DISSERTATION

# FEATURE SELECTION FROM HUGE FEATURE SETS IN THE CONTEXT OF COMPUTER VISION

Submitted by

José Carlos Bins Filho

Department of Computer Sciences

In partial fulfillment of the requirements

for the degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2000

Bell & Howell Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

COLORADO STATE UNIVERSITY

October 27, 2000

WE HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER OUR SUPERVISION BY JOSÉ CARLOS BINS FILHO ENTITLED "FEATURE SELECTION FROM HUGE FEATURE SETS IN THE CONTEXT OF COMPUTER VISION" BE ACCEPTED AS FULFILLING IN PART REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY.

Committee on Graduate Work

Michael Kirby

Charles W Anderson

Bruce A Draper

Adviser

Dale A Smith

Department Head

ABSTRACT OF DISSERTATION

# FEATURE SELECTION FROM HUGE FEATURE SETS IN THE CONTEXT OF COMPUTER VISION

Most learning systems use hand-picked sets of features as input data for their learning algorithms. This is particularly true of computer vision systems, where the number of features that can be computed over an image is, for practical purposes, limitless. Unfortunately, most of these features are irrelevant or redundant to a given task, and no feature selection algorithm to date can handle such large feature sets. Moreover, many standard feature selection algorithms perform poorly when faced with many irrelevant and redundant features. This work addresses the feature selection problem by proposing a three-step algorithm. The first step uses an algorithm based on the well known algorithm called Relief [54] to remove irrelevance; the second step clusters features using K-means to remove redundancy; and the third step is a standard feature selection algorithm. This three-step algorithm is shown to be more effective than standard feature selection algorithms for data with lots of irrelevance and redundancy. In other experiment a data set with 4096 features was reduced to 5% of its original size with very little information loss. In addition, we modify Relief to remove its bias against non-monotonic features and use correlation as the distance measure for K-means.

José Carlos Bins Filho
Department of Computer Sciences
Colorado State University
Fort Collins, Colorado 80523
Fall 2000

## ACKNOWLEDGEMENTS

This work would not be possible without the support and help of many people and institutions. I would particularly like to thank my advisor, Dr. Bruce Draper, for uncountable discussions and revisions of this dissertation; my dissertation committee, Dr. Ross Beveridge, Dr. Charles Anderson and Dr. Michael Kirby for their critiques and suggestions of this work; Dr. Darrell Whitley for his critique and suggestions especially during the proposal of this work; my colleague Kyungim Baek for many discussions and for help running some of the experiments; Wendy Yambor for donation of the Cats and Dogs data set; Robert Duin for donating the Digits data set; my friends at Fort Collins that supported me in my good and bad moments; my parents for their support and love; and the computer science department of Colorado State University, the ADORE project, the Cameron project, and the Pontifícia Universidade Católica do Rio Grande do Sul (Brazil) for financial support.

TABLE OF CONTENTS

LIST OF FIGURES

# LIST OF TABLES

Chapter 1

# INTRODUCTION

The first part of this chapter introduces feature selection in the context of computer vision. It defines feature selection in general and describes computer vision data characteristics and why it is important to do feature selection on this kind of data. It also briefly describes similar problems and their most common solutions. The second part introduces our proposed solution for the problem and gives and small example of how it can be used. Finally, contributions of this work and a brief description of remaining chapters are given.

## 1.1 The problem

Learning agents, in particular artificial learning agents, can be overwhelmed by the shear number of observable features in their environment. Often they must quickly discard many irrelevant and/or redundant features, in order to concentrate their resources on learning a mapping between the remaining, relevant features, and a concept. Thus, the feature selection problem is to find the subset of features of size $n$ that maximizes the system's ability to learn a concept from the selected features.

Formally, feature selection can be defined as the following procedure : given a set of samples $S_n$ ($S_n \subset \Re^n$), where each sample $s_n$ ($s_n \in S_n$) is a vector of $n$ features, and $C_k$ is the set of all possible combinations of $k$ features ($k < n$) from the original set of $n$ features ($|C_k| = \binom{n}{k}$); and given that $E_X(g)$ is the expected value of function $g$ for

the features listed in $X$, $t$ is the target function, $p$ is the function that approximates $t$, and $s_k \in S_k$ ($S_k \subset \Re^k$ where the $k$ dimensions of $\Re^k$ are defined by $c_k$ (or by $c'_k$)). The goal is to select $c_k \in C_k$ such that the expected value of the difference between target and prediction function for $c_k$ is the minimal among all possible combinations of size $k$ ( $E_{c_k}(|t(s_n) - p(s_k)|) = min\ E_{c'_k}(|t(s_n) - p(s_k)|), \quad \forall\ c'_k \in C_k$).

Intuitively, one should never be able to improve performance by discarding information; the more features are used, the more accurate the system should be. In practice, however, the more features there are the more examples (training samples) are needed to induce an accurate classifier. In fact, a smaller set of features can improve the accuracy of the model due to finite sample effects [48]. In addition, more features imply a greater computational cost to acquire and use those features. This burden is especially undesirable if all or most of the class information is associated with a small feature subset.

Feature selection is a difficult problem. Finding the best or approximately best set of features for a given task is usually intractable [57], and many problems related to feature selection have been shown to be NP-hard ([17], [46]). In general, the number of subsets of size $n$ for a set of $m$ features is $\binom{m}{n}$ which is bounded by $(\frac{m}{n})^n > \binom{m}{n} > m^n$. For most practical problems, an optimal solution can only be guaranteed if a monotonic criterion for evaluating features can be found, but this assumption rarely holds in the real-world [56]. Nevertheless, a "good" subset of features can, in most cases, be selected, if the "quality" of the subset is traded for computation time.

We are interested in the problem of feature selection in the context of computer vision. In particular, we are interested in problems with the following characteristics:

- Large number of features: In computer vision the number of possible features is very large (on the order of thousands). At one extreme, every combination of pixels can be considered a feature.

- Irrelevant features: Many of the features computed over an image will have no relevance to the goal. For example, many features computed over the background may be irrelevant to identifying an object.

- Redundant features: Many features used in computer vision are redundant. For example, in color images, the blue, red and green bands and the features computed over them are usually highly correlated.

- Noisy data: The basic data (images) are subject to noise, which will affect all measured features.

- Continuous data: Most of the features computed over an image will generate continuous values.

- Small training samples: Although images are easy to obtain, training samples must be labeled by hand. As a result, the number of samples is normally very small compared to the number of possible features (see above).

- Function Approximation: Most works cited in the literature do feature selection for the purpose of classification. We are interested in optimizing the performance of function approximation (regression) problems. The use of function approximation, instead of classification, makes for a more generic problem.

These characteristics are motivated by the ADORE system (Adaptive Object Recognition [35]), which learns to select vision procedures for object recognition tasks. Since ADORE applies reinforcement learning to samples in the feature space, it has to approximate a Q-function over features. The feature selection techniques here studied can therefore be used to select features for ADORE. Nevertheless, these characteristics are not unique to ADORE. For example Tieu and Viola [101] propose a system were 20 features are selected over a feature set of 45,000 highly selective

features (described in Section 2.3.13). Nevertheless, most object recognition system in computer vision avoid the issue by preselecting a small set of features based on "intuition".

## 1.2 Related Problems

Feature selection is related to four other areas of research: dimensionality reduction [104]; space partitioning [68]; feature extraction and decision trees [87].

Dimensionality reduction is the task of projecting high-dimensional data to a lower dimensional space with a minimal loss of information. The most common procedure used is the Karhunen-Loève Transform (KLT), also known as Principal Components Analysis (PCA). Two types of procedures are normally used to perform PCA: matrix methods ([104] and [53]) and Neural Network methods ([77] and [33]). The principal similarity between dimensionality reduction and feature selection is that both try to remove irrelevant and redundant information. Otherwise the two problems are very different. Dimensionality reduction projects the data into a n-dimensional space, where each dimension is a combination of the original features. As a result, all the original features which are not raw pixels must always be computed. Dimensionality reduction is more commonly used for compression or coding than classification and normally does not utilize abstract features but uses the pixels themselves as features.

Recently, though, dimensionality reduction, in the form of PCA, has become commonly used for object recognition. Murase and Nayar [74] used it as the basis for a classification technique called *appearance matching.* A similar approach for recognition was proposed by Turk and Pentland [104]. One drawback of using PCA for recognition is the implicit assumption that the best features for recognition will be given by the principal eigenvectors. The principal eigenvectors denote the axes of

maximum variation. Although variation implies information, much of this information may be irrelevant to the concept being learned. Another problem with relying on the principal components is that they require precise registration and approximately constant lighting. As a result, when the object is highly specular or has high frequency texture, a small change in pose or illumination can cause dramatic changes on the image brightness and consequently on its position in the eigenspace. For those reasons, this work uses PCA as one technique to generate features, but not as a feature selection technique.

Space partitioning techniques ([68], [103]) try to find regions of the sample or feature space, called hyper-boxes, where the classes are represented with lower ambiguity. The best of these regions are used to create rules for a rule-based classifier. Here, as with dimensionality reduction, the similarity with the feature selection techniques is more in terms of the objective (removing the irrelevant and redundant information) than the method.

In feature extraction, new features are created by combinations and transformations of existing features. In this sense, feature selection can be viewed as a restriction of feature extraction [47]. In a broader sense, even dimensionality reduction and space partitioning can be viewed as restrictions of feature extraction. The projection over a reduced space, used in dimension reduction, is nothing more than a transformation, and the hyper-boxes, used in space partitioning, can be viewed as new features created over existing ones. Nevertheless, feature extraction is such a general problem that no general solutions to it have been proposed.

Inductive decision trees are classifiers that recursively partition the space in order to create classification trees. The simplest of these classifiers, ID3 [85], uses the difference of entropy before and after the partition as the criterion to select the best feature. So, ID3 does feature selection, even if it is disguised as part of the induction process. In [26] this idea is made explicit by discarding the tree generated

by ID3 and only using the subset of features selected as input for a neural net classifier.

## 1.3 The proposed system

### 1.3.1 Introduction

Future object recognition systems in computer vision must be capable of dealing with large numbers of noisy features, most of which are irrelevant and/or redundant. In addition, they should be capable of working with small sets of training samples, since hand-labeled training data is hard to create. The system proposed here tries to fulfill these goals in the context of a function approximation performance task.

The primary motivation for this work comes from the ADORE system [35], which learns object recognition strategies from examples using a Markov process model. As part of its training process, ADORE learns Q-functions that map between the system's states (defined as samples or instances of feature sets) and future expected rewards. In experiments with ADORE, it became clear that the ability to select the proper feature sets prior to training the Q-functions was critical for the success (or failure) of the ADORE system.

Although the motivation came from ADORE, many computer vision recognition systems applications share similar characteristics (e.g. [72],[67]). The number of features that can be computed from an image is, for practical purposes, infinite. For example Tieu and Viola compute 4500 features in [101].

Most feature selection systems reported in the literature work with relatively small number of features (normally less than 100)[76], with the notable exception of automatic text classification systems. Automatic text classification has similarities with computer vision: both have large numbers of features and both have many redundant and irrelevant features. They differ because, normally, the number of

relevant features is still very high. For example, in [60] a feature selection scheme is shown that reduces the number of words in the Reuters1 database [89] from 1675 to 675, a number that is still too high for most object recognition systems in computer vision. In computer vision, it is often hypothesized that the data can be reasonably described by a few features (on the order of tens of features). In addition, automatic text classification systems measure whether or not a word is present, which implies binary features. In most computer vision systems the features are real values.

### 1.3.2 System Description

The goal of our system is to reduce a large set of features (on the order of thousands) to a small subset of features (on the order of tens), without significantly reduce the system's ability to approximate a specific function. Our approach, as shown in Figure 1.1, is a three step process: first the irrelevant features are removed, then the redundant features are removed, and finally a traditional feature selection algorithm is applied to the remaining features. The idea is that each step is a filter that reduces the number of candidate features, until finally only a small subset remains.

Although the purpose of this work is not feature creation, a feature acquisition module was implemented to facilitate the creation of features. Appendix A contains information about the acquisition feature module and the features format.

### 1.3.3 Filtering Features

The first step is to remove irrelevant features using a modified form of the Relief algorithm. Relief uses hits and misses to compute the relevance of features and has been shown to detect relevance even when features interact [23]. Relief will be extensively discused on this work, and more information on it can be found in Section 2.3.6, Chapter 3 and Appendix B. Nevertheless, some modifications were necessary.

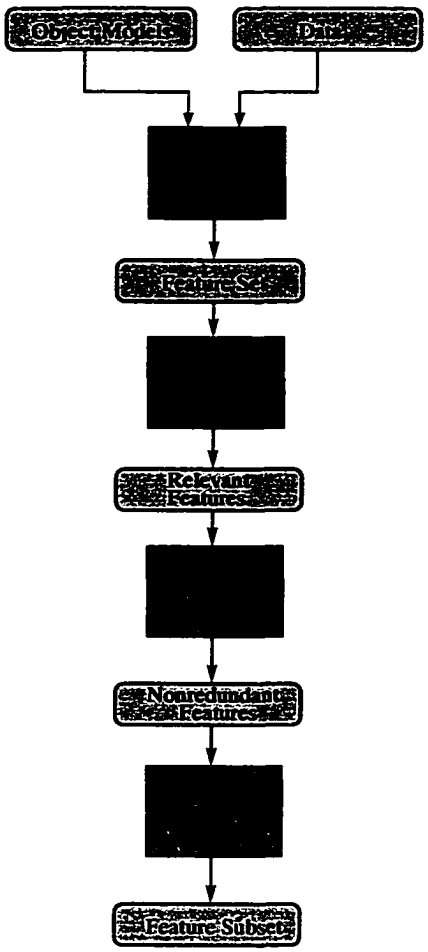



Figure 1.1: High level system architecture.

First, the goal of our system is regression (function approximation) rather than classification, therefore, the way "hits" and "misses" are computed in Relief had to be redefined. Second, Relief is used to remove irrelevant features, rather than to select the best feature set. Third, the Relief algorithm was found to have a bias against non-monotonic features. This bias was reduced by modifying the algorithm to identify non-monotonic features and consider each part of their distribution separately. Our modification considers one-peaked features only. Features with more than one peak/valley are not common in computer vision and their identification much more

complex, so their identification was not attempted at this time. Section 3.4 explain this bias and how it was reduced.

The second step is a redundancy filter. Relief is good at detecting relevance but does not address the problem of redundancy. So, a filter process is applied to remove the redundant features. This filter uses the K-means algorithm [66] to cluster features according to their correlation. By clustering correlated features, each cluster is limited to contribute at most one feature to the final feature subset. This is an unusual application of clustering, and in particular K-means, in that: features are clustered, instead of samples; and correlation is used as the distance measure.

The order of the previous filters is important. Relief complexity depends of the number of samples while the complexity of clustering depends on the number of features. In this work, it is assumed that the number of features is bigger than the number of samples, so the more features are removed the faster the clustering will work. The opposite is not true.

The third and final filter is a traditional feature selection algorithm applied to the remaining features. Because the so-called "wrapper approach" (see Chapter 2) is too expensive for the number of features still remaining, the well-known Sequential Forward Selection (SFFS) and Floating Sequential Backward Selection (SFBS) algorithms [83] were used. Pudil et. al. [83] show that SFFS and SFBS obtain comparable results with Branch and Bound, which is able to find an optimal set, but they are much faster.

### 1.3.4 Example

Consider an object recognition task. Given a region of interest (ROI) and a model of a target object, the goal is to estimate how well the ROI matches the true object position. In particular we want to learn the following target function:

$$f = \frac{|H \bigcap T|}{|H \bigcup T|}$$

where $|X|$ is the cardinality of set X, H is the set of pixels in the mask (Hypothesis) and T is the set of pixels in the real position of the object (Truth). Function $f$ uses only two variables, but one of them (T) is unknown at execution time[1]. So, what is sought is a function $p$ that predicts $f$ based on the features of the ROI. For the sake of illustration, assume that out task is to recognize a style of houses. A possible set of features would be:

1. Area of the mask
2. Sum of squared distances between pixels under the mask and their mean, clipped to a maximum of 1600.
3. Ratio of mask edge to mask area.
4. Number of positive $2^{nd}$ derivatives in dimension x under the mask.
5. Average intensity difference between each pixel under mask and the average of a 5x5 window surrounding the pixel.
6. Minimum overall value under the mask.

Those features where some of the features used in the ADORE system [35]. Each of these features was chosen in a attempt to differentiate between characteristics of one style of houses and characteristics of other house styles or other objects like trees or roads. For example, Feature 3 could be used to differentiate long objects from more compact objects, even if they have the same area, and Feature 2 gives a

[1]Otherwise the function could be computed directly.

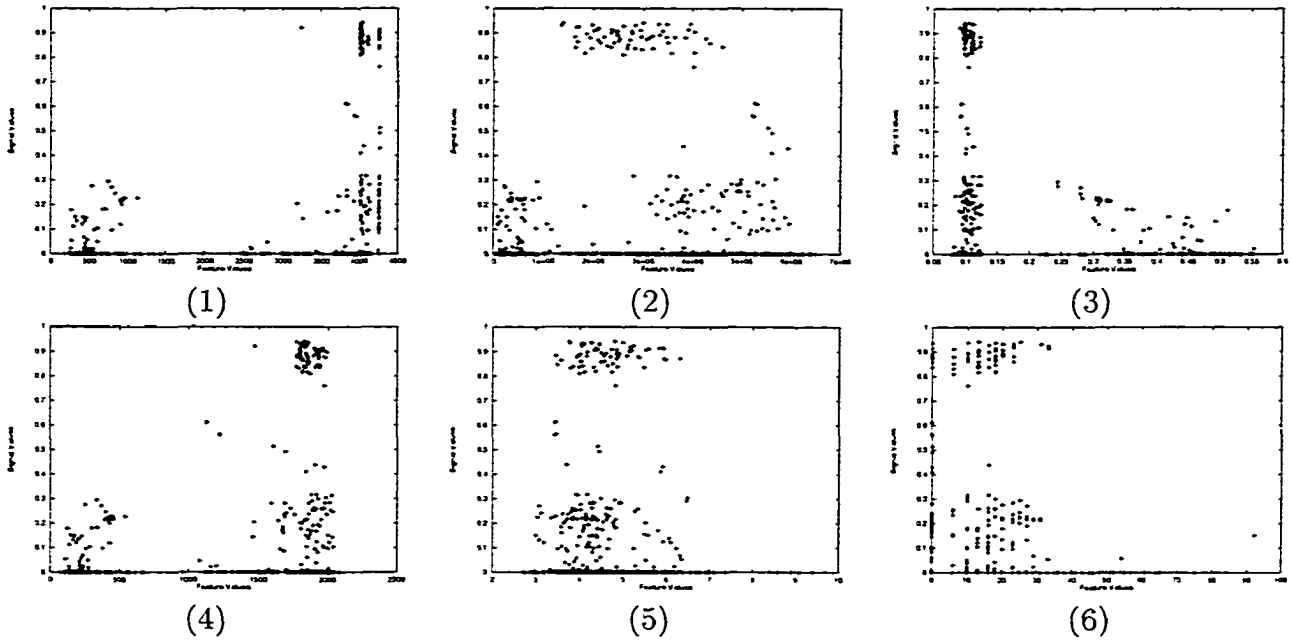



Figure 1.2: Graph of the features. The abscissa correspond to the feature values and the ordinate corresponds to the target function.

measure of the smoothness of the surfaces. Unfortunately, many times our intuition can be misleading and not only some seemly good characteristics are not useful but some unpredictable ones are. Figure 1.2 show a graph of these features against the learning signal used in ADORE.

The standard procedure to select the features would be to apply one of many feature selection algorithms to those features and use the final set to learn the prediction function. This works well if the number of features is small, where small might be as little as 30 for Branch and Bound or as much as 100 for SFBS. Unfortunately, we want a system that deals with thousands of features. The proposed system reduces the number of features until a traditional feature selection algorithm can be applied.

For this example, we start with the original set of features {1,2,3,4,5,6}. The first step is to remove irrelevant features. Features 5 and 6 do not offer much information. This can be intuited by visually inspecting plots 5 and 6 of Figure 1.2. As can be

seen, the high and low values of the learning signal are equally distributed over the feature axis. Despite that, they could have some information when combined with other features. That is not the case. Their low information content can be attested by their low relevance score for the Relief algorithm compared with the score achieved by Feature 1. As a result, features 5 and 6 are irrelevant for the task and are removed. The feature set is now {1,2,3,4}.

The second step removes redundancy by detecting feature clusters. In our example, the k-means algorithm creates 3 clusters: {1,4}, {2}, and {3}. That happens because Features 1 and 4 are highly correlated, see plots 1 and 4 at Figure 1.2. In fact, after normalization, they correlate with each other with a score of 0.9904. Now, each cluster contributes with one feature. Feature 1 is used as representative of Cluster 1 because it has a higher Relief score. We then have a new feature set with {1,2,3}.

A traditional feature selection algorithm, in this case Sequential Floating Backwards Selection (SFBS), is applied to this feature set to select "the best" subset. Assume the feature selection algorithm selects features 1 and 2, those will form the final feature subset and will be the ones used to train the prediction function. This feature selection algorithm mostly differs from Relief in that it computes its evaluation function over various subsets of the feature set. This is computationally much more expensive than Relief, but has the advantage of more accurately evaluating interrelations between features.

### 1.3.5 Contributions

This work differs from other works cited in the literature in three significant ways: the application domain, the irrelevance filter and the clustering of features. The application domain is computer vision, which implies a potentially very large set of candidate features. The number of features itself makes the task hard, but

more important, no other work uses so many redundant and irrelevant features in the original set. The second difference, the irrelevancy and redundancy filters, are a consequence of the first. To handle so many features, the selection process needs to be very fast, but no technique described in the literature has been shown to achieve good results for this number and quality of features. So, the irrelevance filter and redundancy filter are introduced. The techniques used in these steps, Relief and k-means, are not, by themselves, new but the role they play here is. Moreover, both Relief and k-means were modified to conform to the application needs. Relief was modified to remove its bias against non-monotonic functions. K-means was modified to use correlation as its distance measure and to cluster features instead of samples. Many algorithms (eg. Induction trees [85], Space Partitioning [68] and EUBAFES [93]) cluster the data in order to find the best features. Other algorithms (eg. Max-Min [6], GSFS and GSBS [55]) consider the dependency between features, as a way to avoid redundancy for small feature sets. However, no other work explicitly divides the feature space into clusters.

## 1.4 Overview of Remaining Chapters

Chapter 2 reviews the feature selection literature from both the statistics and artificial intelligence communities. Chapter 3 describes the relevance filter plus tests and modification performed in the Relief algorithm, as well as the Relief bias against non-monotonic features. Chapter 4 describes the redundancy filter plus modifications and tests performed for the k-means algorithm. Chapter 5 describes the feature selection module. It also shows tests that indicates that the system is necessary if the assumptions about the data are true and does not compromise the result if those assumptions are false. Appendix A describes the feature acquisition module and the feature file format. Appendix B describes most of the algorithms cited in

this work. Appendix C proves a theorem about quadratic functions used to create the synthetic data. Appendix D contains proofs of some correlation theorems used in the redundancy filter.

