Air Force Institute of Technology
AFIT Scholar

Theses and Dissertations | Student Graduate Works

3-2002

Selecting Salient Features in High Feature to Exemplar Ratio Conditions

Ismail Aslan

Follow this and additional works at: https://scholar.afit.edu/etd

 Part of the Applied Mathematics Commons, and the Operational Research Commons

Recommended Citation

Aslan, Ismail, "Selecting Salient Features in High Feature to Exemplar Ratio Conditions" (2002). *Theses and Dissertations*. 4510.
https://scholar.afit.edu/etd/4510

This Thesis is brought to you for free and open access by the Student Graduate Works at AFIT Scholar. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of AFIT Scholar. For more information, please contact richard.mansfield@afit.edu.

SELECTING SALIENT FEATURES IN

HIGH FEATURE TO EXEMPLAR RATIO CONDITIONS

THESIS

Ismail Aslan, 1st Lt., TUAF

AFIT/GOR/ENS/02-02

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

AFIT/GOR/ENS/02-02

SELECTING SALIENT FEATURES IN HIGH FEATURE TO EXEMPLAR RATIO CONDITIONS

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the
Degree of Master of Science in Operations Research

Ismail Aslan, B.S.

1st Lt., TUAF

March 2002

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

AFIT/GOR/ENS/02-02

SELECTING SALIENT FEATURES IN HIGH FEATURE TO EXEMPLAR RATIO CONDITIONS

Ismail Aslan, B.S.

1st Lt., TUAF

Approved:

__ /s/_____ Mar 13, 2002
Kenneth W. Bauer, Ph.D. (Advisor) date
Professor of Operations Research

__ /s/_____ Mar 13, 2002
Stephen P. Chambal, Capt, USAF (Reader) date
Assistant Professor of Operations Research

Acknowledgements

I believe that this study would have never been completed without the support of many people. I have to mention that there were two people always supporting and encouraging me. The first one is my thesis advisor, Dr Bauer. He encouraged me to work on this topic and always supported me during the study. A great mentor and a great person. Thanks Dr Bauer. The second one is my fiancé. Although she was far away from me, she never left me alone. I also want to thank my academic advisor and the committee member, Capt Chambal.

Ismail Aslan

Table of Contents

Table of Figures

List of Tables

Abstract

This thesis research offers two contributions: (1) a new user-friendly graphical user interface that can be used in a feature screening process, (2) a new algorithm for feature screening in a situation where there is a high ratio of feature to exemplars. The first objective is achieved by creating a MATLAB based graphical user interface, which is named as STNGER. STNGER is evaluated on both abstract and the real life problems and provides promising results. For the second objective a new algorithm is suggested. This new algorithm is based on the SNR screening method. By means of this new algorithm, the SNR screening method can determine the salient features in a situation where there is a high ratio of features to exemplars. The performance of the new algorithm suggests that one can apply the SNR screening method to randomly chosen subsets of the data and retain the best features from each subset for subsequent analysis. In this fashion, noisy features are removed while creating new subsets with salient features. The new algorithm is demonstrated on both real-life and the well-defined abstract problems.

SELECTING SALIENT FEATURES IN

HIGH FEATURE TO EXEMPLAR RATIO CONDITIONS

I. Introduction

In classification, there might be large number of features that can be computed. But it is not necessary or advisable to use all of the features in classification process. If there are too many features and too small a training set, we could obtain a meaningless perfect classification. Some researchers have found that performance actually peaked and subsequently deteriorated as the number of features was increased [1]. Feature selection is one of the basic problems in classification process in pattern recognition. There are many reasons for using feature selection techniques to reduce the number of features. Reasons for using feature selection techniques include:

- Satisfying the general goals of maximizing the accuracy of the prediction function while minimizing the associated measurement costs
- Improving prediction accuracy by reducing irrelevant and possible redundant features
- Reducing the complexity and the associated computational cost of a prediction function

- Reduce the amount of data needed for accurate prediction (i.e. reduce the 'curse of dimensionality')
- Reducing associated data collection and data processing cost
- Improving the chances that a solution will be both understandable and practical
- Improving the possibility of graphical representation of the data [2].

In recent years, there have been numerous efforts to find a process of feature selection. We can divide these studies into two groups: wrappers that use the learning algorithm itself to evaluate usefulness of the features and filters that evaluate features according to heuristic based on general characteristics of the data [3]. Search strategies such as Hill-Climbing and Best-first Search can be mentioned as examples of wrappers; and, one popular filter is to use Pearson correlation coefficient [4]. For practical applications it has been proven that filters are more practical due to being faster [3].

All these methods aim to find an optimal feature set. The inputs included in the optimal set can be named as salient inputs. The process of selecting these is known as saliency screening. The well-known saliency metrics are Ruck's Saliency [5,6], Tarr's Saliency [7], and Signal-to-Noise Ratio Saliency (Bauer) metrics [8]. Each of these metrics can be used by three different methods: Belue-Bauer [9,10], Steppe-Bauer [2, 11,12], and SNR methods. Although first two methods need multiple training runs (30 and 10 runs, respectively) [8], the SNR screening method potentially requires only a single replicate.

In this research, first, we will create some graphical user-friendly MATLAB based programs; second, we will investigate a situation where there is a high ratio of feature to exemplars.

II. Literature Review

Overview

In this chapter we will discuss two classification methods. The first section is a review of the Discriminant Analysis Feature Selection techniques. The second section covers some definitions used when discussing Neural Networks, a summary of multilayer perceptrons, a review of SNR methods, and a brief discussion of generalization and regularization process.

Discriminant Analysis

Discriminant Analysis is statistical technique for classifying individuals or objects into mutually exclusive and collectively exhaustive groups on a basis of a set of independent variables (features). Discriminant Analysis involves deriving linear combinations of the independent variables that will discriminate between the a priori defined groups in such a way that the misclassification error rates are minimized [13]. This is accomplished by maximizing the between group variance relative to the within group variance.

A Non-probabilistic Selection Metric

A common attribute of all selection procedures is an evaluation function used to measure the saliency of features. A large number of metrics have been suggested in the pattern recognition literature each having particular advantages and disadvantages. In this section we will discuss some intraclass distance

measures, which allow for both computational and analytical simplicity. The non-probabilistic feature evaluation metrics are based on rationale that classes should be maximally separated. Generally these types of metrics attempt to maximize the between class distances while minimizing the within class distances. [14] An estimated metric of between class distances, $\hat{S}_b$, defined as

$$\hat{S}_b = \sum_{k=1}^{K} \hat{P}(C_k) \cdot (m_k - m) \cdot (m_k - m)^T$$

where m_k is a M-dimensional mean vector of the k[th] class of the M-dimensional training vectors, m is a M-dimensional mean vector of all the training vectors, M is the number of features in a training vector, $\hat{P}(C_k)$ is the estimated prior probability of class k, denoted as C_k. An estimated matrix of the within class distances, $\hat{S}_w$, is defined as

$$\hat{S}_w = \sum_{k=1}^{K} \hat{P}(C_k) \cdot N_k^{-1} \cdot \sum_{i=1}^{N_i} (x_{ki} - m) \cdot (x_{ki} - m)^T$$

where x_{ki} is the i[th] training vector from the k[th] class and N_k is the number of training vectors in class k. Two possible metrics $D(x)$ which are maximized to find the salient features are

$$D_1(x) = \frac{trS_b}{trS_w}$$

$$D_2(x) = \frac{|\Sigma|}{|S_w|}$$

where Σ is equal to $S_b + S_w$, x is the feature vector. Also, | . | denotes calculation of determinant. The main criticism of these types of metrics they are not closely related to error probability.

These two metrics can be used to determine the salient features by evaluating for each individual feature set and putting in a descending order with respect to $D(x)$.

A general nonlinear metric $D(x)$, which reflects the local probability structure of the data, can be maximized to find a good subset of features.

$$D(x) = \frac{1}{2}\sum_{k=1}^{K}\hat{P}(C_k)\sum_{t=1}^{K}\hat{P}(C_l)\frac{1}{N_k N_l}\sum_{i=1}^{N_k}\sum_{j=1}^{N_l} d(x_{ki}, x_{kj})$$

where $d(x_{ki}, x_{kj})$ is the nonlinear distance metrics between the ith vector in class k and the jth vector in class l. This nonlinear distance equal to a constant H, if its Euclidean distance is above a threshold T, otherwise it is equal to zero. The threshold T represents a safe or effective distance for correctly classifying the two points into separate classes. This nonlinear metric represents a compromise between probabilistic and non- probabilistic feature evaluation metrics.

Selection Methodologies

There are number of optimal and suboptimal search algorithms which are designed to circumvent an exhaustive search procedure. Most search algorithms look for the best features by adding and/or removing features from the current feature set. A forward procedure starts with no features and add a feature at a

time by evaluating the metric at each step. A Backward selection procedure starts with all the candidate features and drops one at a time. There are some alternative algorithms like Genetic Algorithms or Branch and bound algorithms, but these will not be discussed here.

Forward Sequential Selection

Steppe [2] summarized the algorithms step by step that are described in detail by Devijver and Kittler [14].

1. p = 0
2. Set the total number of features to select equal k
3. Select a feature evaluation metric, say $D(x)$
4. Compute $D(x)$ for all feature subsets of size p+1 which include all previously selected features
5. Select the feature set of size p+1 which maximizes (minimizes) $D(x)$
6. Set p = p+1
7. If p < k go to step 2; otherwise go to step 6
8. Stop, k features have now been selected.

Backward Sequential Selection

1. p = 0
2. Set the total number of features to select equal k
3. Select a feature evaluation metric, say $D(x)$

4. Compute $D(x)$ for all feature subsets of size p-1 which do not include all previously selected features

5. Select the feature set of size p-1 which maximizes (minimizes) $D(x)$

6. Set p = p+1

7. If p < k go to step 2; otherwise go to step 6

8. Stop, k features have now been selected.

These algorithms allow just one feature added or selected at a time, but do not allow you to add or remove a feature after a feature is already evaluated. On the other hand generalized sequential forward or backward algorithms, which are called stepwise selection algorithms, allow more than one feature added or removed at a time. This characteristic provides to evaluate the statistical relations between variables.

SAS Feature Selection Algorithms

SAS can be used to apply Discriminant analysis. SAS uses either of the algorithms described above. It uses another metric to enter or remove any feature. This metric is based on the two criteria:

The significance level of an F-test from an analysis covariance, where the variables already chosen act as covariates and the variable under consideration is the dependent variable.

The squared partial correlation for predicting the variable under consideration from the class variable, controlling for the effects of the variables already selected for the model (SAS Help).

Analysis Of Covariance

In this section, we will take a brief look at the analysis of covariance (ANCOVA). To gain a better understanding of ANCOVA, Huitema [15] presents psychomotor test example. In this example, an experimenter is interested in comparing the effects of two drugs on a complex psychomotor test. Obtaining the data after experiment, the experimenter performs an ANOVA and concludes a non-significant F value. The experimenter then point out that there is a great deal of age variability within the two groups and that age is highly related to the scores on the psychomotor test. Analysis of covariance is then computed using age as the so-called covariate. In the present example age is partitioned from the relationship between drug type and psychomotor test scores. The ANCOVA *F* test reveals a statistically significant drug effect.

Computation and Rationale:

The starting point for ANCOVA is exactly the same as for ANOVA; the total sum of squares is computed. The analysis of covariance procedure may be summarized in six steps. [15]

Step 1. Computation of total sum of squares:

$$TotalSS = \sum y_t^2 = \sum Y_t^2 - \frac{\left(\sum Y_t\right)^2}{N}$$

X : Independent variable (in our cases features)

Y : Dependent variables (in our cases classes)

N : Number of data points

x : Deviation score on the covariate

y : Deviation score on the dependent variable

n_t : Number of subject in t different groups

Step 2. Computation of total residuals:

$$TotalresidualSS = \sum y_t^2 - \frac{\left(\sum xy_t\right)^2}{\sum x_t^2} = \left[\sum Y_t^2 - \frac{\left(\sum Y_t\right)^2}{N}\right] - \left[\frac{\left[\sum XY_t - \frac{\left(\sum X_t\right)\cdot\left(\sum Y_t\right)}{N}\right]^2}{\sum X_t^2 - \frac{\left(\sum X_t\right)^2}{N}}\right]$$

Step 3. Computation of within group sum-of-squares:

Within-group SS=

$$\sum y_1^2 + \sum y_2^2 + \sum y_t^2 = \left[\sum Y_1^2 - \frac{\left(\sum Y_1\right)^2}{n_1}\right] + \left[\sum Y_2^2 - \frac{\left(\sum Y_2\right)^2}{n_2}\right] + \left[\sum Y_t^2 - \frac{\left(\sum Y_t\right)^2}{n_t}\right]$$

The within group deviation cross products and sum of squares on X:

$$\sum xy_1 + \sum xy_2 + \sum xy_t = \left[\sum XY_1 - \frac{\left(\sum Y_1\right)\cdot\left(\sum X_1\right)}{n_1}\right] + \left[\sum XY_2 - \frac{\left(\sum Y_2\right)\cdot\left(\sum X_2\right)}{n_2}\right] + \left[\sum XY_t - \frac{\left(\sum Y_t\right)\cdot\left(\sum X_t\right)}{n_t}\right]$$

And within group sum of squares of X:

$$\sum x_1^2 + \sum x_2^2 + \sum x_t^2 = \left[\sum X_1^2 - \frac{\left(\sum X_1\right)^2}{n_1}\right] + \left[\sum X_2^2 - \frac{\left(\sum X_2\right)^2}{n_2}\right] + \left[\sum X_t^2 - \frac{\left(\sum X_t\right)^2}{n_t}\right]$$

Step 4. Computation of within group residual SS (error SS): By subtracting SS due to predictable differences among subjects within groups (sometimes called within-group regression SS) from SS within groups we obtain residuals sum of squares within, which is used as the error SS in ANCOVA.

Step 5. Computation of adjusted treatment affects (AT). By subtracting the SS residual within (step 4) from the total residual (step 2), the adjusted treatment SS is obtained.

Step 6. Computation of F ratio. Step 5 involves the partitioning of the total residuals into the sum of squares residuals within, and the adjusted treatment SS. The latter two correspond directly to within- and between group SS in a simple analysis of variance. Thus F ratio can be obtained by dividing mean square adjusted treatment by mean square error. Degrees of freedom are computed in essentially the same way as in ANOVA except that an additional freedom is lost from the error MS for each covariate employed in the analysis.

Feature Selection And Analysis of Covariance

After explaining the basic rationale of the ANCOVA, we can discuss the significance level of an F-test from an ANCOVA as a Feature Selection metric. The main idea underlying this metric is that every previously chosen feature acts as a covariate and every variable under consideration is a dependent variable. The significance level of each feature will be the metric that helps us to determine the saliency of a feature. The first step is determining the best feature with the most significant F-test, then use this feature as covariate and recalculate the F values again. The stopping criteria might be the number of features, or the significance level of F-test.

Artificial Neural Networks

There is a brief introduction to Artificial Neural Networks (ANN) in the MATLAB Help files [16]. Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the network function is determined largely by the connections between elements. We can train a neural network to perform a particular function by adjusting the values of the connections (weights) between elements.

Typically neural networks are adjusted, or trained, so that a particular input leads to a specific target output. Such a situation is shown below. There, the network is adjusted, based on a comparison of the output and the target, until the network output matches the target. In practice, many such input/target

pairs are used, referred to as "supervised learning", to train a network [16]. This is displayed in Figure II.1.

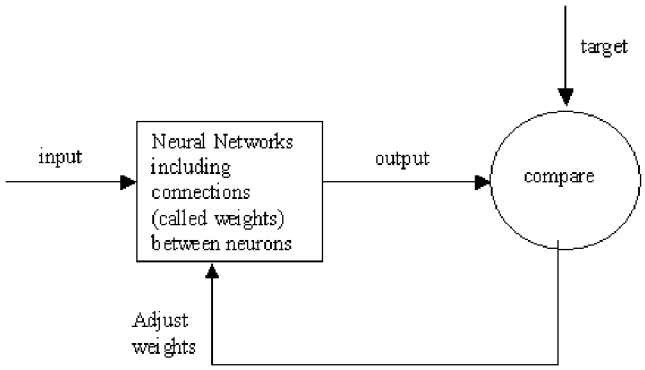


Figure II.1. Supervised Training

Definitions

Before going further, we will present some definitions related to ANN [17].

Artificial Neural Networks (ANN): An information processing system (algorithm) that operates on inputs to extract information and produces outputs corresponding to the extracted information.

Architecture: The topological arrangements of neurons, layers and connections which defines the set of modeling equations available to the ANN.

Backpropagation: A learning algorithm for updating weights in a feed-forward MLP ANN that minimizes the mean squared mapping error.

Epoch: A complete presentation of the data set being used to train the MLP, or equivalently called a training cycle.

Feature: In neural networks, features refer to the input vectors of information which are presumed to have some relation that might be helpful in distinguishing the various output classes. The vector of features is often called an Exemplar.

Feed-forward: Multilayer ANNs whose connections exclusively feed inputs from lower to higher level. In contrast to feedback or recurrent ANN, a feed-forward ANN operates only until all the inputs propagate to the output layer. An example of a feed-forward ANN is the MLP (multilayer perceptron).

Hidden Units: The processing element in MLP ANN that are not included in the input or output layers. This is the part of the neural network located between the input and output layers.

Learning Algorithm: The equations used to modify weights of processing elements in the response to input and output values.

Neuron: The fundamental building block of an ANN. Normally, each neuron takes a weighted sum of its inputs to determine its net input. The net is then processed through its transfer function to produce a single valued output that is broadcast to 'downstream' neurons.

