fig. 5.1. Random probe features [45] fall into the group of pure noise features, and lie in the middle linear region. This conveniently puts all of the redundant features and most of the noise behind the probes in the feature ordering. This approach can be used with PLNs for removal of useless features.

5.2 SFS with Contrast (C-SFS) Algorithm

We take an approach similar to [45] and [50] with our theoretical analysis to remove noise and dependent features quickly. We call our useless feature removal method SFS with Contrast (C-SFS) and summarize it below:

1. Add 9 random probe features to the data.
2. Train a PLN on the data without performing any regularization.

3. Perform SFS on the PLN.
4. Locate where the mid-point, or the fifth feature occurs in the SFS order.
5. Remove all the features occurring after the mid-point feature and all the random probe features.

For data with too many features and too few patterns, the reliability of ranking the probe features goes down significantly. In some of our experiments, the noise features correlated with the output more heavily than any of the real features in the data. This often leads to the false elimination of several features from the data. Since our method is a two stage method, this causes irreversible damage to the remaining feature selection stage. Methods described above like RF-ACE [50] and Boruta [43] deal with this problem by having a user chosen risk parameter that adjust the selectivity of the noise removal method, or they are not very specific about which subset size to keep. Instead of keeping a user chosen parameter, we fix the problem by being being less selective in the first stage for files which have a large N/N_{v} ratio. This does not adversely affect the feature selection outcome as there is a reliable second stage to deal with the useless features left behind from the first stage. We still get the benefits of having a smaller less noisy feature space to deal with during PLOFS. Since we do this only for short files as they are the ones most affected by accidental removal of useful features, it is also not computational burdensome for the second stage to handle the additional features. The feature elimination method for short files is described below:

1. Perform the random probes elimination process to get a list of useless features, but do not remove them from the data.
2. Repeat the above step two more times, but this time using different random features generated using different seeds.

3. Take the intersection of the features eliminated in the three trials of elimination and use it as the final list of features to delete.

With this approach, a feature that has been shown to perform worse than the median performing probe three times, and not once exceeding this level of performance, are deemed useless.

CHAPTER 6

Two Stage Feature Selection

We describe our two-stage algorithm EPLOFS here that solves the problems in section 3.7. Specifically, we use an improved PLN SEF that solves problems P2, P3, and P4, a useless feature removal stage that solves P5 and P6, and an efficient floating search implementation that solves P1.

Since it is difficult to solve the PLN's problems directly, we have developed a smoothed piecewise linear network (SPLN) and algorithms for optimizing its distance measure weights and cluster centers in Chapter 4, thereby solving PLOFS problems P3 and P4.

6.1 SGA Improvements

OLS uses SFS to order basis functions so the SFS component of SFFS is naturally efficient. Let $\mathbf{o}_t$ be the order function at the t^{th} iteration. In the SFS step, the order function for the next iteration is simply obtained by adding the index of the next useful feature n_1 to $\mathbf{o}_t$ as $\mathbf{o}_{t+1} = \{\mathbf{o}_t, n_1\}$. However, the SBS component of SFFS is implemented via exhaustive search. We propose a method to make the SBS step recursive, thereby mitigating PLOFS's inefficiency problem (P1). Extending OLS to SBS requires that we start with an OLS order function $o_t(k)$ for n features and a model for each cell's linear network, with E described by (3.4). Given that we want to eliminate x_m on a trial basis, and evaluate the resulting E, we can (1) Find k such that $o_t(k) = m$, and (2) then $o_{t+1}(i) = o_t(i)$ for $i = 1$ to $k-1$ (3) and replace $o_{t+1}(i)$ by $o_t(i+1)$ for $i = k$ to $n-1$, finding $w'(n)$ and E. Thus we have removed x_m from the feature set, and constructed the corresponding OLS expansion. We can backtrack and repeat these steps for different values of m until we find the best feature to remove in SBS. Since each trial feature removal requires the calculation of only about one half of the basis functions, this version of SBS is more efficient than re-calculating the entire expansion for each trial feature removal. Note that the order in which the $o(n)$ values are generated affects the efficiency of the following trial replacements, leaving room for considerable improvement. If the position of the basis chosen for deletion, k, is close to the end of o_t, fewer forward steps will be needed to build o_{t+1}. We use optimal ordering to build o_{t+1} to ensure that features with a high likelihood of being removed continue to be at the tail end of the order function.

6.2 Complete Extended PLOFS (EPLOFS) algorithm

Putting together the work in previous sections, we present a feature selection tool called Extended-PLOFS (E-PLOFS). Our method uses the novel approach of using PLSFS with probe features to eliminate useless and dependent features in the first stage and the highly efficient and precise floating search in the second stage. In this section, we describe the final algorithm and present simulation results to show that it works. The E-PLOFS algorithm works as follows:

Given training and validation data with N columns,

1. Normalize the inputs and target outputs in the training data. Use the statistics of the training data to normalize the validation data.
2. Insert N_r Gaussian noise probe features to the data changing the dimension of $\mathbf{x}^T$ in $\mathbf{D}$ from N to $N + N_r$ columns wide. $N_r = 9$ is used as the default.
3. Train a PLN on the data with regularization turned off. The initial number of clusters is determined based on the number of inputs and the number of available training patterns. The PLN is pruned using validation data to determine the

optimal cluster size. Once the size is determined, the training and validation data are combined and used to train the weights of a PLN of that size.

4. Perform SFS using the PLN as the SEF to determine the feature order. Note the location of the N_r probes in the feature order.
5. Use the median of the locations of the N_r probes as the cutoff for eliminating noise features.
6. Remove all features worse than the cutoff feature, also remove all the remaining probe features to obtain the feature order for stage 1.
7. Generate new training and validation data files from the features remaining in step 6.
8. Train a PLN on the new training data. The initial number of clusters is recomputed based on the current number of features. The PLN is pruned using validation data to determine the optimal cluster size and the value for the regularization parameter λ.
9. Initialize an SPLN from the PLN of step 8. Optimize the distance measure and center vectors of the SPLN as described in section [97].
10. Using the distance measure and center vectors of the SPLN as the corresponding PLN parameters, generate a new PLN on the training data and delete any empty clusters. Use the validation data to regularize the PLN weights.
11. Run PLOFS using the PLN of step 10 as the SEF and determine optimal subsets of each size up to N. Compute the validation error for each subset size using the validation data.
12. The subset that has the minimum validation error is chosen as the optimum subset.
13. The subset is translated to the original feature numbers using the feature selection order determined in step 6.

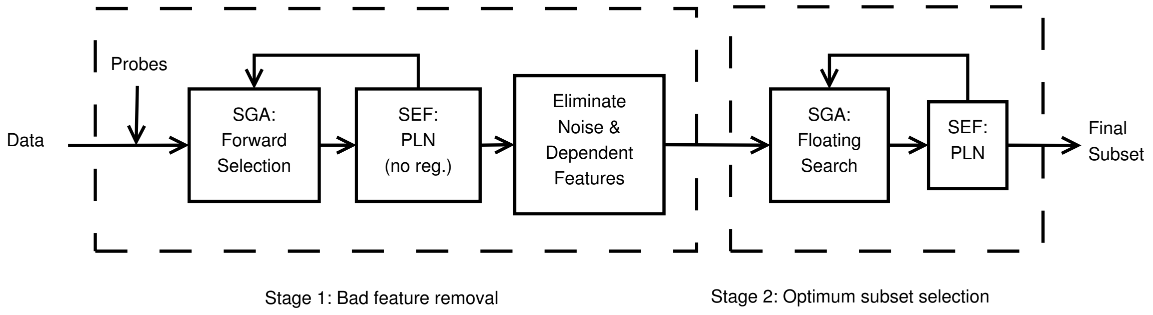

Probes
Data
SGA: Forward Selection
SEF: PLN (no reg.)
Eliminate Noise & Dependent Features
SGA: Floating Search
SEF: PLN
Final Subset
Stage 1: Bad feature removal
Stage 2: Optimum subset selection

Figure 6.1. Two stage feature selection.

The key portions of the two stage algorithm are presented in Fig. 6.1.

6.2.1 Complexity of EPLOFS

EPLOFS runs in two stages. The first stage is a SFS search with an average complexity of $O(N^2)$. The first stage removes redundant features and we can assume that, on average, we are left with half the features we started with. The floating search has no general complexity estimate, but it is safe to assume quadratic complexity. Thus the total complexity can be put at $O(N^2) + O((N/2)^2) \sim O(N^2)$.

6.3 Comparison to PLOFS

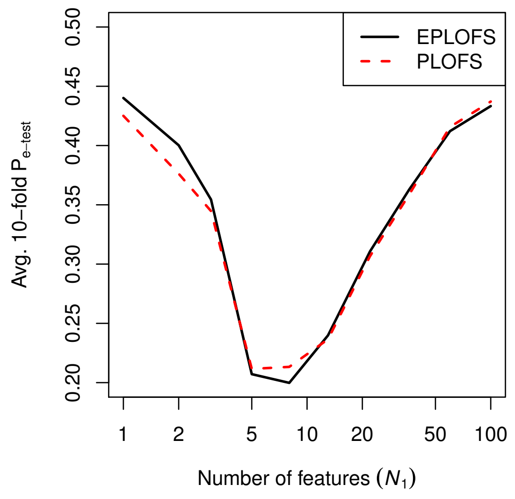
We added $N_{\rm n} = 100$ noise features to a few synthetic datasets and used EPLOFS and PLOFS to produce the optimal subset. The number of real and noise features in the subset were counted. The numbers of retained good features $N'_{\rm g}$ and residual noise features $N'_{\rm n}$ are presented in Table 7.2.

We observe that both EPLOFS and PLOFS are able to detect the useful features and eliminate noise features, but EPLOFS is considerably faster because it eliminates most of the noise features in the first stage. We compare EPLOFS with some other feature selection methods in the next section.

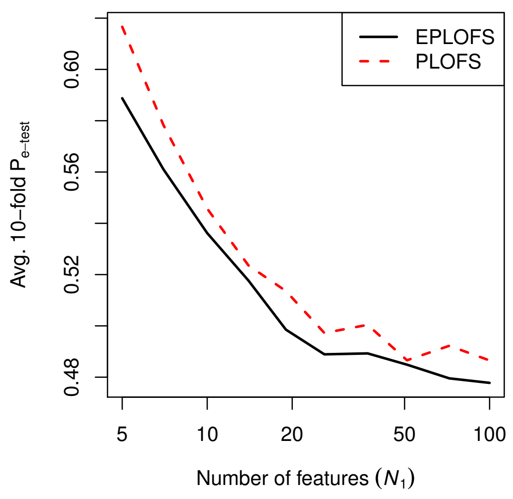
Table 6.1. Numbers of useful and noise features in the best subset generated by EPLOFS and PLOFS.

Data Set	Good features	N_v	EPLOFS N'_g	EPLOFS N'_n	EPLOFS time	PLOFS N'_g	PLOFS N'_n	PLOFS time
Friedman	5	1000	5	0	2.2	5	1	28.3
Reading	2	200	2	0	0.3	2	0	3.4
XOR	2	52	2	0	0.5	2	0	19.8
3-Spirals	2	312	2	2	3.6	2	9	48.4
Hypercube	3	800	3	0	5.7	3	0	57.1

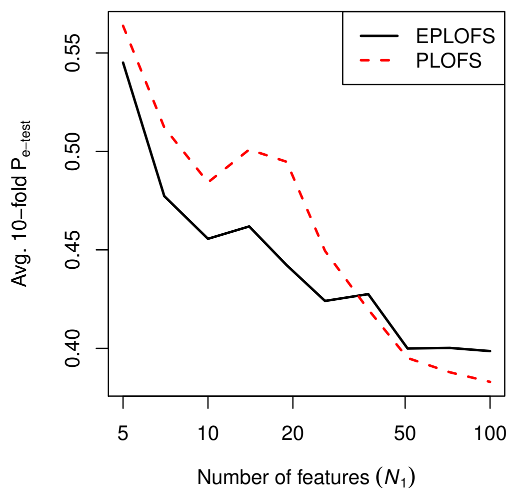
We can say that EPLOFS provides better subsets than PLOFS and takes less time to execute when the number of features is large.



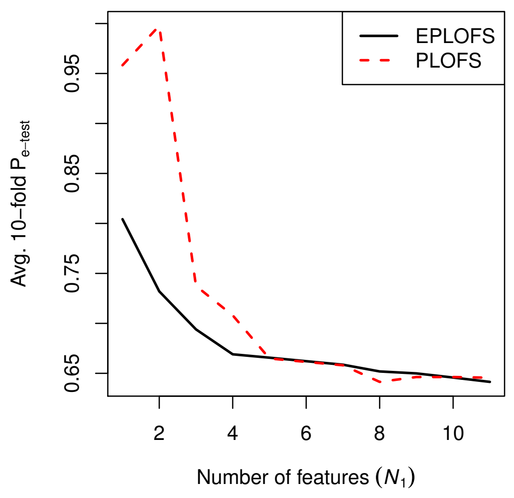
(a) Madelon data (EPLOFS time = 41s, PLOFS time = 842s)



(b) Melting point data (EPLOFS time = 116s, PLOFS time = 705s)



(c) Aquatic toxicity data (EPLOFS time = 53s, PLOFS time = 88s)



(d) White wine data (EPLOFS time = 1s, PLOFS time = 1s)

Figure 6.2. Subset evaluation with EPLOFS and PLOFS..