Chapter 2

# LITERATURE REVIEW

## 2.1 Introduction

The literature covering feature selection is extensive, both in time as in scope. Feature selection is important in many fields, like document classification, data mining, object recognition, biometrics, remote sensing and computer vision. In other words, where the number of features or attributes that can be computed is larger than the ones necessary to the task, but it is not known before hand which of them are the right ones.

Roughly, the literature on feature selection can be extracted from two main areas: statistics and artificial intelligence. Both areas have done extensive studies of the problem, though their approaches, in most cases, are very different. The statistical approach, in general, assumes an a-priori (usually Gaussian) distribution and uses a training set to estimate the parameters of the underlying distribution. Because these methods use the population distribution to estimate model parameters, they are also called parametric methods. In addition, the three components of their algorithms are normally independent modules. See 2.3 for a description of the usual components of a feature selection algorithms. For example, many statistical algorithms can, and many times must, change their evaluation function according to the application. The artificial intelligence approach, like the statistical one, tries to

approximate the data distribution, but in general, it does not assume a model for the distribution. For this reason AI techniques are usually classified as non-parametric statistics. Moreover, their components are much more related and most times it is difficult to separate them. For example, in a node pruning algorithm the evaluation function and the performance function are often the same. Despite these differences in approach, sometimes the solutions found are very close or the same. One of the better examples of independent proposals of the same solution is the proposal of K-means by MacQueen in 1967 [66] and Vector Quantization[1] by Kohonen in 1989 [59]. Although these use different terminology, they are essentially the same algorithm. "MacQueen's on-line k-means algorithm is essentially the same as Kohonen's learning law except that the learning rate is the reciprocal of the number of cases that have been assigned to the winning cluster" [109].

## 2.2 Early Studies

Feature selection can be related back to discoveries on probabilities by Bayes (1702-1761). From a purely Bayesian point of view, the feature selection problem can be easily solved given the joint class and feature distribution. The problem is that this joint distribution is normally not known and so must be approximated. Another important historical reference is Galton. Galton was the first to use correlation between variables, also known as *the regression problem*, as a statistical tool [79]. Nevertheless, in feature selection the number of subsets to be tested is normally so big that a search process is necessary. The first attempt to deal with this combinatorial explosion came in this century when in 1962 Lewis [64] proposed that if a set of

[1]There are three types of Kohonen networks, Vector Quantization (VQ), Self-Organizing Maps (SOM) and Learning Vector Quantization (LVQ). Only VQs are similar to K-means. SOMs provide a topological mapping from the input space to the clusters; and LVQs assigns each codebook vector to one of the target classes.

measurements (features) are statistically independent, then the $n$ best individual features, i.e., the features with highest score for the evaluation function, would form the best subset of size $n$. Cover in 1974 [27] refuted that when he showed that even if features are statistically independent, the two individually best features do not always form the best pair. A stronger result was presented by Toussaint [102], who showed that the best individual feature does not need to be a part of the best pair. This result contributed to the evolution of feature selection algorithms.

## 2.3 Algorithms

Feature selection algorithms are typically composed of three components [3]:

- Search Algorithm: This searches the space of feature subsets.
- Evaluation function: (a.k.a. criterion function), inputs a feature subset and outputs a numeric evaluation of the subset. This is the function that the algorithm tries to maximize.
- Performance function: This is the task to which the resulting subset is applied. In most feature selection studies this task is classification.

As discussed before, statistical and artificial intelligence approaches differ in their assumptions about the a-priori distribution of the data. Nevertheless, feature selection algorithms can be classified also by their use of the evaluation function. In this case, they can be classified as "Filters" or "Wrappers". Filters pose feature selection as a separate process from classification, and so evaluation and performance functions are different entities. Wrappers, proposed in 1995 by Kohavi and John [57], use the performance function (task) as the evaluation function.

The idea behind the wrapper model is that maximizing classification accuracy and identifying which features are relevant to a classifier are two different tasks. As

shown by Kohavi and John [57], the optimal feature set depends on the specific biases and heuristics of the learning algorithm. Therefore, the feature subset performance depends on the classifier being used and should be computed using the characteristics of the classifier. Hence the evaluation and performance functions are the same. For example, Kohavi and John uses standard search algorithms (hill-climbing, best-first, etc) where the evaluation function is a classifier algorithm (decision trees, Relief, etc.).

Filters are generally much less resource intensive than wrappers, because wrappers have to build and train a classifier for each feature set evaluation. On the other hand, filters may result in poor performance if the evaluation criterion does not match the classifier well, even if the subset of features selected is optimal for the criterion function.

Below a brief description of some of the most important techniques of feature selection is given. Many algorithms are independent of the task signal, for example SFFS, while others use the task signal as part of their selection process, for example Relief. Most of the algorithms are typically used as filters, although they usually can be used under the wrapper approach as well. Because of the burden to build and construct classifiers, the wrapper model has been used most often with classifiers that can be induced quickly. Bala, et al. [9] [10], and Vafaie and Iman [106] use decision trees as the fitness function for a GA, and Punch, et al. [84] uses a modified version of K-nearest neighbors.

### 2.3.1 SBS and SFS

Sequential Forward Selection (SFS), proposed by Marill and Green [70] in 1963, is the first algorithm proposed for feature selection. It performs a heuristic-guided Depth-First search on feature space, starting with the empty set. On each iteration,

all features not yet included in the subset of selected features are individually incorporated in the subset and a criterion value computed. The feature that yields the best value is then included in the new subset (see Appendix B for algorithm).

In 1971, Whitney proposed Sequential Backward Selection (SBS) [110]. This algorithm is the top-down equivalent of SFS. It begins with a complete set of features and instead of including a new feature on every iteration it removes one. Although SFS and SBS are very similar, that similarity is misleading. Aha and Bankert [2] argue that SBS outperforms SFS because SBS evaluates the contribution of a given feature in the context of all the other features, while SFS evaluates the contribution of a feature only in the limited context of the previously selected features. On the other hand, SBS is computationally more demanding because it evaluates subsets of much higher dimensionality.

Nevertheless, both algorithms suffer from the so called nesting problem: once a feature is added or removed, this action can never be reversed. For SFS, this means that the best individual feature will always be present in the final set. As proved by Toussaint, this may lead to suboptimal subsets. The opposite is true for SBS. Here, the worst individual feature will never be present in the final set. This is important because, as Toussaint showed [102], not only can the best individual feature be missing in the best feature subset, but the best feature subset may require only the worst individual features.

In 1978, Kittler [55] proposed generalizations for SBS and SFS called GSFS(r) and GSBS(r), where instead of including (or excluding) one feature at each step a subset of $r$ features is included (or excluded). Those algorithms yield better results. However, they still suffer from the nesting problem and they more expensive, since at each step $\binom{n}{r}$ features must be inspected, where $n$ is the total number of features to form subsets from and $r$ is the size of the subsets.

Another variation on SFS and SBS is beam search (BS). This algorithm keeps various subsets of features in an ordered queue and performs the operation (inclusion or exclusion) for the subset that yields the best subset criterion (first in the queue). If there is no limit on the size of the queue, then BS is an exhaustive search. In the trivial case where the queue has size one, beam search is equivalent to SFS (or SBS). By keeping more than one feature subset in the queue, BS improves its chance of finding the optimal solution. Doak showed that the beam search algorithm frequently outperforms SFS [34].

### 2.3.2 PTA(l,r)

In the Plus $l$ - Take Away $r$ algorithm (PTA(l,r)), proposed by Stearns in 1976 [98], at each step $l$ features are individually included and $r$ features individually removed form the selected subset. This solves the nesting problem, but does not account for correlation between features. This was minimized by the proposal of GPTA(l,r) by Kittler [55]. Here, as with GSFS(r) and GSBS(r), features are included in subsets, which allows the correlation between features to influence the selection process. This is true for correlations of size $l$ or $r$. If the correlation is only evident in bigger sets, then it will not be captured.

### 2.3.3 Max-Min

The Max-Min algorithm, proposed by Backer [6] in 1977, assumed that the best feature to include is the one that gives the maximum increase in the criterion function among the features that have the minimum pairwise correlation with the features already in the subset (see Appendix B for the algorithm). Kittler [55] points out that this algorithm is based on very little information, meaning the only available information is the individual and pairwise effectiveness of the features. That is possibly the cause of its poor performance. For example, Jain and Zongker [47]

show that the Max-Min algorithm is very fast, but performs very poorly, even in comparison with SFS and SBS.

### 2.3.4 BB

In 1977, Narenda and Fukunaga [75] applied the Branch and Bound (BB) algorithm to feature selection. The algorithm is very efficient because it avoids exhaustive enumeration by rejecting suboptimal subsets without direct evaluation. The algorithm creates a search tree and uses a clever subset enumeration scheme with nodes at the same level having variable number of terminal nodes and being tested in the order of their number of terminal nodes, allowing suboptimal sequences to be rejected early on in the process (see Appendix B for the algorithm). This algorithm generates optimal results for the sample set used, but requires a monotonic evaluation criterion [2], which is very rarely possible. Note that optimal here does not mean that the resultant classifier will have a 0% error rate. It only means that the subset will have the best possible performance for a given subset size. Hamamoto et al. [42] show that BB finds an optimal or almost optimal result even if the criterion is non-monotonic, but if the cardinality of the desired subset is small the time consumed is very high (worse than exhaustive search). As a result, its use for high dimensional spaces still remains prohibitive. Pudil et al. [83] suggest 30 measurements (features) as the limit for the applicability of the method.

### 2.3.5 SFFS and SFBS

The Sequential Floating Forward Selection (SFFS) and Sequential Floating Backward Selection (SFBS), proposed in 1994 by Pudil et al. [83], are solutions to the nesting problems present in SFS and SBS. The idea behind SFBS and SFFS

[2]Normally an increasing monotonic criteria. For an increasing monotonic criteria any subset of size K has a lower value for the criteria than a subset of size k+1.

is to allow the values of $l$ and $r$, as in PTA(l,r), to vary (float) between stages of the algorithm. In other words the number of features included and/or removed at each stage is not fixed. Although both, SFFS and SFBS, include and remove features at each stage, they differ in the order of inclusion and removal and in their initial set (see Appendix B for the algorithms). The advantage here is that $l$ and $r$ do not need to be fixed compared to PTA(l,r) and GPTA(l,r) which are capable of solving the nesting problem but where there is no theoretical way to predict the values of $l$ and $r$ to achieve the best feature subset. Moreover, the authors claim that SFFS and SFBS are much faster while obtaining results comparable with BB [83].

### 2.3.6 Relief

Relief, proposed by Kira and Rendell [54] in 1992, is a feature weight based algorithm inspired by instance based learning. It computes a global measure of relevance for features, called score, where relevance is based on the distance between each sample and its closest *hit* and *miss*. A hit is a close sample, in the sample space, of the same class as the sample being analyzed and a miss is a close sample of a different class. Relief selects those features whose scores are above a given threshold. Thus, there is no need to set the size of the feature subset before hand, which in most cases is only a guess. Instead, set size is replaced by a relevance threshold which can be computed as a function of the number of samples and the maximum number of false negatives allowed. Relief is able to handle feature interaction but performs poorly in the presence of redundant features. Kononenko [61] extends Relief to handle noisy, incomplete and multi-class data sets. The basic idea is that more than one hit or miss is computed and averaged. For example, for multi-class data sets one miss for each class is found and the distance between the misses and the sample is computed by weight-averaging the misses. The weight used is the a priori probability of each class.

### 2.3.7 Focus

Focus [4], proposed by Almuallin and Dietterich in 1991, is an exhaustive search procedure that examines all possible binary feature subsets of a given size before going on to the next size. It searches for the minimum-size feature subset sufficient to maintain consistency (no conflict) over the training data, where two samples are in conflict if they have the same values for a subset of features but disagree in the class they represent. Unfortunately, it is very sensitive to noise and, being an exhaustive search algorithm, can be computationally prohibitive. Focus2 (1992, [5]) reduces the combinatorial explosion by subdividing the feature space into mutually exclusive subspaces where a subset of features is present and another subset of features is not. With this subdivision, whole subspaces can be discarded once one feature is discovered to be necessary (or unnecessary). Focus2 is much faster than Focus, but it is still based in logical consistency, and therefore performs poorly in the presence of noise.

### 2.3.8 Neural Nets

Artificial Neural Networks (ANN) are parallel computational models comprised of densely interconnected adaptive processing units. An important feature of ANNs is their adaptive nature, where learning by example replaces programming in solving problems [45]. Each of the processing units receives a vector of numbers as input (possibly the output of other units) and produces a single value as output (possibly the input for other units). The number of input nodes used in a neural net reflects the number of features used. The more input nodes, the more computational time and samples are required for training.

Feature selection is done, in the context of neural nets, by pruning input nodes. Pruning input nodes, which can be compared with the backward approach, SBS, in statistical approach, is based on the use of saliency measures. A salience measure

ranks inputs according to the "sensitivity of the network's output". First the net is trained and the saliency of each input node computed. In sequence, the nodes with saliencies under a given threshold are pruned and the process repeats. Examples of such techniques are given by Mao, et. al. [69] and Setiono and Liu [95]. Several different saliency measures are reported by Belue and Bauer [15]. Castellano, et. al. [24] propose a pruning method for adjusting weights in the remaining nodes, after pruning, in such a way that the network performance does not worsen over the entire training set, avoiding additional training. One problem with Node Pruning is that it assumes independent inputs, which cannot be guaranteed for many applications, including computer vision. A second problem is that training nets with several thousand input features is, in many cases, not practical, particularly in computer vision where the number of training samples is small compared with the number of features (inputs). Moreover, this training must be performed several times. Every time it eliminates a group of features the net must be retrained.

### 2.3.9 Genetic Algorithms

Genetic Algorithms (GA) provide an approach to learning based on biological evolution. Hypotheses are described by bit strings (refered to as chromosomes) whose interpretation depends on the application. The search for an appropriate hypothesis begins with a random population of initial hypotheses. Members of the current set of hypotheses give rise to the next population by means of operations such as random mutation and cross-over. At each step, the current population is evaluated relative to a given measure of fitness, with the most fit hypotheses selected probabilistically as seeds for the next generation [73]. The measure of fitness, or fitness function, as it is called in the GA terminology, performs the same role as the evaluation function in feature selection terminology.

When using genetic algorithms for feature selection, a feature set is represented by the chromosome of an individual. Normally, each feature is represented by a binary digit (gene) which indicates the presence or absence of the feature in the set. The fitness function can be of the same kind as used in statistical algorithms such as SBS. In this case, the GA method can be used to search the space of features without any change in the algorithm. The first attempt to use genetic algorithms for feature selection was done by Siedlecki and Sklansky in 1988 [96]. Vafaie and De Yong [105] compare a genetic algorithm approach with the SBS method and show that the GA algorithm outperforms SBS. Bala, et al., [8] use a C4.5 tree induction algorithm as the fitness function for a GA. Guerra-Salcedo and Whitley [39] use features selected by a genetic algorithm to construct ensembles of table based classifiers.

### 2.3.10 Decision trees

Tree induction is a classification technique developed independently by Breiman[19] and Quinlan [85]. It classifies samples by recursively dividing the feature space according to selected feature values until it creates regions of feature space that contain predominantly one class of training samples. Novel samples in this region are then assigned the label of the predominant class. Most classifier methods that build decision trees such as ID3 [85], C4.5 [86] and CART [19] are variations of this approach which implement greedy searches in the space of possible decision trees.

Although induction trees are primarily a classification technique, they also select a subset of features to be used for classification. This subset can be used to train another classifier or function approximator, as done by Cherkauer and Shavlik [26]. As a result, induction trees can also be viewed as feature selection algorithms. In this context, Almuallin and Diettrich point out that the performance of conventional inductive learning algorithms such as ID3 and Fringe [78] is seriously reduced by the presence of irrelevant features. Kohavi and John [57] showed that this is also true for C4.5.

### 2.3.11 EUBAFES

The Euclidean BAsed FEature Selection algorithm (EUBAFES) [93], see Appendix B for the algorithm. The method applies a gradient descent approach to optimize a criterion function with respect to the feature weights. This criterion function is based on the distance between samples in the same class or different classes. EUBAFES uses an almost identical feature measure to that of Relief and both reinforce similarities between the same class and degrade similarities between different classes. However, they differ in the approach and in the optimization technique used. While EUBAFES divides the features into two classes, relevant features and irrelevant features, Relief uses the difference between classes to compute the relevance score of each feature. Another difference is that EUBAFES uses a gradient descent which may be much slower.

### 2.3.12 LVF

The Las Vegas Filter Algorithm (LVF), proposed by Liu and Setiono [65], implements a probabilistic approach to feature selection by randomly generating subsets of features. For each subset, if the number of features is less than the current best set, and if a measure called the inconsistency rate (defined later) is below the current best, then the new subset is saved as the best set. This sampling process is done for a specified number of trials.

The inconsistency rate is the key to LVF. It specifies to what extent the new subset is acceptable. The inconsistency rate is the sum of all inconsistency counts divided by the total number of samples [65], where : two samples are considered inconsistent if they match all but their class labels[3]; and the inconsistency count is the number of all the matching samples minus the largest number of matching

[3]This is similar to the conflict criterion on Focus2.

samples among the different classes. As an example: say there are $n$ matching samples and from those $c_1$ belong to Class 1, $c_2$ to Class 2 and $c_3$ to Class 3. If $c_3$ is the largest of them, then the inconsistency count is $(n - c_3)$ and the inconsistency rate is the sum of the inconsistency count for all groups of matching samples. A version of LVF called the Las Vegas Incremental Algorithm (LVI) generates sets in decreasing order of cardinality.

### 2.3.13 Boosting

Recently Tieu and Viola [101] proposed the use of boosting applied to feature selection. In their work they compute 45,000 highly selective features and reduced this set to 20 features. These highly selective features measure how first order features such as edges and colors are related. Tieu and Viola's work focus on image retrieval. By selecting a small number of features, large sets of images can be represented in an efficient way. Although most of the underlying first order features still must be computed for each image, since the search is done over the selected features it can be done very fast. The subset of features is selected by training weak learners for each feature, selecting the best feature for the task, boosting the dataset and repeating the process. The final feature subset is formed by the union of the features selected at each step of the process. This process is interactive, where the user is able to select images and assign them as positive or negative samples. Their results suggest that using boosting the number of features necessary for retrieval is better than previous methods. We think a similar boosting process can be applied on our proposed system, but because of time constraints the inclusion of this step was deferred as future work.

## 2.4 Evaluation Functions

The algorithms discussed above can be viewed as different approaches to optimize an evaluation function. Evaluation functions estimate the relevance of a feature

or set of features and/or the accuracy that the feature set yields during classification or regression. Filters use evaluation functions that focus on relevancy of features. Those functions depend heavily on the intrinsic properties of the data. There is the implicit assumption that the more relevant the features are, the better will be the accuracy achieved. It is important to note that this assumption is not true, Kohavi and John [57] have shown that these two objectives (relevance and classification accuracy) are not equivalent, but in most cases it is a good enough approximation. Nevertheless, this means that no single evaluation function will be best for all applications. Wrappers use evaluation functions that focus on accuracy. These functions are less dependent of the data structure, but they are computationally costly. These functions normally use the performance function (classifiers) as the evaluation function.

The objective of this section is to introduce evaluation functions that focus on relevancy. The main reasons for this limitation is that they are simpler, compared with classifiers, and have a long history of use for feature selection.

Doak divides filter evaluation functions into four classes [34]:

- Probabilistic distance measures
- Probabilistic dependence measures
- Entropy measures
- Interclass distance measures

Those measures will be further described in the next sections. Briefly, probabilistic distances measures compute the degree to which the probability of a sample, feature set, belonging to a class can be used to distinguish between the classes. Guorong, et. al. [40] use one of these functions (Bhattacharyya) to implement a feature selection algorithm that selects optimal feature sets over normal distributions

when either term of the Bhattacharyya distance is dominant. Probabilistic dependence measures are very close to probabilistic distance measures. They compute the dependence between sets of features and classes, or the ability of predicting a feature vector given the class. Sections 2.4.1 and 2.4.2 show the formulas and further explain these measures. Most probabilistic dependence measures are variations of probabilistic distance measures. Entropy measures use the *a posteriori* probability to compute the dependence between features and classes. They are normally used in decision trees. Almuallin and Dietterich [4] use an entropy measure to create a Mutual Information Greedy algorithm (MIG) which they compare to Focus. Interclass distance measures compute the distance between classes. They are, perhaps, the most used measures in the statistical approach algorithms. For example, the Mahalanobis distance, defined latter, is used in BB [75], SBS, SFS, SFFS, SFBS, PTA and Max-Min [47].

**Notation**

The formulas described on the rest of this chapter use the following notation:

| | | |
|---|---|---|
| Set of features: | $F = \{f_1, f_2, ..., f_p\}$ | $p$: Number of features |
| Subset of features: | $F' = \{f'_1, f'_2, ..., f'_d\}$ | $d$: Number of selected features |
| Sample: | $X = \{x_1, x_2, ..., x_p\}$ | |
| Set of Samples: | $S = \{X_1, X_2, ..., X_n\}$ | $n$: Number of samples |
| Set of Classes: | $C = \{C_1, C_2, ..., C_q\}$ | $q$: Number of classes |
| Probability: | P(.) | |

### 2.4.1 Probabilistic Distance Measures

Probabilistic distance measures use evaluation functions of the form:

$$J(F') = \int f(P(F'|C_i), P(C_i))\ dF'$$

In other words, $J$ is a function of the conditional probability of feature vector $F'$ given class $C_i$ and the probability of class $C_i$. To classify as a probabilistic distance

measure, J must satisfy the following conditions [34]: $J \geq 0$, $J = 0$ when the probability of the feature vector is the same for all classes, J is maximum ($J <= 1$) when the probability of the feature vector is well separated between the classes. Table 2.1 shows examples of these functions for the two-class case. For multi-class cases, the approach may be extended by considering a weighted function of the distances between all pairs of classes. The weight is given by the product of the two classes' a priori probabilities.

| **Probabilistic Distance Measures** | |
|---|---|
| **Name** | **Formula** |
| Bhattacharyya | $J(F') = -\log \int \sqrt{P(F'\|C_1)P(F'\|C_2)}\ dF'$ |
| Matusita | $J(F') = \sqrt{\int \sqrt{P(F'\|C_1)} - \sqrt{P(F'\|C_2)}\ dF'}$ |
| Kullback-Liebler | $J(F') = \int [P(F'\|C_1) - P(F'\|C_2)] \log \frac{P(F'\|C_1)}{P(F'\|C_2)}\ dF'$ |
| Kolmogorov | $J(F') = \int \lvert P(F'\|C_1) - P(F'\|C_2)\rvert\ dF'$ |
| Lissak-Fu | $J(F')_\alpha = \int \lvert P(F'\|C_1) - P(F'\|C_2)\rvert^\alpha\ dF'$ |

Table 2.1: Examples of Probabilistic Distance Measures (adapted from [34] and [11]).