Single Layer Perceptron: A type of ANN algorithm used in pattern classification that is trained using "supervision". Connection weights and thresholds can be fixed or adapted using a number of different algorithms.

Supervised Training: A method of training adaptive ANNs that requires a labeled training data set and external teacher. The teacher knows what the desired response is and thus can provide responses for correct or incorrect classification by the network.

Weight: A processing element (or neuron or unit) need not treat all inputs uniformly. Processing elements receive inputs by means of interconnects (also called 'connections' or 'links'); each of these connections has an associated weight which signifies its strength. The weights are combined to calculate the activations.

Multilayer Perceptrons

To gain a better understanding of Multilayer Perceptrons, it would be better to study the structure of a single perceptron first; then a layer of perceptrons and finally multilayer perceptrons.

Figure II.2 shows the structure of a single perceptron. In this structure, the generalized mathematical notation is:

M is the number of input features

x_i^n is the ith input feature of the nth input vector

x_0^n is the ith input bias term is set to equal one

w_{ij}^n is the weight from input node i to hidden node j at nth layer

$f(.)$ is the activation function

N is the number of exemplars

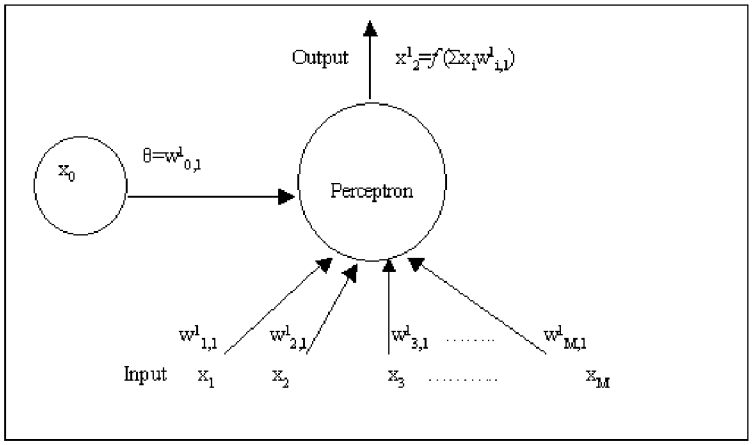


Figure II.2. A Single Perceptron Structure

In a single perceptron structure, as shown in Figure II.2, we have only one layer and one perceptron [17]. The input vector $x = (x_1,\ldots,x_M)$ is linearly combined with the weights [18] and bias term. This sum is fed into a squashing function $f(.)$, or named transfer function, in order to get squashed the sum into the intervals [0,1] or [-1,1] depending upon the characteristic of the transfer function.

A single perceptron structure can partition the feature vector space into two half spaces. In some cases, we need to separate the feature space more than two spaces. Adding one more perceptrons will increase number of the half-

spaces. According to Looney [18], the maximum number of convex regions that could possibly occur is 2^J , where J is the number of perceptrons, while the minimum number of convex regions is J+1.

The network structures discussed previously have a number of limitations in terms of the functions that they could represent. To allow for more general mappings we might consider successive transformations corresponding to networks having several layers of adaptive weights [19]. Bishop presents a graphical example to understand the efficiency of multilayer networks.

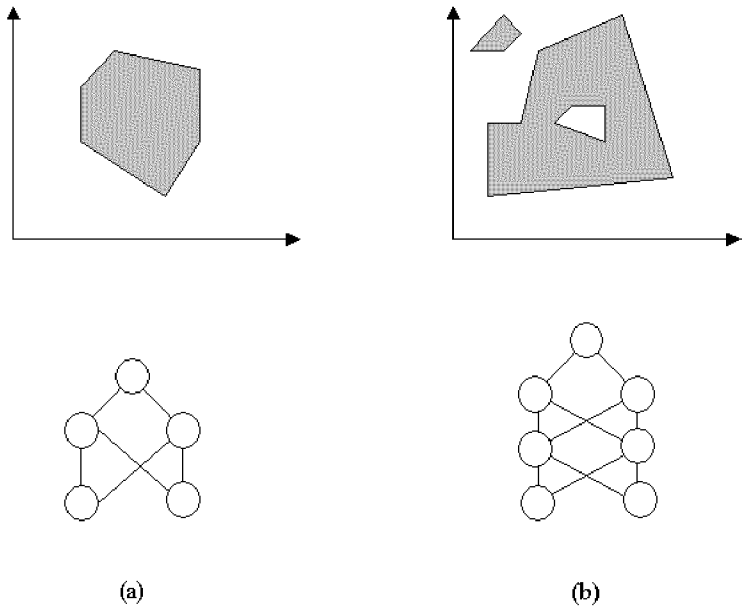


Figure II.3. Illustration of some possible decision boundaries (Bishop, 1995)

One layer of perceptrons can generate decision boundaries which surround a single convex region of the feature vector space, as illustrated in Figure II.3(a). More than one layer networks can generate arbitrary decision

boundaries, which maybe non-convex and disjoint, as illustrated in Figure II.3(b). Although one layer perceptron can create very complex decision boundaries, Gibson and Cowan [20] provided some examples that cannot be generated by one layer perceptrons, as shown in Figure II.4. Therefore, using multilayer perceptron, Figure II.5, provides some advantages in classification process.

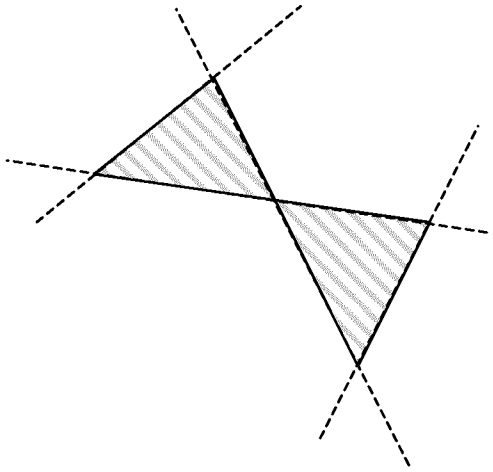


Figure II.4. An example of decision boundaries which can not be produced by a network having two layers of threshold units (Gibson and Cowan, 1990)

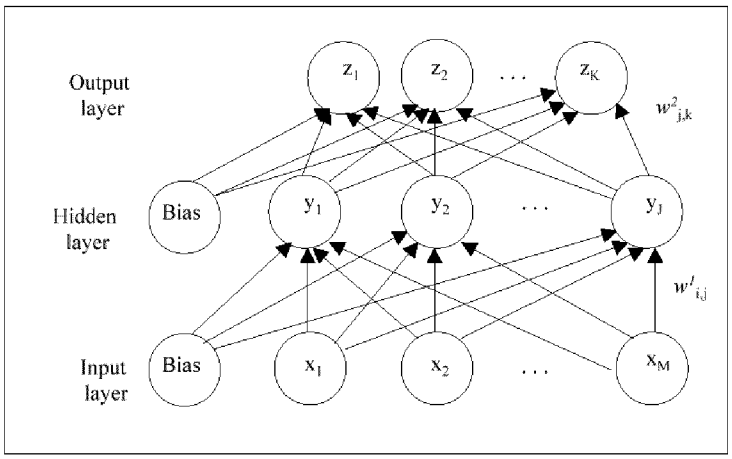


Figure II.5. Multilayer Perceptron

So far we have discussed the single perceptron structure, a layer of perceptrons, the decision boundaries we can create by using multilayer and single layer networks. Now we can take a look at the structure of MLPs, see Figure II.5.

*k*th neural network output

$$z_k^n = f(\sum_{j=1} w_{j,k}^2 \cdot x_j^1)$$

where,

J is the number of hidden nodes,

$f(a) = 1/(1+e^{-a})$,

x_0^1 is the hidden layer bias term and is set to equal to one,

$x_j^1 = f(\sum w_{i,j}^1 \cdot x_i^n)$ is the output of the hidden node j and is summed from,

i=1 to M.

In the backpropagation algorithm learning occurs by changing the weights in a direction that minimizes the sum of the squared errors. The error is basically the squared difference between the network output and the target value. The weights are updated by taking the derivative of the error term with respect to each weight. To be able to do this derivation we prefer activation functions such as sigmoid function, which are easily differentiable.

Signal to Noise Ratio Saliency Method

The rationale behind the Signal to Noise Ratio (SNR) metric is that the salient features should produce larger weights relative to non-salient features. This implies that their sum of squared weights will also be larger. If we inject a noise feature and look at the ratio of each feature's sum of squared weights to noise feature's sum of squared weights, we can rank the features. To place the ratio on a decibel scale we can take the logarithm of this ratio and multiply by ten. The mathematical expression is

$$SNR_i = 10 \cdot \log_{base10}\left(\frac{\sum_{j=1}^{J}(w_{i,j}^1)^2}{\sum_{j=1}^{J}(w_{N,j}^1)^2}\right) \quad [8]$$

where SNR_i is the value of the SNR saliency measure for feature i, J is the number of hidden nodes, $w_{i,j}^1$ is the first layer weight from node i to node j, $w_{N,j}^1$ is the first layer weight from the injected noise node N to node J. The noise feature is created such that its distribution fallows that of a Uniform (0,1) variable.

SNR saliency measure should be significantly larger than 0.0 for salient features and close to or less than 0.0 for non-salient features. The SNR saliency measure can be used to rank order the saliency of the features where higher saliency measure values correspond to higher feature saliency [8].

Improving Generalization

One of the problems we may encounter while training a network is overfitting. While the error on training set getting smaller, the error on test set may be getting larger. If this occurs, this generally means that the network is memorizing the provided training set and losing the power of generalizing the introduced new points.

There are several ways of improving generalization. One of them is selecting a network structure that avoids overfitting. But it is not an easy task to determine the network structure before any training. The number features maybe a good heuristic for number of hidden nodes. But in our case, we may have more features than the exemplars. So this heuristic will not be useful for us. In fact, we will divide the data into the smaller sets and train each data set separately. The details will be discussed in chapter 3.

There is two other methods can be used to improve generalization: "The Bayesian Regularization" and "Early Stopping". In our study we will use the early stopping rule method. According this method, the training set is used to train the net and update the weights. Normally, the error on both sets will

decrease while training continues. If the net starts to memorize data the error on the test set starts to increase. When the test error increases for a specified number of iterations, the training is stopped, and the weights and biases at the minimum of the validation error are returned [16].

III. Methodology

Overview

The following chapter introduces the SRSS algorithm which is an application of the Signal to Noise Ratio Saliency Screening Method to randomly selected subsets. Next we present a new user-friendly tool, STNGER (Signal to Noise Graphical Evaluation Routine), which can be used in feature selection process. In the past several MATLAB functions or other tools have been constructed to do this task. Because feature selection is iterative in nature, it was painful and time consuming to use them; this was especially true if one wanted results with statistical significance. This tool is designed to provide the user an easy way of doing feature selection and producing some useful plots. The first section of this chapter covers the basic algorithms underlying the STNGER, and the second section is a tutorial.

Applying Signal to Noise Ratio to Randomly Selected Subsets (SRSS)

We propose a new method in order to get better results in the feature selection process. According to the new methodology, we separate data into subsets, and apply the SNR method to each subset. In this way we effect the ranking within each group. By recombining the features from each subset we create another data set. We continue this fashion to find a good subset.

The generalized mathematical notation for our method is:

F is the number of features

N is the number of exemplars

$R = \frac{F}{N}$ is the ratio of the number features to the number of exemplars

s is the number of subsets and $i = 1 \ldots s$

S_i is the ith subset

F_i is the number of features in the i^{th} subset

k is the number of features retained from each subset via the SNR method

$r = \frac{F_i}{N}$ is the user defined ratio to create subsets

Figure III.1 depicts our method in three basic steps. The detailed explanations of these steps are:

1. Create subsets from original data randomly according to a user specified ratio r. For example, if there is a data set with $F = 30$ features and $N = 10$ exemplars, a user specified ratio $r = 0.5$ would divide our data set into the $s = 6$ subsets each having $F_i = 5$ features.
2. If any of the subsets have less or equal number of features than k, add the features F_s to F_{s-1}.
3. Execute SNR method to each subset and keep the best k of them.
4. Create another subset from these selected features. The number of features in this set will be $s \times k$.

5. If *r* is smaller than *R*, go to step 1.

6. Execute SNR method on the last subset.

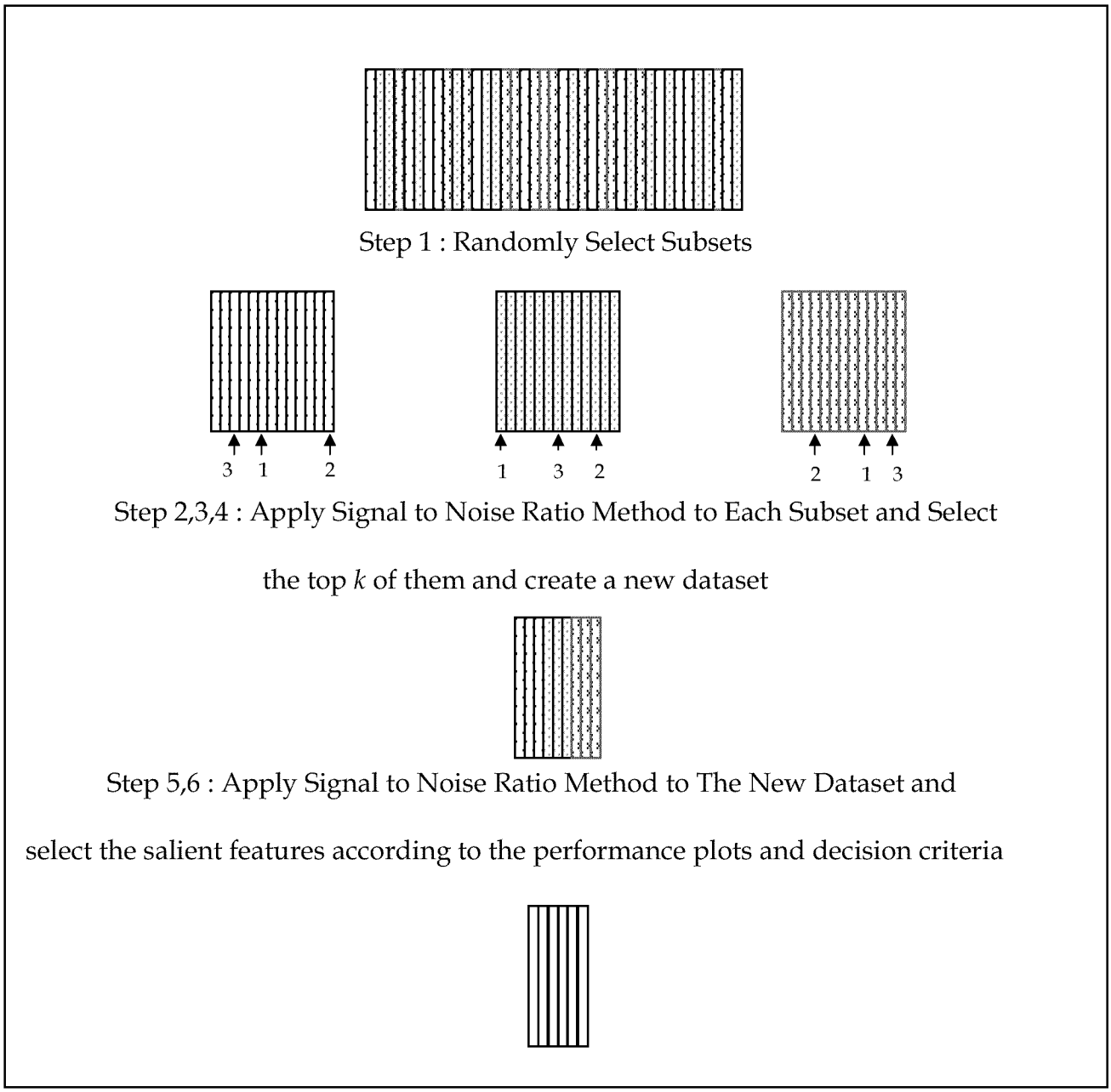


Figure III.1. SRSS Methodology

Signal to Noise Ratio Method

Based on the methodology introduced by Bauer *et al* [8], Lt. Col. Alsing created the MATLAB function that executes the SNR method [21]. The steps of the algorithm are:

1. Create a uniformly distributed noise feature with parameters 0 and 1, and add this to the data as a new feature.
2. Randomly separate data into training set and test set.
3. Standardize the noise-injected data set to zero mean and unit variance.
4. Create a network object that uses a modified version of MATLAB's adaptive learning algorithm (TRAINGDX). The purpose of the modification is to save time by suppressing the built in performance plot. The initial learning rate of the net is set to 0.01. If the last iteration gives a better error rate than the previous one the learning rate will be increased by 1.05. If the last iteration gives a worse error rate than the previous one the learning rate will be decreased by 0.7. All activation functions are sigmoidal. The other user-defined specifications of the net like the maximum number of epochs and the number of the hidden nodes are described in the next section.
5. Create initial weights randomly (uniform) between -0.001 and 0.001.
6. Train network until the SNR stabilizes.

7. Remove the feature with the smallest SNR, calculate classification accuracy with the remaining features, and adjust the network object.
8. If there are features remaining other than noise the feature go to step 6, otherwise go to step 9.
9. To understand the performance of the noise feature, train the net until the SSE stabilizes. This performance is equal to the prior probabilities of the classes.

Sources of Randomness

Randomization is used at four different places in this method.

1. We select each subset from the data randomly.
2. We add random noise to each SNR method iteration.
3. At each of the SNR method iteration a different random weight set is used as an initial weight set.
4. At each of the SNR method iteration, we randomly separate each subset into the training and the test sets.