CHAPTER 7

Simulations

In this section, we compare E-PLOFS to several different FS methods.

7.1 Algorithm evaluation methodology

The quality of subsets generated by a feature selection method can be estimated by measuring the testing error from a model trained on the selected features. To get an unbiased estimate, we adopt a 10-fold testing methodology. The data set is divided into ten disjoint partitions of approximately equal size. Ten training-testing data set pairs were created by using one partition as the test set and the other nine combined as the training set. For each fold, we ran each feature selection algorithm on the training data to generate the optimal subset of a certain size. We then trained an independent classification or regression model (MLP and SMV) using just the chosen features from the training data. The generated classifier is then evaluated on the testing data and the test errors from the ten folds are averaged to get the average testing for each feature selection method for a certain subset size. This error (J_{test} for regression files and $\text{P}_{\text{e-test}}$ for classification files) can be used to evaluate the goodness of each feature selection method. A good feature selection method will always produce the smallest test error for a given subset size. For a dataset with N features, there are N possible subset sizes, and it can be laborious to evaluate every size. We thus choose 10 subset sizes logarithmically distributed between 1 and N. When N is more than 250, we cap the largest subset size at 250 as it already exceeds the complexity requirements for most modeling tasks and it also keeps evaluation times reasonable

[98]. The testing error curves for all the methods are plotted on a graph with the different subset sizes on the x-axis. For EPLOFS components which need validation data, for example to size the PLN, for regularizing, or for determining the best subset size, a validation set was generated by taking out 20% from the training data.

7.1.1 Evaluating testing error for a given subset

It is important that the evaluation function is not tied to the feature selection algorithm to avoid any bias. The multi-layer perceptron (MLP) classifiers and support vector machine (SVM) are widely used classification models with well developed training and parameter selection methods which makes them ideal for evaluating the generated subsets. The same properties apply to MLP approximations and support vector regression (SVR) models for evaluating regression data. We have used the MLP of [99] which can automatically size its hidden layer. The maximum number of hidden units and number of iterations were both set to 100. We used the libSVM library [100] for generating the SVM and SVR models. The SVM and SVR parameters were tuned over a cost grid of 10^{-5} to 10^{+5} and gamma from $\gamma = 0.5$ to $\gamma = 0.9$ with an RBF kernel.

Since we are evaluating six feature selection methods for ten different sizes, for ten folds, we need to make 600 model designs and evaluations for every dataset. This makes the execution time a critical factor in choosing the modeling method, especially for files with a large number of features and patterns. We compare the execution time and model accuracy of the MLP and SVM regression and classification and present our results in Table 7.1.

Although the SVM usually outperforms the MLP, the SVM is slow and its parameters have to be determined via a grid search which further slows it down.

Table 7.1. Test error and training time for MLP and SVM models.

Approx. Datasets	N	M	N_v	MLP J_trg	MLP J_tst	MLP time (s)	SVR J_trg	SVR J_tst	SVR time (s)
Housing	13	1	506	0.0549	0.1634	1.43	0.0413	0.1250	1.64
Red Wine	11	1	1599	0.5166	0.6705	2.00	0.4246	0.6090	5.17
Two-D	8	1	2768	0.2010	0.3259	3.47	0.6370	0.7515	10.92
White Wine	11	1	4898	0.5541	0.6354	3.52	0.0705	0.5461	28.76
Oh7	20	1	10453	0.0052	0.0064	8.15	0.0059	0.0063	86.38
Melting Pt.	202	1	4401	0.3627	0.4796	30.58	0.2898	0.4350	183.2507
CGPS	1000	1	200	0.9987	1.0119	182.33	0.2601	0.8582	23.81
Class. Datasets	N	M	N_v	MLP $\text{P}_\text{e-trg}$	MLP $\text{P}_\text{e-tst}$	MLP time (s)	SVM $\text{P}_\text{e-trg}$	SVM $\text{P}_\text{e-tst}$	SVM time (s)
Grng	16	4	800	0.0033	0.0300	3.74	0.0093	0.0287	2.76
Gongtrn	16	10	6000	0.0476	0.0692	11.95	0.0321	0.0610	32.11
Comf18	18	4	12392	0.1254	0.1379	20.54	0.1212	0.1483	101.90
Madelon	500	2	2600	0.2072	0.4635	123.59	0.1273	0.4019	231.28
Sylva	216	2	14395	0.0021	0.0076	42.23	0.0009	0.0103	1188.70

Because of this limitation, we choose to evaluate the chosen feature subsets using the MLP. The same comments apply for the classification case.

7.1.2 E-PLOFS compared to other methods

Here, we compare EPLOFS to several other well known feature selection methods using the error performance of their generated subsets. We use well known datasets and cover a large number of areas feature selection is frequently applied including bioinformatics data, Quantitative Structure Activity Relationship (QSAR) data, and well known feature selection benchmark problems. Both ReliefF and RReliefF are implemented in the CORElearn R package [101]. The CFS algorithm is available in the Weka toolbox [102] and we have used the R implementations provide in RWeka [103] and FSelector [104] packages. The implementation of mRMR is available in the R package mRMRe [105]. We have used a MATLAB implementation of LOGO provided by the authors. The Boruta package available in R [44].

7.1.3 Removal of noise and dependent features

Methods to determine the smallest optimal subset should remove the noise features. We generated several synthetic datasets with different degrees of linearity and feature dependencies. Using synthetic data ensures that all features are relevant to the outcome in some way. We then augmented the datasets with different numbers of noise features. We ran EPLOFS along with other algorithms described in 7.1.2 and evaluated the final optimal subsets generated by each algorithm for the number of original features retained and the number of useless features left behind. Although ReliefF, CFS, mRMR, and LOGO are feature weighting methods and do not originally provide a cutoff for determining the best subset among the feature rankings, we have used a heuristic that determines the cutoff where there is a sharp drop in the relevance scores to determine the optimal subset. The Boruta method presents an all-relevant subset which is used for that method. EPLOFS uses a validation error to automatically determine the optimal subset.

We added $N_{\mathrm{n}} = 100$ noise features to all the datasets. Then all the feature selection methods were executed on the data to produce the optimal subset. The number of real and noise features in the subset were counted. The numbers of retained good features N'_{g} and residual noise features N'_{n} are presented in Table 7.2.

We see that the Boruta algorithm was the most successful in removing noise features across all files. EPLOFS came in a close second performing the same as Boruta but it had one more residual noise feature than Boruta on the 3-spirals data. The Reading Skills database is a toy example [106] with a spurious correlation - only two out of the three features are really useful in this case but Boruta ends up picking

Table 7.2. Numbers of useful and noise features in the best subset generated by all methods.

Data	Good	N_v	EPLOFS		ReleifF		CFS		Boruta		mRMR		LOGO	
Set	features		N'_g	N'_n	N'_g	N'_n	N'_g	N'_n	N'_g	N'_n	N'_g	N'_n	N'_g	N'_n
Friedman	5	1000	5	0	1	1	1	0	5	0	1	0	-	-
Reading	2	200	2	0	1	0	2	16	2	2	1	0	-	-
XOR	2	52	2	0	2	99	1	0	0	1	0	1	1	1
Twonorm	20	7400	20	0	8	0	20	0	20	0	20	0	20	1
3-Spirals	2	312	2	2	2	0	2	0	2	1	1	0	2	1
Hypercube	3	800	3	0	3	0	3	0	3	0	1	0	2	0
Banana	2	5300	2	0	1	0	2	0	2	2	1	0	2	1

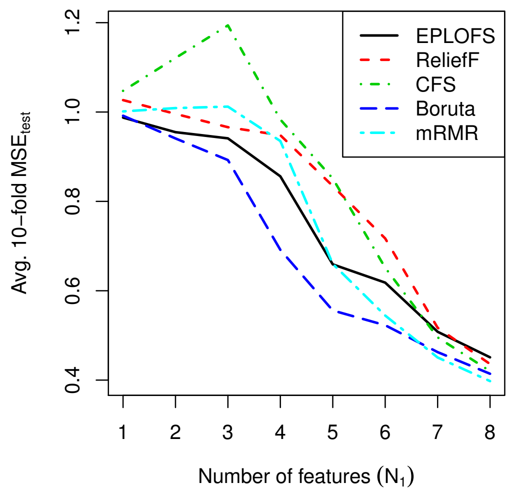
all three, fooled by the third spurious input. The ReleifF and CFS methods worked very well on the three classification datasets, but failed to find the useful features in the Friedman dataset for the approximation of a high degree polynomial. LOGO had one leftover feature for both Twonorm and 3-Spirals data. mRMR did not find the all the useful features on three of the four datasets tested.

7.1.4 Quality of generated subsets

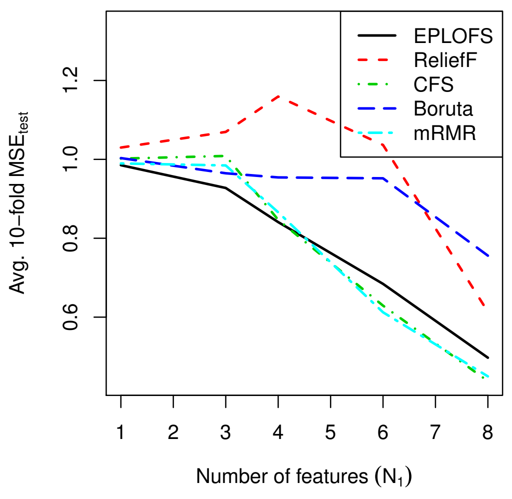
7.1.4.1 Approximation Datasets

Since most feature selection methods described here can only work with one regression target, we present results on the first target column of all multi-target files. PLOFS does not have such a limitation and can generate a unified model for feature selection in multi-target data.

1. **Two-D data** With the small number of features there isn't a major distinction in the performance of feature selection methods and Boruta outperforms all other feature selection methods, but when 8 additional dependent features are



(a) Original features

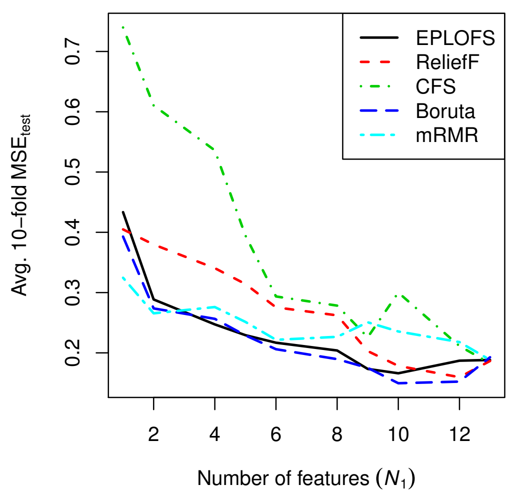


(b) With dependent features

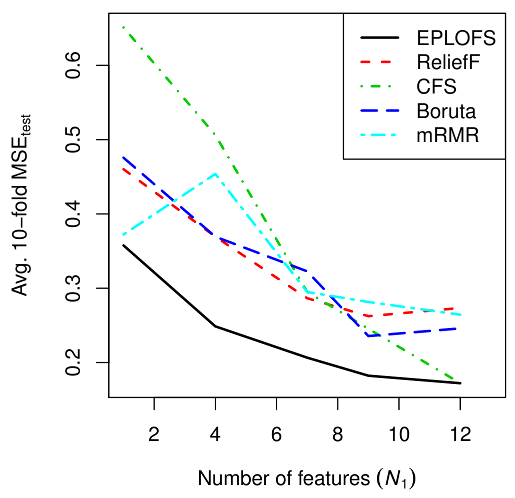
Figure 7.1. Subset evaluation for Two-D data ($N = 8$) for (a) original features (b) added dependent features.

added, Releif and Boruta methods are not able to perform as well as EPLOFS, CFS, mRMR.