### 2.4.2 Probabilistic Dependence Measures

Probabilistic dependence measures use evaluation functions of the form:

$$J(F') = \int f(P(F'|C_i), P(F'))\ dF'$$

Dependence measures, also known as association measures or correlation measures, differ from distance measures in that they use the probability of a feature vector P(F') instead of the probability of a class $P(C_i)$. The underlying premise is that if $P(F'|C_i)$ and $P(F')$ are equal the feature vector does not differentiate between

the classes. In other words, the feature vector and the classes are independent. On the other hand, if they are different, then some information is added when the class is given, or the feature vector and the class have some interdependency. With that, feature vectors can be ordered under the assumption that the bigger the difference between the two probabilities, the greater the dependence and consequently the better the features predict the class. Examples of probabilistic dependence measures are given in Table 2.2. The classical correlation

$$J(F') = \frac{\sum_{x=1}^{n}[(f(x) - \overline{f(x)})\ (g(x) - \overline{g(x)})]}{\sigma_f\ \sigma_g}$$

is a measure of dependence between two numerical variables. It is often used to correlate two features, for example in the Max-Min algorithm [6]. Unfortunately, features and classes cannot be correlated directly, since class membership is a binary variable and therefore its mean has no meaning. One alternative is to compare features to class prototypes.

### 2.4.3 Entropy Measures

Entropy measures differ from probabilistic measures in that they use the *a posteriori* probability $P(C_i|F')$, instead of the *a priori* probability $P(F'|C_i)$. In the classical entropy measure, the greater the gain the bigger the dependence between the class and the feature vector, where the gain for class $i$ is expressed as:

$$Gain_i(F') = [-P(C_i)\ log_2 P(C_i)]\ -\ [-P(C_i|F')\ log_2 P(C_i|F')],$$

[4]Correlation coefficient between a word and a class of texts, where N is the number of texts, $N_{r+}$ ($N_{n+}$) is the number of relevant (non-relevant) texts in which the word occurs, and $N_{r-}$ ($N_{n-}$) is the number of relevant (non-relevant) texts in which the word does not occurs (from [76]).

| Probabilistic Dependence Measures | |
|---|---|
| **Name** | **Formula** |
| Bhattacharyya | $J(F') = \sum_{i=1}^{q} -\log[P(C_i) \int \sqrt{P(F'\|C_i)P(F')}\, dF']$ |
| Matusita | $J(F') = \sum_{i=1}^{q} [P(C_i) \sqrt{\int \sqrt{P(F'\|C_i)} - \sqrt{P(F')}\, dF'}]$ |
| Joshi | $J(F') = \sum_{i=1}^{q} [P(C_i) \int [P(F'\|C_i) - P(F')] \log \frac{P(F'\|C_i)}{P(F')}\, dF']$ |
| Kolmogorov | $J(F') = \frac{1}{2} \sum_{i=1}^{q} [P(C_i) \int \|P(F'\|C_I) - P(F')\|\, dF']$ |
| Patrick-Fisher | $J(F') = \sum_{i=1}^{q} P(C_i) \int \sqrt{P(F'\|C_i) - P(F')}\, dF'$ |
| Word Correlation[4] | $J(F') = \frac{(N_{r+}N_{n+} - (N_{r-}N_{n-})\sqrt{N}}{\sqrt{(N_{r+} + N_{r-})(N_{n+} + N_{n-})(N_{r+} + N_{n+})(N_{r-}N_{n-})}}$ |

Table 2.2: Examples of Probabilistic Dependence Measures (adapted from [34], [11] and [76]). $q$ is the number of classes

and the total gain is

$$Gain(F') = -\sum_{i=1}^{q} P(C_i)\ log_2 P(C_i) + \sum_{i=1}^{q} P(C_i|F')\ log_2 P(C_i|F').$$

where q is the number of classes. Because $\sum_{i=1}^{q} P(C_i)\ log_2 P(C_i)$ does not depend on $F'$, the gain is normally computed as

$$Gain(F') = -\sum_{i=1}^{q} P(C_i|F')\ log_2 P(C_i|F')$$

Some authors prefer to use the average gain computed using the weighted sum

$$AverageGain(F') = -\sum_{i=1}^{q} P(C_i)\ P(C_i|F')\ log_2 P(C_i|F')$$

Examples of entropy measures are given in Table 2.3.

| Entropy Measures | |
|---|---|
| **Name** | **Formula** |
| Shannon | $J(F') = -\sum_{i=1}^{q} P(C_i\|F') \log P(C_i\|F')$ |
| Quadratic | $J(F') = \sum_{i=1}^{q} P(C_i\|F')(1 - P(C_i\|F'))$ |
| Daroczy | $J(F')_\alpha = \frac{1}{2^{1-\alpha}-1}(\sum_{i=1}^{q} P(C_i\|F')^\alpha - 1)$ |
| Renyi | $J(F')_\alpha = \frac{1}{1-\alpha} \log \sum_{i=1}^{q} P(C_i\|F')^\alpha$ |
| Gain Ratio | $J(F') = \frac{-\sum_{i=1}^{q} P(C_i\|F') \log P(C_i\|F')}{-\sum_{j=1}^{v} \frac{u_j}{\sum_{l=1}^{v} u_l} \log \frac{u_j}{\sum_{l=1}^{v} u_l}}$ |
| Normalized Gain | $J(F') = \frac{-\sum_{i=1}^{q} P(C_i\|F') \log P(C_i\|F')}{\log q}$ |

Table 2.3: Examples of Entropy Measures (adapted from [34], [11] and [50]). $q$ is the number of classes, $v$ is the number of distinct values for $F'$, and $u_j$ is the number of samples for which $F'$ has value $a_j$.

### 2.4.4 Interclass Distance Measures

The interclass distance measures attempt to divide a feature space into clusters and assume that different classes will be represented by distinct clusters. To do that, the average pairwise distance between training examples of different classes for various feature subsets is calculated. The bigger the distances, the better the feature subset divides the space and the easier it is to predict the class. Examples of interclass measures are given in Table 2.4.

| Interclass Measures | |
|---|---|
| **Name** | **Formula** |
| Euclidean Distance | $J(X_i, X_j, F') = \sqrt{\sum_{l \in F'} (x_{i,l} - x_{j,l})^2}$ |
| Average Pairwise Distance | $J(X_i, X_j) = P(C_1)P(C_2)\frac{1}{N_1 N_2}\sum_{l \in F'}\sum_{t \in F'} dist(x_{1,l}, x_{2,t})$ |
| Mahalanobis Distance | $J(F') = (\mu_1 - \mu_2)^t \sum^{-1} (\mu_1 - \mu_2)$ |

Table 2.4: Examples of Interclass Measures (adapted from [34] and [11]). $X_{i,l}$ is the value of feature $l$ for sample $i$, $N_i$ is the number of samples (instances) of class $i$, dist is any distance measure, $\mu_i$ is the sample mean for class $i$ (for features in $F'$), $\sum$ is the covariance matrix for the two classes also for features in $F'$

## 2.5 Conclusion

This chapter reviews important algorithms and evaluation functions used in feature selection. It is not exhaustive, but it gives an idea of the state of the art in the field. Many of the topics are particularly important in this work and will be further studied. Among them are the following: Relief is used because of its potential to find relevance between features; SFFS and SFBS are used to select features in the system as well as for performance comparison, since they are standard feature selection algorithms; Mahalanobis and Entropy distances are used in the selection; and Correlation is used to filter redundancy. These algorithms and measures and our use of them will be described in subsequent chapters. Other techniques, although not used explicitly in the system, were useful as insight on what could and could not work.

Chapter 3

# RELEVANCE

## 3.1 Introduction

The irrelevance filter is an essential part and the first step of the three-step system (irrelevance filter, redundancy filter, feature selection filter) used for feature selection. The objective is to eliminate features that are irrelevant, and by doing so significantly reduce the number of features that must be handled by subsequent modules. It does that by using a modified version of Relief [54] that adapt it to regression problems with large numbers of irrelevant features. This chapter describes the original Relief algorithm, the modified Relief and and how to select Relief's parameters. Also described is a new way of identifying and eliminating bias against non-monotonic functions (i.e., functions whose max/min do not occur at the extremes of their ranges).

Relief was selected because: 1) it is computationally efficient; 2) it is able to detect relevance even when features interact [23]; and 3) it returns a degree of relevance, allowing features to be ranked. The main disadvantages of Relief are that it was designed to work for two-class classification problems only. It does not deal with redundancy, and it may fail for noisy or uncharacteristic samples.

In this chapter there are four additional sections: Section 3.2 explains the Relief algorithm, modifications, parameters and the result of its test for simple features;

Section 3.4 discusses the bias, the proposed modifications to deal with it and results before and after modifications; Section 3.5 shows some results for real features; and Section 3.6 summarizes the results and contributions of our study of Relief.

## 3.2 Relief

Relief is a relevance-based function selection algorithm inspired by instance-based learning techniques. It defines relevance in terms of consistency. It assumes that samples belonging to the same class should be closer, in terms of their feature values, than samples from different classes. As a result, if two instances (samples) belong to the same class and are generally similar, the similar features for the two instances are rewarded and the different ones are punished. When the samples belong to different classes the opposite occurs (see algorithm below).

One important question addressed here is whether Relief is able to identify relevance when the learning task is function approximation. We propose modifications to deal with function approximation. We also use some modifications proposed by Kononenko [61] that generalize the algorithm to multiple classes and reduce its sensitive to noise. As part of this process, we need to test different versions of the modified algorithm and tune its parameters to best perform in our context.

The basic Relief algorithm selects samples randomly from the sample space. For each sample, it finds the closest sample of the same class (hit) and the closest sample of a different class (miss). The difference in feature values between the sample and the hit and between the sample and the miss are then used to update the weights of each feature (see formula below). If the hit difference is smaller than the miss difference the weight will increase, indicating that the feature is a good one. The pseudo-code below formalizes the algorithm; different versions are generated by modifying subroutines (steps of the algorithm), such as how samples are selected. In this way,

the pseudo-code covers the original Relief algorithm, Kononenko's versions and our version. We call this space of algorithms FARelief (Function Approximation Relief).

**Basic algorithm**

▷ Given:

- A set of classes $S = \{c_1, c_2, ..., c_q\}$, where $q$ is the number of classes;
- A set of samples $S = \{s_1, s_2, ..., s_n\}$, where $n$ is the number of samples;
- A set of features $F = \{f_1, f_2, ..., f_p\}$, where $p$ is the number of features;
- A set of values $X = \{x_j^i \mid i = 1, 2, ..., p \text{ and } j = 1, 2, ..., n\}$;
- Class instances $c_i = \{s_j \mid s_j \in S \text{ and } s_j \text{ is associated with } c_i\text{'s label}\}$;
- Sample instances $s_i =< x_i^1, x_i^2, ..., x_i^p >$;
- Feature instances $f_i =< x_1^i, x_2^i, ..., x_n^i >$; and
- A set of feature weights $W = \{w_1, w_2, ..., w_p\}$, where weight $i$ is associated with feature $i$

do:

▷ 0: Set weights to zero ($w_i = 0 \quad \forall w_i \in W$)

▷ 1: Select $S'$, subset of $S$ ($S' \subseteq S$).

▷ 2: For all $s_i$ in $S'$ do:

- ▷ 2.a: Select $S''$, subset of $S$ ($S'' \subseteq S$).
- ▷ 2.b: Divide $S''$ in classes.
- ▷ 2.c: Find the closest samples in $S''$ of the same class of $s_i$ (hits)

$$Distance(h_i, s_i) = min \quad Distance(h_j, s_i) \quad \forall h_j \in S''$$

such that $s_i, h_j$, and $h_i \in c_k$ (other versions select the $r$ closest hits or select one hit randomly among the $r$ closest hits)

- ▷ 2.d: Find the closest samples in $S''$ of different class of $s_i$ (misses).

$$Distance(m_i, s_i) = min \quad Distance(m_j, s_i) \quad \forall m_j \in S''$$

such that $s_i \in c_k$, $h_j$ and $h_i \in c_m$, and $k \neq m$ (same as for hits)

- ▷ 2.e: Compute the Euclidean distance in the feature space between the hits and $s_i$ and the misses and $s_i$ and normalize it by the a priori probability of the hit or miss class.
- ▷ 2.f: Use the distances to update each feature's weight by:

$$w_j = w_j - (x_i^j - hit_i^j)^2 + (x_i^j - miss_i^j)^2 \quad \forall w_j \in W$$

▷ 3: Normalize the weights using the max and min distances for each feature and the number of samples used

$$W_j = \left( \frac{\sqrt{W_j} - Mindist_j}{Maxdist_j - Mindist_j} \right)^2 \frac{1}{n}$$

▷ 4: Rank the features according to their weights.

▷ 5: Select the best features.

Note that in our system the features are normalized between [0,1] before running Relief. This makes the Max distance and Min distance be, in practice, always zero and one respectively. Consequently the normalization at step 3 of the algorithm simplifies to:

$$W_j = \frac{W_j}{n}$$

The algorithms' performance depends on how each step is implemented. Five steps are manipulated to generate different versions of the algorithm: Step 1) how to select samples to apply the algorithm (select $S'$); Step 2.a) how to select samples to consider for hits and misses (select $S''$); Step 2.b) how to divide samples ($S''$) into classes; and Steps 2.c-d) how to select the hits and misses. Each as explained below.

**Selecting samples to apply the algorithm.**

$S'$ is a subset of $S$, such that every sample $S'$ will be compared to its nearest hit and miss. Intuitively, the more data that used, the better the final feature ranking. So, ideally $S' = S$, but because the complexity of the algorithm is dependent on the square of the size of $S'$, some versions randomly sample $S$ if too many samples are available.

**Selecting samples to consider for hits and misses.**

$S''$ contains the samples that will be used to select hits and misses. $S''$ is also a subset of $S$. Nevertheless, the situation is quite different here. Using the whole set $S$ can have undesirable consequences depending on how the best hit and miss are

selected (see below). It is therefore useful to sample $S$ before computing the best hit and miss. In FARelief, for implementation purposes, we decided to make $S''$ not only a subset of $S$, but also a subset of $S'$. Intuitively, that choice is of no consequence for the performance of the algorithm. In practice, no effect related to this decision was noticed.

**Dividing samples ($S''$) into classes.**

Relief assumes the data is divided into two classes; Kononenko extended that to $n$ classes. In function approximation tasks, however, the data is not divided in discrete classes a-priori. Consequently, three different ways to divide the set into classes were implemented and tested (see Figure 3.1). The first use a threshold over the learning signal to divide the sample set into two classes. This is a direct adaptation of Relief to function approximation. The second, for each sample $X_i$ in $S'$ divide $S''$ into two classes according to the learning signal; one class of all samples that are inside a neighborhood around $X_i$, and another class of all samples outside the neighborhood. This approach seems, intuitively, to be the best option for function approximation, because it provides information about how to differentiate samples associated with a value (and close values) from the rest of the samples. The third divides the samples into a set of classes depending on their associated value (learning signal). In this case, the signal is equally divided in $n$ ranges (classes), but the classes may have different numbers of samples, particularly if the signal is not uniformly distributed.

**Selecting hits and misses.**

For every sample in $S'$, Relief selects the closest "hit" and "miss" sample in $S''$ and uses them to update the feature weights. Unfortunately, this strategy is sensitive to noise. To see why, consider Figure 3.2, which shows two disjoint classes, each with one aberrant point. Although classifying the samples should be simple,

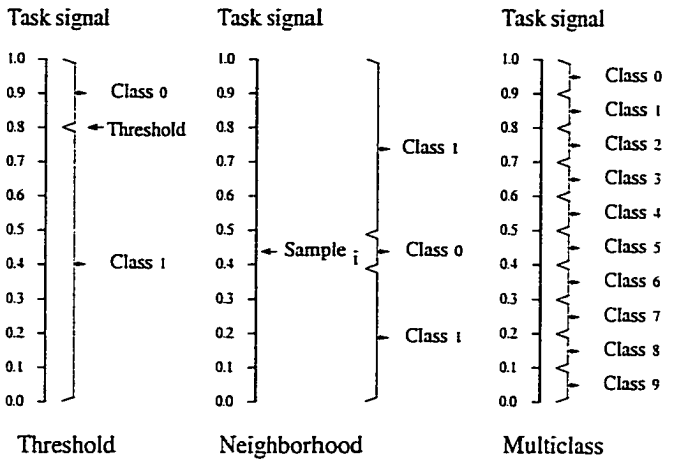



Figure 3.1: Three ways to select classes.

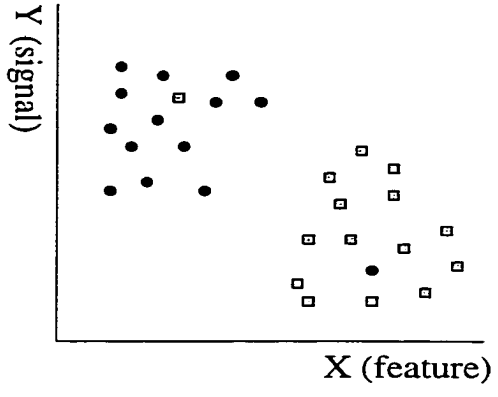



Figure 3.2: Two classes with noisy data.

the two noisy points will always be selected the best hit and miss for other points, and consequently Relief does not consider the feature to be relevant.

As an alternative, Kononenko computes the best $n$ hits and misses for every sample $X_i$ then updates the weights according to the average difference with $X_i$. This option is included in FARelief (see Section 3.3.1). Another option selects $n$ hits and misses, as in Kononenko's, but then randomly selects one hit or miss from each class for updating the feature weights. A third option reduces the effect of noise by selecting a single hit and miss for each class from a random subset of the sample set.

## 3.3 Selecting Algorithm Versions and Parameters

As described above, FARelief is less an algorithm than an algorithm space; it includes a variety of options and parameters for adapting Relief to specific tasks. In practice, a good set of options and parameters must be chosen for each task. Our method of adapting Relief is to generate a synthetic data set with properties that mimic real data as closely as possible, but where a ground-truth ranking of relevance is available. Various versions of FARelief are then tested on the data set, selecting the options and parameters that produce rankings that most accurately match the ground-truth rankings. To check the results, the selected version of FARelief is then applied to a real data set. Although the true relevance of real features is unknown for any specific task, we were able to apply some weak statistical tests to show that Relief's ranking of real features are at least better than random.

The synthetic data set must have two properties: it must contain the types of features we expect to encounter in real data; and the relevance of the features to the task must be known a-priori. To create realistic features, synthetic signal values were selected from a uniform probability distribution between zero and one. Each signal value represents the "true" value of the target function for one sample. A total of 630 signal values were generated, since this is the number of samples in the ADORE data set (used in Section 3.5 and evaluated more throughly in Section 5.4).

The synthetic signal must have the property that it is a function of the synthetic features. At the same time, many features must be generated for a single signal, since every sample has many features but only one signal value. To do this, a selected type for each feature, either linear or quadratic (see Figures 3.3 and 3.4), is used to generate a function by randomly selecting parameters, for example $a$, $b$ and $c$ for $ax^2 + bx + c - signal = 0$. The equation is then solved and the feature values computed. Equations without linear roots are discarded, as are quadratic functions

that are degenerate in the sense of being approximately linear. See Appendix C for how to detect approximately linear functions.

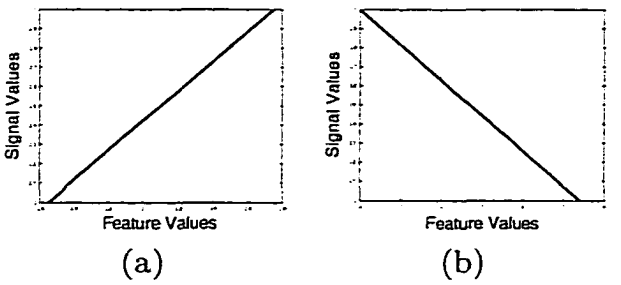



Figure 3.3: Synthetic Features: Linear.

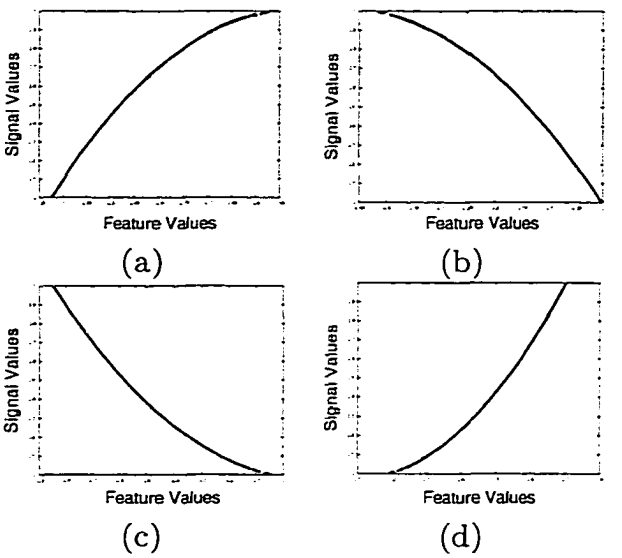



Figure 3.4: Synthetic Features: Quadratic.

Quadratic functions are as relevant as linear functions. Since both types of functions are invertible, the signal can, in principle, be reconstructed from either type of function, although some regression techniques may prefer one over the other. What makes a function more or less relevant is the amount of noise added to the feature values. Here, each randomly generated function is replicated eleven times, each time increasing the amount of noise added to the feature values. The noise levels

range from 0.0 to 0.5, where the noise level indicates the amount of the standard deviation of a random zero-mean Gaussian function that is added to every feature value. Figure 3.5 shows examples of all eleven versions of one quadratic function. After each randomized feature is created, it is normalized to the range [0,1], as a preliminary step of the Relief algorithm.

For each randomly selected function, the ground-truth ranking is therefore known since we assume that the more noise is added to a function, the less relevant it becomes. Six hundred feature sets were created, each containing eleven features with the same underlying function, but with varying levels of noise. Two hundred of those sets were linear and four hundred quadratic, since there are two types of linear features and four types of quadratic (see figures 3.3 and 3.4).

The features in the resulting synthetic feature sets, although noisy, are not yet a good reflection of the types of features expected in real data. Those features are all monotonic, while many computer vision features are non-monotonic, with a single peak or valley somewhere between the extremes of their range. This will be addressed in section 3.4.

Given these synthetic feature sets, every version of FARelief was applied to each feature set and the rankings produced by FARelief compared to the ground-truth rankings using the Spearman's Rank-Order Correlation [82]. Specifically the follow steps were used:

- Create synthetic data.
    - Randomly create a learning signal.
    - Randomly create a set of features with increasing levels of noise
    - Normalize the features.