The randomization provides us the means to attach statistical significance to the analysis.

Tutorial of STNGER

In this section, a user manual for STNGER will be presented in a step-by-step fashion.

Installation and Requirements

The STNGER is designed in MATLAB. MATLAB Release 12 must be loaded on the computer with the Neural Network Toolbox Version 4.0 and the Statistics Toolbox Version 3.0 available. In order to run STNGER more conveniently, it would be better to add the STNGER folder to the MATLAB search path.

1. Select File / Set Path… From command window menus
2. Add With Subfolders and point the application folder
3. Save and Close

Preparing Data

Before running the STNGER, the data file must be ready. There are basically three data requirements:

1. The data file must be written in a tab delimited text (`*.txt`) format and there should be feature titles (tab delimited as well) for each column of data.
2. The data must reside under the 'data' folder which is in the `stnger` folder
3. The target vector must be the first column of the data file, and the classes can be indicated as 1,2 or 1,0 or -1,1.

Running STNGER

After preparing the data for analysis, we can run the STNGER by typing `stnger` on the command window. This command will bring up the main menu (Figure III.2).

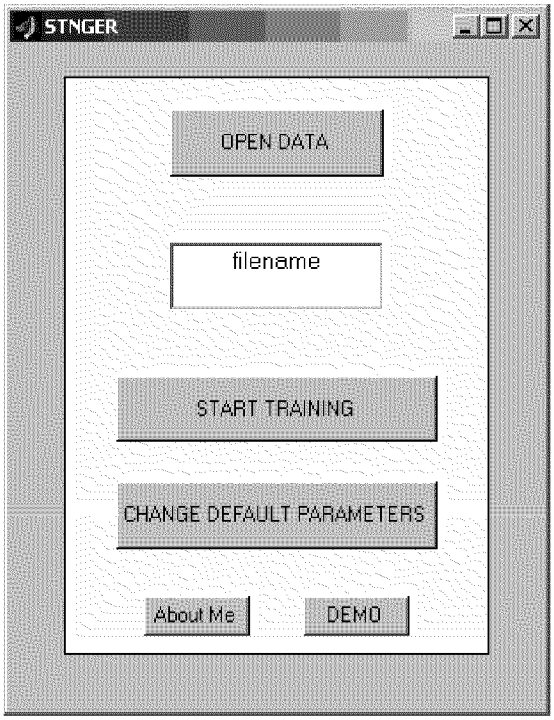


Figure III.2. STNGER Main Menu

The **Open Data** button will be used in order to browse and select the data file that we have already prepared. When you hit this button the next window will be as reflected in Figure III.3. After selecting the data file, which is already placed under the Data folder, a window as shown in Figure III.4 will appear. This window is mainly divided into two parts: (1) Input and Output Selection and (2) Data Fractioning for Training. The upper part is used for choosing the

features we wish to investigate for the saliency. The lower part allows user to determine *r* and *k*.

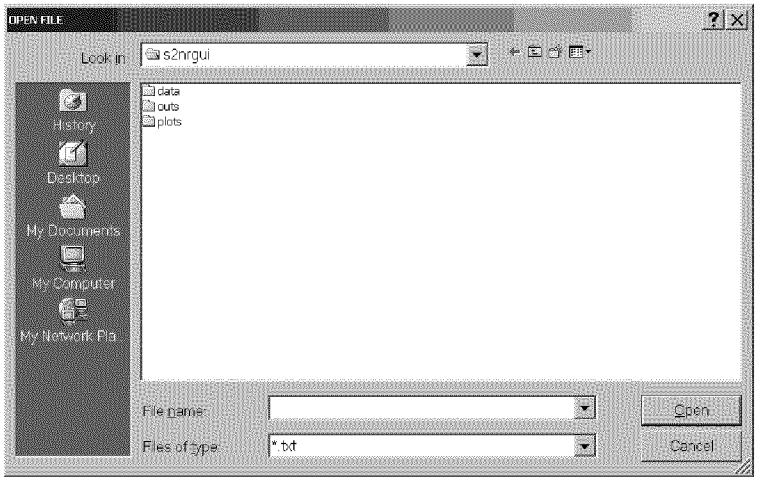


Figure III.3. Open File Menu

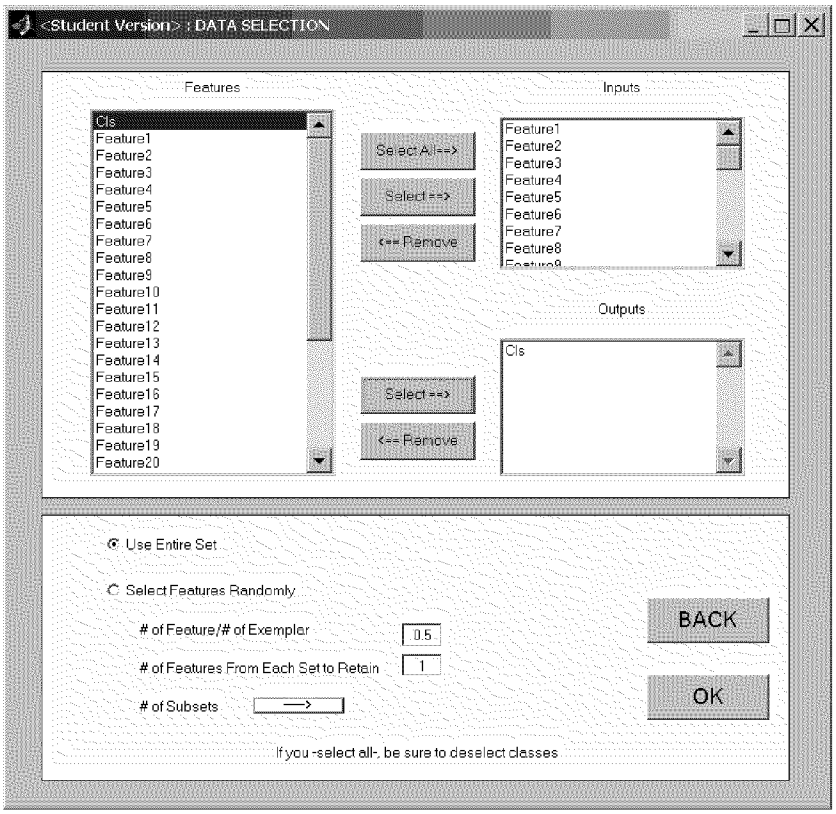


Figure III.4. Input Output Selection

There are two exclusive options in the lower part of the "Data Selection" window. If we select **Use Entire Set** option, SNR method will be applied to entire data set without doing any subset creation. This option might be used if there is small number of features with respect to number of exemplars.

The second option is used to apply our method. **The ratio of the number of features to the number of exemplars**, *r*, is the parameter used to determine the size of each subset. For example, if there are *N* exemplars and the ratio is r,

then each subset will have $N \times r$ features. The general formula regarding to number of features for all subsets except the last one will be

$$F_i = N \times r$$

The general formula for the last subset will be

$$F_s = F - \sum_{i=1}^{s-1} F_i$$

If the last subset has less or equal number of features than k, $F_s \leq k$, the last subset will be added to the previous subset,

$$F_{s-1} = F_{s-1} + F_s$$

So, the minimum number of features in any subset will be $k+1$. This is the case where the last subset S_s has $k+1$ number of features. The maximum number of features in any subset will be $(N \times r) + k$. This is the case where the last subset S_s has equal number of features to the k. The user can calculate the number of subsets by hitting the **Number of Subsets** button.

After selecting the inputs/outputs and after the fractioning is determined, we return to the main window. The data file name will be shown in the Figure III.2. **Start Training** will start the SNR method with the default parameters. We can see and change the default values of parameters, Figure III.5 by hitting the **Change Default Parameters** button. The explanations of the parameters are presented below.

Fraction For Training: The training fraction of the data. The remaining part is the test set and Classification Accuracy is calculated on the test set.

Number of Hidden Nodes: The number of the hidden nodes at the first layer of the network. The user can determine the number of hidden nodes or number of features is used as the number of hidden nodes by default. For each data set the number of hidden nodes are held constant. In other words, for each data set, after removing a feature we don't remove one of the nodes. The injected noise counts as a feature.

Min Epochs to Feature Removal: Minimum number of epochs before any feature is removed.

Number of Epochs to Check SNR: Number of epoch to check for SNR stabilization. For example, the default is to check the SNR at every of the 50 iterations.

Epoch to Check SSE: After removing all features, check SSE after this number of epochs for stabilization. This is done for each SNR method.

Criteria For SNR Change: Train network until the maximum change in the SNR is less than or equal to the determined criteria. The formula of this is

$$abs\left(\frac{SNR_{i+1,j} - SNR_{i,j}}{SNR_{i,j}}\right)$$

where SNR_i is the SNR value for feature j at i^{th} epoch.

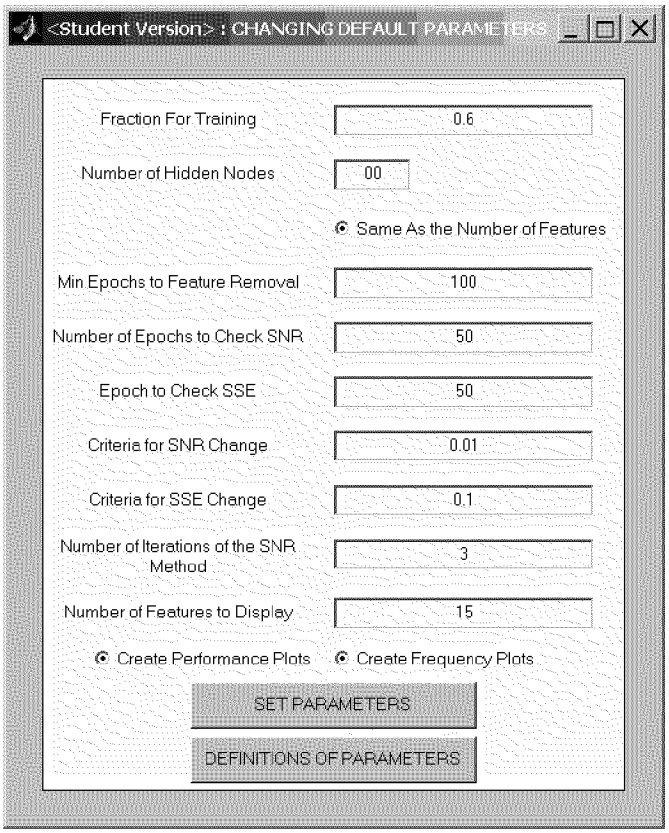


Figure III.5. Changing Default Parameters

Criteria For SSE Change: Train network until the change in the SSE is less than or equal to the determined criteria. The SSE change can be written as

$$abs\left(\frac{SSE_{i+1,n} - SSE_{i,n}}{SSE_{i,n}}\right)$$

where SSE_i is the performance of the net with the noise feature n at i^{th} epoch.

Number of Iterations of the SNR Method: The number of iterations we want to run SNR method.

Number of Features to Display: How many of the of the frequency plots that will be shown at the end of the iterations is determined by the user. The

performance and the sorted rank plots will also display this number of features ranks and performances. This will help to present a more understandable plot.

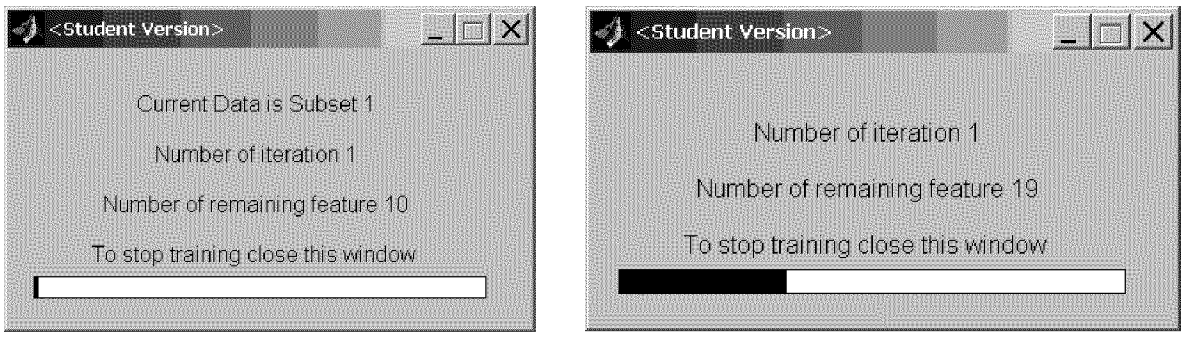
Create Frequency Plots: Creates the rank frequency plot for every feature.

Create Performance Plots: Creates a sorted rank plot, a rank plot with standard deviation, and a mean performance plot. These plots will be explained in detail in the Outputs section.

The **Definitions of the Parameters** button will bring up a window that shows a brief definition of the parameters.

Outputs

When we hit the **Set Parameters** button we will be ready for training. The **Start Training** button at the main window will be used to start training. While the data is prepared for training; procedures such as standardization, adding noise feature etc., a small window will appear with a "Please wait…" message. The message will change when the training is started. If we apply fractioning, the current subset number will be shown in the window, see Figure III.6(a). By closing this window we can stop training and lose all information gathered until this point. But if we do not apply fractioning we will not see any indication of the subset number, Figure III.6(b). We can stop training at this point and the outputs will be created according to the data gathered until closing the wait window, or we can use Ctrl+Pause keys to end the process.



(a) (b)

Figure III.6. Wait Windows

When the training ended, there will be seven types of outputs.

1. Rank Frequency Plots: These plots show the frequency of a feature's rank during the SNR method. For example, Figure III.7 shows that the intelligence feature is ranked 1 time as the forth feature, and 4 times as the sixth feature.

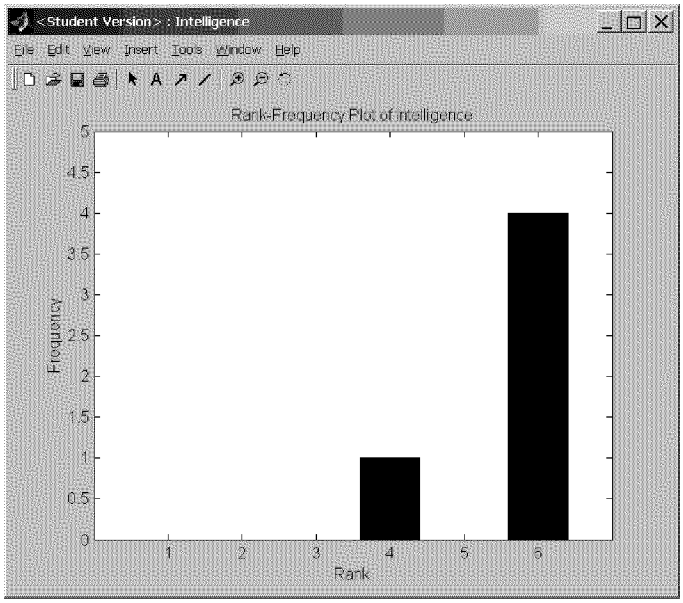


Figure III.7. Rank-Frequency Plot

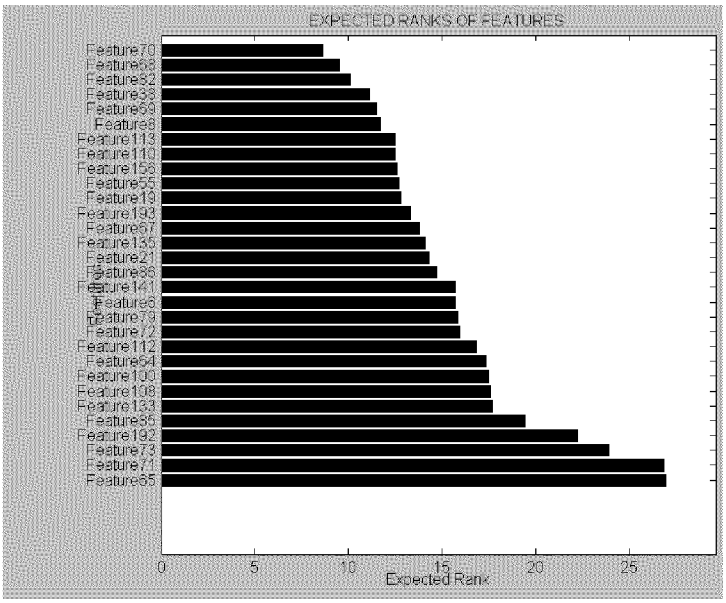


Figure III.8. Expected Ranks of the Features

2. The Sorted Expected Ranks of Features: The plot shown in Figure III.8 visualizes the ranking within the features. The top feature is ranked as the best according to the expected ranks.

3. Expected Ranks of the Features with Standard Deviations: The plot shown in Figure III.9 helps to gain a better understanding of the ranking. Each point represents the corresponding features expected rank, and the lines shows the feature's standard deviation. The best feature is shown at the left most part of the plot. The features are listed in a descending order with respect to their expected ranks.