2. **Housing data** We see that EPLOFS, mRMR, and Boruta perform well on the original dataset. But EPLOFS has much better performance than all other methods when dependent features are added.
3. **Red wine data** For this dataset, all the tested methods perform well on the original set of features. With dependent features, the ReliefF method loses some of its efficacy.
4. **White wine data** Again, we see that EPLOFS performs well on both the original dataset and the one with dependent features, but the performance of Boruta and ReleifF degrades in the presence of dependent features.
5. **Oh7 data** This file has a large number of training examples, therefore less chances of memorization due to the non-linearity of the problem. All methods

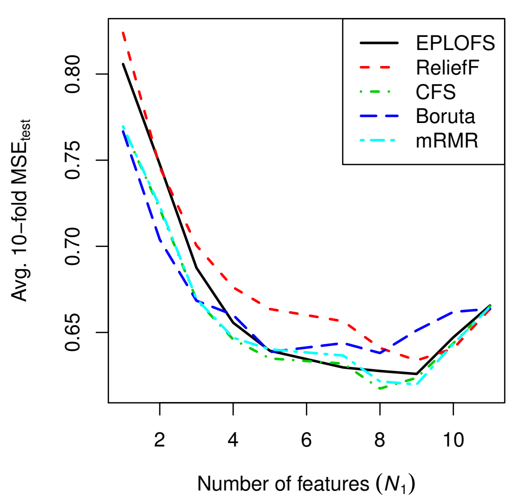


(a) Original features

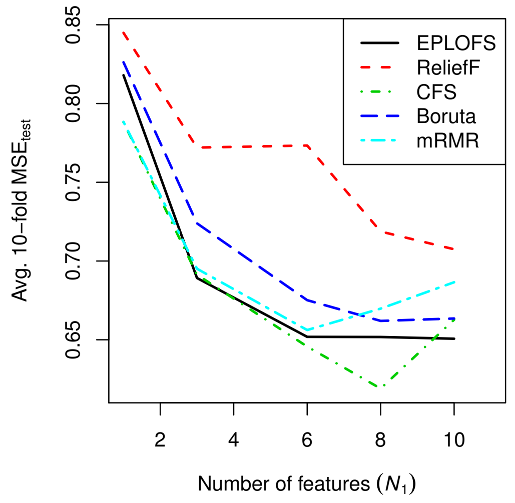


(b) With dependent features

Figure 7.2. Subset evaluation for Housing data ($N = 13$) for (a) original features (b) added dependent features.

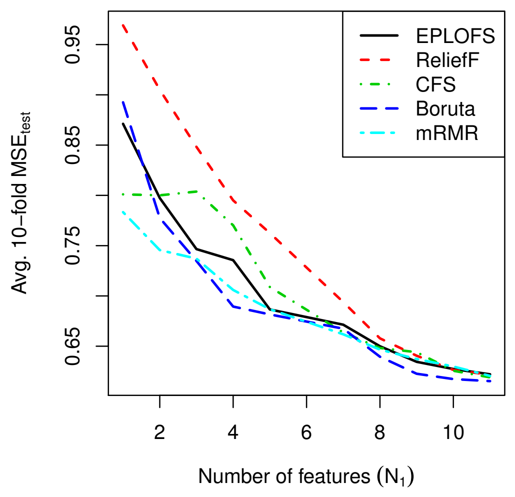


(a) Original features

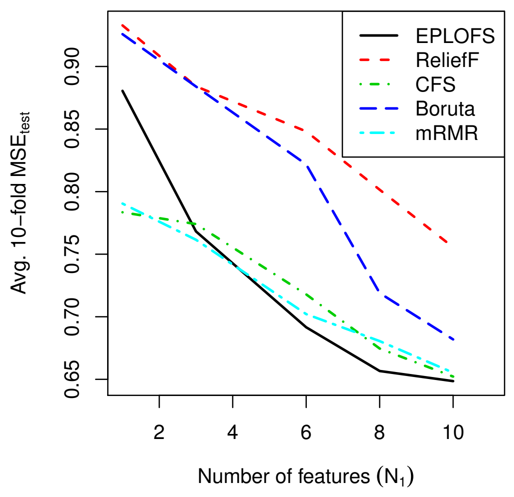


(b) With dependent features

Figure 7.3. Subset evaluation for Red wine data ($N = 11$) for (a) original features (b) added dependent features.



(a) Original features



(b) With dependent features

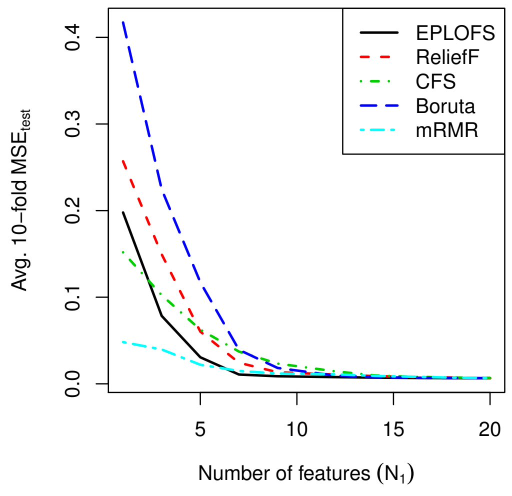
Figure 7.4. Subset evaluation for White wine data ($N = 11$) for (a) original features (b) added dependent features.

perform well on this data, but Boruta's performance deteriorates significantly when dependent features are present.

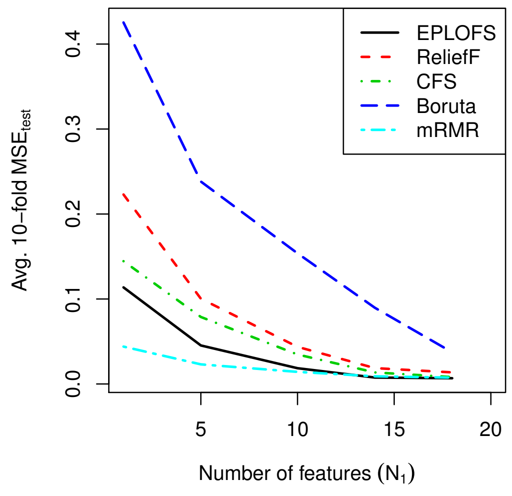
6. **Melting point data** For this file, EPLOFS performs better than all the other feature selection methods. Even in the presence of dependent features, EPLOFS and mRMR give good performance.

7. **Aquatic toxicity data** For this file, EPLOFS performs better than all the other feature selection methods. Since this dataset can be easily memorized ($N = 500$, $N_v = 220$), the results for ReliefF quickly deteriorate when dependent features are added.

8. **CGPS data** The file only has a 100 training patterns, so it was evaluated in 4-CV folds instead of the 10 used in other data. EPLOFS does not generate the best subset of every size, but it is able to seek out one with the minimum error. We did not evaluate this dataset with dependent features due to its size.

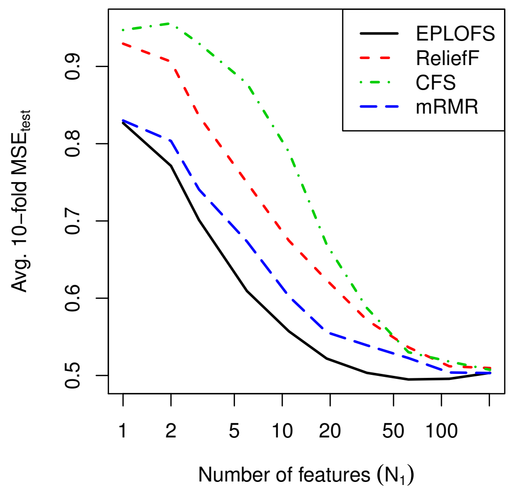


(a) Original features

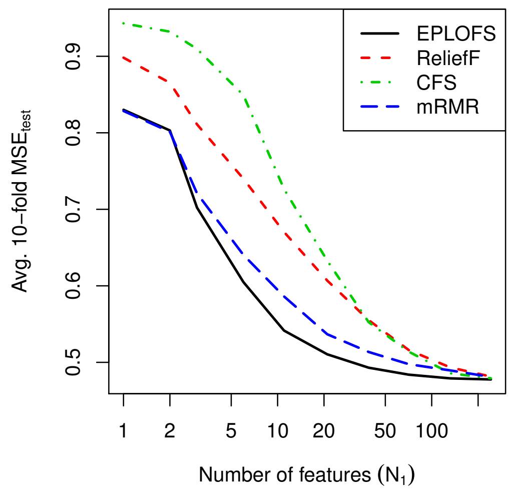


(b) With dependent features

Figure 7.5. Subset evaluation for Oh7 data ($N = 20$) for (a) original features (b) added dependent features.

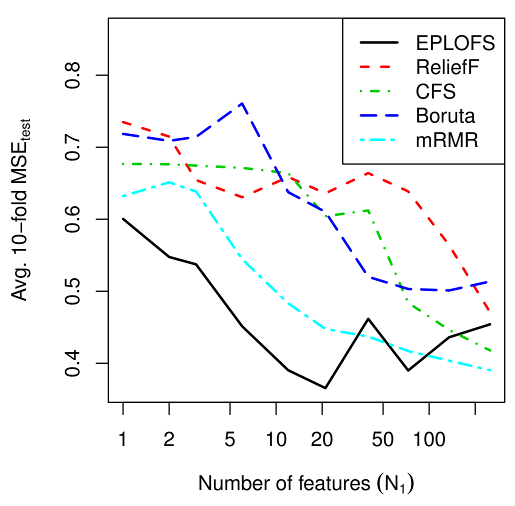


(a) Original features

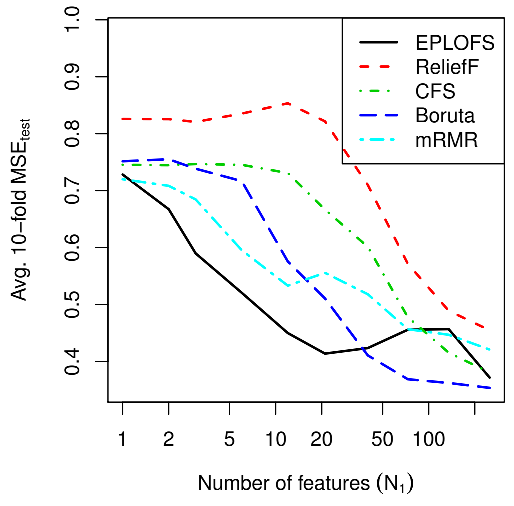


(b) With dependent features

Figure 7.6. Subset evaluation for Melting Point data ($N = 202$) for (a) original features (b) added dependent features.



(a) Original features



(b) With dependent features

Figure 7.7. Subset evaluation for Aquatic toxicity data ($N = 500$) for (a) original features (b) added dependent features.

7.1.4.2 Classification datasets

1. **Comf18 data** All methods are able to provide good subsets for this file. There is a slight separation in the curves when dependent features are added and EPLOFS, CFS and mRMR tend to perform better.
2. **Grng data** EPLOFS has one of the lowest error curves for this file for both the original feature set and one with dependent features added.
3. **Gongtrn data** In presence of dependent features, CFS performs the best.
4. **Breast cancer data** This is a difficult dataset due to the large number of features, and only 65 examples. EPLOFS is able to produce the best results on this dataset with only mRMR approaching its level of performance, although the subset size produced by EPLOFS is larger ($N_1 \approx 11$ compared to $N_1 \approx 5$).
5. **ADA data** All methods perform well on this dataset, but with dependent features, EPLOFS, mRMR, and CFS produce the lower error curves.

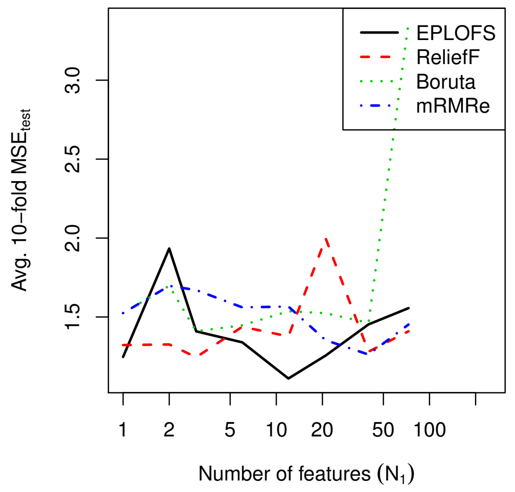
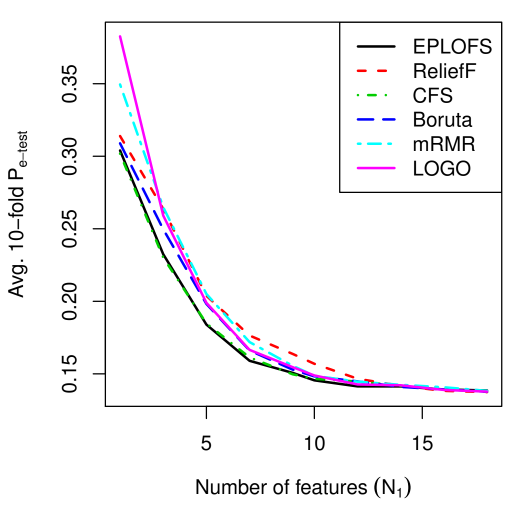
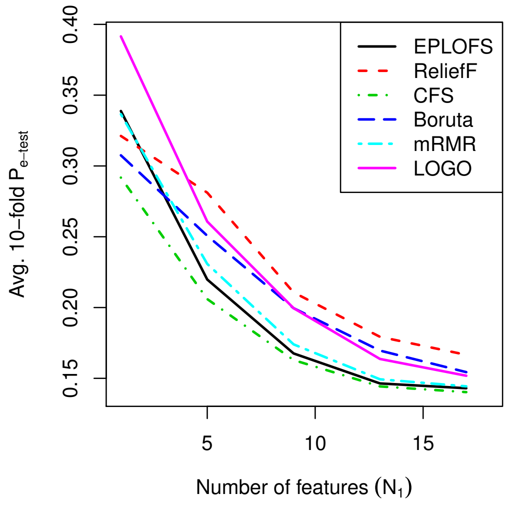


Figure 7.8. Subset evaluation for CGPS data ($N = 1000$).