- Run FARelief
    - Run FARelief with different options on the synthetic data, recording the relevance measure.
- Analysis of the results.
    - For each type of feature, rank the features according to their noise level ("true ranking")
    - For each type of feature, rank the features according to the relevance measure computed by FARelief.
    - Correlate the FARelief rank with the true feature ranking.
    - Select the FARelief options which rank the features most accurately.
    - Select one or more options.

Our goal is to find which version of FARelief performs well on the synthetic data and to analyze the algorithms' sensitivity to each aspect. The next two sections describe in detail the versions and parameters tested for FARelief, and how the results were analyzed.

### 3.3.1 Running FARelief

FARelief includes options to define: how to divide the samples in classes; how to sample the classes; and how to find the hit/miss for a sample (see page 38). These options are encoded in a series of parameters:

1. **Class definition**: This parameter sets the way the classes will be defined on Step 2.b of the algorithm (page 37) and Figure 3.1.

    1.1 **Threshold** indicates that there will be only two classes and that a threshold will divide the data into these two classes depending of the signal value for each sample. Thresholds 0.5 and 0.8 were tested.

1.2 **Multiple class**: The data is divided into $n$ classes associated with different ranges of values in the learning signal. Multi-class with 3, 5, and 9 classes were tested.

1.3 **Neighborhood** For each sample, the data is divided into two classes. One class includes the sample and its neighbors defined by a neighborhood size over the learning signal, the other includes everything else. Neighborhoods of size 0.05 and 0.1 were tested.

2. **Sampling**: This parameter defines how the training set is sampled in Step 2.a of the algorithm (page 37).

   2.1 If the option is set to **All**, all the samples will be used to find the best hit/miss.

   2.2 If the option is set to **Perc**, a randomly selected percentage of the data is used. Note that using all samples is different from randomly sampling 100% because sampling is done with replacement[1]. Percentages of 0.5 and 0.8 were tested.

3. **Hit/Miss selection**: This parameter defines how the best hit and miss are defined in Step 2.c-d of the algorithm (page 37).

   3.1 **Best**: The best hit and miss for each sample are used to update the features weights. This is standard Relief [54].

   3.2 **Perc**: Instead of looking for the best hit and miss for all samples of each class, a randomly selected subset of the class is used to select the hit

[1]Three different random generators were tested, rand() from C++ library, Knuth's random generator and Park and Miller with Bays-Durham shuffle random generator, with the last two extracted from "Numeric Recipes in C" [82]. One thousand numbers between 0 and 1000 were generated. The number of different values generated for any of the generators tested fell between 628 to 650. So, the number of repetitions can be very high.

and miss samples. This minimizes the problem described above, in Figure 3.2, because a noisy point will only be present in some of the subsets. Hence, the probability of a noise point be selected depends of the degree of noise and of the percentage of the class used. Percentages of 0.5 and 0.8 were tested.

3.3 **Number**: The **n** best hits and **n** best misses are computed and one of each is selected randomly to update the features. Tests were done with $n = 5$, 10, and 20.

3.4 **Average**: The **n** best hits and **n** best misses are computed and the average of each is used to update the features. This is Kononenko's version of Relief [61]. Tests were done with $n = 5$, 10, and 20.

The options tested were all combinations of the 3 parameters for the values listed above. FARelief was run on 600 synthetic feature sets for each of the parameter settings. If a parameter setting includes random selections at any stage of the process, then that parameter setting was run five times and the results were averaged.

### 3.3.2 Analysis of the results

The result of FARelief is a set of ranked features. To evaluate this result, a rank correlation is computed between the FARelief ranking and the true noise-based ranking of the features. The correlation method used is Spearman's Rank-Order Correlation [82]. In this method, a linear correlation coefficient $r$ is computed as:

$$r = \frac{\sum_i (R_i - \overline{R})(S_i - \overline{S})}{\sqrt{\sum_i (R_i - \overline{R})^2} \sqrt{(\sum_i (S_i - \overline{S})^2}}$$

where $R_i$ is the real rank for feature $i$, $\overline{R}$ is the average rank for the true ranked features, $S_i$ is the computed rank for feature **i** and $\overline{S}$ is the average rank for the computed rank features.

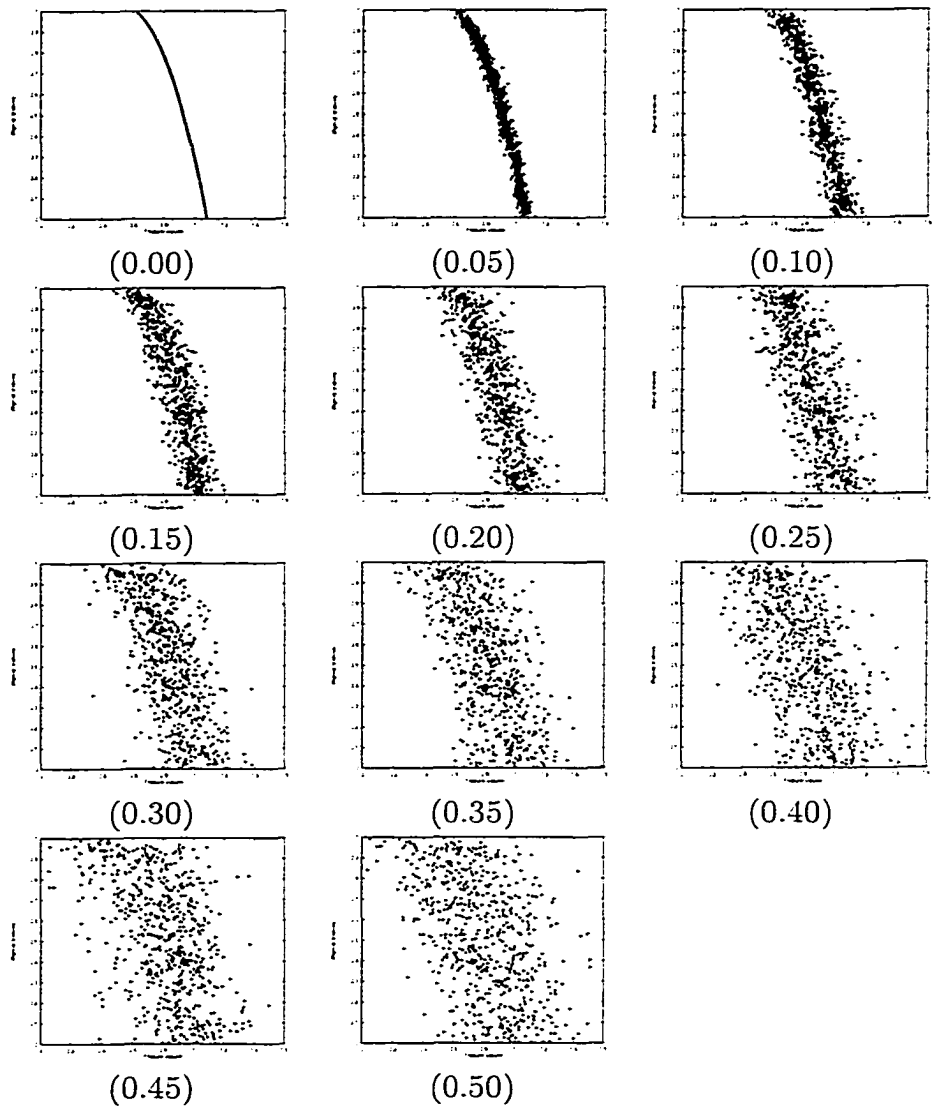



Figure 3.5: Noise level for a quadratic function. The number below the graph indicates the noise level.

The significance of this value, if $r \neq 0$ and $r \neq 1$, is given by:

$$t = r\sqrt{\frac{N-2}{1-r^2}}$$

where N is the total number of features. $t$ is distributed approximately as Student's distribution with N - 2 degrees of freedom. One important point is that this approximation does not depend on assumptions about the original distribution of the features.

Spearman's correlation gives a measure of quality for each run of the system. The question is whether any set of parameters performs consistently better than the others across all 600 feature sets. One of the first observations is that some parameter settings perform erratically, producing good results on some feature sets while producing poor results on others. This is clearly undesirable, since in the final system the irrelevance filter will only be applied once and we want the performance of the system to be reliable. Therefore, any parameter setting with a variance in correlation of more than 0.01 across the feature sets was discarded. Ideally, one version of Relief would be optimal for all feature types. Unfortunately, an ANOVA test between parameter settings feature sets indicates that they are not independent. This is a set back, but not an insurmountable obstacle, since our objective here is only to remove irrelevant features. Later stages in the process will select the best feature set. So, we only need a parameter setting that works well for most feature types. To better analyze this, graphs of the average rank correlation measure for the class definition parameter and for the hit/miss selection parameter were created for each feature type. These two parameters have the most influence on the performance of the algorithm. Figure 3.6 shows the graphs after removing the parameter settings with high variance. To make the graphs easy to read, each parameter setting was given a number. These values are listed in table 3.1. The numbers between parenthesis in the caption are the number of function types and the number of instances of functions

| Horizontal Axis | | | Lines | | |
|---|---|---|---|---|---|
| Axis Position | Hit/Miss Selection | Value | Line Number | Class Definition | Value |
| 0 | Best | — | 0 | Threshold | 0.50 |
| 1 | Percentage | 0.50 | 1 | | 0.80 |
| 2 | | 0.80 | 2 | Neighborhood | 0.05 |
| 3 | | 5 | 3 | | 0.10 |
| 4 | Number | 10 | 4 | | 3 |
| 5 | | 20 | 5 | Multi-class | 5 |
| 6 | | 5 | 6 | | 9 |
| 7 | Average | 10 | | | |
| 8 | | 20 | | | |

Table 3.1: Axis position and lines legend for graphs at figure 3.6

of each type. For example, in Item b of Figure 3.6 "(4x100)" means that there are 4 basic types of quadratic functions (see Figure 3.4) and 100 functions of each type.

Figure 3.6 shows that for linear and quadratic functions the parameters are independent in the region where the best performance occurs (upper right corner). This allows each parameter to be selected independently. In order to reinforce this result and to handle cases were independence can not be established, another kind of analysis is used. A run of FARelief is considered a success if the correlation between ranked features and ground-truth is greater or equal to a threshold of 0.9 (this threshold yields a probability of more than 0.9995 that the two ranks have the same distribution). An example of feature rank which has 0.9 correlation with the true ranking of (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11) is (1, 2, 3, 4, 5, 7, 6, 11, 10, 9, 8). Any pair of parameters can then be tested to see if they have significantly different probabilities of success using the binomial test [63]:

$$\frac{\frac{x}{n} - \frac{y}{m}}{\sqrt{\frac{\left(\frac{x+y}{n+m}\right)\left(1 - \frac{x+y}{n+m}\right)(n+m)}{nm}}} \quad \begin{cases} \leq -z_{\alpha/2} \\ or \\ \geq +z_{\alpha/2} \end{cases}$$

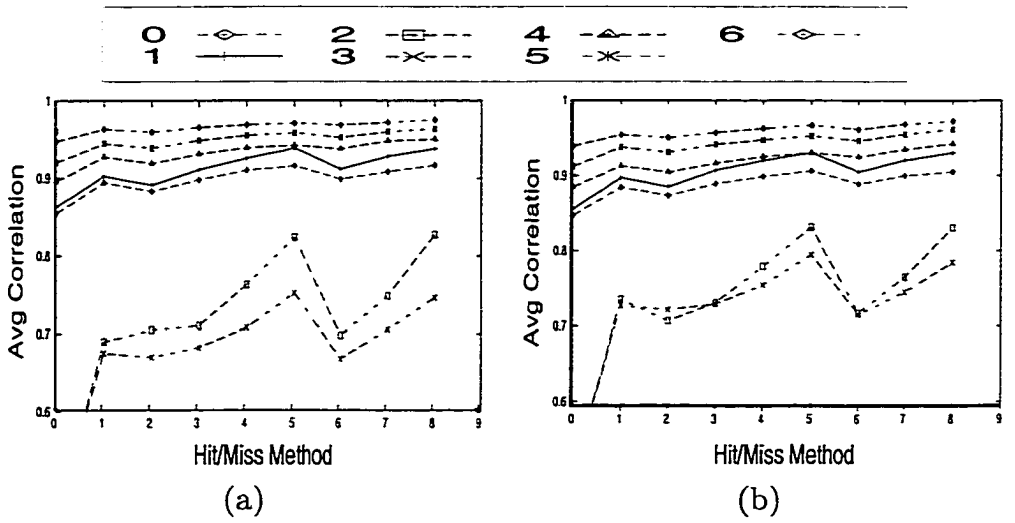



Figure 3.6: Average correlation score, after pruning, for all files for combinations of the class definition parameter and hit/miss selection parameter. a) Linear (2x100); b) Quadratic (4x100);

where **x** and **y** are the number of successes for the parameter settings being tested,**n** and **m** are the number of runs for the parameter settings, **z** is the Standard Normal and $\alpha$ is the level of significance.

The binomial test allows discarding parameter settings that have lower average success rates than the best setting, to within a level of significance $\alpha$. After removing those parameters the intersection of the remaining parameter settings is selected, if such intersection exists. Table 3.2 shows the results. Columns correspond to options of how to select hits and misses and rows correspond to options of how to define classes (respectively parameters 3 and 1), as in figure 3.6. The Max value is the number of runs over the threshold for the best parameter setting for the feature type. The "cut point" is the minimum number of successes a parameter setting has to have in order to be retained. Underlined values are parameters settings that were not discarded for an $\alpha$ of 0.005.

According to Table 3.2 the parameter settings than can not be discarded are:

- for linear functions:

| | | Linear (2x100) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Best (B) | Perc (P) | | Number(N) | | | Average (A) | | |
| | | | 0.5 | 0.80 | 5 | 10 | 20 | 5 | 10 | 20 |
| Threshold (T) | 0.50 | 73 | 112 | 91 | 112 | 135 | 146 | 107 | 135 | 145 |
| | 0.80 | 85 | 121 | 112 | 125 | 153 | 178 | 131 | 158 | 174 |
| Neighborhood (N) | 0.05 | 0 | 0 | 0 | 0 | 0 | 7 | 0 | 0 | 10 |
| | 0.10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Multi-class (M) | 3 | 104 | 161 | 145 | 166 | 182 | 183 | 171 | 184 | 183 |
| | 5 | 143 | 182 | 169 | 186 | 195 | 197 | 189 | 193 | 195 |
| | 9 | 185 | 197 | 197 | 199 | 198 | 199 | 196 | 197 | 200 |
| | | Max: 200 Cut Point: 193 | | | | | | | | |

| | | Quadratic (4x100) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Best (B) | Perc (P) | | Number(N) | | | Average (A) | | |
| | | | 0.5 | 0.80 | 5 | 10 | 20 | 5 | 10 | 20 |
| Threshold (T) | 0.50 | 117 | 178 | 156 | 194 | 231 | 251 | 190 | 227 | 241 |
| | 0.80 | 136 | 223 | 187 | 253 | 295 | 336 | 248 | 300 | 334 |
| Neighborhood (N) | 0.05 | 3 | 0 | 0 | 0 | 1 | 22 | 0 | 3 | 20 |
| | 0.10 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 3 | 1 |
| Multi-class (M) | 3 | 196 | 275 | 237 | 283 | 319 | 333 | 303 | 337 | 350 |
| | 5 | 262 | 353 | 337 | 364 | 378 | 391 | 368 | 378 | 388 |
| | 9 | 352 | 381 | 380 | 392 | 397 | 398 | 389 | 398 | 399 |
| | | Max: 399 Cut point: 392 | | | | | | | | |

Table 3.2: Number of successful runs for each parameter setting (correlation threshold of 0.9). Underlined values are statistically equivalent according to the two sample binomial test ($\alpha = 0.005$). Columns correspond to ways of selecting hits/misses (parameter 3) and rows correspond to ways to dividing the samples into classes (parameter 1)

- { (M 5, N 10), (M 5, N 20), (M 5, A 10), (M 5, A 20), (M 9, N 5), (M 9, N 10), (M 9, N 20), (M 9, A 5), (M 9, A 10), (M 9, A 20) };

- for quadratic functions:

  - { (M 9, N 5), (M 9, N 10), (M 9, N 20), (M 9, A 10), (M 9, A 20) };

Note that all selected parameter settings for quadratic functions are also selected for linear functions. That indicates that they work well to detect relevance for both types of features and that they can not be statistically differentiated from each other. In this case we can use any of the options listed.

Qualitatively the results can be summarized by:

1. Multi-class is the best way to divide the samples.

2. The more classes, hits and misses used, the better the results achieved.

3. Randomly selecting hits and misses or averaging them yields similar results.

### 3.3.3 Evaluating set and class sizes.

To test assertions that "the more classes, hits and misses used, the better the result" another test was designed where only option "Multi-class" (M) and option "Average hits and misses" (A) were used. The goal was to test how results can be affected by the size of the sample set and the ratio $R$ of the number of hit/miss samples to the number of samples in each class. Four sample set sizes were used: 500; 1000; 1500; and 2000. For each of these set sizes, 300 functions (100 linear, 200 quadratic) were run for a combination of values of the number of hits and misses averaged and number of classes. Table 3.3 shows the number of classes used for for each of the sample sets.

Running the sample sets for the number of classes shown in Table 3.3 allows the data to be organized in two different ways.

| Sample set size | | | |
|---|---|---|---|
| 500 | 1000 | 1500 | 2000 |
| 2 | 2 | 2 | 2 |
| 3 | 4 | 4 | 4 |
| 4 | 6 | 6 | 8 |
| 5 | 8 | 9 | 9 |
| 9 | 9 | 12 | 12 |
| 12 | 10 | 20 | 16 |
| 15 | 12 | 36 | 20 |
| 20 | 15 | 45 | 36 |
| | 18 | 60 | 48 |
| | 20 | | 60 |
| | 24 | | 80 |
| | 30 | | |
| | 40 | | |

Table 3.3: Number of classes run for each sample set size.

- Test 1: Fixed number of classes but variable ratio $R$ (Figures 3.7 and 3.8). Each graph corresponds to one set size. Each line corresponds to one number of classes. For example, line 0 has 2 classes whether it is on the set with 500 samples or 2000 samples (see Table 3.4). The horizontal axis (from 0 to 8) corresponds to the number of hits/misses averaged (respectively 1, 2, 3, 5, 9, 13, 17, 21, and 25). This means that any point in the graph has a different ratio $R$ depending of the set size. For example, $x = 8$ for line 4 corresponds to 20 classes and 25 hits/misses. That is a ratio of 1 for the set of size 500, 0.5 for a set of size 1000, 0.33 for the set of size 1500, and 0.25 for the set of size 2000.

- Test 2: Fixed ratio but variable number of classes (Figures 3.9 and 3.10). Here each line corresponds to one ratio. That means that the number of classes is changed so as to keep the ratio fixed. Table 3.4 shows the number of classes used and their corresponding values in the graphs.

| | Test 1: Equal # of classes | | | | Test 2: Equal ratios ($R$) | | | |
|---|---|---|---|---|---|---|---|---|
| | Sample set size | | | | Sample set size | | | |
| line | 500 | 1000 | 1500 | 2000 | 500 | 1000 | 1500 | 2000 |
| 0 | 2 | 2 | 2 | 2 | 2 | 4 | 6 | 8 |
| 1 | 4 | 4 | 4 | 4 | 3 | 6 | 9 | 12 |
| 2 | 9 | 9 | 9 | 9 | 4 | 8 | 12 | 16 |
| 3 | 12 | 12 | 12 | 12 | 5 | 10 | 15 | 20 |
| 4 | 20 | 20 | 20 | 20 | 9 | 18 | 27 | 36 |
| 5 | | | | | 12 | 24 | 36 | 48 |
| 6 | | | | | 15 | 30 | 45 | 60 |
| 7 | | | | | 20 | 40 | 60 | 80 |

Table 3.4: Number of classes for each sample set size for each of the tests. The left most column shows the corresponding line number used in each graph.

Test 1 (Figures 3.7 and 3.8) confirms that for linear and quadratic functions, the more classes, hits and misses the better the result. Test 2 (Figures 3.9 and 3.10) shows that the ratio $R$ does not have a significant influence in the result for these functions, since if the ratio were important there should be a significant difference between them.

It is important to note that these results were computed for monotonic functions where the learning signal is uniform. This result can not be generalized without further tests. In fact, we will show later that for non-monotonic functions the ratio will have a great influence.

## 3.4 Bias: Non-monotonic Functions

### 3.4.1 Data and Evaluation of Non-monotonic Functions

The parameter settings selected in Section 3.3.2 allow a good ranking of monotonic functions. Nevertheless, in computer vision it is common to have features that are not monotonic. For example, consider a feature such as size. If the size of the image is known, then the best hypothesis is the one whose size matches the size of

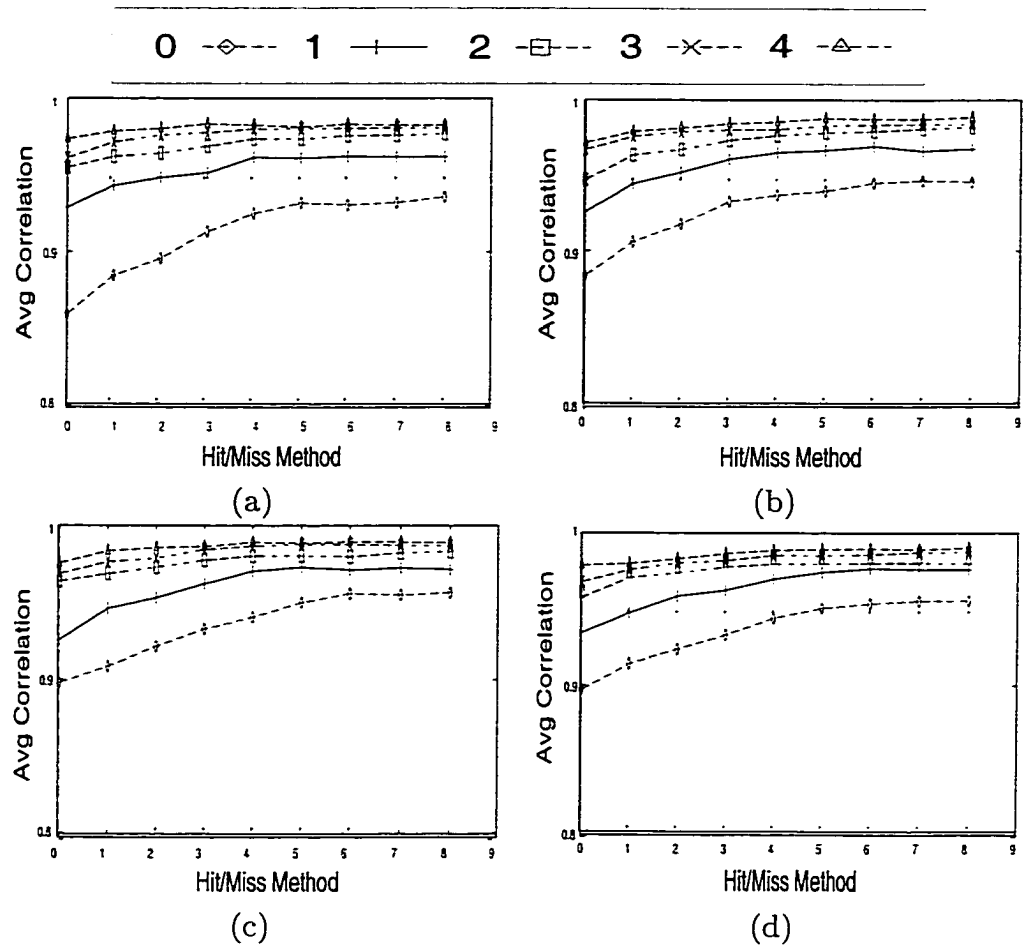



Figure 3.7: Linear features: Average correlation score for files for combinations parameters for equal number of classes for all sample sizes (see table 3.4). a) Sample size 500; b) sample size 1000; c) sample size 1500; d) sample size 2000.

the target object. Smaller or larger hypotheses are either fragmented, under segmented or do not represent the object and will therefore have lower signal values. An idealized case might look like the feature in Figure 3.11.