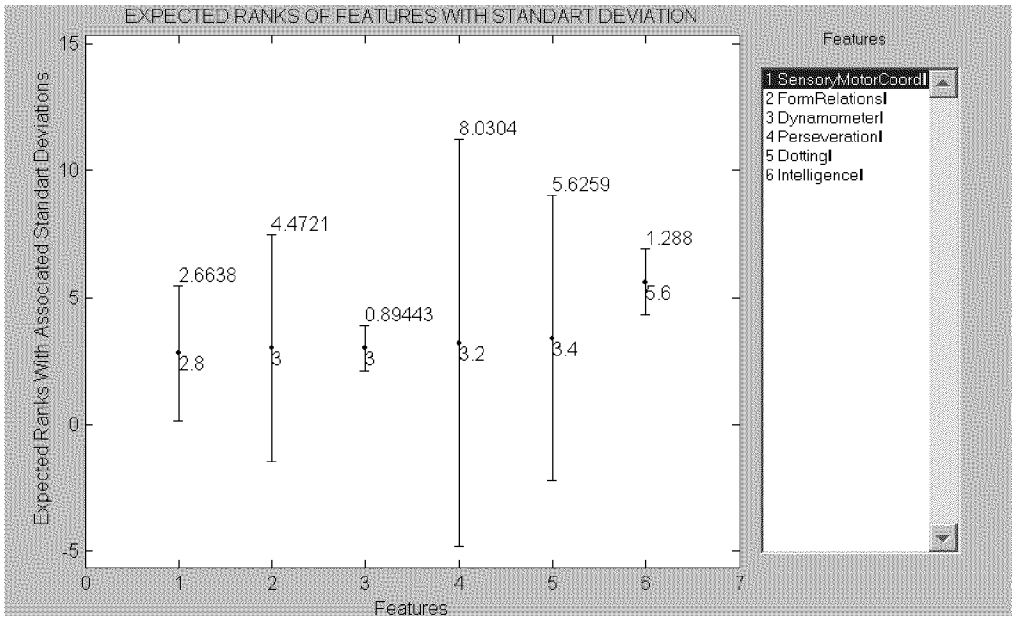


Figure III.9. Expected Ranks of the Features with Standard Deviations

4. The Mean Performance Plot:

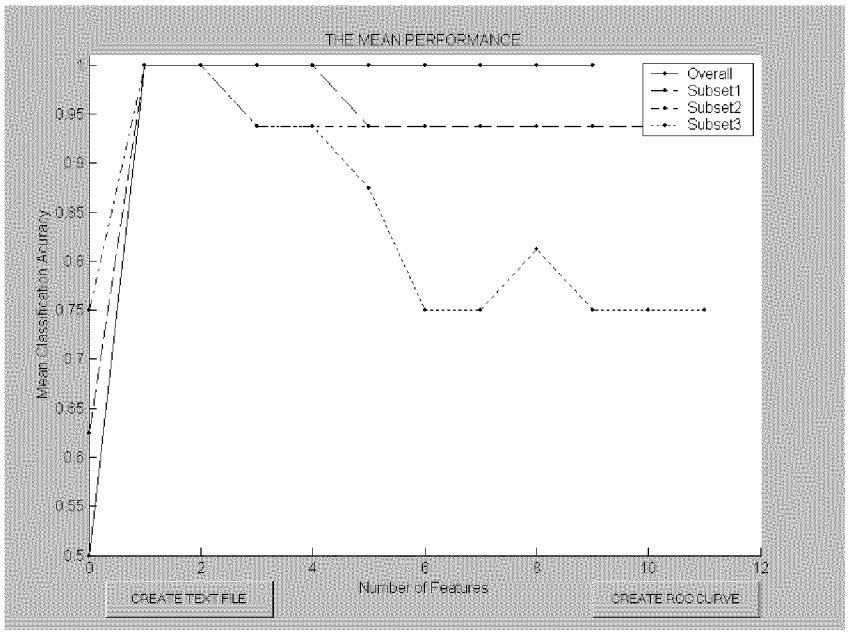


Figure III.10. The Mean Performance Plot

The mean performance plot, Figure III.10, shows the mean classification accuracy with respect to number of features. If the SNR Method applied based on different subsets, the mean performance plot will also show the performance on these subsets. One can determine the *k*, by looking at the subsets performances. For example, most of the features in subset 3 are hurting the mean classification accuracy. One can choose select *k* as 3 for subset 3. In a future version of the STNGER, different *k* values can be chosen for different subsets by looking at the performances of the subsets before creating and training the new dataset.

There are two buttons on this plot. One of them, **Create Text File**, is used to create a new data text file that has the features in a ranked order. **The Number of Features to Display** will be used to determine the number of features in this text file.

The other button, **Create ROC Curve**, is used to create a ROC curve with the new feature subsets, see Figure III.11. The parameters of the net used to create the ROC curve are:

Max number of epochs: 500

Number of Hidden Nodes: Number of features

Validation Set: The fraction for test/training is used to create validation set out from training set.

Maximum number of failures on validation set: To improve generalization, a validation set is used. The maximum number of failures on the validation set during the training will not be more than 5% of the number of exemplars in training set. In other words, if the error increases for a certain number of epochs, which is defined as the maximum number of failure in MATLAB, on the validation set the training will be stopped. The weights at the minimum error will be returned. The details of the generalization are discussed in Chapter 2 under the topic of "Improving Generalization".

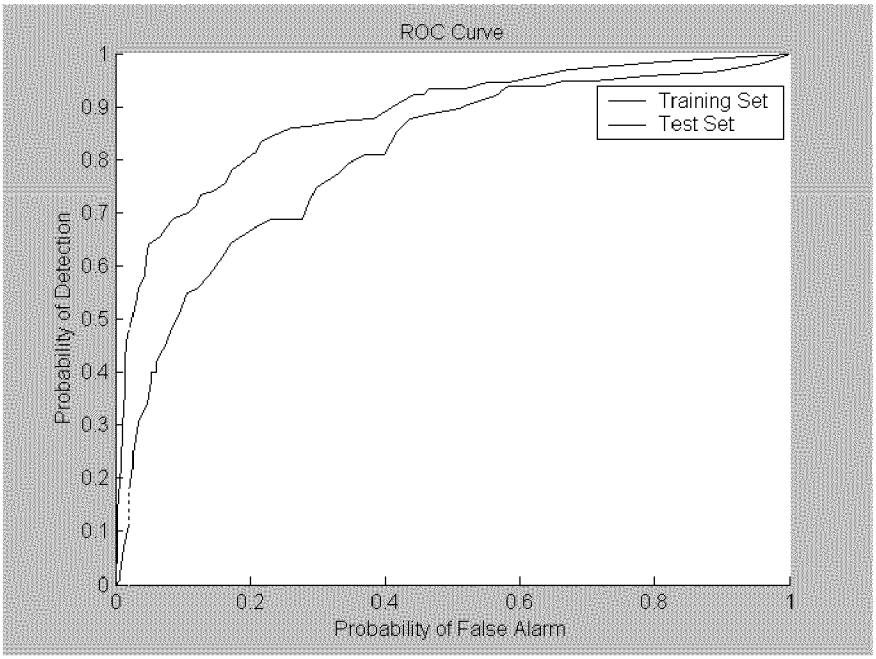


Figure III.11. ROC Curve

The modified version of TRAINGDX is used as the training function with a learning rate of 0.01, and the other parameters are the same as the parameters used during the SNR method.

IV. Results

Overview

In the following chapter, the results of three empirical studies will be presented. The first one is a well-defined abstract problem with known salient features. The second one is the corrupted version of the Fisher's famous iris classification problem. The third application is from a drug database with 222 features and 891 exemplars. These applications will provide sufficient evidence of the merit of STNGER. In chapter 3, we discussed the new methodology used by STNGER. Recall that the basic idea of this method is applying SNR Method to Randomly Selected Subsets (SRSS).

Abstract Problem

In this section we will run the STNGER with a well-defined abstract problem in order to examine whether the STNGER is able to identify salient features by using SNR Method.

In a real word data, three kinds of features can be seen. The first group is the salient features. The salient features will improve a classifiers ability to determine the classes effectively. The second group, redundant features, in some way convey the same kinds of information as salient features do. The last group is the noisy features. Noisy features will lessen the efficiency of the classifiers.

To examine the merits of STNGER, we used a data set known as the Block-C data set. The Block-C data can be defined as:

$$\text{Target} = \begin{cases} \text{True} & \text{if} \quad (x_{1i} > 2\cdot(b-a)/3) \text{ and } ((b-a)/3 < x_{2i} < 2\cdot(b-a)/3) \\ \text{False} & \text{Otherwise} \end{cases}$$

where x_1 and x_2 uniformly distributed between a and b, $i = 1...N$, N number of exemplars.

We defined $a = 0, b = 1$ and $N = 60$, the classes are evenly distributed. Figure IV.1 is a graphical representation of the data defined above.

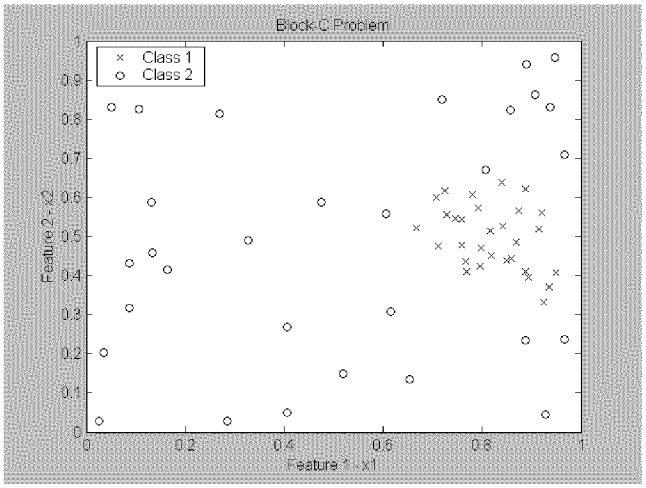


Figure IV.1. Block-C Data

We generated 5 uniformly distributed (0,1) noisy features. We added one redundant feature by using the formula below

$$r = \left(\frac{x_1}{S_{x_1}} \right) \cdot 0.5 + N(0,1)$$

where S_{x_1} is the standard deviation of x_1, $N(0,1)$ is the random noise, 0.5 is chosen as the correlation between *Redundant* feature and x_1, but we ended up with a correlation of .43 because of random noise. Subsequently, we had a dataset with 60 exemplars, 30 of them belong to Class 1 and 30 of them belong to Class 2. We had 8 features; 2 of them are significant features (x_1, x_2), 5 of them are noise features (n_1, n_2, n_3, n_4, n_5), and one of them is the redundant feature, r.

Figure IV.2 shows the ranking plot produced by STNGER with the default parameters.

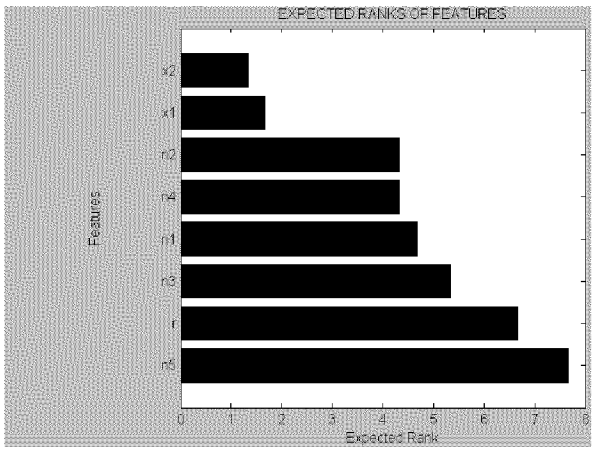


Figure IV.2. Expected Ranks of Features of Block-C Problem

These results indicate that STNGER successfully applies SNR Method to a feature selection problem.

Noise Corrupted Version of Fisher's Iris Classification Problem

In order to investigate the consistency of the SRSS method, we applied the SRSS method on a noise-corrupted version of Fisher's Iris classification problem. We also demonstrate the SRSS algorithm using this experiment.

There are three classes in Fisher's Iris classification problem: iris setosa, iris versicolor, and iris virginica. There are four input features: sepal length, sepal width, petal length, and petal width. There are 150 exemplars. The classes are evenly distributed. To increase the ratio of features to exemplar, we generated 12 noisy features and 4 redundant features. Each three of the noisy features are uniformly distributed between the minimum and the maximum values of one of the known features. Each redundant feature has a correlation of 0.6 with one of the known features. All generated features are discrete. Table IV.1 present a brief definition of each feature.

Table IV.1. The Descriptions of the features of a noise corrupted version of Fisher's Iris classification problem

Feature	Feature Representative	Description
Sepal length	F1	A known feature
Sepal width	F2	A known feature
Petal length	F3	A known feature
Petal width	F4	A known feature
Noise 1	N11	Discrete Uniform (min(F1), max(F1))
Noise 2	N12	Discrete Uniform (min(F1), max(F1))
Noise 3	N13	Discrete Uniform (min(F1), max(F1))
Noise 4	N21	Discrete Uniform (min(F2), max(F2))
Noise 5	N22	Discrete Uniform (min(F2), max(F2))
Noise 6	N23	Discrete Uniform (min(F2), max(F2))
Noise 7	N31	Discrete Uniform (min(F3), max(F3))
Noise 8	N32	Discrete Uniform (min(F3), max(F3))
Noise 9	N33	Discrete Uniform (min(F3), max(F3))
Noise 10	N41	Discrete Uniform (min(F4), max(F4))
Noise 11	N42	Discrete Uniform (min(F4), max(F4))
Noise 12	N43	Discrete Uniform (min(F4), max(F4))
Redundant 1	R1	It has a correlation of 0.6 with F1
Redundant 2	R2	It has a correlation of 0.6 with F2
Redundant 3	R3	It has a correlation of 0.6 with F3
Redundant 4	R4	It has a correlation of 0.6 with F4

We used MATLAB's Neural Network Toolbox to create a single hidden layer network with the following parameters. The number of hidden nodes is equal to number of features. 60% of the dataset was used for training. All activation functions were sigmoidal.

Train Function : Traingdx.m (An adoptive learning algorithm of MATLAB)
Learning Rate : 0.01
Performance goal (sse) : 0.0010
Learning Decrease : 0.7000
Learning Increase : 1.0500

Maximum performance increase: 1.04
Momentum Constant : 0.9000
Minimum Gradient : 1.0000e-006

The SRSS parameters were chosen to create 5 random subsets out of the original dataset. So the ratio of features to exemplar was set to 0.03 (r =0.03). We applied SNR method to each subset 3 times. The best 2 (k = 2) features from each of the subset were retained to create a new dataset.

Table IV.2. The subset with member features and their expected values within subsets

	Subset 1			
Features	N31	**N32**	**N11**	N22
Expected Ranks	2.6	**2**	**2.3**	3

	Subset 2			
Features	N41	R1	**F1**	**R3**
Expected Ranks	3.6	3.3	**1**	**2**

	Subset 3			
Features	N13	N21	**R4**	**F4**
Expected Ranks	2.6	4	**2.3**	**1**

	Subset 4			
Features	N42	**N12**	**R2**	N23
Expected Ranks	3.3	**2**	**1.6**	3

	Subset 5			
Features	**F3**	N33	N43	**F2**
Expected Ranks	**1**	3.3	3.3	**2.3**

Table IV.2 shows the randomly created subsets and the expected values of the features after 3 SNR iterations. By retaining the best 2 features from each subset a 10-feature subset was created. The ratio of this new dataset was still

larger than 0.03. So we created 2 subsets one of which having 4 features and the other one having 6 features. The expected ranks of these features are presented in Table IV.3. We combined the best two features from these subsets and trained again. After 3 SNR method iterations, we determined the rankings given in Figure IV.3.

Table IV.3. Second level subsets

	Subset 1			
Features	**F3**	R2	N11	**R4**
Expected Ranks	**1**	3.3	3.3	**2.3**

	Subset 2					
Features	**F4**	**F2**	F1	R3	N12	N32
Expected Ranks	**1**	**2.3**	3.3	3.6	4.6	6

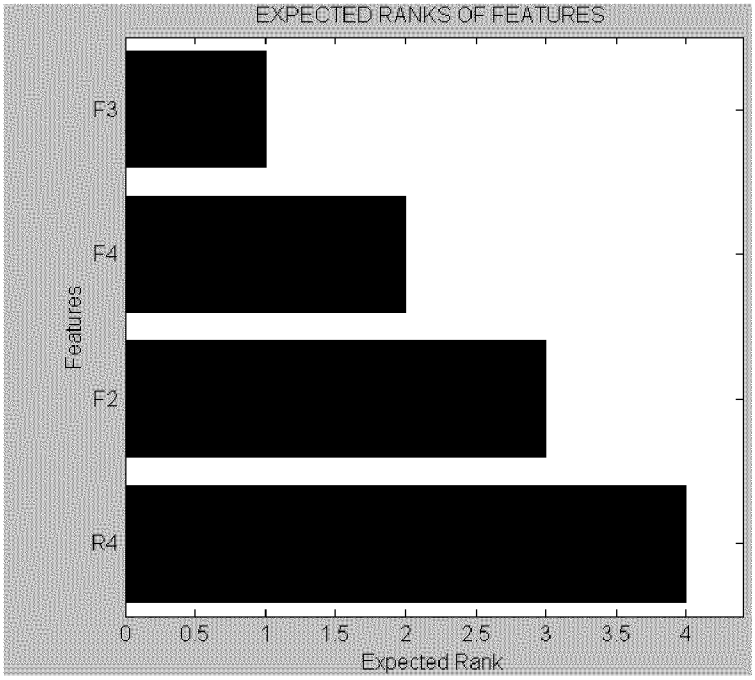


Figure IV.3. Expected ranks of the salient features

SRSS algorithm determined the salient features easily although there were 12 noisy and 4 redundant features. For all 3 SNR method iterations the same ranking was produced, see Table IV.1. In this experimental study, since we knew the salient features, we didn't measure the performance of the selected salient feature subset. We only compare the salient features determined by the SRSS with the known salient features.

It may be concluded that how many noisy feature are in the original dataset does not that important as far as we properly determine the ratio of features to exemplar, *r*, and the number of features to retain from each subset, *k*.