6. **Madelon data** This is a synthetic dataset with 500 features, of which only five are really useful. Most methods are able to infer this and LOGO seems to be the best. Surprisingly, mRMR produces bad subsets on this data.
7. **Sylva data** Here EPLOFS performs well, but is beaten by mRMR in determining the minimal optimal subset.
8. **Gina data** EPLOFS finds the subset with the best error, but mRMR and CFS perform better for smaller subset sizes. Dependent features cause performance of LOGO and ReliefF to deteriorate.

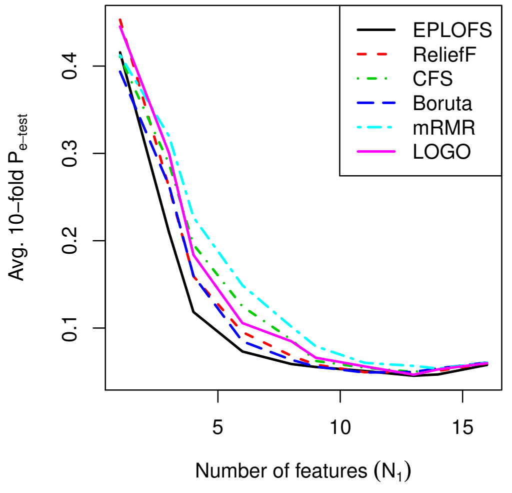


(a) Original features

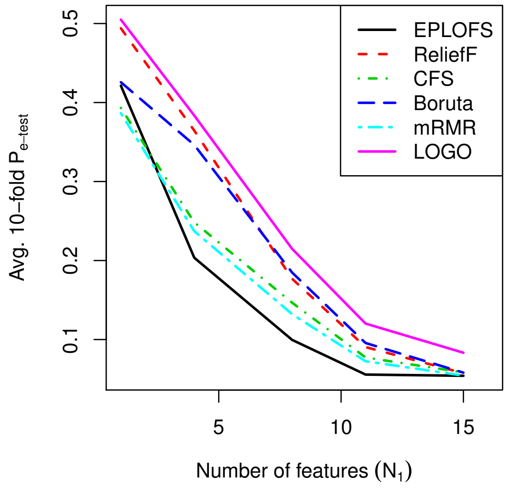


(b) With dependent features

Figure 7.9. Subset evaluation for Comf18 data ($N = 18$) for (a) original features (b) added dependent features.

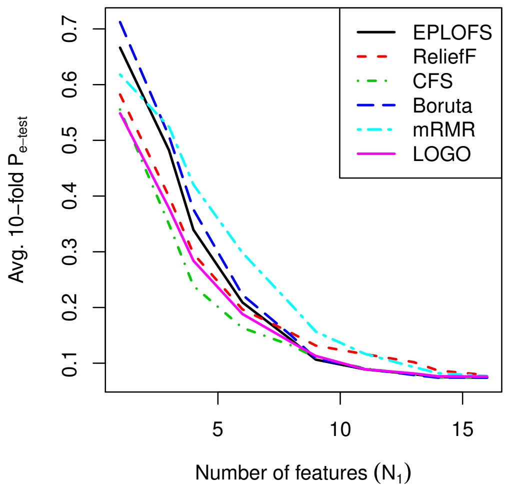


(a) Original features

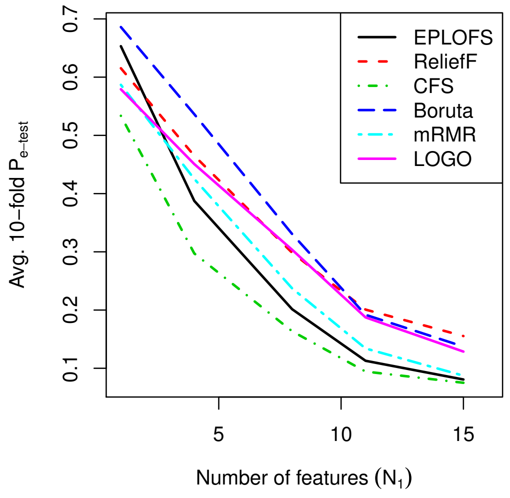


(b) With dependent features

Figure 7.10. Subset evaluation for Grng data ($N = 16$) for (a) original features (b) added dependent features.

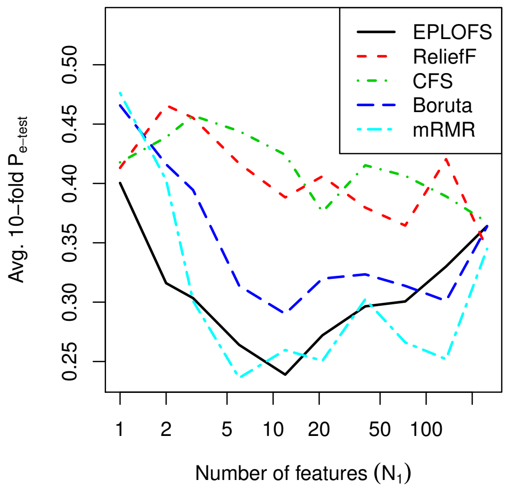


(a) Original features

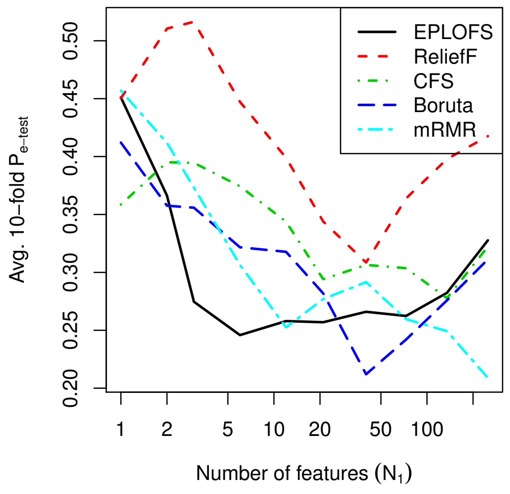


(b) With dependent features

Figure 7.11. Subset evaluation for Gongtrn data ($N = 16$) for (a) original features (b) added dependent features.

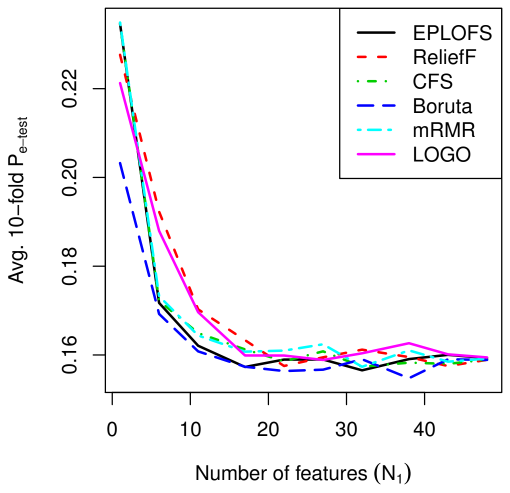


(a) Original features

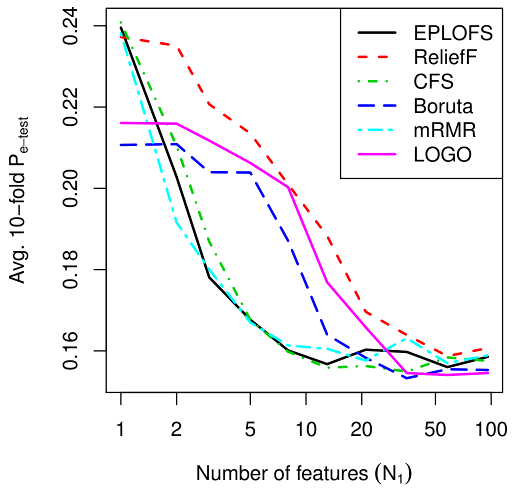


(b) With dependent features

Figure 7.12. Subset evaluation for Breast cancer data ($N = 989$) for (a) original features (b) added dependent features.

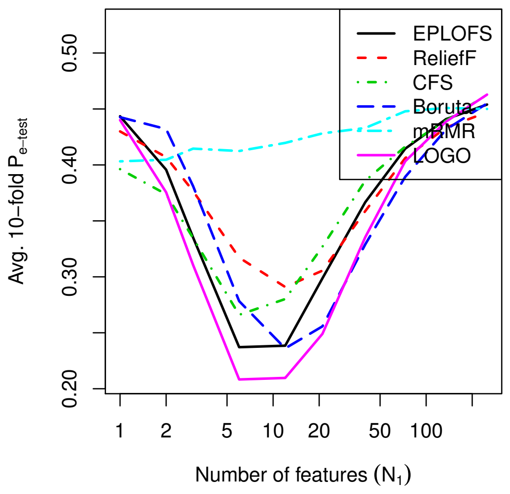


(a) Original features

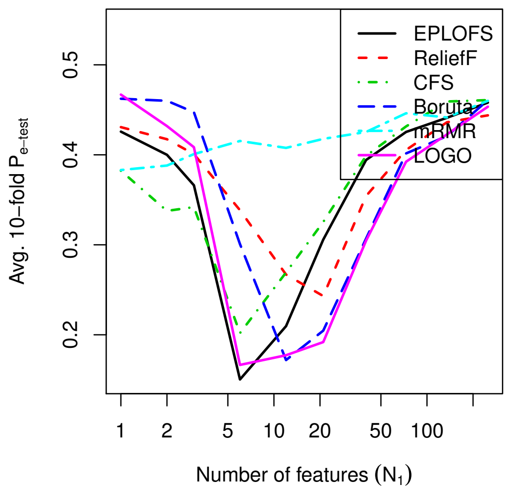


(b) With dependent features

Figure 7.13. Subset evaluation for ADA data ($N = 48$) for (a) original features (b) added dependent features.

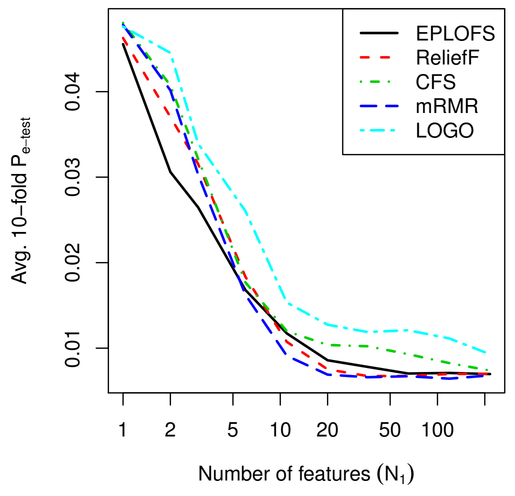


(a) Original features

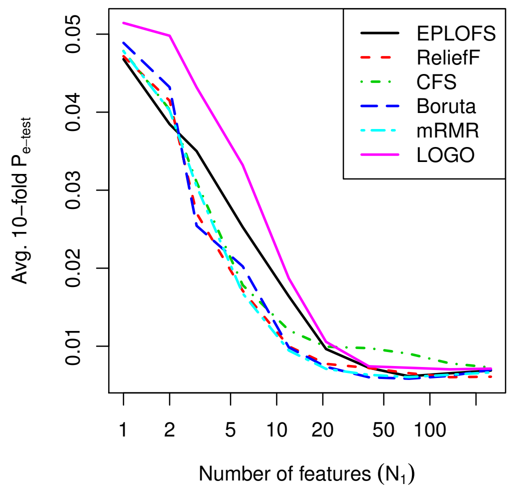


(b) With dependent features

Figure 7.14. Subset evaluation for Madelon data ($N = 500$) for (a) original features (b) added dependent features.

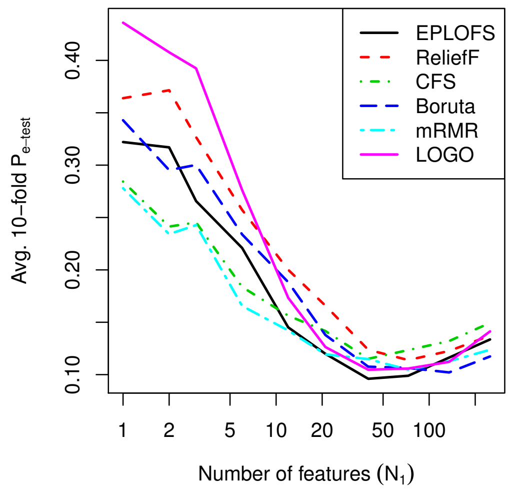


(a) Original features

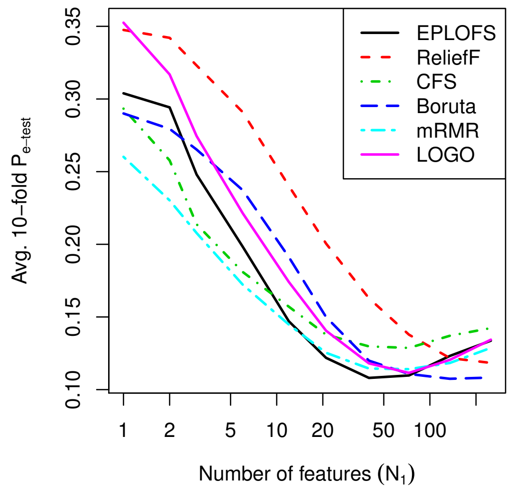


(b) With dependent features

Figure 7.15. Subset evaluation for Sylva data ($N = 216$) for (a) original features (b) added dependent features.