In order to test how FARelief performs for these kinds of functions, we created a new set of non-monotonic synthetic functions. These functions are generated by reflecting one of the monotonic functions about the midpoint of the feature axis[2].

[2]It is important to note that these are functions of the features (f: feature $\rightarrow$ signal), but they are not invertible (i.e. $\nexists\, f^{-1}$: signal $\rightarrow$ feature). This poses a problem since what we have available

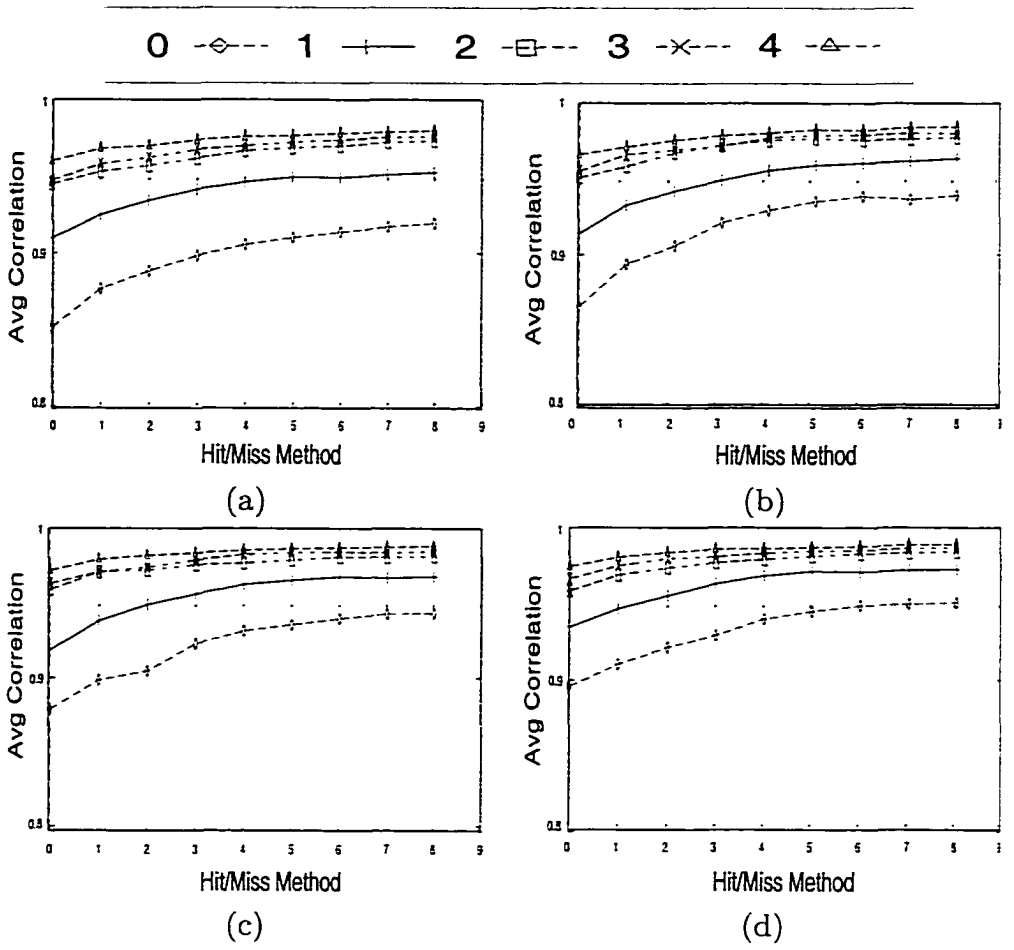



Figure 3.8: Quadratic features: Average correlation score for files for combinations of parameters for equal number of classes for all sample sizes (see table 3.4). a) Sample size 500; b) sample size 1000; c) sample size 1500; d) sample size 2000.

Figures 3.11 and 3.12 show examples of these functions. In addition, the amount of noise for non-monotonic features is divided by two so that the signal to noise ratio remains the same as on monotonic features.

The same tests previously performed for linear and quadratic functions were performed for the non-monotonic function using the methodology presented in Section 3.2. The parameters tested are also the same as in Table 3.1. The results, however,

at this stage is the signal. So, to compute the non-monotonic features for the signal set we compute the two possible feature values (one in each side of the midpoint) and select one randomly.

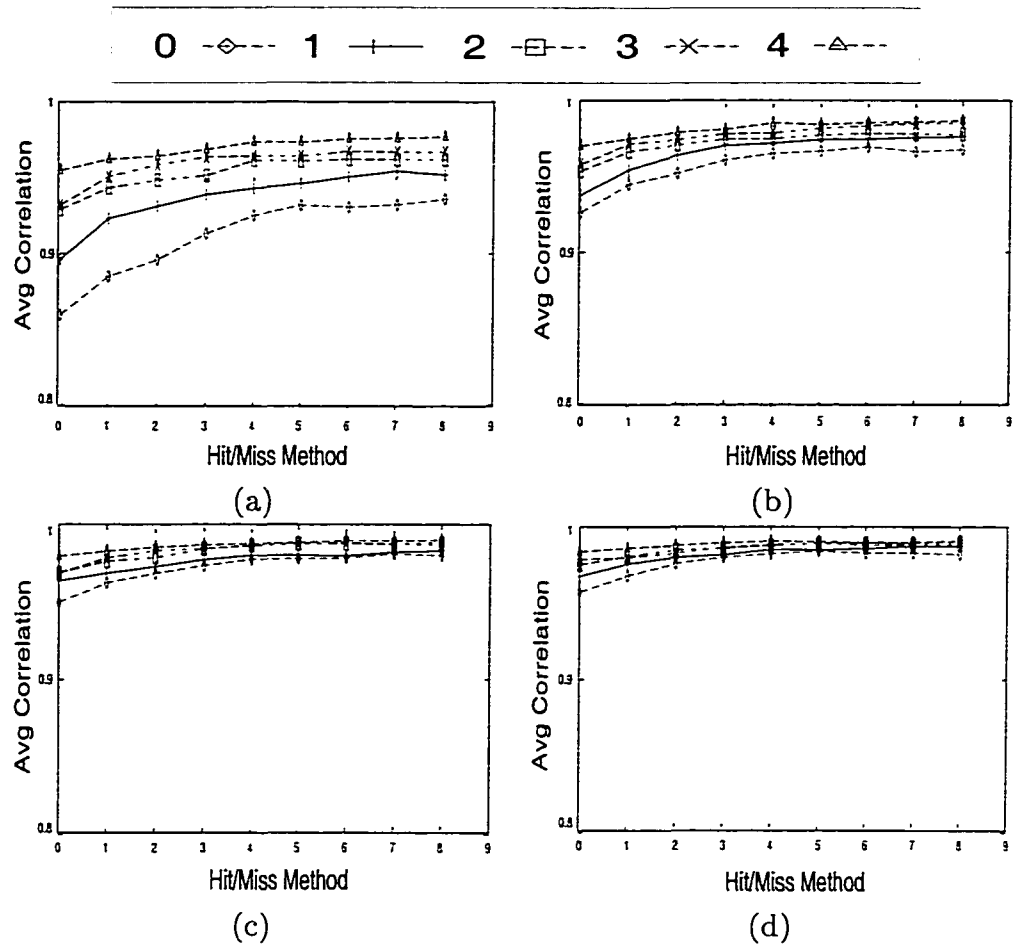



Figure 3.9: Linear features: Average correlation score for files for combinations of parameters for the same ratio $R$ (see table 3.4). a) Sample size 500; b) sample size 1000; c) sample size 1500; d) sample size 2000.

are very different. Table 3.5 shows the results of the same binomial test summarized in Table 3.2, but this time for the non-monotonic features.

As can be seen, the best parameter settings for non-monotonic features are different from the best parameter settings for monotonic ones. Non-Monotonic features prefer fewer classes and fewer hits/misses. In fact, as shown in Figure 3.13, the ratio $R$ of the number of hit/miss samples to the number of samples in each class, varies from one graph to the other. This indicates that $R$ has a great influence over the result. This is in contrast to Figure 3.8, where no such influence exists.

| | | Non-monotonic Linear (2x100) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Best (B) | Perc (P) | | Number(N) | | | Average (A) | | |
| | | | 0.5 | 0.80 | 5 | 10 | 20 | 5 | 10 | 20 |
| Threshold (T) | 0.50 | 113 | 102 | 114 | 74 | 33 | 9 | 79 | 44 | 24 |
| | 0.80 | 116 | 108 | 108 | 110 | 103 | 83 | 107 | 99 | 80 |
| Neighborhood (N) | 0.05 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0.10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Multi-class (M) | 3 | 129 | 120 | 139 | 97 | 44 | 8 | 178 | 180 | 190 |
| | 5 | 126 | 101 | 120 | 72 | 22 | 0 | 166 | 178 | 183 |
| | 9 | 92 | 46 | 84 | 33 | 2 | 0 | 145 | 150 | 154 |
| | | Max: 190 Cut Point: 175 | | | | | | | | |

| | | Non-monotonic Quadratic (4x100) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Best (B) | Perc (P) | | Number(N) | | | Average (A) | | |
| | | | 0.5 | 0.80 | 5 | 10 | 20 | 5 | 10 | 20 |
| Threshold (T) | 0.50 | 96 | 47 | 76 | 36 | 8 | 0 | 48 | 12 | 0 |
| | 0.80 | 138 | 96 | 116 | 71 | 31 | 9 | 77 | 37 | 11 |
| Neighborhood (N) | 0.05 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0.10 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Multi-class (M) | 3 | 130 | 49 | 89 | 28 | 2 | 0 | 154 | 112 | 84 |
| | 5 | 128 | 28 | 55 | 10 | 0 | 0 | 114 | 95 | 69 |
| | 9 | 58 | 11 | 22 | 3 | 0 | 0 | 65 | 59 | 59 |
| | | Max: 154 Cut point: 120 | | | | | | | | |

Table 3.5: Non-monotonic Features: Number of successful runs (correlation threshold of 0.9). Underlined values are statistically equivalent according to the two sample binomial test ($\alpha = 0.005$). Columns correspond to ways of selecting hits/misses and rows correspond to ways to dividing the samples into classes

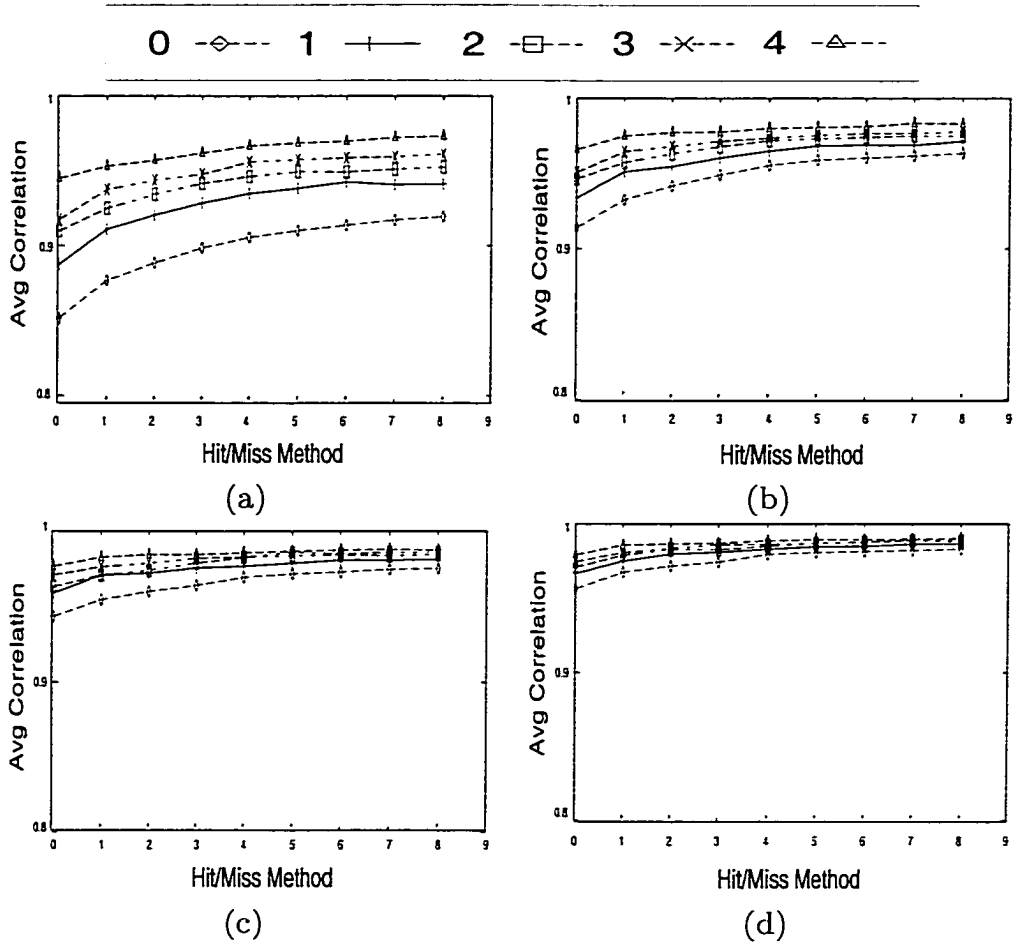



Figure 3.10: Quadratic features: Average correlation score for files for combinations of parameters for the same ratio $R$ (see table 3.4). a) Sample size 500; b) sample size 1000; c) sample size 1500; d) sample size 2000.

More importantly, the overall results are worse for non-monotonic features. For example, the best result achieved for quadratic functions was 399 out of 400, while the best result for the non-monotonic quadratic was 154 out of 400. This indicates a bias against non-monotonic functions. To confirm this bias, new data sets with 8 monotonic features plus non-monotonic versions of the same 8 features were created[3]. The average Relief scores for the monotonic features and their non-monotonic coun-

[3]Since a function can be reflected to either the left or right, there are two non-monotonic features for each monotonic feature.

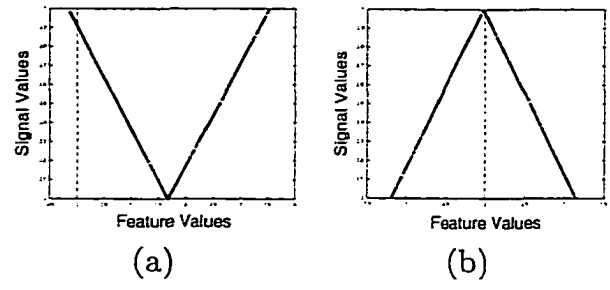



Figure 3.11: Non-monotonic Synthetic Features: Linear.

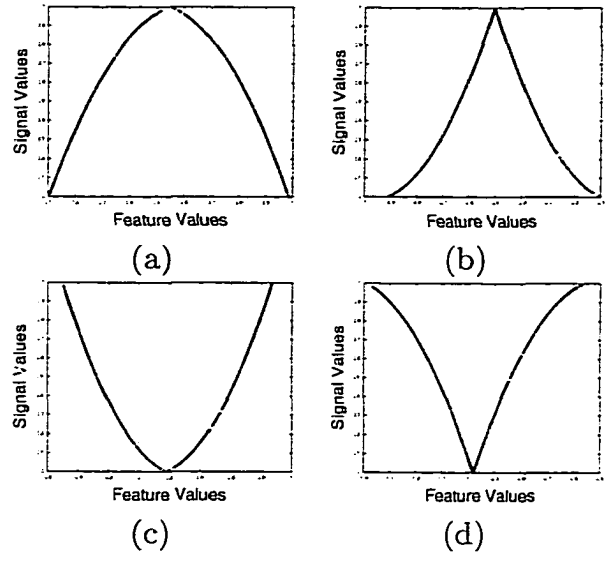



Figure 3.12: Non-monotonic Synthetic Features: Quadratic.

terparts are given in Table 3.6. The table shows a large bias against non-monotonic features. In some cases, the score for a non-monotonic feature is in excess of 1000 times smaller than the corresponding monotonic feature.

A visual confirmation of the bias is shown in Figures 3.14 and 3.15, where a set of 60 functions (10 linear, 20 quadratic, 10 non-monotonic linear and 20 non-monotonic quadratic) with noise level ranging from zero to 0.4 [4] were ranked by FARelief using

[4]Here, as in the previous tests, the non-monotonic functions have half the amount of error to compensate for the fact that they are formed by two distributions.

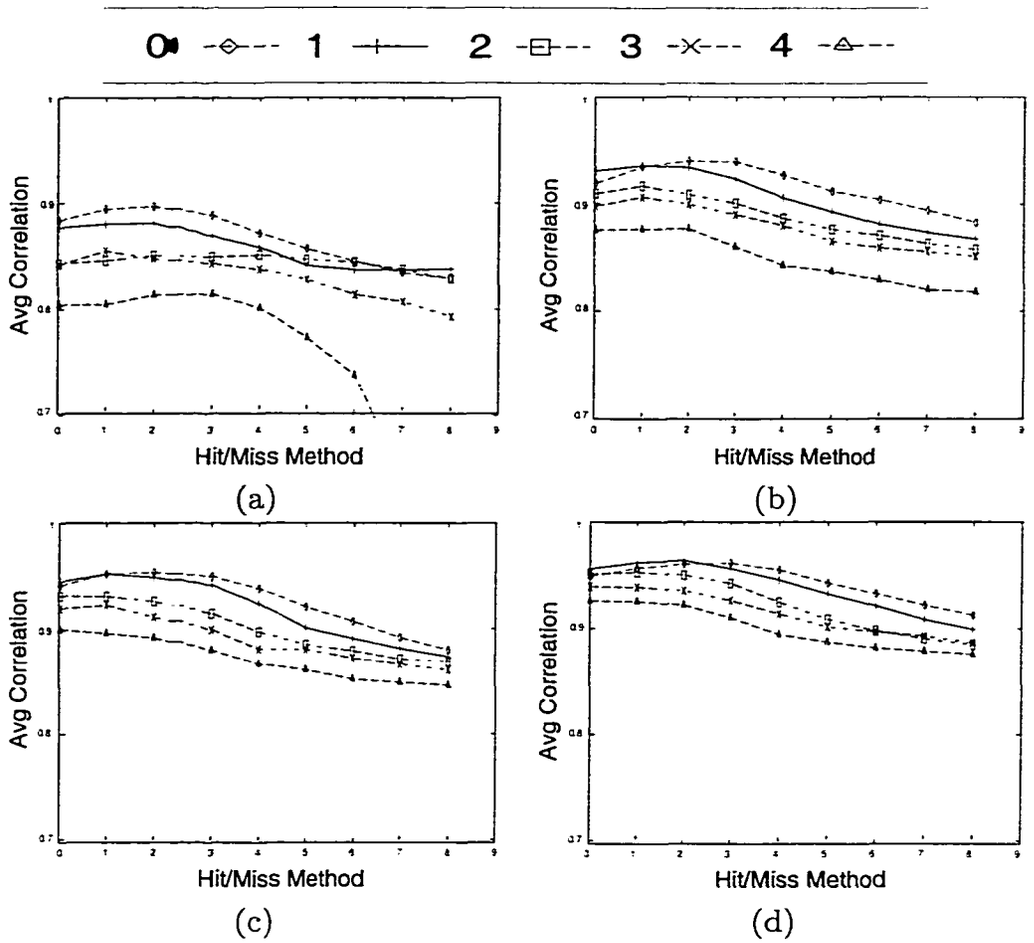



Figure 3.13: Non-Monotonic quadratic features: Average correlation scores for combinations of parameters across all sample sizes (see table 3.4). a) Sample size 500; b) sample size 1000; c) sample size 1500; d) sample size 2000.

9 classes and 20 hits/misses for average. Non-Monotonic functions with noise zero are ranked in the same level as monotonic functions with 0.2 noise level.

### 3.4.2 Identifying the causes of bias against non-monotonic features

The bias against non-monotonic features can be traced to two main causes: 1) the deterioration of the distance measure; and 2) the non-correlation of similar non-monotonic features.