Table IV.4. Salient features with expected ranks and associated standard deviations

Features	F3	F4	F2	R4
Expected Ranks	1	2	3	4
Standard deviations of the expected values	0	0	0	0

Drug Data Set

As mentioned before, this data set is from a drug database. The drug producers are interested in predicting a potent or not potent compound by examining various chemical properties. Their main objective was to minimize the percentage of the incorrectly predicted non-potents and to maximize the correctly predicted potents by employing the minimum number of features.

Table IV.5 shows the interest areas of the confusion matrix in this application [17].

Table IV.5. Confusion Matrix

	Actual Target	Actual Clutter
Predict Target (Potent)	**True Positive (TP)**	**False Positive (FP)**
Predict Clutter (Non-Potent)	False Negative (FN)	True Negative (TN)

In other words, we wanted to generate a classifier with a minimal feature that gives the maximum rate of True Positives (TP) for a certain value of False Positives (FP). Their specific objective is getting as many TPs as possible when the FPs are held to 2%. The drug company was already getting 43% TP with 4% FP on the test by using 70% of the data.

Data Preparation For Analysis

The drug company didn't want to share its experimental result with the public. Hence, the first change in the dataset was to rename the features as Feature1, Feature2, etc. During the preprocessing of the data we kept track of the features so we could identify the salient features' names at the end of the analysis.

I will use an example dataset (Table IV.6) to justify the changes. All features and values are fictitious in this dummy set. There were 5 types of situations we had to deal with:

Table IV.6. Dummy Dataset

	Class	Feature1	Feature2	Feature3	Feature4	Feature5	Feature6
1	1	ID 1		0	10	0.67	-0.300
2	1	ID 2		0	10.1	0	-1.278
3	1	ID 3		0	0	0	
4	1	ID 4		0	0	0	1.276
5	1	ID 5		0	0	0	1.198
6	1	ID 6		0	0	0.45	1.733
7	2	ID 7		0	0	0	-2.184
8	2	ID 8		0	10.3	0.56	-0.234
9	2	ID 9		0	0	0	1.095
10	2	ID 10		0	0	0	
11	2	ID 11		0	0	0.69	-0.690
12	2	ID 12		0	9.4	0	-1.690

1. Some features were identification numbers for each exemplar. Therefore these features are not really features. Feature1 in the dummy problem represents these features.
2. Some features were blank like Feature2 in our example.
3. Some features had only zero values. Feature3 is an example of such a feature.

4. Some features had many zeros with respect to nonzero exemplars for a certain feature. During the training we divide data into training and test sets randomly. If there is constant value for a feature in the training set (like Feature3 in dummy dataset), we cannot calculate SNR values for each feature. This is because we cannot standardize such data. Also a constant feature does not provide useful information as a network input. For example if we select the exemplars 3,4,5,6,7,9,10,11 or 2,3,4,5,7,9,10,12 from the dummy set as a training set, we will end up with a zero valued feature.

Keeping this in mind and looking at Figure IV.4, we decided to eliminate features that had less than 100 values different from zero.

Table IV.7 shows the excluded features with respect to the categories represented above.

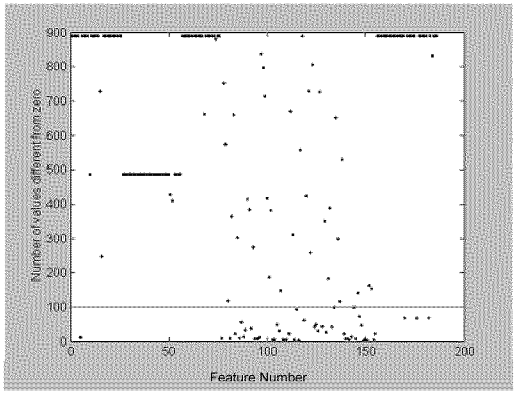


Figure IV.4. Number of values different from zero by features

Table IV.7. Excluded Features

The Reason of Excluding	Dummy dataset examples	Excluded Features
1	Feature1	1,3,4
2	Feature2	10,11,33
3	Feature3	94, 96, 98, 137, 138, 145, 146, 147, 151, 158, 162, 166, 167, 168, 172, 173, 176, 177, 178, 181, 186, 187, 188, 189, 190, and 191.
4	Feature4 and Feature5	12, 13, 107, 120, 171, 7, 84, 88, 91, 93, 95, 97, 99, 102, 104, 105, 106, 114, 115, 116, 117, 119, 121, 122, 123, 126, 127, 128, 131, 136, 139, 140, 142, 144, 150, 152, 157, 159, 160, 161, 163, 164, 165, 170, 174, 175, 179, 180, 184, 185, 206, 212, and 218.
5	Exemplar 4,11	Exemplar 326,346,356

5. Some features had several missing values for some exemplars. We deleted these exemplars. In the dummy example exemplars 4 and 11 have no values for Feature6. So these exemplars will be removed.

By removing all these features and missing values we end up with 137 features and 888 exemplars.

Parameter Settings

This was the first time we used STNGER on a real data set. Before running the STNGER to the end, we made several "warm up" runs. The main

purpose of these runs was to get some idea about the responses of the STNGER to the dataset and to determine which parameters should be included in the experiment and at what level.

The trial runs showed that the criterion for SNR change has a huge impact on the run time. Another factor that suggested we take the criterion for SNR as a design parameter is the question of whether setting this parameter to a smaller value - like 0.01 - may hinder generalization or not. A larger value may help net to improve generalization.

Basically, we were wondering what the effect would be of applying the SNR method to randomly selected subsets. The drug company determines the fractioning of the training set, therefore, this was held constant. The number of hidden nodes was selected as the number of features.

The other parameters are shown in Table IV.8.

Table IV.8. Parameter Settings

Parameters	
Fraction for training	0.7
Number of hidden nodes	Number of Features
Min Epochs to Feature Removal	100
Number of epochs to check SNR	50
Epoch to check SSE	100
Criteria for SSE change	0.01
Number of iterations for SNR method	30
Number of features to display	30
Number of features from each subset to retain	10

The designed experiment had two factors, "Criterion for SNR Change" and "Using the entire dataset. These were set at two levels 0.01 or 0.1 for "Criteria for SNR Change", and TRUE or FALSE for "Using the entire dataset". The combinations of those factors and levels resulted in four experimental four experimental runs given in Table IV.9

Table IV.9. Experimental Parameter Settings

Parameters

Run	1	2	3	4
Fraction for training	0.7	0.7	0.7	0.7
Number of hidden nodes	Same as the number of features			
Min epochs to feature removal	100	100	100	100
Number of epochs to check SNR	50	50	50	50
Epoch to check SSE	100	100	100	100
Criteria for SSE change	0.01	0.01	0.01	0.01
Number of iterations for SNR method	30	30	30	30
Number of features to display	30	30	30	30
Number of features from each subset to retain	-	10	-	10
Criteria for SNR change	**0.01**	**0.01**	**0.1**	**0.1**
Use entire dataset	**TRUE**	**FALSE**	**TRUE**	**FALSE**
Use subsets	**FALSE**	**TRUE**	**FALSE**	**TRUE**

Those runs represent four areas of interest to this research.

During the results section, these experimental runs will be identified as Run1, Run2, Run3, and Run4.

Results

In this section, we will talk about the performances of the different runs, Run1 through Run4. The details of the selected features and the relations

between these feature subsets are not explained due to the sensitivity of the results. The ranking plots are presented at Appendix A.

Looking at Figure IV.5, Run 1, an immediate observation is the performance increase as the number of features increases. Actually this is a typical characteristic of such performance plots. If we divide the plot into three different regions, we see that, in region 1 the performance is increasing very fast. In the second region we realize an increase but not as much as in the first region. In the third region, the performance is almost at the same level. By looking at this plot one can reach a conclusion that keeping 40 features will give us almost the best performance, approximately 74% classification accuracy.

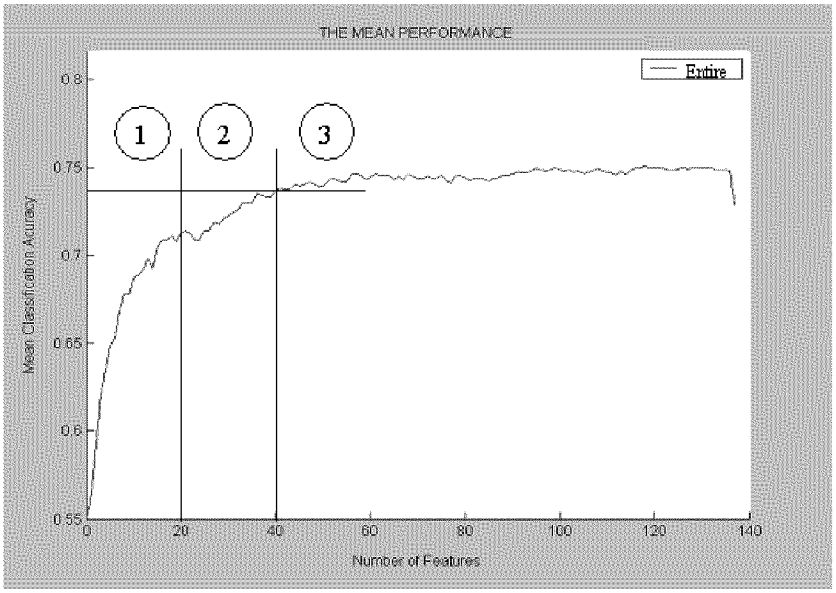


Figure IV.5. Entire dataset with SNR change 0.01

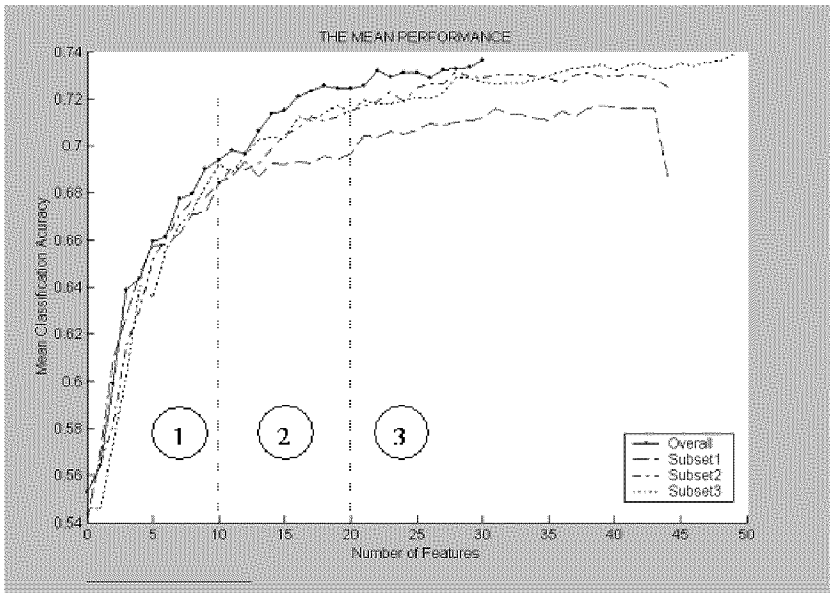


Figure IV.6. Three subsets with SNR change 0.01

Figure IV.6, Run2 shows the performance of the 3 subsets and the dataset created by taking the top 10 features of these 3 subsets. In the mathematical notation of chapter 3

$s = 3$	number of subsets
$F_1 = F_2 = 44$	number of features in subset 1 and subset 2
$F_3 = 49$	number of features in subset 3
$k = 10$	number of features to retain from each subset

In region 1 all three subsets' performances are increasing very fast. In region 2 this increase is smaller, and in region 3 the performances are almost at the same level. By looking at these performances we can decide the number of features to be retained from each dataset. The overall dataset represents the dataset created by taking top 10 features from each subset. It seems that the new

data set, which is created by retaining top 10 features from each subset, has a better performance than all other subsets. The performance with these 30 features is almost at the same level with the performance of the 40 features found in setting 1.

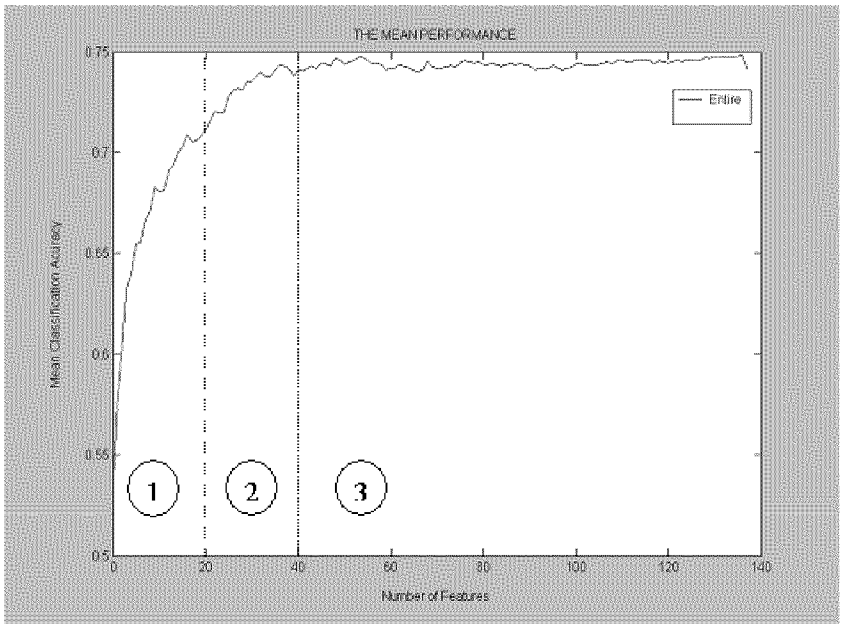


Figure IV.7. Entire dataset with SNR change 0.1

By contrasting the performance plots of Run1, Figure IV.5, and Run3, Figure IV.7, we can say that there is no significant difference between the results of Run1 and Run3. The only difference between these two runs settings is the SNR change criteria. The same results can be deduced by looking at the results of Run2 and Run4 (Figure IV.6, Run2, and Figure IV.8, Run4). For this dataset, it can be concluded that the SNR change criteria does not impact the classification

accuracy. In other words, setting SNR change criteria to a larger value –like 0.1– does not help to improve the generalization, or, setting SNR change criteria to a smaller value –like 0.01– does not help to get a better subset of salient features. In the conclusions section this will be discussed in more detail.

On the other hand, as we experienced during the warm up runs the SNR change criteria had a huge impact on the different runs. Using SNR change criteria as 0.01 increased the run time. For this particular problem, Run3 took only 40% of the run time of Run1. This is also true for Run4 and Run2. Run time of Run4 was about 30% of the run time of Run2.

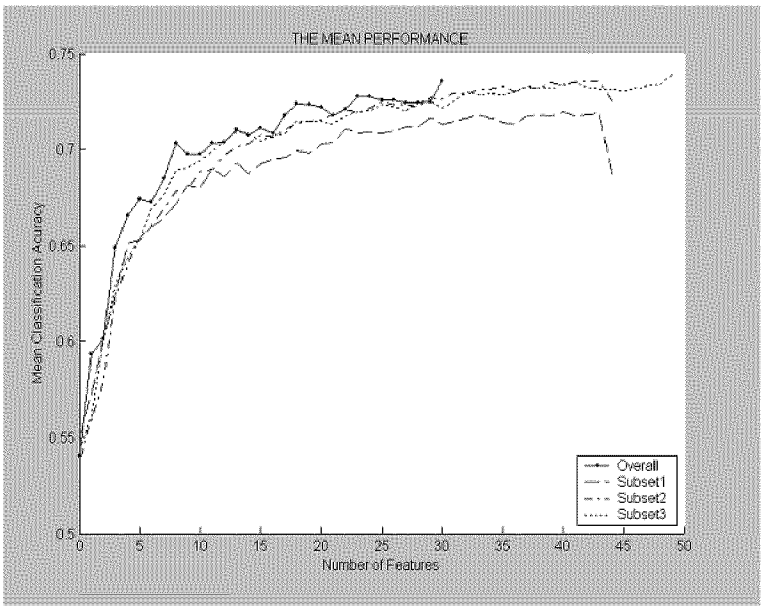


Figure IV.8. Three subsets with SNR change 0.1

According to the results provided until this point, one can conclude that applying SNR Method to Randomly Selected Subsets provided a better feature

subset in terms of mean classification accuracy. In other words, reducing the ratio of number of features to number of exemplars of a dataset, on which the SNR method is applied, will help the SNR method to determine the salient features. By dividing the data set into subsets we reduce the number of noise features in a data set. Having less – in an absolute sense – number of noise features in a data set will increase the SNR method's performance.

ROC Curve Values as a Function of Number of Features

In some cases, classification accuracy does not provide sufficient information as a measure of performance. As we mentioned while introducing the data set, the drug company is interested in TPs and FPs rather than the overall classification accuracy. This motivation forced us to create a plot that helps us to see TP and FP as a function of number of features.

A mathematical notation, which can help us to understand how we gathered these informations, might be

θ : Decision threshold

Pfa_{Train} : Probability of false alarm on training set

Pfa_{Test} : Probability of false alarm on test set

Pd_{Train} : Probability of detection on training set

Pd_{Test} : Probability of detection on test set.

The goal is to maximize TPs for a certain value of FP. The algorithm below is used to evaluate different TP values for different numbers of features.