(a) Original features



(b) With dependent features

Figure 7.16. Subset evaluation for Gina data ($N = 970$) for (a) original features (b) added dependent features.

CHAPTER 8

Conclusions

In order to improve upon the SEF used by PLOFS we mapped the PLN to a SPLN. Then a second order embedded feature selection was used to generate improved distance measure weights. Next, a second order method for positioning center vectors was developed. The distance measure weights and improved center vectors are mapped back to the PLN, resulting in improved performance.

We analyze the behavior of noise and dependent features in OLS and use the results to develop a reliable method of eliminating these useless features, thereby extending PLOFS to problems with larger numbers of features. We augment the data with artificial random features as probes and use piecewise linear sequential forward search to identify the useless features and remove them from the data. A two-stage feature selection method has been developed which removes useless features and then generates subsets of different sizes of the remaining features using floating search. Our two stage feature selection approach helps eliminate the ill-effects of too many useless features in the final piecewise linear model allowing EPLOFS to be applicable to larger datasets.

Through the use of ten-fold testing errors in MLPs, we have evaluated EPLOFS and compared its performance to those of several other methods. In the presence of a large number of noise features, EPLOFS consistently produced the optimal subset with only the useful features and no noise features, while other methods struggled to find all the useful features and keep out the noise features. Subsets of various sizes produced by EPLOFS often have smaller testing errors compared to subsets of the same size produced by other methods. The presence of dependent features further deteriorated performance of filter methods while the performance of EPLOFS remained largely unaffected.

8.1 Future Work

In future work, we plan to implement an ensemble method to improve the robustness of our algorithm by running it across several cross-validation or bootstrap iterations and combining the feature selection results. We will also improve the PLN SEF by the same principle. We will also attempt to merge the feature importance determined from the distance measure weight optimization of the SPLN with our feature selection results to get better subsets. To further improve the SPLN, we will implement SPLN pruning, improved SPLN output weight training, and SPLN classifiers.

APPENDIX A

SPLN DERIVATIONS

A.1 Calculations for distance measure optimization

Taking the gradient of the error in equation (4.5) with respect to the distance measure weight change element $e_b(v)$,

$$\frac{\partial E}{\partial e_b\left(v\right)} = -\frac{2}{N_\mathrm{v}}\sum_{p=1}^{N_\mathrm{v}}\sum_{i=1}^{M}\left[t_p\left(i\right) - y_p\left(i\right)\right]\frac{\partial y_p\left(i\right)}{\partial e_b\left(v\right)} \tag{A.1}$$

where

$$\frac{\partial y_p\left(i\right)}{\partial e_b\left(v\right)} = \sum_{k=1}^{K}\frac{\partial \theta\left(k\right)}{\partial e_b\left(v\right)}y_{\mathrm{pk}}(i)$$

Here, we represent $d_p\left(k\right)$ with d_k to improve readability.

$$\frac{\partial \theta\left(k\right)}{\partial e_b\left(v\right)} = \frac{D\frac{\partial d_k^{-a}}{\partial e_b(v)} - d_k^{-a}\sum_{m=1}^{K}\frac{\partial d_m^{-a}}{\partial e_b(v)}}{D^2}$$

$$\frac{\partial d_k^{-a}}{\partial e_b\left(v\right)} = -\frac{a}{d_k^{a+1}}\frac{\partial d_k}{\partial e_b\left(v\right)}$$

$$\frac{\partial d_k}{\partial e_b\left(v\right)} = \left(x_p\left(v\right) - m_k\left(v\right)\right)^2$$

The elements of the Gauss-Newton Hessian matrix $\mathbf{H}_b$ are calculated as

$$h_b\left(u,v\right) = \frac{\partial^2 E}{\partial e_b\left(u\right)\ \partial e_b(v)} = \frac{2}{N_\mathrm{v}}\sum_{p=1}^{N_\mathrm{v}}\sum_{i=1}^{M}\frac{\partial y_p\left(i\right)}{\partial e_b\ \left(u\right)}\frac{\partial y_p\left(i\right)}{\partial e_b\ \left(v\right)} \tag{A.2}$$

A.2 Calculations for center vector optimization

The gradient of the SPLN error from equation (4.5) with respect to the u^{th} cluster's center vector element $\mathbf{m}_u(v)$ is calculated as:

$$g_m\left(u,v\right) = \frac{\partial E}{\partial m_u\left(v\right)} = -\frac{2}{N_\mathrm{v}}\sum_{p=1}^{N_\mathrm{v}}\sum_{i=1}^{M}\left[t_p\left(i\right) - y_p\left(i\right)\right]\frac{\partial y_p\left(i\right)}{\partial m_u(v)} \tag{A.3}$$

where

$$\frac{\partial y_p\left(i\right)}{\partial m_u(v)} = \sum_{k=1}^{K}\frac{\partial \theta\left(k\right)}{\partial m_u\left(v\right)}\ y_{\mathrm{pk}}(i)$$

$$\frac{\partial\theta\left(k\right)}{\partial m_u\left(v\right)}=\frac{\delta\left(u-k\right)\ D\frac{\partial d_u^{-a}}{\partial m_u(v)}\ \ -d_k^{-a}\frac{\partial d_u^{-a}}{\partial m_u(v)}}{D^2}$$

$$\frac{\partial d_u^{-a}}{\partial m_u\left(v\right)}=\frac{2\ a}{d_u^{a+1}}\ b(v)\left(x_p\left(v\right)-m_u\left(v\right)\right)$$

The elements of the Gauss-Newton Hessian matrix $\mathbf{H}_m$ are given as

$$h_m\left(u,v\right)=\frac{\partial^2 E}{\partial z_m\left(u\right)\ \partial z_m(v)}=\frac{2}{N_{\mathrm{v}}}\sum_{p=1}^{N_{\mathrm{v}}}\sum_{i=1}^{M}\frac{\partial y_p\left(i\right)}{\partial z_m(u)}\frac{\partial y_p\left(i\right)}{\partial z_m(v)} \tag{A.4}$$

and the gradient of the error with respect to the learning factor elements

$$g_{zm}\left(u\right)=\frac{\partial E}{\partial z_m\left(u\right)}=-\frac{2}{N_{\mathrm{v}}}\sum_{p=1}^{N_{\mathrm{v}}}\sum_{i=1}^{M}\left[t_p\left(i\right)-y_p\left(i\right)\right]\frac{\partial y_p\left(i\right)}{\partial z_m\left(u\right)} \tag{A.5}$$

where

$$\frac{\partial y_p\left(i\right)}{\partial z_m\left(u\right)}=\sum_{k=1}^{K}\frac{\partial\theta\left(k\right)}{\partial z_m\left(u\right)}y_{\mathrm{pk}}(i)$$

$$\frac{\partial\theta\left(k\right)}{\partial z_m(u)}=\frac{\delta\left(u-k\right)\ D\frac{\partial d_u^{-a}}{\partial z_m(u)}\ \ -d_k^{-a}\frac{\partial d_u^{-a}}{\partial z_m(u)}}{D^2}$$

$$\frac{\partial d_u^{-1}}{\partial z_m(u)}=\frac{2\ a}{d_u^{a+1}}\sum_{n=1}^{N}b(v)\left(x_p\left(v\right)-m_u\left(v\right)\right)g_m(u,v)$$

APPENDIX B

MEMORIZATION IN LINEAR NETWORKS

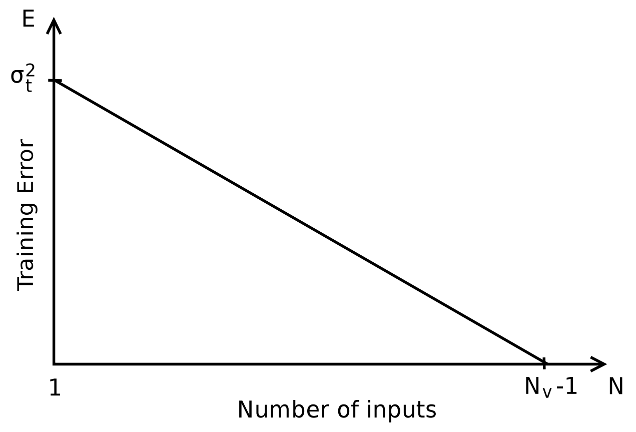


Figure B.1. Error as function of number of basis functions for noise inputs.

In this appendix, we investigate memorization in linear networks. Let $\mathbf{D}$ be a data file of random inputs of dimensions $N_{\mathrm{v}} \times N$ and $\mathbf{t}$ be the vector of desired outputs of dimensions $N_{\mathrm{v}} \times 1$. We create an augmented data file $\mathbf{D}_a$ of dimensions $N_{\mathrm{v}} \times (N+1)$ with an additional column set to one as

$$\mathbf{D}_a = [\mathbf{1} : \mathbf{D}] \tag{B.1}$$

The weight row vector $\mathbf{w}_a$ of dimensions $1 \times (N+1)$ connecting the inputs to the outputs satisfies

$$\mathbf{D}_a \mathbf{w}_a^T = \mathbf{t} \tag{B.2}$$

The plot of the error as the number of inputs is varied is shown in Fig. B.1. When the number of inputs $(N+1)$ exceeds N_{v}, the error goes to zero and memorization is seen.

Let $\mathbf{t} = \bar{\mathbf{t}} + \tilde{\mathbf{t}}$, where $\bar{\mathbf{t}}$ is the vector made of mean values of $\mathbf{t}$ and $\tilde{\mathbf{t}}$ is the remaining variable component. The first input data column that is made up of ones maps to the mean of the output vector. When solving for a solution to equation (B.2), for example, using orthogonal least squares, the bias input maps the $\bar{\mathbf{t}}$ component of $\mathbf{t}$, and the $\tilde{\mathbf{t}}$ component is mapped through equation (B.3).

$$[1 : \mathbf{D}]\, \mathbf{w}_a^T = \bar{\mathbf{t}} + \tilde{\mathbf{t}}$$

$$\mathbf{1}{\cdot}w_a(1) = \bar{\mathbf{t}}$$

$$\mathbf{D}\mathbf{w}^T = \tilde{\mathbf{t}} \tag{B.3}$$

where $\mathbf{w} = \mathbf{w}_a(2 \ldots N+1)$. When the matrix $\mathbf{D}$ is empty, the mean square error for the mapping in equation (B.2) becomes the variance of the output vector $\mathbf{t}$. This can be seen in Fig. B.1. Pre-multiplying both sides of equation (B.3) by $(1/N_\mathrm{v})\ \mathbf{D}^T$ gives

$$(1/N_\mathrm{v})\mathbf{D}^T\mathbf{D}\mathbf{w}^T = (1/N_\mathrm{v})\mathbf{D}^T\tilde{\mathbf{t}} \tag{B.4}$$

$$\mathbf{R}\mathbf{w}^T = \mathbf{c} \tag{B.5}$$

$$E = \frac{1}{N_\mathrm{v}}\left[\mathbf{D}\mathbf{w}^T - \tilde{\mathbf{t}}\right]^T\left[\mathbf{D}\mathbf{w}^T - \tilde{\mathbf{t}}\right]$$

which reduces to

$$E = E_t - 2\mathbf{w}\mathbf{c} + \mathbf{w}\mathbf{R}\mathbf{w}^T$$

where E_t is the variance of $\mathbf{t}$ given by $(1/N_\mathrm{v})\tilde{\mathbf{t}}^T\tilde{\mathbf{t}}$. When E is minimized with respect to $\mathbf{w}$, this equation reduces to

$$E = E_t - \mathbf{w}\,\mathbf{c} \tag{B.6}$$

Let the data matrix $\mathbf{D}$ have a singular value decomposition (SVD) of $\mathbf{D} = \mathbf{U\Sigma V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices of dimensions $N_\mathrm{v} \times N_\mathrm{v}$ and $N \times N$ respectively and $\mathbf{\Sigma}$ is a $N_\mathrm{v} \times N$ diagonal matrix. From (B.2), the matrix $\mathbf{R}$ can be rewritten as,

$$\mathbf{R} = (1/N_\mathrm{v})\mathbf{D}^T\mathbf{D}$$

$$= (1/N_\mathrm{v})(\mathbf{U\Sigma V}^T)^T(\mathbf{U\Sigma V}^T)$$

$$= (1/N_\mathrm{v})(\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T)(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)$$

$$= (1/N_\mathrm{v})(\mathbf{V}\mathbf{\Sigma}^T\mathbf{I}_{N_\mathrm{v}}\mathbf{\Sigma}\mathbf{V}^T)$$

which gives

$$\mathbf{R} = (1/N_\mathrm{v})(\mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{V}^T) \tag{B.7}$$