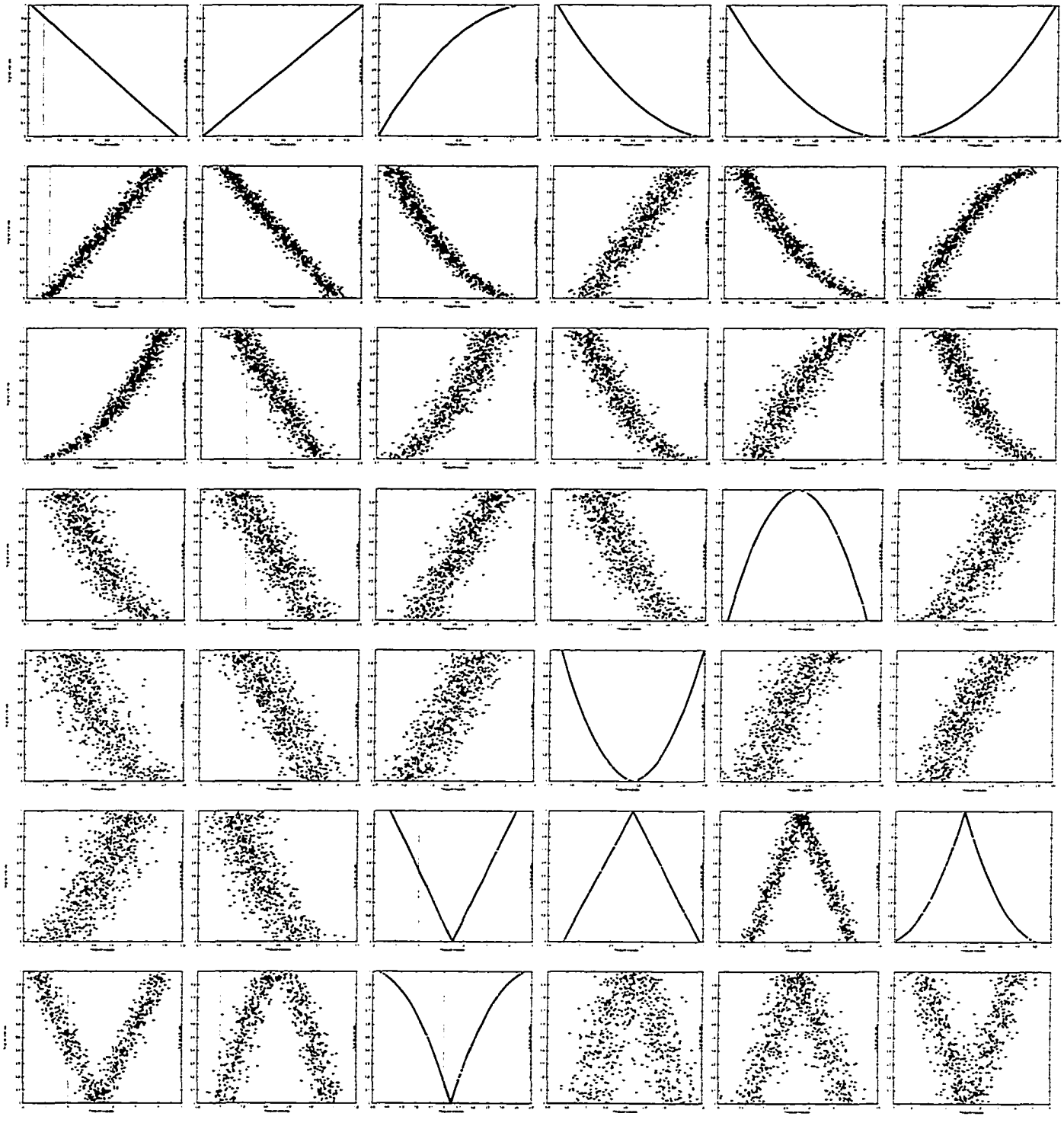


Figure 3.14: First 42 features of 60 (10 linear, 20 quadratic, 10 non-monotonic linear, and 20 non-monotonic quadratic) ordered, left to right, top to bottom, by using 9 classes and average of 20 hits/misses.

| F | Monotonic | | Non-monotonic | | | |
|---|---|---|---|---|---|---|
| | Noise | | Left Ref. | | Right Ref. | |
| | | | Noise | | Noise | |
| | No | Yes | No | Yes | No | Yes |
| L1 | 0.1636 | 0.1223 | 0.0028 | 0.0016 | 0.0042 | 0.0020 |
| L2 | 0.1636 | 0.1212 | 0.0036 | 0.0020 | 0.0027 | 0.0011 |
| Q1 | 0.1282 | 0.0977 | 0.0001 | -0.0006 | 0.0066 | 0.0031 |
| Q2 | 0.1285 | 0.0975 | 0.0067 | 0.0026 | 0.0002 | -0.0002 |
| Q3 | 0.1225 | 0.0931 | 0.0054 | 0.0019 | 0.0008 | 0.0003 |
| Q4 | 0.1219 | 0.0915 | 0.0006 | 0.0000 | 0.0052 | 0.0022 |
| G1 | 0.0666 | 0.0537 | 0.0044 | 0.0026 | 0.0000 | -0.0001 |
| G2 | 0.0666 | 0.0543 | 0.0002 | 0.0002 | 0.0047 | 0.0025 |

Table 3.6: Relief score (weight) for 8 features and their reflections. Rows L1 and L2 correspond to linear functions; rows Q1 to Q4 to quadratic functions; and rows G1 and G2 to Gaussian functions.

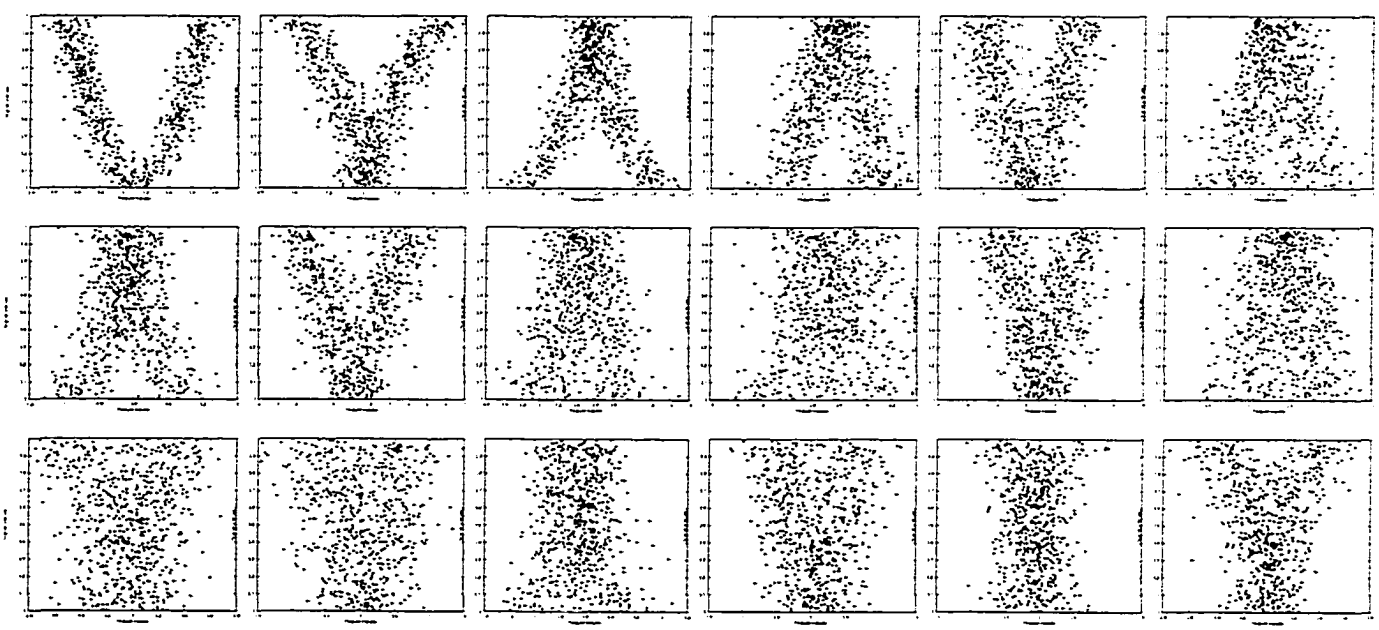


Figure 3.15: Remaining 18 features of 60 features (10 linear, 20 quadratic, 10 non-monotonic linear and 20 non-monotonic quadratic) ordered by using 9 classes and average of 20 hits/misses. Features are ordered left to right, top to bottom.

Relief uses a distance measure over the sample space to select "similar" hits and misses. If a data set is mainly formed by features irrelevant to the task, the distance measure will be determined by those features rather than the relevant ones. A badly degraded distance measure implies that the selected hit/miss instances are simply randomly selected instances from the same/different class. Therefore, the basic assumption that the hits are close to the current sample and the misses are distant from the current sample does not hold for datasets with a large numbers of irrelevant features.

Nevertheless, monotonic functions are much less affected than non-monotonic functions. Figure 3.16 shows the effect of increasingly degrading the distance measure. In this case, the distance measure is degraded by increasing the number of random features in the set applied to FARelief. Two different learning signal sets were used: 1) a synthetic signal which is uniformly distributed; 2) a real learning signal taken from one of the tasks used in the ADORE system. This signal is very non-uniform with 67% of the samples in class 0 when 9 classes are used.

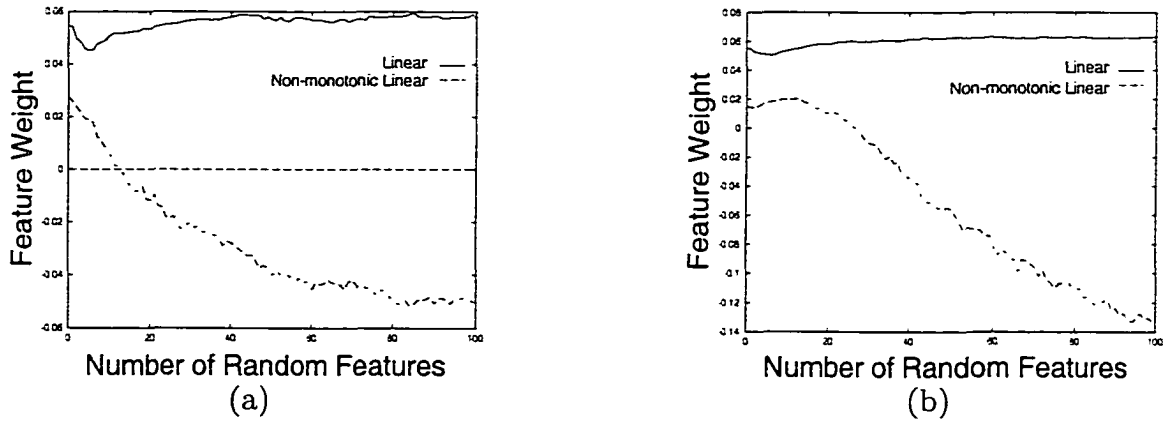



Figure 3.16: Relief score (weight) value for a linear and a non-monotonic linear feature for increasing number of random features. a) uniform learning signal; and b) non-uniform learning signal.

As can be seen, for the linear function the weight value computed by FARelief varies randomly but remains close to the initial value, which is computed without any random features. This occurs because, for monotonic functions, the maximum distance between two features of the same class is bounded by the size of the classes (see Figure 3.17). On the other hand, for non-monotonic functions, the maximum distance between two features of the same class is bounded only by the range of the function (see Figure 3.17). That makes the computed weight of these functions much more dependent on the quality of the distance measure. This is shown in the reduction of the weight seen in Figure 3.16. The distance degradation is even more crucial in our system, where thousands of features are expected and many, if not most, of them may be irrelevant.

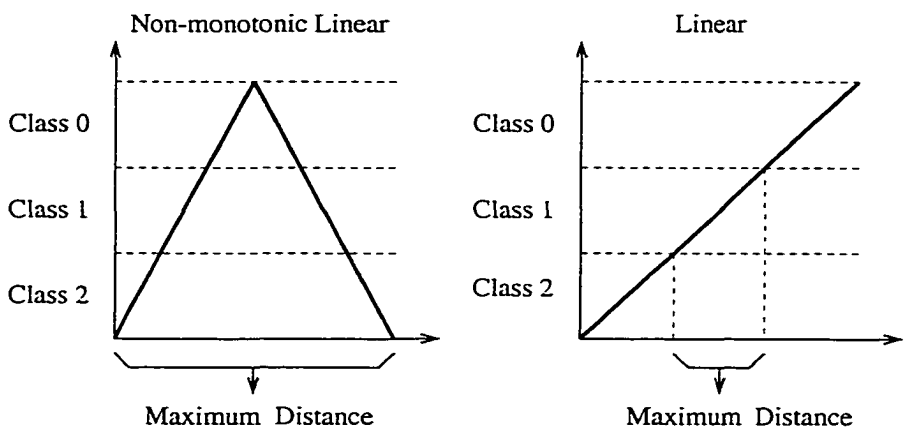



Figure 3.17: Maximum distance for non-monotonic linear and linear features using 3 classes.

The other main cause of bias is that apparently similar non-monotonic features may not correlate well with each other. In general, two linear functions computed using the same learning signal have correlation 1.0, except for noise. On the other hand, two non-monotonic linear functions computed over the same signal can have a very low correlation. An example of correlation of two non-monotonic linear functions

computed for the synthetic signal was 0.02256. This is due to the fact that the feature values may represent the same underlying function, but samples may be extracted from different sides of the distribution. Figure 3.18 illustrates the problem.

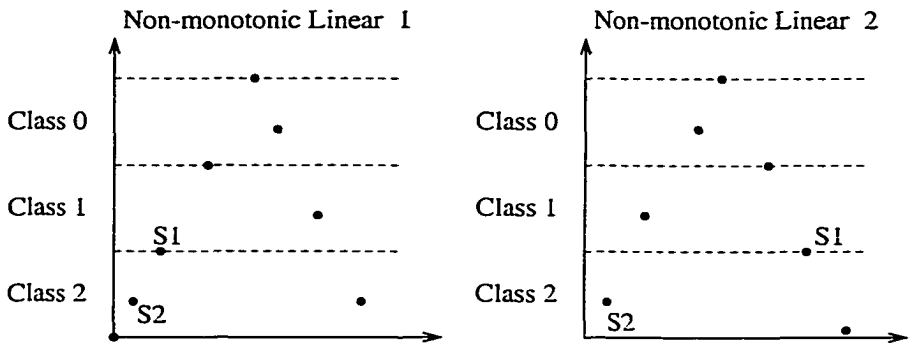



Figure 3.18: Two sets of points (samples) that represent the same underlying function but have poor correlation. Due to the discretization required by Relief.

In Figure 3.18, samples S1 and S2 of Feature 1 are close to each other, which indicates that one is a good hit for the other. On the other hand, for Feature 2, this is not true because S1 was selected on the other side of the distribution. The consequence of this behavior is that the distance measure is again degraded. Figure 3.19 shows that the weight computed for one non-monotonic linear feature reduces when more functions computed using the same signal and underlying function were included. This is probably the cause of the bias on Table 3.6 where no irrelevant features were included.

### 3.4.3 Eliminating the bias

To eliminate the bias against non-monotonic functions, three modifications of Relief are proposed: 1) identify the non-monotonic functions and compute the reflection point which divides the function into two distributions; 2) update each feature using only hits and misses that are on the same side of the reflection point; and 3) weigh differences between samples and their hits/misses using the reflection point.

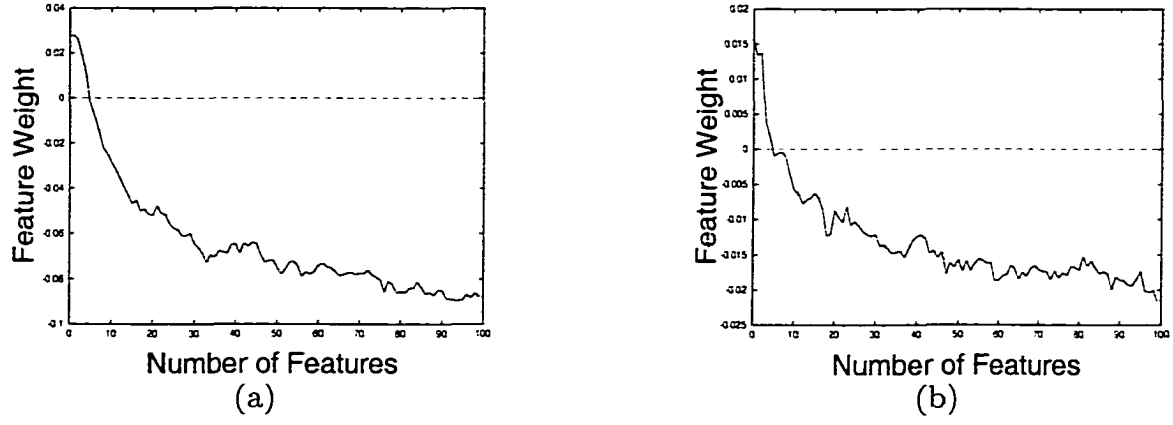



Figure 3.19: Weight value for a non-monotonic linear feature for increasing number of similar features. a) uniform learning signal; and b) non-uniform learning signal.

The idea is to treat each side of the distribution as one feature. That limits the distance between samples as in a monotonic feature (see Figure 3.17). Nevertheless, by doing so the distance is excessively limited because each side of the distribution, after normalization, can be much steeper than for a monotonic feature, and consequently the maximum distance is reduced. To correct this, each side of the distribution is normalized using the reflection point. This creates comparable max distances.

### Identifying peeks and valleys

We want to find if a function has one peak or valley (is non-monotonic) and the position of the peak/valley. The method used here is simple but is able to correctly identify the reflection point in many cases, even in the presence of noise. More general and complex methods to identify peaks and valleys in a function exist. Many have been proposed for the specific case of histograms (see [43], [58], and [16]) but many of those methods can be extended for more general functions. Our method does occasionally find peaks when none are present. However, the important point is not to miss the presence of a peak (false negative), or else the bias will remain. As will

be shown, inserting a false peak (false positive) has a smaller effect on the relevance score.

**Algorithm to identify peaks/valleys**

For each feature do:

- ▷ Sort the feature values
- ▷ Slide a *1* x $n$ window over the values and compute the sum
- ▷ Sort the sums
- ▷ Finds the first (FM) and last (LM) windows that are within a threshold ($th$) of the minimum sum
- ▷ Finds the first (FX) and last (LX) windows that are within a threshold ($th$) of the maximum sum
- ▷ If the order of the extreme points is :
  - ▷ LM FX : the function is increasing.
  - ▷ LX FM : the function is decreasing.
  - ▷ FM FX LX LM : the function is non-monotonic with a peak.
  - ▷ FX FM LM LX : the function is non-monotonic with a valley.
- ▷ If the function is non-monotonic, then the reflection point is the value of the central element of the central window between the two internal limits.

In the tests shown here $n = 21$ and $th = 0.15$.

### Updating weights

When updating the relevance weights, only hits and misses that are on the same side of the reflection point as the target sample are considered. This means that in some cases fewer hits and misses will be considered than specified by the hit/miss selection parameter (A: Average in Table 3.5) . In fact, in some extreme cases, some of the classes will not contribute any hits or misses. That implies that each class weight[5] must be recomputed for each sample.

### Weighting differences

Relief normalizes the differences between samples and hits and misses into the interval [0,1]. In our system this is not always necessary because the features are normalized before Relief is applied. Nevertheless, by using only hits and misses that are on the same side of the peak as the target sample our modified version of Relief reduces the maximum and average distances between samples for the non-monotonic features. To compensate for this, the hit and miss distances are divided by the size of the distribution in which they reside. In other words, if the sample is to the left of the reflection point the distances are divided by the square of the reflection point value; if the sample is to the right they are divided by the square of one minus the reflection point value.

## 3.4.4 Results of Bias Elimination

The modified algorithm is re-tested for bias, and the results are shown in Table 3.7 (this compares to Table 3.6). The maximum difference between the relevance of a feature and the relevance of its reflection is less than 0.009, or approximately 7%. When noisy features are used this difference is 0.025, or 27%. We also repeated

[5]For Kononenko's version of Relief these weights are the a-priori probability of each class. We normalize this weight using only the classes that effectively contribute with hits/misses.

| F | Non-refl. | | Left Ref. | | Right Ref. | |
|---|---|---|---|---|---|---|
| | Noise | | Noise | | Noise | |
| | No | Yes | No | Yes | No | Yes |
| L1 | 0.1636 | 0.1210 | 0.1630 | 0.1135 | 0.1615 | 0.1146 |
| L2 | 0.1636 | 0.1213 | 0.1615 | 0.1141 | 0.1634 | 0.1151 |
| Q1 | 0.1299 | 0.0991 | 0.1317 | 0.0801 | 0.1222 | 0.0694 |
| Q2 | 0.1271 | 0.0972 | 0.1181 | 0.0742 | 0.1291 | 0.0776 |
| Q3 | 0.1231 | 0.0920 | 0.1195 | 0.0738 | 0.1226 | 0.0730 |
| Q4 | 0.1224 | 0.0921 | 0.1221 | 0.0732 | 0.1187 | 0.0662 |
| G1 | 0.0666 | 0.0535 | 0.0642 | 0.0470 | 0.0640 | 0.0536 |
| G2 | 0.0666 | 0.0529 | 0.0637 | 0.0539 | 0.0642 | 0.0475 |

Table 3.7: Modified Relief score (weight) for 8 features and their reflections. Rows L1 and L2 correspond to linear functions; rows Q1 to Q4 to quadratic functions; and rows G1 and G2 to Gaussian functions.

the tests of Figures 3.16 and 3.19 with the modified algorithm, verifying that non-monotonic functions scores are no longer degraded by including irrelevant features or other non-monotonic features (see Figures 3.20 and 3.21).

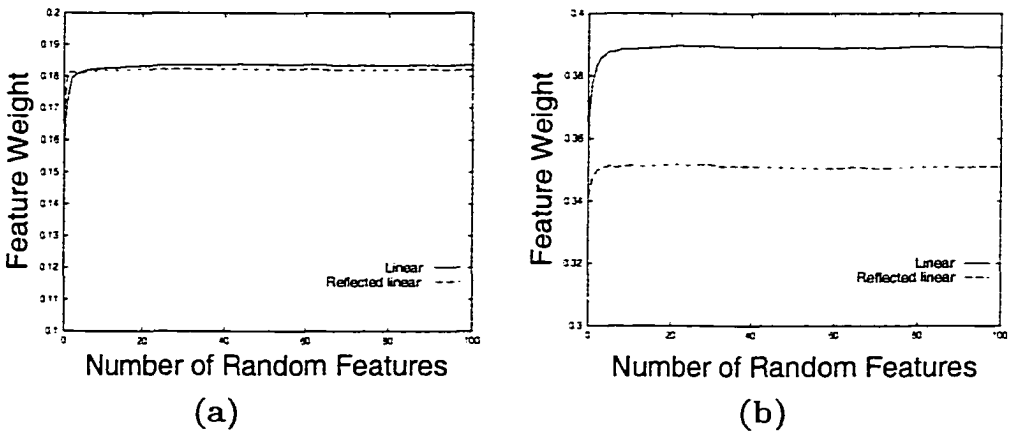



Figure 3.20: Modified Relief weight value for a linear and a non-monotonic linear feature for increasing number of random features. a) uniform learning signal; and b) non-uniform learning signal.