1. Set number of features f to 30, set Pfa_{Train} to 0.02.

2. Set counter equal to 1. (We averaged across 30 ROC curves for each feature subset)

3. Create a single hidden layer network object with the parameters:

Train Function : Traingdx.m (An adoptive learning algorithm of MATLAB)
Learning Rate : 0.01 (Default)
Maximum number of epochs: 350
Performance goal (mse) : 0.0010
Learning decrease : 0.7000 (Default)
Learning increase : 1.0500 (Default)
Maximum number of fails on validation set: 25
Maximum performance increase: 1.04 (Default)
Momentum Constant : 0.9000 (Default)
Minimum Gradient : 1.0000e-006 (Default)
Plot show : 25 (Default)
Maximum time : Infinity (Default)

4. Train network.

5. Simulate network and create ROC curve for training set.

6. Determine θ and Pd_{Train} for $Pfa_{Train} = 0.02$.

7. Create ROC curve for test set by simulating the network.

8. Determine Pd_{Test} and Pfa_{Test}.

9. Increase counter by 1

10. If counter is smaller than 31 go to step 3.

11. Calculate means of θ, Pd_{Train}, Pd_{Test}, and Pfa_{Test}

12. If $f > 1$, remove f^{th} feature, and go to step 2.

Figure IV.9 visualizes the steps of the algorithm. At this point, we have to talk about the validation set. There are a number of built-in stopping criteria in MATLAB's traingdx.m function. One of them is we create a validation set and monitor the error on the validation set. If the error increases for a certain number of iterations on the validation set the training will be stopped and the weights and biases at the minimum of the validation error are returned [16]. To be able to apply the same criteria we selected 20% of the training set randomly and assigned it as a validation set. In this way, we would stop training if the error increases on the training set for a certain number of epochs. We determined the maximum number of epochs and maximum number fails from "warm up" runs.

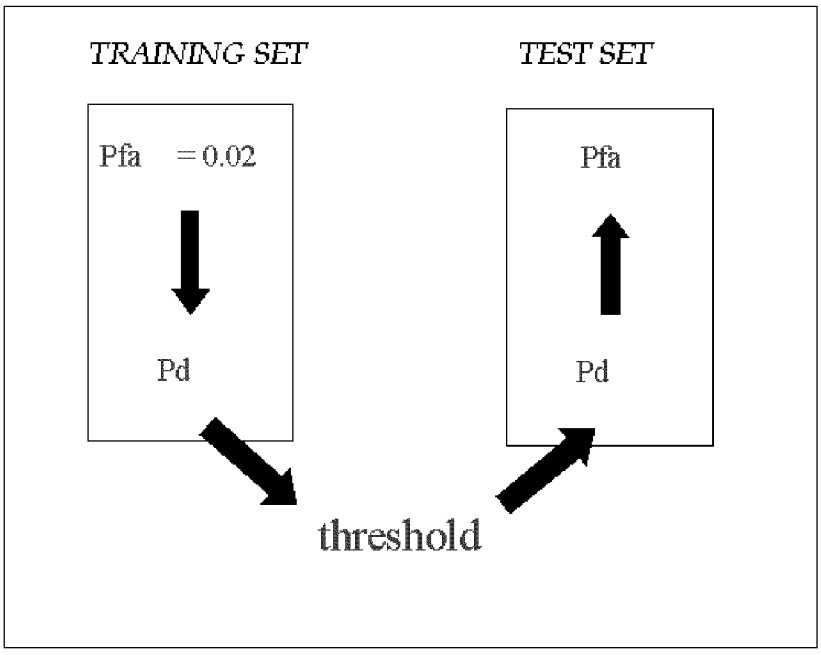


Figure IV.9. Collecting ROC Information

Based on these values, we created four plots representing the four runs outlined in Table IV.9. There are 6 different lines in these plots:

 : Threshold

 : Probability of detection on train set

 : Probability of detection on test set

 : Probability of false alarm on train set

 : Representing the 2% and the 4% probability of false alarm rates

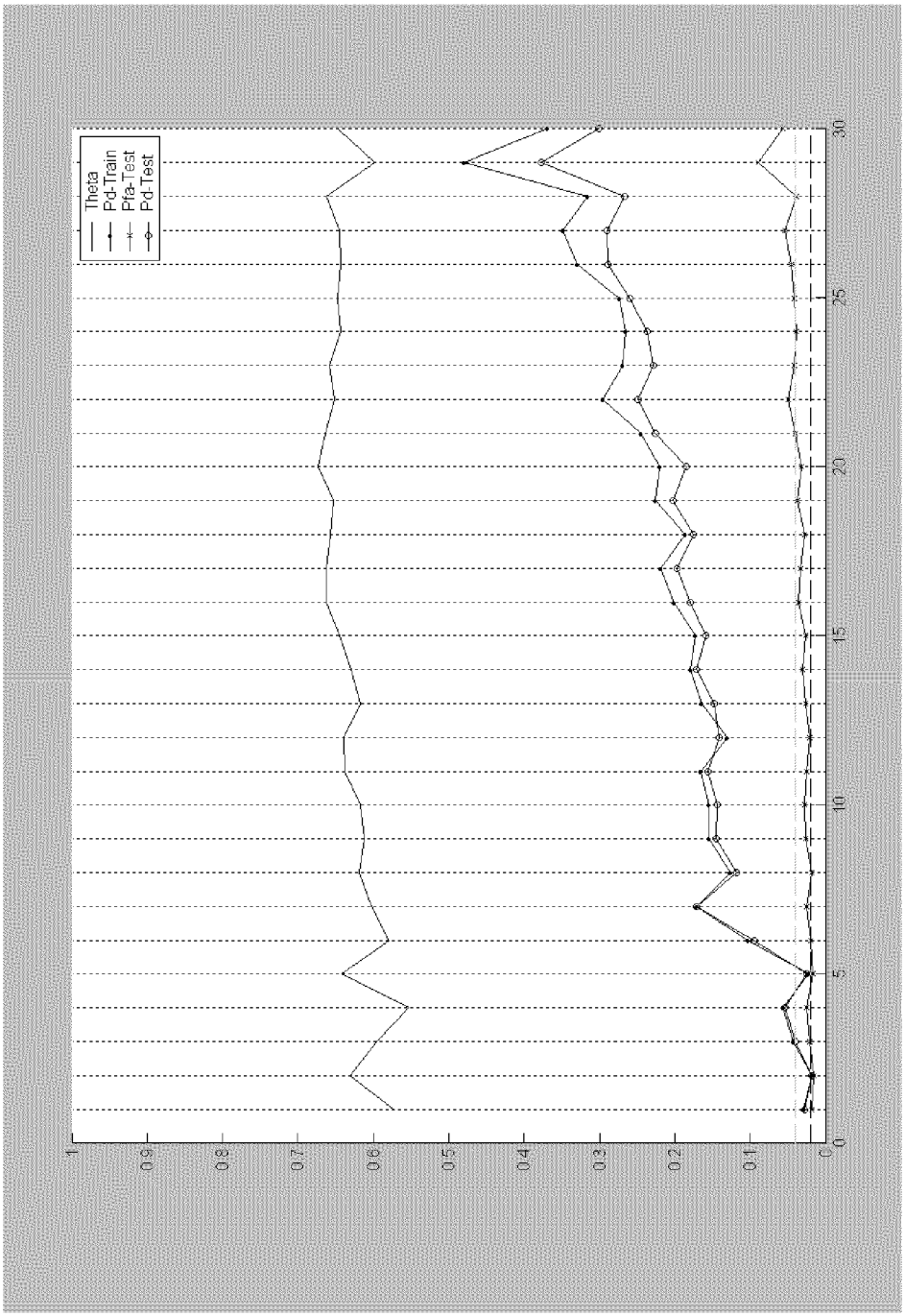


Figure IV.10. Entire Dataset with SNR Change Criteria 0.01

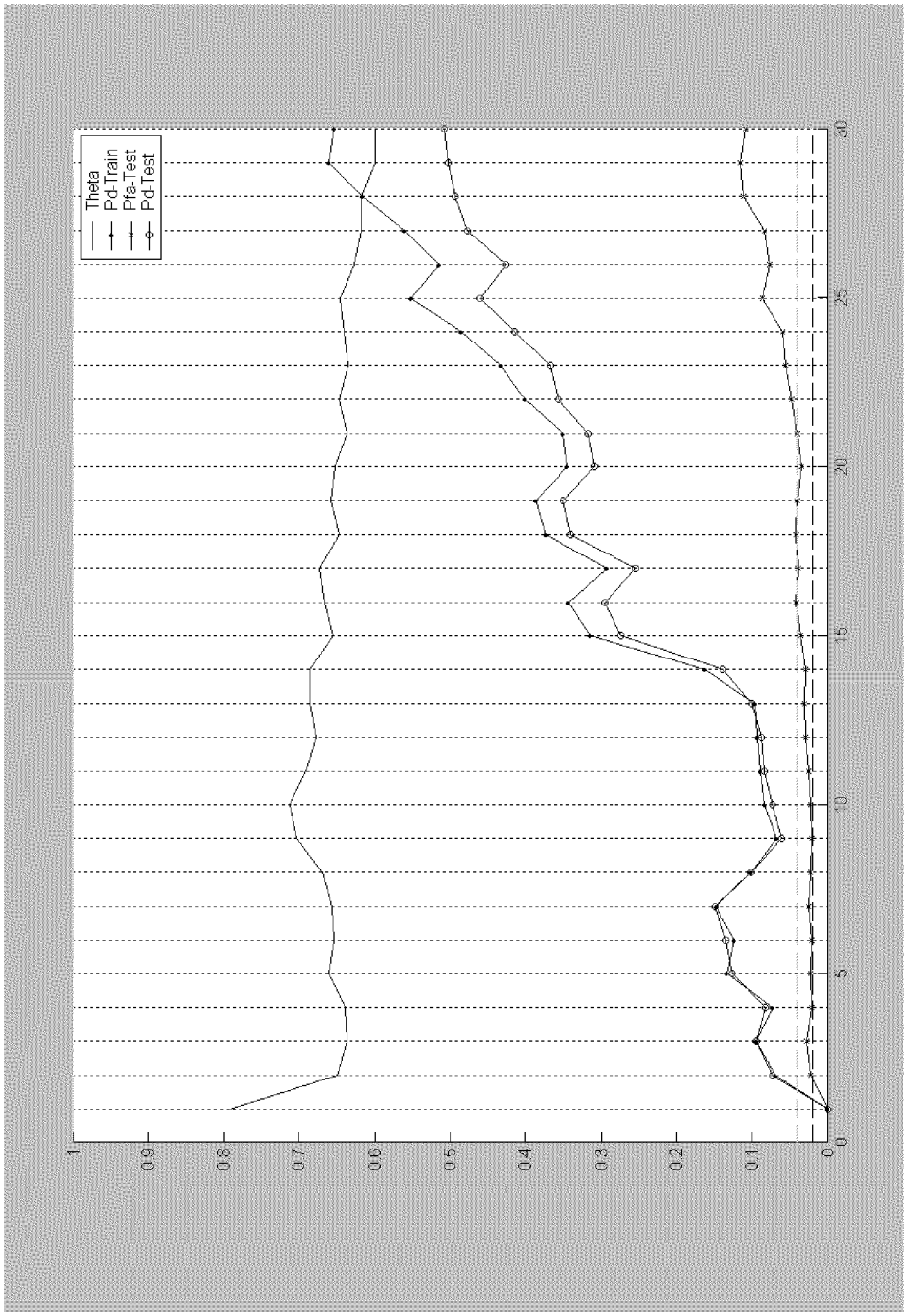


Figure IV.11. Three Subsets with SNR Change Criteria 0.01

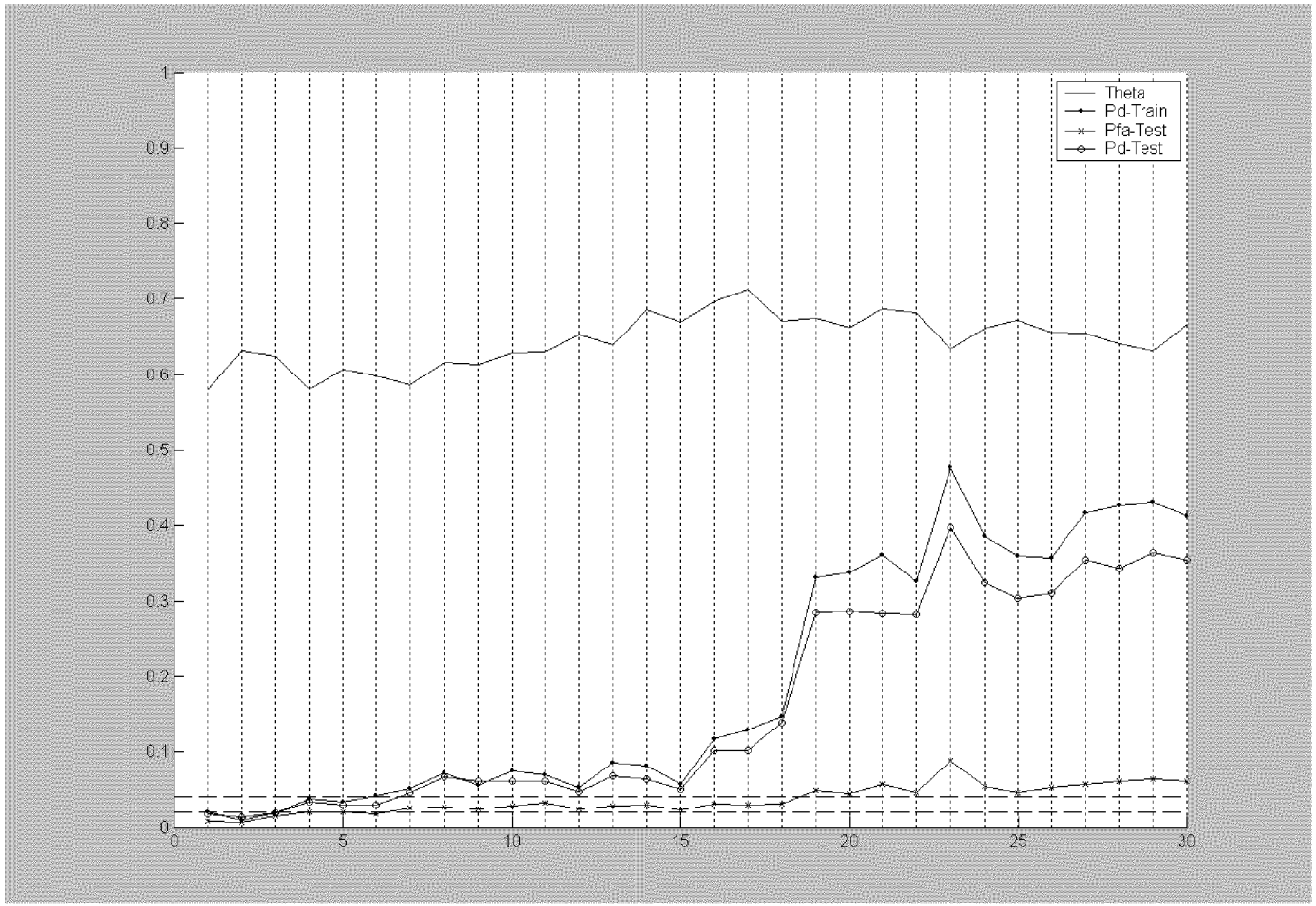


Figure IV.12. Entire Dataset with SNR Change Criteria 0.1

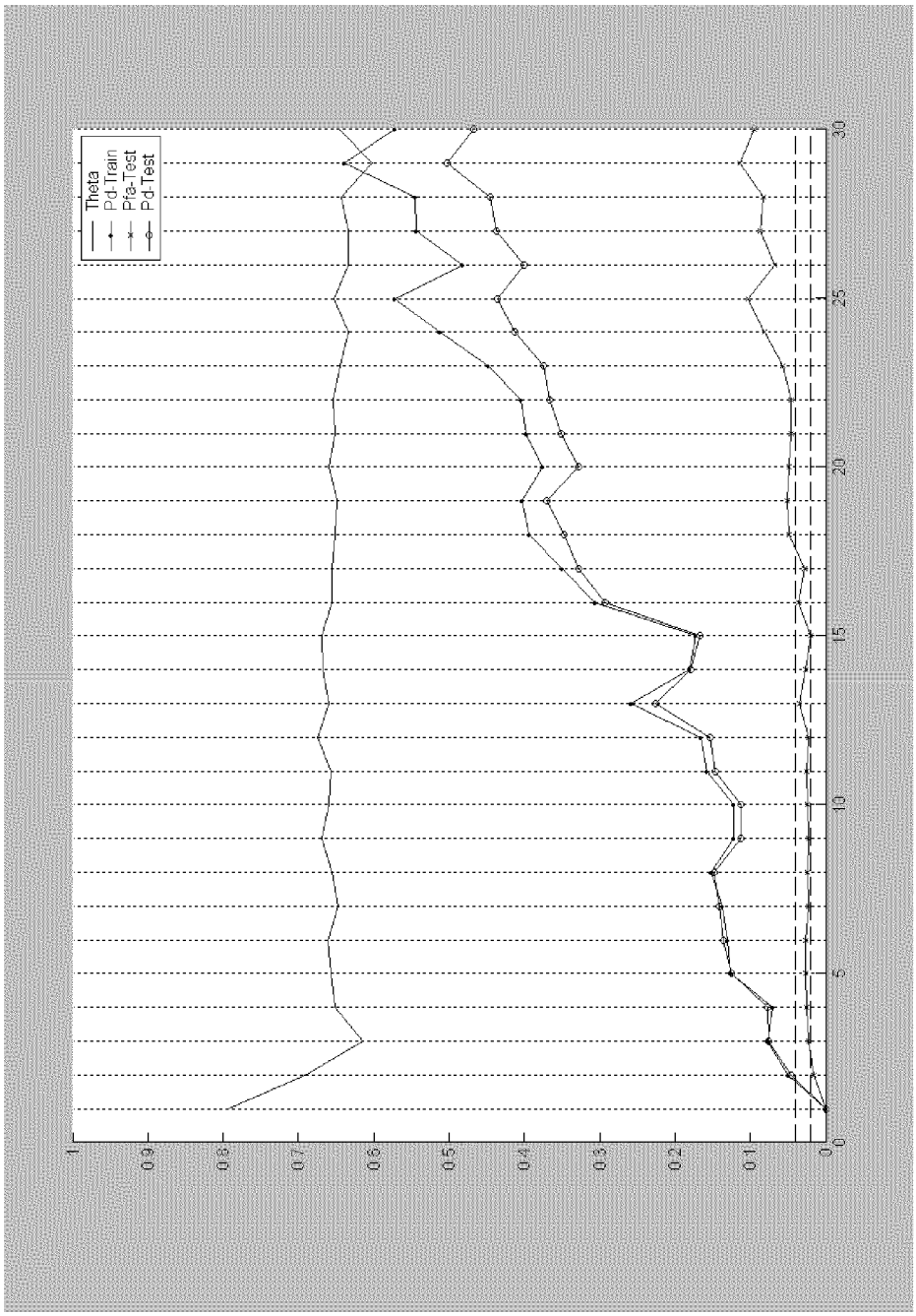


Figure IV.13 Three Subsets with SNR Change Criteria 0.1

Conclusions

An immediate observation might be the threshold values are almost at the same level for all settings.