Similarly, the matrix $\mathbf{c}$ can be written as

$$\mathbf{c} = (1/N_\mathrm{v})\mathbf{D}^T\tilde{\mathbf{t}} = (1/N_\mathrm{v})(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)^T\tilde{\mathbf{t}}$$

$$\mathbf{c} = (1/N_\mathrm{v})(\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\tilde{\mathbf{t}}) \tag{B.8}$$

The solution for $\mathbf{w}$ from (B.2) can be written using (B.8) and (B.7) as

$$\mathbf{w}^T = \mathbf{R}^{-1}\mathbf{c}$$

$$= \{(1/N_\mathrm{v})(\mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{V}^T)\}^{-1}(1/N_\mathrm{v})(\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\tilde{\mathbf{t}})$$

$$= (\mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{V}^T)^{-1}(\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\tilde{\mathbf{t}})$$

$$= \mathbf{V}(\mathbf{\Sigma}^T\mathbf{\Sigma})^{-1}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\tilde{\mathbf{t}}$$

or

$$\mathbf{w}^T = \mathbf{V}(\mathbf{\Sigma}^T\mathbf{\Sigma})^{-1}\mathbf{\Sigma}^T\mathbf{U}^T\tilde{\mathbf{t}} \tag{B.9}$$

$\mathbf{\Sigma}$ is a rectangular matrix with a non-zero main diagonal. Let the square submatrix of $\mathbf{\Sigma}$ that forms this main diagonal be denoted by $\mathbf{\Sigma}_S$, so $\mathbf{\Sigma} = [\mathbf{\Sigma}_S^T : \mathbf{0}]^T$. Then the product $\mathbf{\Sigma}^T\mathbf{\Sigma}$ is the square diagonal matrix $\mathbf{\Sigma}_S^2$ of dimensions $N \times N$. Thus (B.9) can be written as

$$\mathbf{w}^T = \mathbf{V}(\mathbf{\Sigma}_S^2)^{-1}\mathbf{\Sigma}^T\mathbf{U}^T\tilde{\mathbf{t}}$$

$$\mathbf{w} = \tilde{\mathbf{t}}^T\mathbf{U}\mathbf{\Sigma}(\mathbf{\Sigma}_S^2)^{-1}\mathbf{V}^T \tag{B.10}$$

The expression for error in equation (B.6) can be written using (B.8) and (B.10) as

$$E = E_t - \mathbf{w}\mathbf{c}$$

$$= E_t - \tilde{\mathbf{t}}^T \mathbf{U\Sigma}(\mathbf{\Sigma}_S^2)^{-1}\mathbf{V}^T(1/N_\text{v})(\mathbf{V\Sigma}^T\mathbf{U}^T\tilde{\mathbf{t}})$$

$$= E_t - (1/N_\text{v})\tilde{\mathbf{t}}^T \mathbf{U\Sigma}(\mathbf{\Sigma}_S^2)^{-1}\mathbf{V}^T\mathbf{V\Sigma}^T\mathbf{U}^T\tilde{\mathbf{t}}$$

$$= E_t - (1/N_\text{v})\tilde{\mathbf{t}}^T \mathbf{U\Sigma}(\mathbf{\Sigma}_S^2)^{-1}\mathbf{\Sigma}^T\mathbf{U}^T\tilde{\mathbf{t}} \tag{B.11}$$

Consider the product

$$\mathbf{P} = \mathbf{U\Sigma}(\mathbf{\Sigma}_S^2)^{-1}\mathbf{\Sigma}^T\mathbf{U}^T = \mathbf{U\Sigma\Sigma}_S^{-1}\mathbf{\Sigma}_S^{-1}\mathbf{\Sigma}^T\mathbf{U}^T$$

$$= \mathbf{U}[\mathbf{\Sigma}_S^T : \mathbf{0}]^T\mathbf{\Sigma}_S^{-1}\mathbf{\Sigma}_S^{-1}[\mathbf{\Sigma}_S^T : 0]\mathbf{U}^T$$

$$= \mathbf{U}\mathbf{I}_{N_\text{v}N}\mathbf{I}_{NN_\text{v}}\mathbf{U}^T$$

$$= \mathbf{U}\mathbf{I}_Q\mathbf{U}^T$$

where $\mathbf{I}_Q$ is of dimensions $N_\text{v} \times N_\text{v}$ made up of the identity matrix in its top left $N \times N$ quadrant, and zero elsewhere. The product becomes

$$\mathbf{P} = \mathbf{U}\mathbf{I}_Q\mathbf{I}_Q\mathbf{U}^T$$

When $\mathbf{U}$ is post multiplied by $\mathbf{I}_Q$, Only the first N columns in $\mathbf{U}$ remain. Similarly, when $\mathbf{U}^T$ is pre-multiplied by $\mathbf{I}_Q$, only the first N rows in $\mathbf{U}^T$ remain. Thus, the product reduces to

$$\mathbf{P} = [\mathbf{u}_1, \mathbf{u}_2, ..\mathbf{u}_N : \mathbf{0}][\mathbf{u}_1, \mathbf{u}_2, ..\mathbf{u}_N : \mathbf{0}]^T$$

$$p\,(m,n) = \sum_{k=1}^{N} u\,(m,k)\,u(m,k)$$

Since the data matrix $\mathbf{D}$ has the SVD decomposition $\mathbf{D} = \mathbf{U\Sigma V}^T$, then $\mathbf{D} = \mathbf{U}'\mathbf{\Sigma}'\mathbf{V}'^T$ is also a valid SVD where $\mathbf{U}' = \mathbf{UT}$, $\mathbf{\Sigma}' = \mathbf{T\Sigma S}$, $\mathbf{V}' = \mathbf{SV}^T$, and $\mathbf{T}$ and $\mathbf{S}$ are unitary matrices. Given $\mathbf{T}$, $\mathbf{S}$ can be found so that $\mathbf{\Sigma}'$ is a diagonal matrix and has the singular values of $\mathbf{\Sigma}$ re-arranged.

Therefore,

$$E\,[p\,(m,n)] = \frac{1}{N_\text{v}}\sum_{i=1}^{N_\text{v}}\sum_{k=1}^{N} u'\,(m,i)\,u'\,(n,i)$$

$$= \frac{1}{N_{\mathrm{v}}} \sum_{k=1}^{N} \sum_{i=1}^{N_{\mathrm{v}}} u'(m,i)\, u'(n,i) = \frac{N}{N_{\mathrm{v}}} \delta(m,n)$$

so $E[\mathbf{P}] = (N/N_{\mathrm{v}})\mathbf{I}_{N_{\mathrm{v}}}$

Substituting $\mathbf{P}$ back in (B.6), we can write $E[E]$ as

$$E[E] = E_t - (1/N_{\mathrm{v}})\tilde{\mathbf{t}}^T E[\mathbf{P}]\tilde{\mathbf{t}}$$

$$= E_t - (1/N_{\mathrm{v}})\tilde{\mathbf{t}}^T (N/N_{\mathrm{v}})\, \mathbf{I}_{N_{\mathrm{v}}}\tilde{\mathbf{t}}$$

$$= E_t - N/N_{\mathrm{v}}^2 \tilde{\mathbf{t}}^T\tilde{\mathbf{t}}$$

$$= E_t - N/N_{\mathrm{v}} E_t$$

$$= E_t[1 - N/N_{\mathrm{v}}]$$

This is the expected error curve of Fig. B.1.

To test this result, we generated a data file with $N = 225$ inputs and $N_{\mathrm{v}} = 200$ rows, and trained linear networks with different number of inputs $1 \leq N_1 \leq N$. The error observed for the linear networks, and the one expected from our hypothesis are shown in Fig. B.2. The observed curve comes from an average of 10 trials on a data set with 200 Gaussian random features. We see that both are in close agreement.

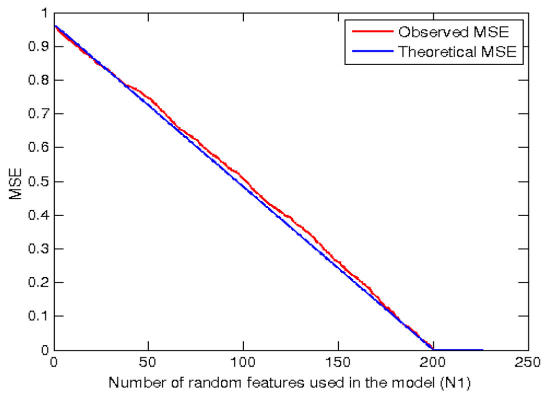


Figure B.2. Error as function of number of basis functions for noise inputs for simulated data.

APPENDIX C

DESCRIPTION OF DATASETS

C.1 Regression datasets

1. **Red and White Wine quality data sets**: The two datasets are related to red and white variants of the Portuguese "Vinho Verde" wine [107]. Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.).
2. **Two-D data set**: This training file is used in the task of inverting the surface scattering parameters from an inhomogeneous layer above a homogeneous half space, where both interfaces are randomly rough. The data file contains 2,768 patterns. It has eight inputs and seven outputs [92, 93].
3. **Housing dataset** This data comes from the UCI repository and is about housing prices in Boston suburbs [108]. It has 13 inputs and 1 output - the median value of homes in \$1000's.
4. **Oh7 Data**: This data set is given in [94]. The training set contains VV and HH polarization at L-band 30°, 40°, C-band 10°, 30°, 40°, 50°, 60°, and X-band 30°, 40°, 50° along with the corresponding unknowns rms surface height, surface correlation length, and volumetric soil moisture content in g/cm^3. The file has 20 inputs, 3 outputs and 10,453 training patterns.
5. **Melting Point Data**: This data set comes from [95] where a robust and general model is developed for the prediction of melting points. It has a diverse set of 4401 examples of compounds with 202 descriptors that capture molecular physicochemical and other graph-based properties. It is included in the R package QSARdata [109].
6. **Aquatic Toxicity Data**: These data were compiled and described by He and Jurs [110]. The data set consists of 322 compounds that were experimentally

assessed for toxicity. The inputs used here are the Moe2D molecular descriptors (220) of the compounds. The outcome is the negative log of activity.

7. **CGPS data** This dataset is a part of the large pharmacogenomic dataset published by Garnett et al. within the Cancer Genome Project (CGP). This dataset contains gene expression of 200 cancer cell lines for which sensitivity (IC50) to Camptothecin was measured (release 2). The inputs are expressions of 1000 genes; cell lines in rows, genes in columns and the output is drug sensitivity measurements (IC50) for Camptothecin. It is available in the R package mRMRE [105].
8. **Three Spirals Data**: This is a synthetic data set used in [111], which consists of three two-dimensional spirals, each labelled for a different class. It was converted to a regression problem by decoding the classes as binary outputs. It is available at [112].
9. **Concrete Data**: This data set predicts compressive strength of high performance concrete from its components and age. It comprises of eight inputs, the first seven being the quantities of cement, slag, fly ash, water, superplasticizer, coarse aggregate, and fine aggregate in Kg/m^3. The eighth input is age in days. The output variable is the compressive strength in MPa [96].

C.2 Classification datasets

1. **COMF18 data**: The training data file is generated from segmented images. Each segmented region is separately histogram equalized to 20 levels. Then the joint probability density of pairs of pixels separated by a given distance and a given direction is estimated. We use 0, 90, 180, 270 degrees for the directions and 1, 3, and 5 pixels for the separations. The density estimates are computed for each classification window. For each separation, the co-occurrences for for

the four directions are folded together to form a triangular matrix. From each of the resulting three matrices, six features are computed: angular second moment, contrast, entropy, correlation, and the sums of the main diagonal and the first off diagonal. This results in 18 features for each classification window [113].

2. **Grng data set**: The geometric shape recognition data file consists of four geometric shapes, ellipse, triangle, quadrilateral, and pentagon [114]. Each shape consists of a matrix of size 64×64. For each shape, 200 training patterns were generated using different degrees of deformation. The deformations included rotation, scaling, translation, and oblique distortions. The feature set is ring-wedge energy, and has 16 features.
3. **Gongtrn data set**: The raw data consists of images from hand printed numerals collected from 3,000 people by the Internal Revenue Service. We randomly chose 300 characters from each class to generate 3,000 character training data. Images are 32 by 24 binary matrices. An image scaling algorithm is used to remove size variation in characters. The feature set contains 16 elements. The 10 classes correspond to 10 Arabic numerals. For more details concerning the features, see [115].
4. **Breast cancer spectroscopy data**: This data has near infrared spectroscopy (NIRS) data for 989 wavelengths from ex-vivo breast tissue. The measured spectral range was from 475-1100 nm with a spectral resolution of ~4 nm (using 100 um slit width) [116]. The tissue is labeled as cancer (invasive ductal carcinoma) or non-cancer. It has 62 observations.
5. **ADA database** The task of ADA is to discover high revenue people from census data. This is a two-class classification problem. The raw data from the census bureau is known as the Adult database in the UCI machine-learning repository. The 14 original attributes (features) include age, workclass, edu-

cation, education, marital status, occupation, native country, etc. This data comes from the NIPS2007 agnostic learning challenge [117]. It has 48 inputs and 2 output classes.