To test how sensitive the modified algorithm is to our ability to find peaks in functions, we assigned the monotonic functions false reflection points (located at 0.5). As can be seen from table 3.8, the functions are overestimated by amounts

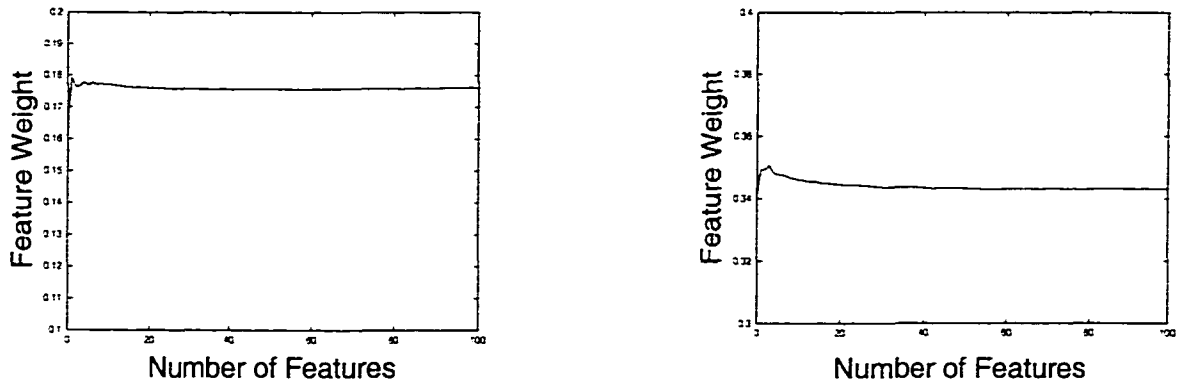



Figure 3.21: Modified Relief weight value for a non-monotonic linear feature for increasing number of similar features. a) uniform learning signal; and b) non-uniform learning signal.

| | L1 | L2 | Q1 | Q2 | Q3 | Q4 | G1 | G2 |
|---|---|---|---|---|---|---|---|---|
| monotonic | 0.1636 | 0.1636 | 0.1299 | 0.1271 | 0.1231 | 0.1224 | 0.0666 | 0.0666 |
| non-monotonic | 0.1927 | 0.1927 | 0.1842 | 0.1840 | 0.1723 | 0.1731 | 0.1222 | 0.11222 |

Table 3.8: Modified Relief score (weight) for 8 features when they are classified as non-monotonic or not. Columns L1 and L2 correspond to linear functions; columns Q1 to Q4 to quadratic functions; and columns G1 and G2 to Gaussian functions.

ranging from 20% (linear functions) to 60% (Gaussian functions). This represents a bias in favor of monotonic features misinterpreted as non-monotonic features, but the bias is much less than the previous bias against non-monotonic features (which was in excess of 5,000%). As a result, even if every monotonic feature is assigned an erroneous reflection point, the bias is still less than for other versions of Relief.

For real data it is difficult to find features which are non-monotonic exactly on the center of the feature range. Moreover, irregularly centered features can further test the limits of the algorithm. Consequently, as a second test, features with reflection points not at the center of the feature range were created and tested. Example of the features are shown in Figure 3.22. The results are close to the centered reflections, as shown in Table 3.9.

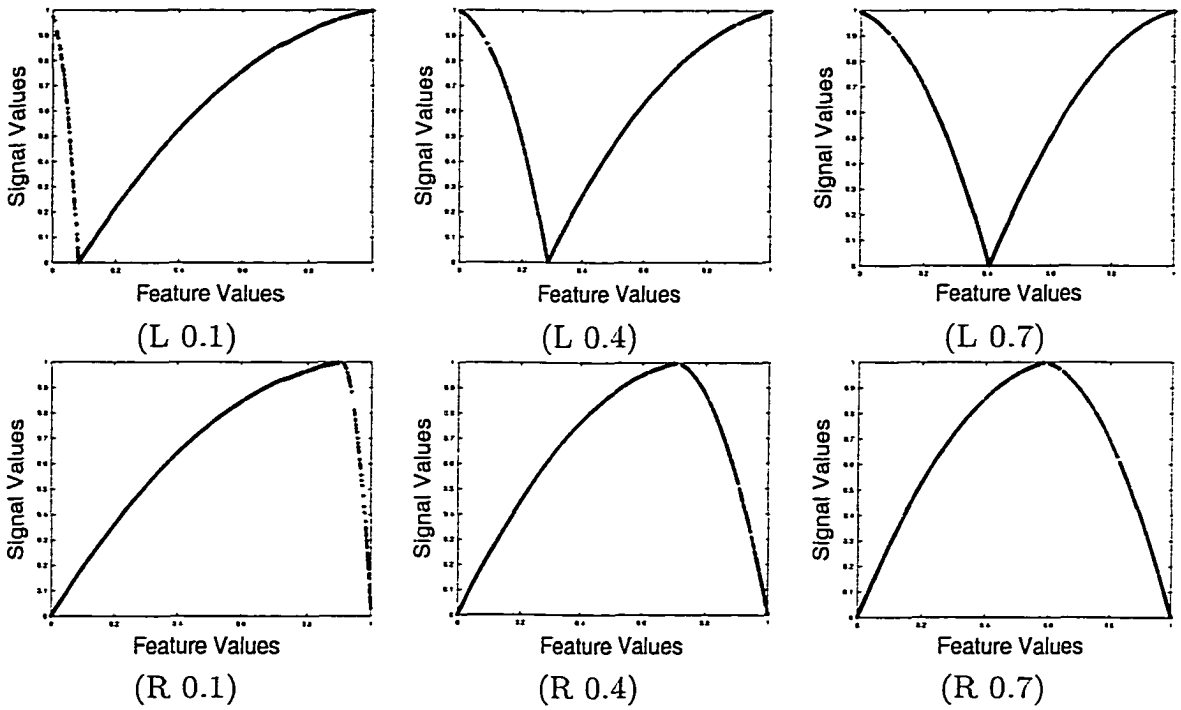



Figure 3.22: Synthetic Features: Tilted features. The value below each graph specifies the type of reflection (left or right) and the amount of tilting.

The modified algorithm was able to eliminate most of the bias against non-monotonic features. Visual confirmation is given by Figures 3.23 and 3.24, where the non-monotonic features and monotonic features are approximately ranked by their noise. The bias against non-monotonic functions is no longer present.

## 3.5 Real data

### 3.5.1 ADORE Data

There are a lot of data sets used in the literature to test feature selection. Unfortunately, few data sets have the necessary characteristics sought in this work. Therefore, to test Relief and its modification on real data we used data from the ADORE system [35].

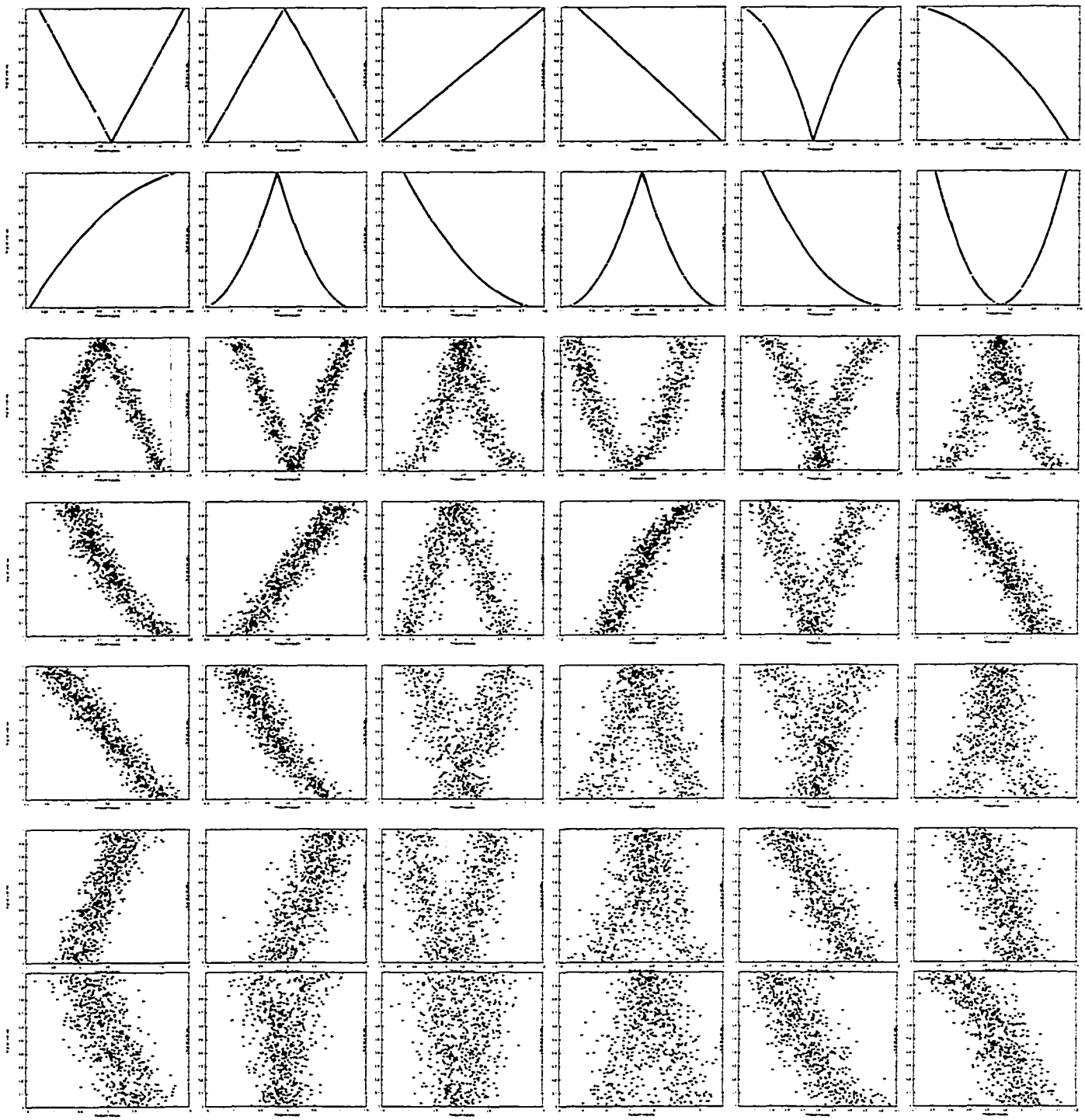


Figure 3.23: First 42 features of 60 (10 linear, 20 quadratic, 10 non-monotonic linear, and 20 non-monotonic quadratic) ordered, left to right, top to bottom, by using 9 classes and average of 20 hits/misses on the modified algorithm.

| | Left Reflections | | | Right Reflections | | |
|---|---|---|---|---|---|---|
| F | 0.1 | 0.4 | 0.7 | 0.1 | 0.4 | 0.7 |
| L1 | 0.1564 | 0.1675 | 0.1655 | 0.1272 | 0.1673 | 0.1641 |
| L2 | 0.1303 | 0.1678 | 0.1641 | 0.1518 | 0.1673 | 0.1658 |
| Q1 | 0.1122 | 0.1328 | 0.1329 | 0.1035 | 0.1465 | 0.1314 |
| Q2 | 0.1081 | 0.1467 | 0.1293 | 0.1194 | 0.1318 | 0.1313 |
| Q3 | 0.1359 | 0.1407 | 0.1289 | 0.1048 | 0.1286 | 0.1273 |
| Q4 | 0.0877 | 0.1256 | 0.1240 | 0.1249 | 0.1389 | 0.1252 |
| G1 | 0.0950 | 0.1068 | 0.0826 | 0.0590 | 0.0739 | 0.0695 |
| G2 | 0.0575 | 0.0740 | 0.0683 | 0.0988 | 0.1059 | 0.0815 |

Table 3.9: Modified Relief score (weight) for 8 features and their reflections. Reflection points are not centered. The numbers in the title line shows how much the non-monotonic side of the functions was compressed. Rows L1 and L2 correspond to linear functions; rows Q1 to Q4 to quadratic functions; and rows G1 and G2 to Gaussian functions.

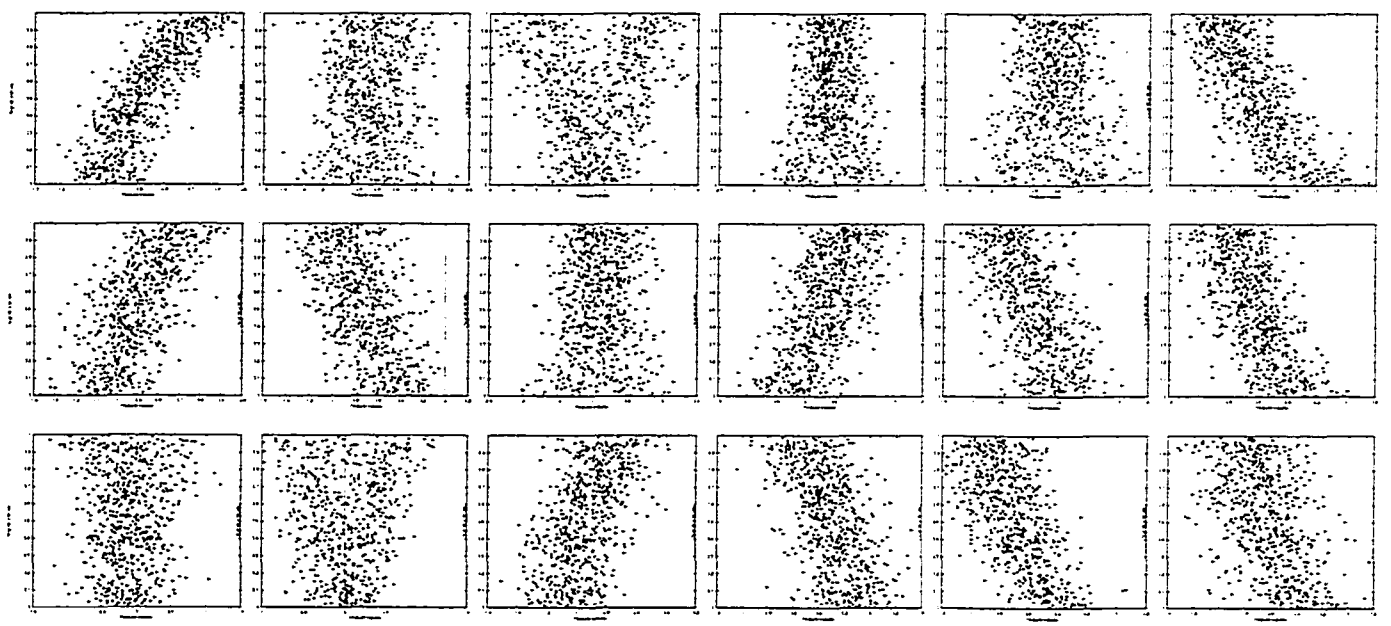


Figure 3.24: Remaining 18 features of 60 features (10 linear, 20 quadratic, 10 non-monotonic linear and 20 non-monotonic quadratic) ordered by using 9 classes and average of 20 hits/misses on the modified algorithm. Features are ordered left to right, top to bottom.

The ADORE data set is a set of aerial images of Fort Hood, Tx, collected by the U.S. Army Topological Engineering Center. There are 20 black and white images, each from a nadir[6] view perspective. Each image is 7000 x 7000 pixels. To test the ADORE system, one of these images was selected and divided in eight pieces (tiles) of size less than or equal to 512 x 512 pixels. Figure 3.25 shows one of these tiles.

Figure 3.25: Example of tile of ADORE data, Fort Hood, Tx. (half size).

The images contain houses of five different styles. Each house in each tile was identified and their pixels manually marked. Figure 3.26 show the same tile as Figure 3.25 with the true position of one style of houses marked.

The ground-truth image in Figure 3.26 and the corresponding ground-truth images for the other four styles of houses allow the computation of the recognition error. For each tile, 20 regions of interest (ROI) are computed using cross correlation. Each of these ROIs is associated with a model of each house and used as data. Figure 3.27 shows two examples of ROIs. These ROIs are submited to several operators that modify the hypotheses to better match positions and shapes of real objects.

[6]Vertically downward from the observer.

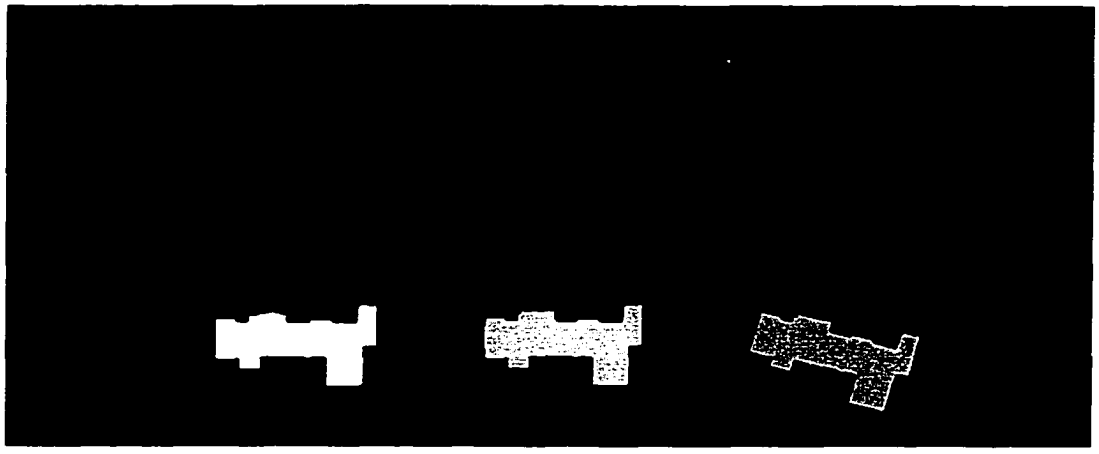


Figure 3.26: Example of truth tile for style duplex of ADORE data (half size).

In this way, a total of 800 basic hypotheses (8 tiles x 5 styles x 20 ROIs) are created, plus a variable number of modified hypotheses. Each hypothesis is compared to the truth tiles and a score is computed. This score is $\frac{|H \bigcap T|}{|H \bigcup T|}$, where $H$ is the set of pixels that defines the hypothesized object (in red on Figure 3.27) and $T$ is the set of pixels that defines the true position of the model of the object in the tile (in white on Figure 3.26). The features used on the Relief tests were created by computing 1069 features for each hypothesis for one of the styles. A total of 630 samples (160 basic hypothesis + 470 modified hypothesis) were generated.

### 3.5.2 Tests

When testing on real data, the true relevance of features is not known a-priori. This eliminates the methodology used to evaluate the algorithm on synthetic features. Instead, we tested the quality of the resulting feature ranking through random sampling. In particular, we randomly selected 300 features and trained function approximators for each one to infer the signal from the feature value. Back-propagation neural networks were used since they may be considered as representative of non-

Figure 3.27: Examples of ROIs. A positive (left) and a negative (right) example for style duplex are shown. To facilitate visualization, each hypothesis is overlaid over the ROI chip.

parametric function approximators. We then selected another feature whose relevance score was better (or worse) than the original sample by a fixed amount, and trained a second neural net using that feature. We then compared the mean squared errors of the nets to determine which feature was more relevant.

Since back-propagation with random initialization is nondeterministic, this is a highly noisy evaluation mechanism. It also evaluates features in isolation from each other, so dependency between features are ignored. Nonetheless, when two features have a difference in relevance scores (as computed by modified Relief) of 0.00 to 0.01, the higher scoring feature produces a better neural net 55.3% of the time; when the score difference is between 0.01 and 0.02, the higher ranking features produces a better neural net 61% of the time; and 94%, for score differences between 0.08 and 0.09. While this is an admittedly imperfect evaluation, the binomial test suggests that a difference in Relief relevance score of 0.00-0.01 is significant with a probability of 0.964, while a difference in score of 0.01-0.02 is significant with a probability greater than 0.9999.

Interestingly, the best feature found for the ADORE data using modified Relief was a non-monotonic feature not selected using unmodified Relief (see Figure 3.28

to some of the features selected by modified Relief), and not used by the authors of ADORE. Hopefully, identifying this feature will improve the performance of that system. A new version of ADORE which incorporates the three-step feature selection algorithm proposed in this work is being planned . This will allow the comparison of the performance of ADORE with and without the modification here proposed.

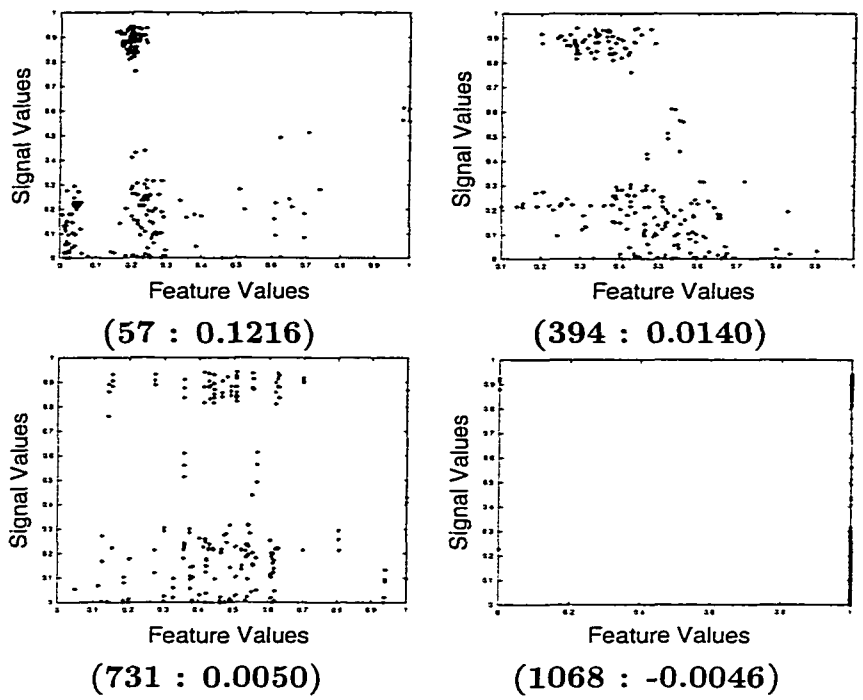



Figure 3.28: Examples of real features. The numbers below the feature represents the order of ranking and the Relief score.

## 3.6 Conclusion

The literature has claims like: "[Relief] is able to detect feature relevance even when features interact" [23]. This was one of the main reasons we selected Relief, even though most of the tests that led to this result use boolean features and assume classification as the learning task. Nevertheless, our tests on real features using function approximation as the learning task seems to support that claim. Moreover,

we have identified two sources of bias in Relief against non-monotonic features, and we have proposed a simple modification to Relief that eliminates these sources of bias. It should be noted that we have not removed all sources of bias; Relief still prefers simple functions (first and second order) to more complex functions (e.g. Gaussian; see Table 3.7). Partly for this reason, we do not even consider multi-peaked functions (Rendell and Seshu (1990) do consider such functions in the context of feature construction). Given the extent of noise in our data, we do not think we could identify such functions anyway. Instead, we propose a simple modification to Relief that extends its capabilities from evaluating strictly increasing/decreasing functions to evaluating functions with a single peak or valley. We also confirm the efficiency of Kononenko's modifications of Relief, and selected the parameters of Relief to be used for the rest of this work.

Chapter 4

# REDUNDANCY

## 4.1 Introduction

Clustering is an unsupervised learning technique used to divide data sets into groups or clusters. These clusters can be viewed as a group of elements which are more similar to each other than elements belonging to other groups. An alternative definition of a cluster is "a region with a relatively high density of points, separated from other clusters by a region with a relatively low density of points" [49].

In our system, clustering has an unusual objective and use. First, it clusters features instead of clustering samples. Second, clustering is used as a means to find redundancy instead of as a means of classification. The goal is to eliminate redundant features from the set of candidate features, prior to the final selection stage. What is at stake here is the feasibility of performing a combinatorial feature selection algorithm at the next step of the system, which depends of the number of features and hence the number of final clusters.

It is important that reducing the size of the candidate set does not reduce the performance of the final feature set by removing informative features. The decrease in performance as a result of removing features, if present, must be small. The next sections will explain the consequences of our unusual objectives on the clustering algorithm.