Comparison of the Performances regarding to r ratios: To see the distinctions in more detail we depicted the same kind of information for different runs on the same plot. Three plots were created for each combination showing the probability of detection on training set (bottom), probability of false alarm on test set (middle), and probability of detection on test set (top).

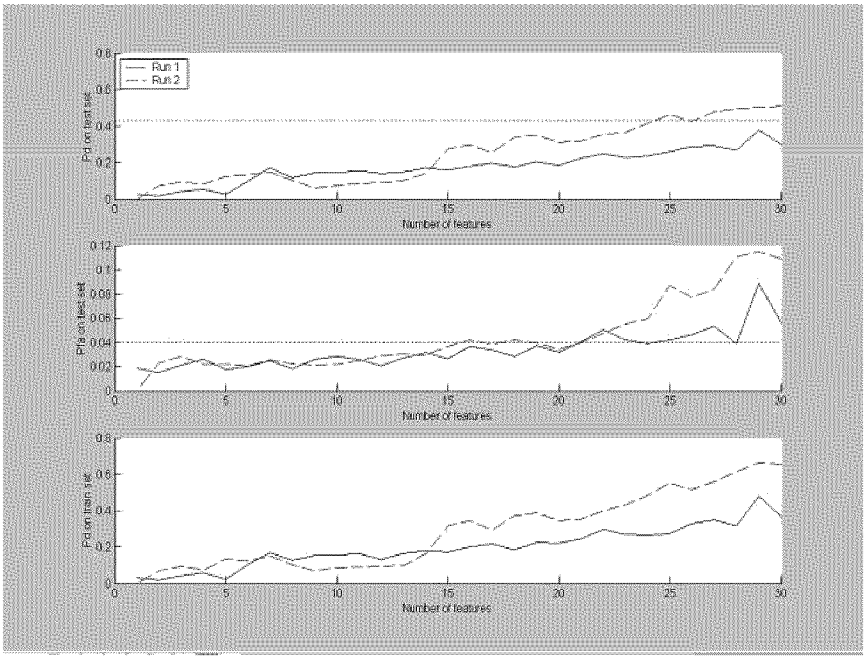


Figure IV.14. Comparison Plot of The Performances of Run1 and Run2

Figure IV.14 compares Run1 and Run2. Recall that we used SNR Change Criteria as 0.01 at Run1 and Run2. We used entire dataset at Run1 and created

three subsets at Run2. It seems that applying SNR method gives better results when the r ratio is smaller. Especially when the number of features is more than 15, subset creation is helping to determine the salient features. We notice that Pd_{Test} for Run1 is less than Pd_{Test} for Run2, but unfortunately Pfa_{Test} grows as Pd_{Test} grows. But this increase does not lessen the merit of SRSS. Table IV.10 provides a couple of points from this plot.

Table IV.10. A couple of performance examples from Run1 and Run2

	Number of features	θ	Pd_{train}	Pfa_{test}	Pd_{test}
RUN 1	25	0.65	0.27	0.04	0.26
	29	0.60	0.48	0.09	0.38
RUN 2 (SRSS)	19	0.66	0.39	0.04	0.35
	27	0.62	0.56	0.08	0.48

It is obvious that SRSS is providing better performance with fewer features than applying SNR method to entire dataset in terms of Pfa and Pd.

Figure IV.15 shows a comparison of Run3 and Run4 , similar to Figure IV.14 for Run1 and Run2. Remember that we used an SNR Change Criteria as 0.1 at Run3 and Run4. We used entire dataset at Run3 and created three subsets at Run4.

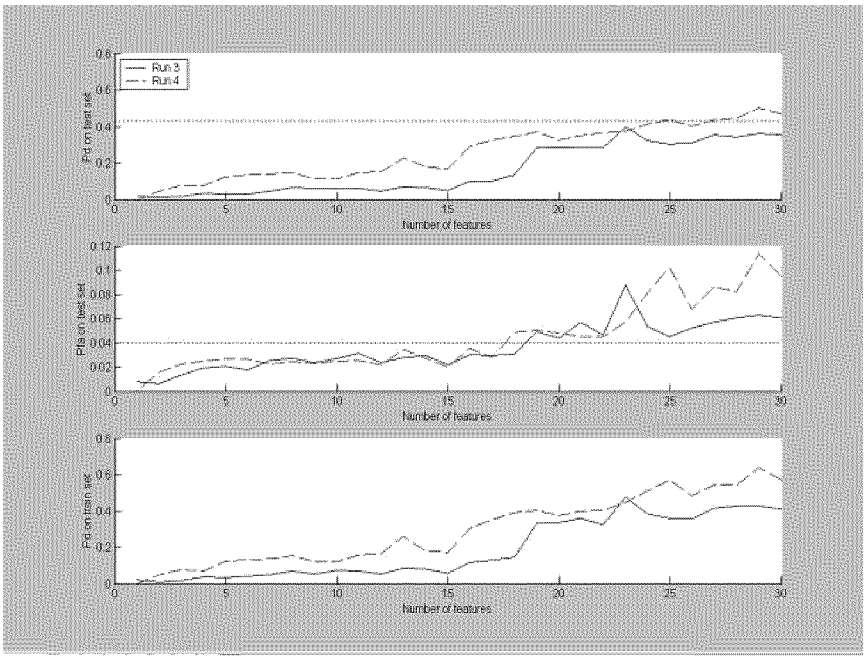


Figure IV.15. Comparison Plot of The Performances of Run3 and Run4

It seems that Run4 always gives better subsets than Run3.

Table IV.11. A couple of performance examples from Run3 and Run4

	Number of features	θ	Pd_{train}	Pfa_{test}	Pd_{test}
RUN 3	25	0.672	0.360	0.045	0.303
	27	0.655	0.417	0.057	0.354
RUN 4 (SRSS)	22	0.654	0.404	0.045	0.366
	23	0.644	0.449	0.057	0.375

Among all runs, Run2 gives the best performance with the same number of features. The performance values and number of features that would be retained from these subsets are always subject to the decision maker's criteria.

The visual comparison has provided that the amount of data or the ratio of the number of features to the number of exemplars, *r*, has an impact on

selecting salient features. At this point, we investigated if there is a statistical difference between applying SNR method to different datasets that have different r ratios.

The *paired t confidence interval* is one of the statistical approaches that can be used in comparing two different systems. The only required assumption is the pairwise differences must be normally distributed [22]. The good thing with this approach is we don't have to assume that two systems are independent and have equal variances [22,23].

A $(1-\alpha)100\%$ *paired t confidence interval* is given by

$$\bar{d} \pm t_{\alpha/2} \cdot \left(\frac{S}{\sqrt{n}} \right)$$

where $d_i = X_{1i} - X_{2i}$, X_{ji} is a random variable, $j = 1,2$, $S^2 = \dfrac{\sum_{i=1}^{n} d_i^2 - \frac{1}{n} \cdot \left(\sum_{i=1}^{n} d_i \right)^2}{n-1}$,

$t_{\alpha/2}$ is such that $P\{T > t_\alpha\} = \alpha$ for a *t* distribution with *n-1* degrees of freedom.

Our null hypothesis and the alternative hypothesis will be

Null Hypothesis: $H_o : \mu_D = 0$: There is no difference between performances at $\alpha = 0.05$ level where $\mu_D = \mu_i - \mu_j$.

Alternative Hypothesis: $H_a : \mu_D \neq 0$: There is a difference between performances.

We will assume that if the confidence interval covers zero then the null hypothesis cannot be rejected. Otherwise we will reject the null hypothesis and conclude that there is difference between two systems. If the confidence interval leads to a positive interval we can say that system1 is better than system2. If the confidence interval leads to a negative interval we can say that system2 is better than system1 [23].

We summarized the results of hypothesis test results in the below table. Table IV.12 shows the number of times each run gives the better result.

Table IV.12. SNR Criterion=0.01

	RUN 1	RUN 2 (SRSS)	No statistical difference
Pd_{train}	6	18	6
Pfa_{test}	1	4	25
Pd_{test}	6	20	4

One important result of the Table 4.8 is although there is not much difference between Pfa_{Test} for both methods, SRSS is giving better results for Pd_{Test} and Pd_{Train}.

Table IV.13. SNR Criterion=0.01

	RUN 3	RUN 4 (SRSS)	No statistical difference
Pd_{train}	1	23	6
Pfa_{test}	1	6	23
Pd_{test}	1	25	4

Again, Run4 is giving better results than Run3.

Comparison of Performances Regarding to SNR Change Criteria:

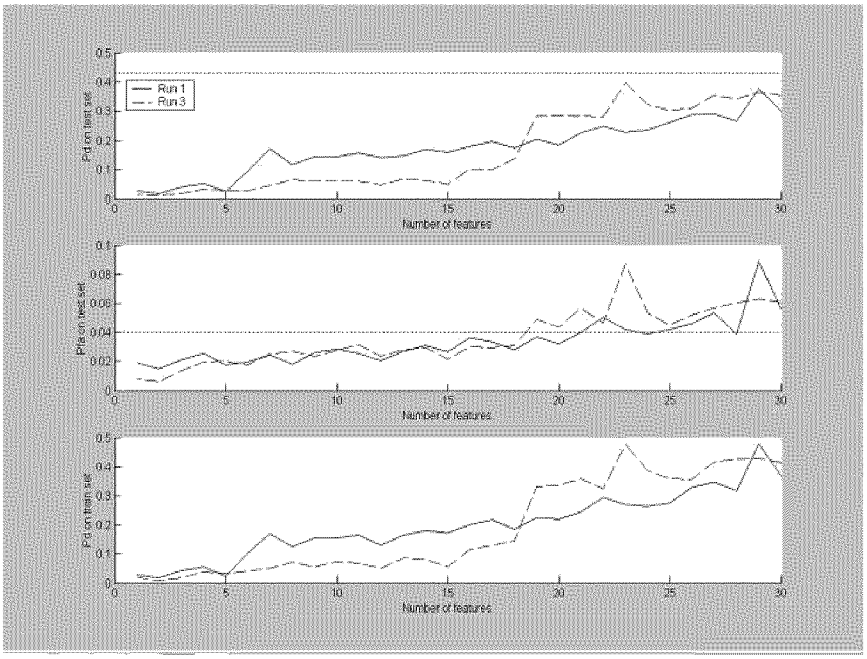


Figure IV.16. Comparison Plot of The Performances of Run1 and Run3

Figure IV.16 compares Run1 and Run3. Visually one might notice that for small number of features Run1 gives better results than Run3. However, when the number of features is more than 18, Run3 provides a better performance than Run1. This is statistically true only 4 times for Pd_{Test} and Pd_{Train}. The smaller SNR change criteria –0.01 – gives better results than the SNR change criteria – 0.1 – when the SNR method applied to the entire dataset or the r ratio is large.

Table IV.14. SNR change criteria in parentheses

	RUN 1 (0.01)	RUN 3 (0.1)	No statistical difference
Pd_{train}	14	4	12
Pfa_{test}	1	1	28
Pd_{test}	14	4	12

Table IV.15. Design Parameters in Parentheses

	RUN 2 (SRSS, 0.01)	RUN 4 (SRSS, 0.1)	No statistical difference
Pd_{train}	1	5	24
Pfa_{test}	1	0	29
Pd_{test}	1	7	22

However, Figure IV.17 and confidence intervals shown in

Table IV.15 indicate that SRSS is very robust to different SNR change criteria.

Although there is break point on performances between 15 features and 20 features, see Figure IV.10, Figure IV.11, Figure IV.12, and Figure IV.13, for other three runs, Run1 has a steadily increasing rate of probability of detection and probability of false alarm. This change brings to mind a possible interaction between features ranked within first 15 and between 15-30.

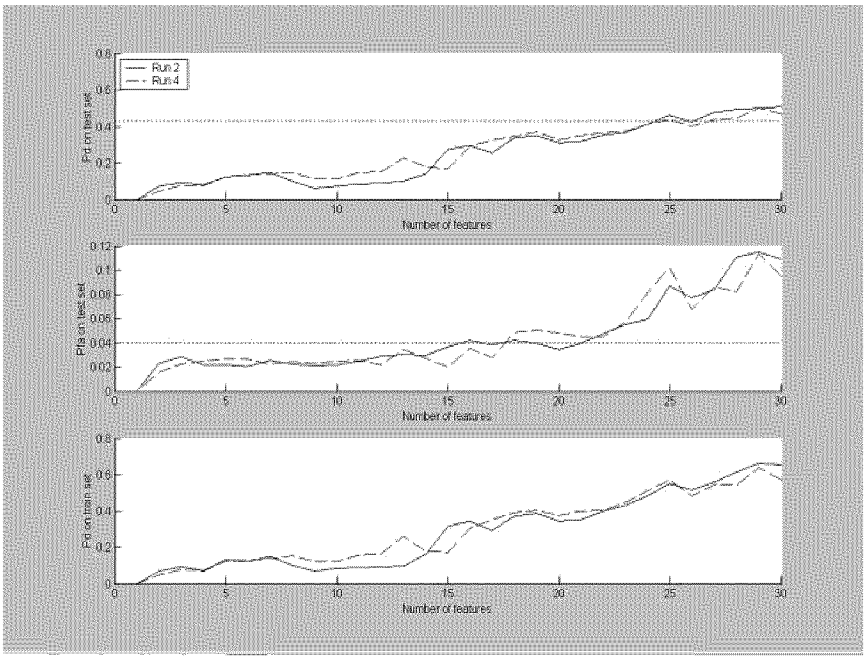


Figure IV.17. Comparison Plot of The Performances of Run2 and Run4

With 19 features, Run2 (Figure IV.11) gives a 3.9% Pfa with a 35% Pd on test set. Recall that the drug company results were 4% Pfa, 43% Pd with 137 features. Therefore, we see that the use of SRSS has resulted in a situation which nearly duplicates the performance of the drug company but with for greater efficiency. Further, if we accept a 10% Pfa, we can get a 50.7% Pd on test set with 30 features.

In some studies, decreasing the number of features might be very helpful to improve the chances that a solution will be both understandable and practical. The drug company may now focus on these features to improve the solution.

V. Summary and Recommendations

Overview

In this thesis study, we had two objectives. The first one was creating a graphical user interface that can be used in feature selection process. The second objective was to select salient features in a situation where there is a high ratio of feature to exemplars. To achieve both objectives, we used the Signal-to-Noise saliency method [8].

Selecting salient features in a situation, where there is a high ratio of feature to exemplars, was our second objective. In chapter three, we presented a new idea method for applying the SNR method. The new methodology suggested that, briefly, one could apply the SNR method to randomly selected subsets of a data set, and recombine the best k features from each subset. Then, apply the SNR method to newly created dataset. The new methodology, SRSS (SNR with Randomly Selected Subsets), was applied to real dataset from a drug study. The results of this experiment presented in chapter 4. SRSS produced a better feature subset than the current application with regards to classification accuracy, probability of detection and probability of false alarm.

In chapter three we introduce STNGER (Signal to Noise Graphical Evaluation Routine) as a graphical interface for feature saliency screening. The basic algorithms and a user manual of STNGER are presented in chapter 3. Basically, we put SNR method into an easy to use format. The output plots are

designed to visualize screening results. STNGER was evaluated on a well-known abstract problem, the corrupted version of a well-known real world problem and a real world problem with unknown salient features. All results were promising.

As an addition to STNGER, we put SRSS method as an option. The STNGER can operate on both an entire dataset and randomly created subsets.

Overall, STNGER and SRSS seem very reliable in feature screening process. STNGER did very well on both the well-defined abstract problem and real dataset. The statistical analysis of the real world problem provides sufficient evidence to the merit of STNGER.

Limitations and Recommendations

The only limitation to the SRSS Method might arise from placing features, which have an interaction between them, into different groups.

In mathematical notation, say, we have N exemplars and M features. According to a given ratio, r, we created s subsets. Let's assume that, the interaction of the features, say, x_1 and x_2 determines the classification rule. If we happen to put these two features into different groups, we would lose some information about the saliency of these features.

To investigate this situation we defined a problem with a known classification rule. We created 8 features, each of which is a uniformly

distributed (-1,1) random variable, with 100 exemplars. The classification rule is determined as

$$\text{Target=} \begin{cases} \text{True} & \text{if } (x_{1i} \cdot x_{2i} > -0.1) \text{ and } (x_{3i} \cdot x_{4i} > -0.1) \\ \text{False} & \text{otherwise} \end{cases}$$

where $i = 1...N$, N is the number of exemplars. The classes are evenly distributed.