6. **Madelon data**: MADELON is an artificial dataset, which was part of the NIPS 2003 feature selection challenge. This is a two-class classification problem with continuous input variables. The difficulty is that the problem is multivariate and highly non-linear. It has 500 attributes and 4400 examples. It is an artificial dataset containing data points grouped in 32 clusters placed on the vertices of a five dimensional hypercube and randomly labeled $+1$ or -1. The five dimensions constitute 5 informative features. 15 linear combinations of those features were added to form a set of 20 (redundant) informative features. Based on those 20 features one must separate the examples into the 2 classes (corresponding to the two labels). We added a number of distractor feature called 'probes' having no predictive power. The order of the features and patterns were randomized [118].

7. **SYLVA data**: The task of SYLVA is to classify forest cover types. The forest cover type for 30×30 meter cells is obtained from US Forest Service (USFS) Region 2 Resource Information System (RIS) data. We brought it back to a two-class classification problem (classifying Ponderosa pine vs. everything else). The agnostic learning track data consists in 216 input variables. Each pattern is composed of 4 records: 2 true records matching the target and 2 records picked at random. Thus half of the features are distracters [118].

8. **Gina data**: This data comes from the NIPS2007 agnostic learning challenge [117]. The task of GINA is handwritten digit recognition. It presents the problem of separating two-digit odd numbers from two-digit even numbers. Only the unit digit is informative for that task, therefore at least half of the features

are distracters. Additionally, the pixels that are almost always blank were removed and the pixel order was randomized to hide the feature identity. This is a two class classification problem with sparse continuous input variables, in which each class is composed of several clusters. It is a problems with heterogeneous classes.

REFERENCES

[1] I. Guyon and A. Elisseeff, “An introduction to variable and feature selection,” *The Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[2] Y. Sun, S. Todorovic, and S. Goodison, “Local-Learning-Based Feature Selection for High-Dimensional Data Analysis,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1610–1626, Sept. 2010.

[3] A. Yennu, R. Rawat, M. T. Manry, R. Gatchel, and H. Liu, “Investigation of human frontal cortex under noxious thermal stimulation of temporo-mandibular joint using functional near infrared spectroscopy,” in *SPIE BiOS*. International Society for Optics and Photonics, 2013, pp. 857 804–857 804. [Online]. Available: http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=1672620

[4] M. Eklund, U. Norinder, S. Boyer, and L. Carlsson, “Choosing feature selection and learning algorithms in qsar,” *Journal of chemical information and modeling*, vol. 54, no. 3, pp. 837–843, 2014.

[5] C.-F. Lin, A. K. LeBoulluec, L. Zeng, V. C. P. Chen, and R. J. Gatchel, “A decision-making framework for adaptive pain management,” *Health Care Management Science*, vol. 17, no. 3, pp. 270–283, Aug. 2013. [Online]. Available: http://link.springer.com/article/10.1007/s10729-013-9252-0

[6] A. K. LeBoulluec, L. Zeng, V. C. Chen, J. M. Rosenberger, and R. J. Gatchel, “Outcome and State Transition Modeling for Adaptive Interdisciplinary Pain Management,” in *IIE Annual Conference. Proceedings*. Institute of Industrial Engineers-Publisher, 2013, p. 1400. [Online]. Available: http://search.proquest.com/openview/9adec6c9adeb981d88deb26bc7fe3460/1?pq-origsite=gscholar

[7] R. Rawat, K. Vora, M. Manry, and G. Eapi, “Multi-variable Neural Network Forecasting Using Two Stage Feature Selection,” in *2014 13th International Conference on Machine Learning and Applications (ICMLA)*, Dec. 2014, pp. 243–250.

[8] R. Bellman, “Dynamic Programming, Princeton,” *NJ: Princeton UP*, 1957.

[9] S.-M. Zhou and J. Q. Gan, “Low-level interpretability and high-level interpretability: a unified view of data-driven interpretable fuzzy system modelling,” *Fuzzy Sets and Systems*, vol. 159, no. 23, pp. 3091–3131, 2008.

[10] R. Daz-Uriarte and S. A. d. Andrs, “Gene selection and classification of microarray data using random forest,” *BMC Bioinformatics*, vol. 7, no. 1, p. 3, Jan. 2006. [Online]. Available: http://www.biomedcentral.com/1471-2105/7/3/abstract

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[12] H. Lee, P. Pham, Y. Largman, and A. Y. Ng, “Unsupervised feature learning for audio classification using convolutional deep belief networks,” in *Advances in neural information processing systems*, 2009, pp. 1096–1104.

[13] H.-F. Yu, H.-Y. Lo, H.-P. Hsieh, J.-K. Lou, T. G. McKenzie, J.-W. Chou, P.-H. Chung, C.-H. Ho, C.-F. Chang, Y.-H. Wei, *et al.*, “Feature engineering and classifier ensemble for kdd cup 2010,” in *Proceedings of the KDD Cup 2010 Workshop*, 2010, pp. 1–16.

[14] S. Scott and S. Matwin, “Feature engineering for text classification,” in *ICML*, vol. 99, 1999, pp. 379–388.

[15] R. Kohavi and G. H. John, “Wrappers for feature subset selection,” *Artificial intelligence*, vol. 97, no. 1, pp. 273–324, 1997.

[16] C. Yu, M. T. Manry, J. Li, and P. Lakshmi Narasimha, "An efficient hidden layer training method for the multilayer perceptron," *Neurocomputing*, vol. 70, no. 13, pp. 525–535, Dec. 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231206000476

[17] J. R. Vergara and P. A. Estvez, "A review of feature selection methods based on mutual information," *Neural Computing and Applications*, vol. 24, no. 1, pp. 175–186, 2014. [Online]. Available: http://link.springer.com/article/10.1007/s00521-013-1368-0

[18] Y. Zhang, C. Ding, and T. Li, "Gene selection algorithm by combining reliefF and mRMR," *BMC Genomics*, vol. 9, no. Suppl 2, p. S27, Sept. 2008. [Online]. Available: http://www.biomedcentral.com/1471-2164/9/S2/S27/abstract

[19] A. Gallant and H. White, "There exists a neural network that does not make avoidable mistakes," in *, IEEE International Conference on Neural Networks, 1988*, July 1988, pp. 657–664 vol.1.

[20] M. A. Hall, "Correlation-based feature selection for machine learning," Ph.D. dissertation, The University of Waikato, 1999. [Online]. Available: https://www.lri.fr/~pierres/donn%E9es/save/these/articles/lpr-queue/hall99correlationbased.pdf

[21] D. Koller and M. Sahami, "Toward Optimal Feature Selection," Feb. 1996. [Online]. Available: http://ilpubs.stanford.edu:8090/208/

[22] C. Ding and H. Peng, "Minimum redundancy feature selection from microarray gene expression data," *Journal of bioinformatics and computational biology*, vol. 3, no. 02, pp. 185–205, 2005. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/s0219720005001004

[23] Y. Sun, "Iterative RELIEF for Feature Weighting: Algorithms, Theories, and Applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1035–1051, June 2007.

[24] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik, "Feature selection for SVMs," in *NIPS*, vol. 12. Citeseer, 2000, pp. 668–674. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.470.3628&rep=rep1&type=pdf

[25] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene Selection for Cancer Classification using Support Vector Machines," *Machine Learning*, vol. 46, no. 1-3, pp. 389–422, Jan. 2002. [Online]. Available: http://link.springer.com/article/10.1023/A%3A1012487302797

[26] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996. [Online]. Available: http://www.jstor.org/stable/2346178

[27] J. H. Friedman, "Multivariate adaptive regression splines," *The annals of statistics*, pp. 1–67, 1991. [Online]. Available: http://www.jstor.org/stable/2241837

[28] K. Fukunaga, *Introduction to statistical pattern recognition.* Academic Press, New York, 1990.

[29] J. Kittler and others, "Feature set search algorithms," *Pattern recognition and signal processing*, pp. 41–60, 1978.

[30] P. Pudil, J. Novoviov, and J. Kittler, "Floating search methods in feature selection," *Pattern recognition letters*, vol. 15, no. 11, pp. 1119–1125, 1994.

[31] J. Li, M. T. Manry, P. L. Narasimha, and C. Yu, "Feature selection using a piecewise linear network," *IEEE Transactions on Neural Networks*, vol. 17, no. 5, pp. 1101–1115, 2006.

[32] H. Peng, F. Long, and C. Ding, “Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, Aug. 2005.

[33] H. Yuan, S.-S. Tseng, W. Gangshan, and Z. Fuyan, “A two-phase feature selection method using both filter and wrapper,” in *1999 IEEE International Conference on Systems, Man, and Cybernetics, 1999. IEEE SMC ’99 Conference Proceedings*, vol. 2, 1999, pp. 132–136 vol.2.

[34] Y. B. Kim and J. Gao, “Unsupervised gene selection for high dimensional data,” in *Sixth IEEE Symposium on BioInformatics and BioEngineering (BIBE’06)*. IEEE, 2006, pp. 227–234.

[35] P. M. Narendra and K. Fukunaga, “A branch and bound algorithm for feature subset selection,” *Computers, IEEE Transactions on*, vol. 100, no. 9, pp. 917–922, 1977.

[36] S. D. Stearns, “On selecting features for pattern classifiers,” in *Proceedings of the 3rd International Joint Conference on Pattern Recognition*, 1976, pp. 71–75.

[37] R. A. Fisher, “The use of multiple measurements in taxonomic problems,” *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[38] I. Kononenko, E. imec, and M. Robnik-ikonja, “Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF,” *Applied Intelligence*, vol. 7, no. 1, pp. 39–55, Jan. 1997. [Online]. Available: http://link.springer.com/article/10.1023/A%3A1008280620621

[39] K. Kira and L. A. Rendell, “The feature selection problem: Traditional methods and a new algorithm,” in *AAAI*, vol. 2, 1992, pp. 129–134. [Online]. Available: http://www.aaai.org/Library/AAAI/1992/aaai92-020.php

[40] M. Robnik-ikonja and I. Kononenko, “Theoretical and empirical analysis of ReliefF and RReliefF,” *Machine learning*, vol. 53, no. 1-2, pp. 23–69, 2003. [Online]. Available: http://link.springer.com/article/10.1023/A:1025667309714

[41] J.-Y. Wang, J. Yao, and Y. Sun, “Semi-supervised local-learning-based feature selection,” in *2014 International Joint Conference on Neural Networks (IJCNN)*, July 2014, pp. 1942–1948.

[42] K. Kira and L. A. Rendell, “A practical approach to feature selection,” in *Proceedings of the ninth international workshop on Machine learning*, 1992, pp. 249–256. [Online]. Available: https://books.google.com/books?hl=en&lr=&id=jEejBQAAQBAJ&oi=fnd&pg=PA249&dq=kira+relief&ots=EQ3zgnNj5q&sig=CmlCpQd4cITn75osbONPzOSWLQk

[43] M. Kursa and W. Rudnicki, “Feature Selection with the Boruta Package,” *Journal of Statistical Software*, vol. 36, no. 1, pp. 1–13, 2010. [Online]. Available: http://www.jstatsoft.org/index.php/jss/article/view/v036i11

[44] M. B. Kursa and W. R. Rudnicki, “Boruta: Wrapper Algorithm for All-Relevant Feature Selection,” Dec. 2014. [Online]. Available: http://cran.r-project.org/web/packages/Boruta/index.html

[45] H. Stoppiglia, G. Dreyfus, R. Dubois, and Y. Oussar, “Ranking a random feature for variable and feature selection,” *The Journal of Machine Learning Research*, vol. 3, pp. 1399–1414, 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=944980

[46] I. Inza, P. Larrañaga, R. Blanco, and A. J. Cerrolaza, “Filter versus wrapper gene selection approaches in dna microarray domains,” *Artificial intelligence in medicine*, vol. 31, no. 2, pp. 91–103, 2004.

[47] S. Aitken, T. Jirapech-Umpai, and R. Daly, "Inferring gene regulatory networks from classified microarray data: Initial results," *BMC Bioinformatics*, vol. 6, no. 3, p. 1, 2005.

[48] A. Bordes, S. Ertekin, J. Weston, and L. Bottou, "Fast kernel classifiers with online and active learning," *Journal of Machine Learning Research*, vol. 6, no. Sep, pp. 1579–1619, 2005.

[49] P. Lin, J. Zhang, and R. An, "Data dimensionality reduction approach to improve feature selection performance using sparsified SVD," in *2014 International Joint Conference on Neural Networks (IJCNN)*, July 2014, pp. 1393–1400.

[50] E. Tuv, A. Borisov, G. Runger, and K. Torkkola, "Feature Selection with Ensembles, Artificial Variables, and Redundancy Elimination," *J. Mach. Learn. Res.*, vol. 10, pp. 1341–1366, Dec. 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1577069.1755828

[51] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, "Classification and regression trees." *Wadsworth, Belmont, CA*, 1984.