The purpose of the experiment was to answer the question:

Can we use within group mean rankings as a global metric between the subsets? For example, if a feature's (say, x_1) mean ranking determined as 1.4, and another feature's (say, x_2) mean ranking determined as 1.7, both features from different subsets, could we say that x_1 is more salient than x_2? Actually, the main point is not so much ranking features according to their subgroup mean rankings. But rather, we want to catch the differences between rankings in different subgroups at different iterations (at each iteration the member features of subsets are changing). If a feature is ranked significantly higher in a specific relative to other subsets, it could indicate this feature has an interaction with at least one other feature. This suggests ways that we might rearrange our SRSS algorithm to avoid missing separated but interacting features.

We designed an experiment with three settings. The first run included the entire dataset. For the second and third runs we divided the entire set into to subsets. First subset was created by choosing x_1, x_3, $noise_1$, and $noise_3$. The

second subset was created by choosing x_2, x_4, $noise_2$, and $noise_4$. We investigated the different rankings for different subsets.

We used single hidden layer network architecture with the following parameters:

Train Function : Traingdx.m (An adoptive learning algorithm of MATLAB)
Learning Rate : 0.01 (Default)
Maximum number of epochs : inf
Performance goal (sse) : 0.00001
Learning Decrease : 0.7000 (Default)
Learning Increase : 1.0500 (Default)
Maximum performance increase : 1.04 (Default)
Momentum Constant : 0.9000 (Default)
Minimum Gradient : 1.0000e-006 (Default)
Plot show : 25 (Default)
Maximum time : Infinity (Default)
Number of hidden nodes : Number of features
SNR change criteria : 0.01
Number of SNR method : 30
Train/Test Fraction : 60/40

Table V.1. The expected ranks of separate but interacting features

	Run 1								Run 2				Run 3			
Features	x_1, x_2, x_3, x_4, n_1, n_2, n_3, n_4								x_1, x_3, n_1, n_3				x_2, x_4, n_2, n_4			
Ranking	x_4, x_3, x_2, x_1, n_1, n_2, n_3, n_4								x_3, x_1, n_3, n_1				x_4, n_2, x_2, n_4			
Expected Ranks	x_1	X_2	X_3	x_4	n_1	n_2	n_3	n_4	x_1	x_3	n_1	n_3	x_2	x_4	n_2	n_4
	4.16	3.23	3.03	2.86	5.36	5.46	5.56	6.3	2.33	2.26	2.96	2.43	2.6	2.3	2.33	2.76

Table V.1 summarizes the experiment and results. The last three columns correspond to the three runs. The first row shows the features that are included in a run. The second row shows the ranking for a specific run. For example, for

run 1, x_4 is ranked as the first and x_3 is the second and so on. The third row shows the expected rankings of each feature. For example, the expected rank of x_1 in run 2 is 2.33. Surprisingly, although the interaction features are not included, SNR method detected the salient features in run 2. But we did not see the same performance in run 3.

If we take the expected ranks for run 2 and run 3 and sort in an ascending order, we create Table V.2.

Table V.2. Combined expected ranks of features from different subsets

Features	x_3	x_4	x_1	n_2	n_3	x_2	n_4	n_1
Expected Ranks	2.26	2.3	2.33	2.33	2.43	2.6	2.76	2.96

It is very interesting that, the SNR method found the 75% of the salient features although interaction terms are in different subsets.

The results above might suggest some changes in the SRSS methodology. Further research might investigate the effectiveness of the suggestions given below.

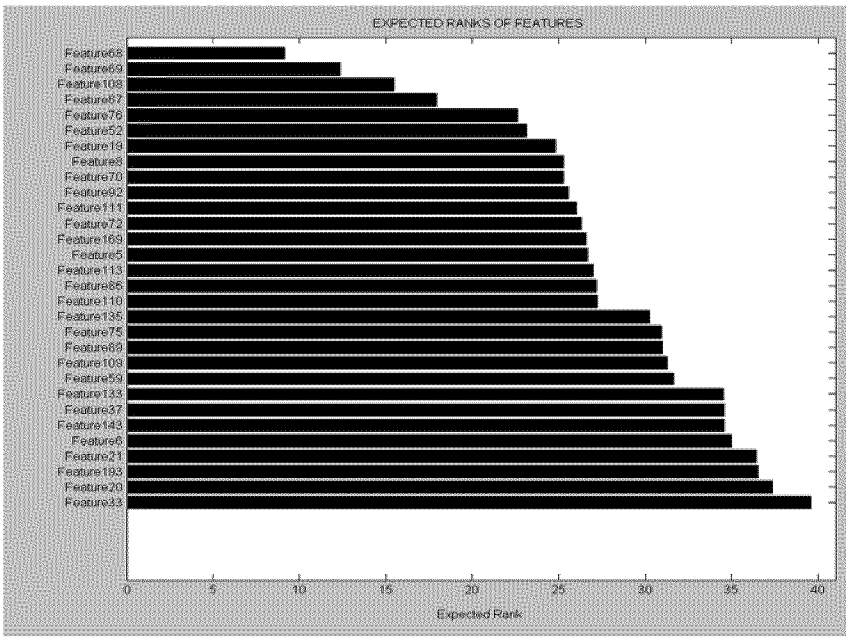
1. Instead of creating subsets only one time and applying SNR method to each subset *t* time, one can create subsets *t* times and applies SNR method for each subset. If the assumption of global ranks are acceptable, some evidence of this is given in the above experiment, one can calculate the rankings based on rankings within each subset for

each of the iterations. One can create subsets randomly or with a given rule for all iterations.

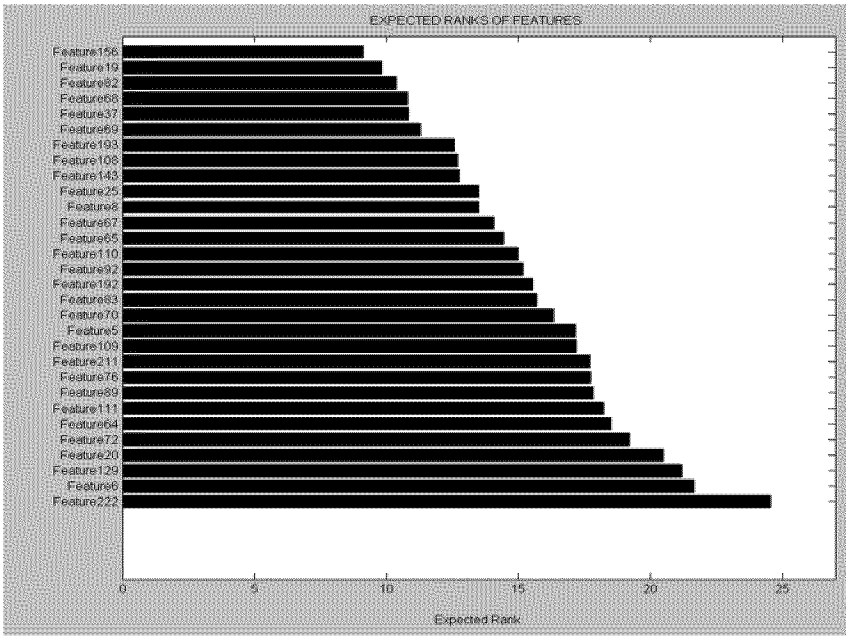
2. Allowing user to select different k values for different subsets by looking at the performances plots of these subsets, then creating and training the new dataset. This will avoid the setting k without any prior knowledge about performance of features, and choosing the same k for all subsets.
3. The effect of random components, which are defined in chapter 3, on screening performance might provide some more insight on the screening experience.

Appendix A: Expected ranks of features

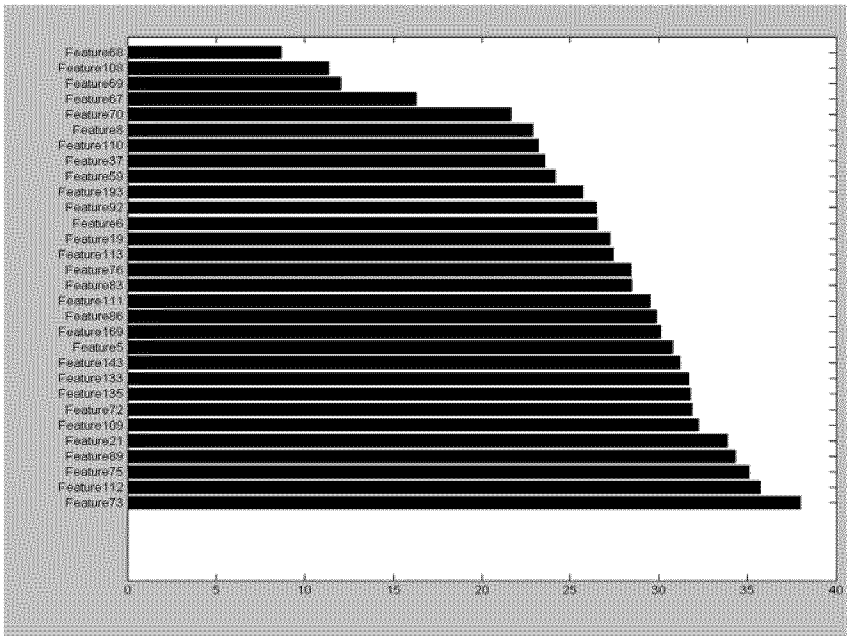
Run 1:



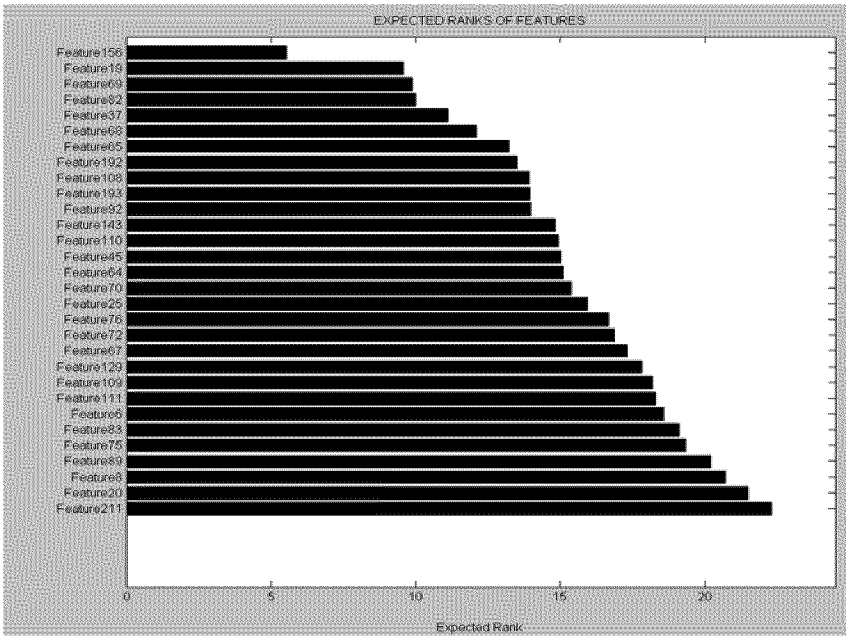
Run 2:



Run 3:


Feature68
Feature108
Feature69
Feature67
Feature70
Feature8
Feature110
Feature37
Feature59
Feature193
Feature92
Feature6
Feature19
Feature113
Feature76
Feature83
Feature111
Feature96
Feature189
Feature5
Feature143
Feature133
Feature135
Feature72
Feature109
Feature21
Feature89
Feature75
Feature112
Feature73
0
5
10
15
20
25
30
35
40

Run 4:


EXPECTED RANKS OF FEATURES
Feature156
Feature19
Feature69
Feature82
Feature37
Feature68
Feature65
Feature192
Feature108
Feature193
Feature92
Feature143
Feature110
Feature45
Feature64
Feature70
Feature25
Feature76
Feature72
Feature67
Feature129
Feature109
Feature111
Feature6
Feature83
Feature75
Feature89
Feature8
Feature20
Feature211
0
5
10
15
20
Expected Rank

Bibliography

1. Nadler, Morton and E. P. Smith. *Pattern Recognition Engineering*. New York: John Willey & Sons. 1993.

2. Steppe, Jean M. *Feature and Model Selection in Feedforward Neural Networks.* PhD Dissertation, Air Force Institute of Technology, Wright Patterson AFB OH, June 1994.

3. Hall, Mark A. "Correlation Based Feature Selection For Discrete and Numeric Class Machine Learning". *Proc. 17th International Conf. on Machine Learning*, Morgan Kaufmann, San Francisco CA, 2000.

4. Kohavi, Ron. "*Wrappers For Performance Enhancement and Oblivious Decision Graphs*". Ph.D. Thesis, Stanford University, Computer Science Department. 1995.

5. D.W. Ruck. *Characterization of Multilayer Perceptrons and Their Applications to Multisensor Automatic Target Detection*, Ph.D. Dissertation, Air Force Institute of Technology, WPAFB OH, 1990.

6. D.W. Ruck, S.K. Rogers, M.Kabriski. Feature Selection Using A Multilayer Perceptron, *J. Neural Network Comput.* 2(2), 1990.

7. G.L. Tarr. *Multilayer Feedforward Neural Networks For Image Segementation*, Ph.D. Dissertation, Air Force Institute of Technology, WPAFB OH, 1991.

8. Bauer, Jr., K.W., Stephen G. Alsign, and Kelly A. Greene. "Feature Screening Using Signal to Noise Ratio" *Neurocomputing* 31(2000) 29-44.

9. L. M. Belue, K.W. Bauer. Methods of Determining Input Features for Multilayer Perceptrons, *Neurocomputing* 7 (2) (1995) 111-121.

10. L.M.Belue, *Multilayer Perceptrons for Classification*, M.S. Thesis, Air Force Institute of Technology, WPAFB OH, 1992.

11. J.M. Steppe, K.W. Bauer, Improved Feature Screening in Feedforward Neural Networks, *Neurocomputing* 13 (1996) 47-58.

12. J.M. Steppe, K.W. Bauer Jr., S.K. Rogers. Integrated Feature and Architecture Selection, *IEEE Trans. Neural Networks* 7 (4) (1996) 1007-1014.

13. Dillon, W.R and M. Goldstein. *Multivariate Analysis Methods and Applications*, John Willey & Sons, New York, 1984.

14. Devijver, P.A. and J. Kitler. *Pattern Recognition*, Prentice Hall, Englewood Cliffs NJ, 1982.

15. Huitema, B.E. *The Analysis of Covariance and Alternatives*, John Willey & Sons, New York, 1980.

16. MATLAB Help Files, Release 12.

17. Bauer, K. W. "*OPER 685, Applied Multivariate Data Analysis*," Spring 2001. Air Force Institute of Technology, WPAFB OH.

18. Looney, C.G. *Pattern Recognition Using Neural Networks*, Oxford University Press, New York, 1997.

19. Bishop, C.M. *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford, 1995.

20. Gibson, G.J. and Cowan, C.F.N., On The Decision Boundaries of Multilayer Perceptrons, *Proceedings of the IEEE* 78 (10), 1590-1594, 1990.

21. S. G. Alsing, *The Evaluation of Competing Classifiers*, PhD Dissertation, Air Force Institute of Technology, WPAFB OH, March 2000.

22. Mendenhall, William and others. *Mathematical Statistics with Applications* (Fifth Edition), Duxbury Press, Belmont, 1996.

23. A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis*, McGraw-Hill Companies, New York, 2000.

Vita

1st Lieutenant Ismail Aslan was born in Mersin, Turkiye. He graduated from the Turkish Air Force Academy in 1997 with a Bachelor of Science degree in Industrial Engineering. He was assigned to 1th Main Jet Base as an Intelligence officer.

He was elected to enter the Air Force Institute of Technology at Wright-Patterson AFB, Ohio in 2000 and graduated in 2002 with a Masters Degree in Operations Research

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to an penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
20-03-2002	Master's Thesis	1 Mar 2001 – 20 Mar 2002

TITLE AND SUBTITLE
SELECTING SALIENT FEATURES IN HIGH FEATURE TO EXEMPLAR RATIO CONDITIONS

5a. CONTRACT NUMBER
5b. GRANT NUMBER
5c. PROGRAM ELEMENT NUMBER

AUTHOR(S)
Aslan, Ismail, 1st Lt, TUAF

5d. PROJECT NUMBER
5e. TASK NUMBER
5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 P Street, Building 640
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GOR/ENS/02-02

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSOR/MONITOR'S ACRONYM(S)
11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This thesis research offers two contributions: (1) a new user-friendly graphical user interface that can be used in a feature screening process, (2) a new algorithm for feature screening in a situation where there is a high ratio of feature to exemplars. The first objective is achieved by creating a MATLAB based graphical user interface, which is named as STNGER. STNGER is evaluated on both abstract and the real life problems and provides promising results. For the second objective a new algorithm is suggested. This new algorithm is based on the SNR screening method. By means of this new algorithm, the SNR screening method can determine the salient features in a situation where there is a high ratio of features to exemplars. The performance of the new algorithm suggests that one can apply the SNR screening method to randomly chosen subsets of the data and retain the best features from each subset for subsequent analysis. In this fashion, noisy features are removed while creating new subsets with salient features. The new algorithm is demonstrated on both real-life and the well-defined abstract problems

15. SUBJECT TERMS
Feature selection, Feature Screening, SNR Screening Method, High Ratio features to exemplars conditions

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Kenneth W. Bauer, Jr.
U	U	U	UU	97	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-6565, ext 4328 Kenneth.Bauer @afit.af.edu

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18