[52] J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, Oct. 2001. [Online]. Available: http://www.jstor.org/stable/2699986

[53] R. Rawat, "An Efficient Piecewise Linear Network," Master's thesis, UNIVERSITY OF TEXAS AT ARLINGTON, 2009. [Online]. Available: http://hcstt-mlp-pln.googlecode.com/svn/trunk/CD/REF/Rawat_uta_2502M_10503.pdf

[54] H. Chandrasekaran, J. Li, W. H. Delashmit, P. L. Narasimha, C. Yu, and M. T. Manry, "Convergent design of piecewise linear neural networks," *Neurocomputing*, vol. 70, no. 4, pp. 1022–1039, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231206002372

[55] T. Kohonen, “The self-organizing map,” *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[56] D. Arthur and S. Vassilvitskii, “K-means++: The Advantages of Careful Seeding,” in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA ’07. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035. [Online]. Available: http://dl.acm.org/citation.cfm?id=1283383.1283494

[57] S. CHEN, S. A. BILLINGS, and W. LUO, “Orthogonal least squares methods and their application to non-linear system identification,” *International Journal of Control*, vol. 50, no. 5, pp. 1873–1896, Nov. 1989. [Online]. Available: http://dx.doi.org/10.1080/00207178908953472

[58] J. W. Dettman, *Mathematical methods in physics and engineering.* Dover Publications, 1988.

[59] F. J. Maldonado and M. T. Manry, “Optimal pruning of feedforward neural networks based upon the Schmidt procedure,” in *Asilomar Conference on Signals Systems and Computers*, vol. 2. IEEE; 1998, 2002, pp. 1024–1028.

[60] Q. Du, V. Faber, and M. Gunzburger, “Centroidal Voronoi Tessellations: Applications and Algorithms,” *SIAM Review*, vol. 41, no. 4, pp. 637–676, Jan. 1999.

[61] S. Aksoy, R. Haralick, F. Cheikh, and M. Gabbouj, “A weighted distance approach to relevance feedback,” in *International Conference on Pattern Recognition*, vol. 15, 2000, pp. 812–815.

[62] A. N. Tikhonov and V. I. Arsenin, *Solutions of ill-posed problems.* Winston, 1977.

[63] S. Z. Selim and M. A. Ismail, "K-means-type algorithms: A generalized convergence theorem and characterization of local optimality," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 1, pp. 81–87, 1984.

[64] G. W. Milligan and M. C. Cooper, "An examination of procedures for determining the number of clusters in a data set," *Psychometrika*, vol. 50, no. 2, pp. 159–179, 1985.

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

[66] R. P. Lippmann, "Neural networks, bayesian a posteriori probabilities, and pattern classification," in *From Statistics to Neural Networks.* Springer, 1994, pp. 83–104.

[67] D. W. Ruck, S. K. Rogers, M. Kabrisky, M. E. Oxley, and B. W. Suter, "The multilayer perceptron as an approximation to a Bayes optimal discriminant function," *Neural Networks, IEEE Transactions on*, vol. 1, no. 4, pp. 296–298, 1990.

[68] T. Fujisawa and E. S. Kuh, "Piecewise-linear theory of nonlinear networks," *SIAM Journal on Applied Mathematics*, vol. 22, no. 2, pp. 307–328, 1972. [Online]. Available: http://dx.doi.org/10.1137/0122030

[69] M.-J. Chien and E. Kuh, "Solving nonlinear resistive networks using piecewise-linear analysis and simplicial subdivision," *IEEE Transactions on Circuits and Systems*, vol. 24, no. 6, pp. 305–317, Jun 1977.

[70] J. Li, "Subset Selection for Local Processors: Methodologies and Applications," PhD Dissertation in Electrical Engineering, University of Texas at Arlington, Aug. 2004.

[71] M. Al-Abed, M. Manry, J. R. Burk, E. A. Lucas, and K. Behbehani, "A method to detect obstructive sleep apnea using neural network classification of time-

frequency plots of the heart rate variability," in *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, 2007, pp. 6101–6104.

[72] J. Li, J. Yao, R. M. Summers, N. Petrick, M. T. Manry, and A. K. Hara, "An efficient feature selection algorithm for computer-aided polyp detection," *International Journal on Artificial Intelligence Tools*, vol. 15, no. 06, pp. 893–915, 2006.

[73] P. L. Narasimha, M. T. Manry, and F. Maldonado, "Upper bound on pattern storage in feedforward networks," *Neurocomputing*, vol. 71, no. 1618, pp. 3612–3616, Oct. 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231208002506

[74] M. L. Raymer, W. F. Punch, E. D. Goodman, L. A. Kuhn, and A. K. Jain, "Dimensionality reduction using genetic algorithms," *Evolutionary Computation, IEEE Transactions on*, vol. 4, no. 2, pp. 164–171, 2000.

[75] J. Fan and R. Li, "Statistical challenges with high dimensionality: Feature selection in knowledge discovery," *arXiv preprint math/0602133*, 2006. [Online]. Available: http://arxiv.org/abs/math/0602133

[76] J. Fan, Y. Fan, and J. Lv, "High dimensional covariance matrix estimation using a factor model," *Journal of Econometrics*, vol. 147, no. 1, pp. 186–197, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304407608001346

[77] R. J. Samworth, "Optimal weighted nearest neighbour classifiers," *The Annals of Statistics*, vol. 40, no. 5, pp. 2733–2763, Oct. 2012. [Online]. Available: http://projecteuclid.org/euclid.aos/1359987536

[78] S. Bandyopadhyay and U. Maulik, "An evolutionary technique based on K-Means algorithm for optimal clustering in RN," *Information Sciences*, vol. 146, no. 1-4, pp. 221–237, 2002.

[79] B. Hammer and T. Villmann, "Generalized relevance learning vector quantization," *Neural Networks*, vol. 15, no. 8, pp. 1059–1068, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608002000795

[80] A. J. Shepherd, *Second-order methods for neural networks: Fast and reliable training methods for multi-layer perceptrons.* Springer Science & Business Media, 2012.

[81] X. Cai, K. Tyagi, and M. T. Manry, "An optimal construction and training of second order RBF network for approximation and illumination invariant image segmentation," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*, 2011, pp. 3120–3126.

[82] M. Lichman, *UCI Machine Learning Repository.* University of California, Irvine, School of Information and Computer Sciences, 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[83] R. Turner, *deldir: Delaunay Triangulation and Dirichlet (Voronoi) Tessellation.* CRAN, 2016, r package version 0.1-12. [Online]. Available: https://CRAN.R-project.org/package=deldir

[84] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of applied mathematics*, vol. 2, no. 2, pp. 164–168, 1944.

[85] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[86] J. M. Ortega and W. C. Rheinboldt, *Iterative solution of nonlinear equations in several variables.* Siam, 1970, vol. 30.

[87] Y.-J. Wang and C.-T. Lin, "A second-order learning algorithm for multilayer networks based on block hessian matrix," *Neural Networks*, vol. 11, no. 9, pp. 1607–1622, 1998.

[88] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the marquardt algorithm," *IEEE transactions on Neural Networks*, vol. 5, no. 6, pp. 989–993, 1994.

[89] Z. Q. Luo and P. Tseng, "On the convergence of the coordinate descent method for convex differentiable minimization," *Journal of Optimization Theory and Applications*, vol. 72, no. 1, pp. 7–35, 1992. [Online]. Available: http://dx.doi.org/10.1007/BF00939948

[90] G. Chen and M. Teboulle, "A proximal-based decomposition method for convex minimization problems," *Mathematical Programming*, vol. 64, no. 1, pp. 81–101, 1994. [Online]. Available: http://dx.doi.org/10.1007/BF01582566

[91] T. H. Cormen, *Introduction to algorithms.* MIT press, 2009.

[92] M. S. Dawson, J. Olvera, A. K. Fung, and M. T. Manry, "Inversion of surface parameters using fast learning neural networks," in *IGARSS'92*, 1992, pp. 910–912.

[93] M. S. Dawson, A. K. Fung, and M. T. Manry, "Surface parameter retrieval using fast learning neural networks," *Remote Sensing Reviews*, vol. 7, no. 1, pp. 1–18, 1993.

[94] Y. Oh, K. Sarabandi, and F. T. Ulaby, "An empirical model and an inversion technique for radar scattering from bare soil surfaces," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 30, no. 2, pp. 370–381, 1992.

[95] M. Karthikeyan, R. C. Glen, and A. Bender, "General melting point prediction based on a diverse compound data set and artificial neural networks," *Journal*

of chemical information and modeling, vol. 45, no. 3, pp. 581–590, 2005. [Online]. Available: http://pubs.acs.org/doi/abs/10.1021/ci0500132

[96] I.-C. Yeh, "Modeling of strength of high-performance concrete using artificial neural networks," *Cement and Concrete research*, vol. 28, no. 12, pp. 1797–1808, 1998. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0008884698001653

[97] R. Rawat and M. Manry, "Second order training of a smoothed piecewise linear network (in publication)," *Neural Processing Letters*, 2016.

[98] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artificial Intelligence*, vol. 97, no. 1, pp. 245 – 271, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370297000635

[99] R. Rawat, J. Patel, and M. Manry, "Minimizing validation error with respect to network size and number of training epochs," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, Aug. 2013, pp. 1–7.

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

[101] M. Robnik-Sikonja and P. S. with contributions from John Adeyanju Alao, *CORElearn: Classification, Regression and Feature Evaluation*, 2016, r package version 1.48.0. [Online]. Available: https://CRAN.R-project.org/package=CORElearn

[102] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[103] H. K, B. C, and Z. A, *Open-Source Machine Learning: R Meets Weka*. Computational Statistics, 2009, vol. 24. [Online]. Available: http://doi.org/10.1007/s00180-008-0119-7⟩

[104] P. Romanski and L. Kotthoff, *FSelector: Selecting Attributes*, 2016, r package version 0.21. [Online]. Available: https://CRAN.R-project.org/package=FSelector

[105] N. De Jay, S. Papillon-Cavanagh, C. Olsen, N. El-Hachem, G. Bontempi, and B. Haibe-Kains, "mrmre: an r package for parallelized mrmr ensemble feature selection," *Bioinformatics*, vol. 29, no. 18, pp. 2365–2368, 2013.

[106] T. Hothorn, K. Hornik, and A. Zeileis, "Unbiased recursive partitioning: A conditional inference framework," *Journal of Computational and Graphical statistics*, vol. 15, no. 3, pp. 651–674, 2006.

[107] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decision Support Systems*, vol. 47, no. 4, pp. 547–553, Nov. 2009.

[108] D. Harrison and D. L. Rubinfeld, "Hedonic housing prices and the demand for clean air," *Journal of environmental economics and management*, vol. 5, no. 1, pp. 81–102, 1978. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0095069678900062

[109] M. Kuhn, *QSARdata: Quantitative Structure Activity Relationship (QSAR) Data Sets*, 2013, r package version 1.3. [Online]. Available: https://CRAN.R-project.org/package=QSARdata

[110] L. He and P. C. Jurs, "Assessing the reliability of a QSAR model's predictions," *Journal of Molecular Graphics and Modelling*, vol. 23, no. 6, pp. 503–523, June 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1093326305000173

[111] H. Chang and D.-Y. Yeung, "Robust path-based spectral clustering," *Pattern Recognition*, vol. 41, no. 1, pp. 191–203, Jan. 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320307002038

[112] Speech and I. P. Unit, "Clustering datasets," 2016. [Online]. Available: http://cs.joensuu.fi/sipu/datasets/

[113] R. R. Bailey, E. J. Pettit, R. T. Borochoff, M. T. Manry, and X. Jiang, "Automatic recognition of usgs land use/cover categories using statistical and neural network classifiers," in *Optical Engineering and Photonics in Aerospace Sensing.* International Society for Optics and Photonics, 1993, pp. 185–195.

[114] H.-C. Yau and M. T. Manry, "Iterative improvement of a nearest neighbor classifier," *Neural Networks*, vol. 4, no. 4, pp. 517–524, 1991.

[115] M. T. Manry, "Non-gaussian feature analyses using a neural network," *Progress in Neural Networks*, vol. 2, p. 253, 1994.

[116] V. Sharma, S. Shivalingaiah, Y. Peng, D. Euhus, Z. Gryczynski, and H. Liu, "Auto-fluorescence lifetime and light reflectance spectroscopy for breast cancer diagnosis: potential tools for intraoperative margin detection," *Biomedical optics express*, vol. 3, no. 8, pp. 1825–1840, 2012.

[117] I. Guyon, A. Saffari, G. Dror, and G. Cawley, "Agnostic learning vs. prior knowledge challenge," in *2007 International Joint Conference on Neural Networks.* IEEE, 2007, pp. 829–834.

[118] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror, "Result analysis of the nips 2003 feature selection challenge," in *Advances in neural information processing systems*, 2004, pp. 545–552.

BIOGRAPHICAL STATEMENT

Rohit Rawat was born in New Delhi, India, in 1986. He received his bachelor's degree in electronics and communication engineering from Indraprastha University, New Delhi, India, in 2007. He obtained his master's and doctorate degrees in electrical engineering from The University of Texas at Arlington (UTA) in 2009 and 2016, respectively. He worked for Motorola as an intern in the summers of 2008 and 2009, where he worked on developing embedded software. He has also taught undergraduate courses in signal processing at UTA. He served as the president of the Linux user group at UTA for several years and actively promoted and tutored about the Linux operating system. He has also served as the president and in other official capacities for the UTA chapters of Tau Beta Pi and Eta Kappa Nu engineering honor societies. His research interests include machine learning, data science, and computer vision.