## 4.2 Redundancy Definition and Test

### 4.2.1 Introduction

The first step in adapting clustering algorithms for use as redundancy filter is to define what is meant by redundancy. The intuitive idea is that two features are redundant if adding one feature will not improve the performance of any function approximation algorithm that already has access to the other feature. Note that this wrapper-like definition is impractical for our purposes since it can only be checked by training function approximators for every combination of features. This defies our purpose of removing the redundant features before feature selection occurs.

There is no way to predict with certainty whether an unspecified function approximation technique can take advantage of the minor differences between two features, and therefore whether two features are effectively redundant. Nevertheless, there are many ways to define redundancy measures that will approximate the concept. Two such definitions are: 1) the amount of information of a feature; and 2) correlation.

The amount of information can be computed by Shannon's entropy measure. However, it is difficult to see how this can be used in a clustering algorithm. The difference in the amount of information between two features states how good the features are, not how similar, and so could not be used as a redundancy measure. Alternatively, one could compute Shanon's information measure for every pair of features and compare that to the information content of individual features. If the information for the pair of features is no greater than the component features then the features are redundant. The problem is that this measure can only be computed between features and not between a feature and a cluster center, because there is no way to compute the class probabilities for the center of the cluster. Consequently, no clustering mechanism that uses a cluster center can be used. Moreover, Shanon's

entropy has the disadvantage that it needs the feature probabilities, which can not be computed from real values. This means that the values need to be discretized, and this introduces one more parameter. In light of all this, we decided to not attempt to use entropy as a distance measure when clustering.

Instead, we use correlation to measure redundancy between features. Correlation, although not a good indicator of general dependence, does recognize linear dependence [63]. Therefore, we define redundancy in terms of correlation. Hall [41] recently also uses the idea of correlation as a measure of redundancy, but instead of using it as distance measure in a clustering algorithm, he includes it in the weight update procedure of an algorithm similar to Relief.

If two features correlate completely (correlation score equals 1), then their values are exactly the same and at most one of them needs to be included in the final set. However, if two features correlate well but not completely, then they may or may not be included in the final feature set, depending on where the difference occurs and whether this difference is relevant to the task. Figure 4.1 shows two pathological cases. Features A and B have all points coincident except the one of class 2, hence they correlate well. However, while Feature A is able to linearly separate the two classes, Feature B is not. At the same time, features A and C have no coincident points and will not correlate well. Nevertheless, the features still separate the classes similarly.

Although figure 4.1 shows that even well-correlated features can yield very different results, these are pathological cases and should be rare. Moreover, intuitively, the better distributed the data is between classes, the smaller the probability that well correlated features have significantly different results.

### 4.2.2 Clustering features

Standard clustering algorithms are used to cluster points in a sample space, where each point is a series of measurements or features. In this case, the distance

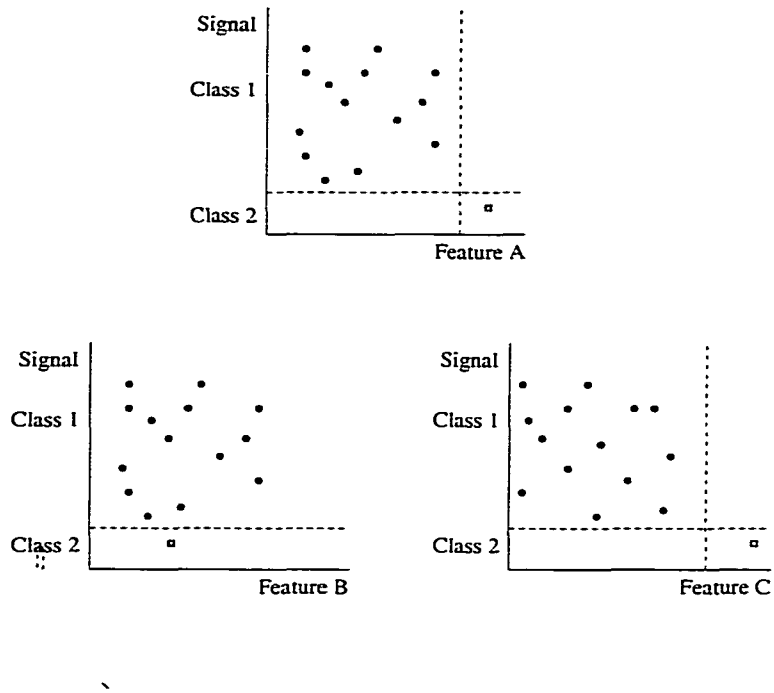



Figure 4.1: Two pathologic cases: Feature A and B have good correlation but have very different information about the task. Features A and C do not have good correlation but have essentially the same information about the task.

between two points has a clear intuitive meaning. For example, if all houses in a block are measured in terms of the number of rooms, number of bathrooms, inside area, yard area, etc, the distance measure gives an idea of how different they are, though not an exact explanation of which features they differ on. However, when features are clustered, the notion of distance is different. For example, what is the meaning of the difference between the number of rooms and the yard area of houses? There may be a relation between these two variables, but this relation is much less obvious than when clustering samples. For example, houses with bigger yard area may have fewer rooms, if the total land area is the same. Now the distance measure indicates how likely it is that a house with similar room count will have similar yard area.

The use of correlation as distance measure between features not only gives a measure of redundancy but it gives an interpretation for the distance between two

features. So the distance between two features is now a measure of how much they vary together.

### 4.2.3 Use of clustering

In our system, clustering eliminates features so that combinatorial feature selection algorithms become practical. Most feature selection algorithms are practical when applied to on the order of one hundred features. The more features are eliminated, the faster the feature selection algorithm will run. Of course, features that might improve the prediction function should not be eliminated, since the final objective of the system is to select the feature set that will best predict the target function. So, the goal is to maximize the number of redundant features clustered together, without clustering non-redundant features together. This brings a different approach to clustering. The important aspect is not how many clusters are created, but the maximum distance between the elements of the cluster and the cluster center (radius). Here, as in standard clustering algorithms, clusters with large number of elements (size) are desirable, but unlike standard clustering algorithms, this size is subordinated to the radius of the cluster. This means that what drives the algorithm is the maximum radius allowed, not the number of clusters to be created. Instead of fixing the number of clusters to be found and accepting clusters of any size, the maximum cluster radius is fixed and as many clusters as necessary to cover the whole feature space are accepted.

## 4.3 K-means

Recent studies in cluster analysis suggest that : 1) every clustering algorithm will find clusters in a data set whether they exist or not; and 2) there is no "best" clustering algorithm ([49]). That means that the characteristics of the dataset must be known before-hand if an optimal clustering algorithm and, in many cases, the

number of clusters is to be selected. In our case, we want a general clustering algorithm. Consequently we know beforehand that it will not be optimal for every data set. This suggests that our search for a clustering algorithm should concentrate in finding algorithms that perform well over a wide range of datasets.

Two algorithms that have been applied with good results for a wide variety of clustering problems [36] are K-means ([66], [28], [36] and [51]) and Expectation-Maximization (EM) ([31] and [14]). Both algorithms are quite similar to each other, and execute a two step process. The fist step assigns data points to clusters. The second estimates the density model for each data point/cluster instance. The difference between the algorithms is that k-means makes a hard assignment at step one while EM makes a soft one. Formally, given data points $x = \{x_0, ..., x_n\}$, clusters $\gamma_0, ..., \gamma_k$, and distribution models $f_{\gamma_j}(x_i)$, the first step of both algorithms assigns a vector to each point $A(x_i) = \{A_{\gamma_0}(x_i), ..., A_{\gamma_k}(x_i)\}$. In k-means, $A(x_i)$ is defined as:

$$A_{\gamma_j}(x_i) = \begin{cases} 1 & for\ j = \{y \mid \forall z : f_{\gamma_y}(x_i) >= f_{\gamma_z}(x_i)\} \quad (1) \\ 0 & otherwise \quad (2) \end{cases}$$

If more than one j satisfies condition (1) only one of them can be selected and assigned to 1. In EM, $A(x_i)$ is defined as:

$$A_{\gamma_j}(x_i) = P(\gamma_j | x_i)$$

K-means, therefore, can be classified as a Winner-Takes-All (WTA) version of EM. Many attempts have been made to improve K-means, but many of these so-called enhancements are computationally demanding and require additional user-specified parameters for which no guidelines are available [49]. In view of this and because there is no general solution for the problem and k-means have been shown

to perform well to a variety of problems, we decided to use standard k-means as our clustering algorithm.

It is important to prepare the algorithm for the consequences of clustering features to detect redundancy. We modified K-means to use correlation as the distance measure, and to enforce a threshold on the minimum correlation between two features considered as redundant (see section 4.3.3). Other issues that have to be addressed are: initialization of the clusters; selection of a representative for each cluster; and anti-correlation.

### 4.3.1 K-means algorithm

The standard k-means algorithm is as follows:

- Select an initial set of K cluster centers.
- While the criterion function can be improved
  - Assign each observation (sample) to its closest cluster center.
  - Compute new cluster centers.

On-line versions of K-means are similar, except that instead of computing a two-step assignment and update process, cluster centers are modified each time a sample is assigned. How much a cluster center is modified is defined by a learning rate.

### 4.3.2 Correlation as distance measure

The K-means algorithm must be modified to use correlation as the distance measure. Standard K-means uses the average of the samples assigned to a cluster as the center of the cluster. That is the point where the sum of all distances between the center and the sample's measurements is smallest. In particular, when only two samples are used, the center is equidistant from both samples.

When correlation is used as the distance measure, the average of the features values will not create a cluster center where the sum of the distances is minimized. For example, given three features A, B and $C = \frac{A+B}{2}$, the correlation between A and C is not likely to be the same as the correlation between B and C. The center in terms of Euclidean distance between two vectors is not the center in terms of correlation, and therefore the center does not represent the cluster. The distance between center and features has lost its meaning. To address this problem, a new method for computing clustering centers, in terms of correlation, is defined. We prove that for two features it performs as expected. The proofs for the theorems are fully developed in Appendix D.

**Theorems:**

Theorem 1: If A and B are random variables, and $C = \sigma(A)\ B + \sigma(B)\ A$, where $\sigma(A)$ is the standard deviation of A, then the correlation between A and C is equal to the correlation between B and C.

Lemma 1: If A and B are random variables, and $C = \sigma(A)\ B + \sigma(B)\ A$, and x is a scalar, then any D of the form $D = x\ \ \sigma(A)\ B + x\ \sigma(B)\ A$ has the same correlation with A and B as C does. In other words, $Corr(A, C) = Corr(B, C) = Corr(A, D) = Corr(B, D)$.

Theorem 2: Given random variables A and B and scalars x and y, any D of the form $D = x\ \ \sigma(A)\ B + y\ \sigma(B)\ A$ has a smaller or equal average correlation with A and B than $C = \sigma(A)\ B + \sigma(B)\ A$.

Theorem 1 allows for the calculation of a cluster center for two features in such a way that the center is equidistant to both features. It can be expanded for more than two variables (features) by:

$$C = \sum_{i=1}^{n} \frac{\prod_{j=1}^{n} \sigma(V_j) \quad V_i}{\sigma(V_i)}$$

Of course, for more than two variables, it is unlikely that all the correlations between the center and the features will be the same, but the same is true for Euclidean distance. Nevertheless, unlike Euclidean distance, correlation does not follow the triangle inequality property.

There is also no guarantee that C will be close in terms of individual values with the features. In fact, when the number of features is large, we can have an underflow problem. To prevent underflow we normalize C. That can be done because of the property that multiplying a random variable by any value other than zero does not affect its correlation with other variables (see B2 in Appendix D). So the final expression for C is:

$$C = \frac{\sum_{i=1}^{n} \frac{\prod_{j=1}^{n} \sigma(V_j) \quad V_i}{\sigma(V_i)}}{\sum_{i=1}^{n} \frac{\prod_{j=1}^{n} \sigma(V_j)}{\sigma(V_i)}} \tag{4.1}$$

Lemma 1 states that multiplying the weights of each variable by the same scalar does not change the final result. Hence, the only way to change the result is by multiplying by different scalars. Theorem 2 proves that this will yield a "center" for which average correlation with the features will be smaller than the ones given by Theorem 1, and we want to maximize the correlation. Consequently, the best solution is equation 4.1.

### 4.3.3 Correlation Threshold

Most k-means implementations require the number of clusters to be predetermined. Their goal is to have clusters that represent the natural classes within the data. This implies a good knowledge of the data distribution and a good initialization algorithm. Recent work in initialization algorithms is presented in [18]. Our purpose when using clustering is very different, however. It is important that no clusters have features that differ greatly from each other, rather than the clusters represent natural classes in the data. So, instead of guiding the algorithm by the number of possible classes in the data, the algorithm is guided by a distance threshold between prototypes and members of a cluster. As a secondary advantage, this threshold is much less dependent upon the data and its distribution. In Table 4.1, Section 4.4, it is shown that for correlation intervals over [0.94,0.95) using a non-redundant feature is better than using a redundant one. So, we fixed our threshold at 0.97. This means that all elements in a cluster will have a maximum correlation difference of 0.03 with the center of the cluster.

This difference in focus has other implication for the algorithm. In this case the algorithm is started with a small number of clusters and more clusters are added as needed. The refined algorithm is now:

- Select an initial set of K cluster centers.
- While the criterion function can be improved
  - While clusters are not stable
    - Assign each observation (sample) to its closest cluster center.
    - Compute new cluster centers.
  - Divide clusters whose correlation distance are over the threshold

The difference is that the clusters are divided after the clusters stabilize. A cluster is stable if no feature will change clusters no matter how many interations the algorithm is run. One way to do that is to check if no cluster center has changed. In practice, it is sufficient to allow the algorithm to run for some predefined number of iterations before dividing the clusters. Pelleg and Moore [80] also use splitting of clusters as a way to find the real value of K, although their work uses a Bayesian information criteriun and has different goals from ours.

### 4.3.4 Initialization

Although the main reason for our study on clustering is not to find a good initilization procedure, the results obtained with the standard method, random initialization, were very poor, so we studied alternative methods. Bradley and Fayyad [18] propose a refinement process to find the initial points or centers for k-means. Basically, their method selects a small number of initial random sets by subsampling the data space. These sets are used to cluster the data. The initial points are the average of the centers that are close together. This method has proven to work better than random initialization [18]. Nevertheless, this method tries to find the centers of natural classes in the data. It does not take into account the radius of the cluster, which is the most important issue for our application. Instead, we are looking for clusters that are far apart from each other and have small radii. One method to find those centers, which we called threshold initialization, is to get random centers that are at least a mininum distance (threshold) from each other. Another method, called farthest initialization, is to select one center randomly and iteratively select the farthest points from the previous centers in the data space. Most applications of K-means do not use these methods because they have a tendency to select isolated points as center of clusters. This is not a problem in our case. Those points will

probably be clusters of one element only, which is the desired behaviour for isolated points in our use of clustering.

To test which of these methods creates a better clustering, we run clustering many times with both initialization techniques, theshold and farthest. Each cluster initialization method was run with an initial number of clusters ranging from 2 to 80. The features clustered were the 300 best features selected by Relief from the 891 features of the ADORE data set. Two parameters were relevant to the test, the maximun distance within a cluster and the minimum distance between clusters. The maximum distance within a cluster is used to divide large clusters, and was set to 0.03. Meaning that the correlation between each data point and its cluster center can not be less than 0.97. The minimum distance between clusters is used during theshold initialization to eliminate cluster centers that are too close to other centers. If no center passes the threshold test, then the next choice is to eliminate clusters that are too close to all other clusters, except for one, and so on. The minimum distance between clusters is 0.2. Figures 4.2, 4.3 and 4.4 show comparisons of, respectevely, number of clusters created, average distance within clusters, and number of features per cluster.

In Figure 4.2 each line shows the number of clusters generated for threshold (TI) and farthest (FI) initialization methods applied from 2 to 80 initial centers. Figure 4.3 shows the average cluster distance and the maximum distance for each of the runs. The average cluster distance is the average distance between each feature and its cluster center and the maximum distance is the maximum distance between each feature and its cluster center. Figure 4.4 shows the average number of features per cluster and the number of features of the biggest cluster.

As can be seen, the FI initialization method creates a stable number of clusters as the number of initial centers grows, while the number of clusters created by the threshold initialization grows almost exponentially. This means that clusters that are

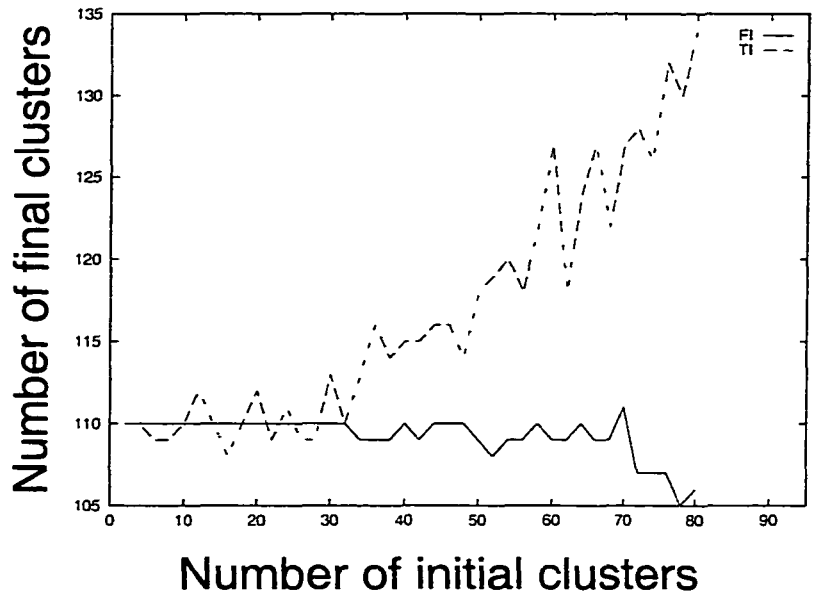



Figure 4.2: Number of clusters generated using threshold initialization (TI) and farthest initialization (FI) from 2 to 80 initial centers.

not needed are being generated and in consequence features that should be clustered together are being clustered apart. More important, the average cluster distance and average number of features in the FI initialization remains basically the same as the ones for TI initialization. The maximum distance and maximum number of features are better for the TI initialization than the FI initialization in most cases, but that is at the expense of creating many more clusters and therefore is not a good solution.

## 4.4 Redundancy test.

Our modified version of k-means cluster features according to how well they correlate, with the idea that only one feature from each cluster will be passed to the final selection filter. This design is based on the intuition that it is better to include two distinct features (non-correlated), even if one is less relevant than the other, than two nearly identical features. The goal of this section is to test this intuition.

In particular, our informal hypothesis is that two redundant features, A and B, should be less effective at predicting the target than two non redundant features

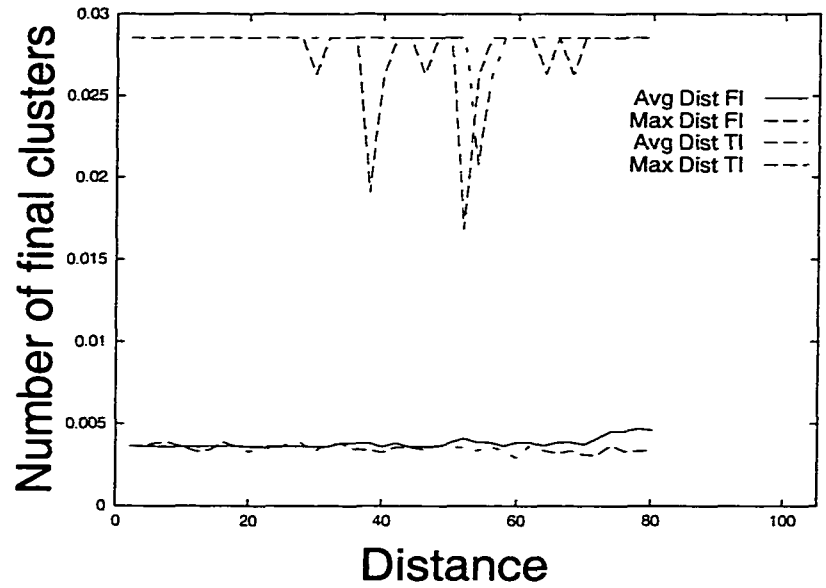



Figure 4.3: Average and maximum distance within cluster for threshold Initialization (TI) and farthest Initialization (FI) from 2 to 80 initial centers.

A and C. Moreover, we predict that this should be true even if C, by itself, is less relevant than B.

To test this, we select correlation intervals ranging from [0.93, 0.94) to [0.99, 1.00). For each of these intervals, 300 sets of three features $\{A, B, C\}$ are randomly selected, with the following characteristics: 1) Features A and B have correlation within the current correlation interval; 2) Features A and C have a low correlation, between [-0.3, 0.3]; and 3) Relief score for Feature C is between [0.01, 0.02] less than Relief score for Feature B[1]. We then create neural nets for feature sets $\{A, B\}$ and $\{A, C\}$. Our hypothesis is that $\{A, C\}$ should be better than $\{A, B\}$, even though B is better than C. The data used was the ADORE data described in section 3.5 and the specific steps followed are:

- Normalize the data.
- Run Relief over the data and save the score for each feature.

[1]In section 3.5 it was demonstrated that a range between 0.01 and 0.02 is statistically relevant

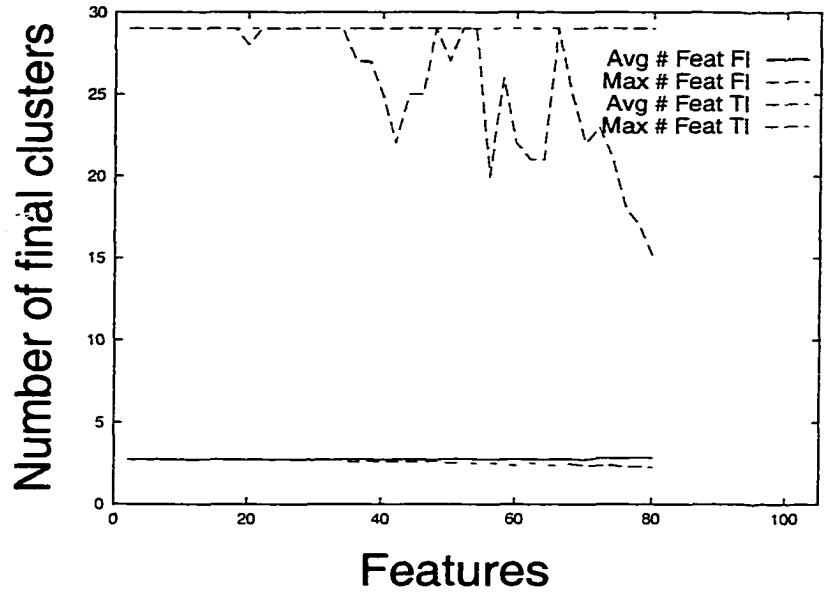



Figure 4.4: Average number of features per cluster and number of features of the biggest cluster generated for threshold Initialization (TI) and farthest Initialization (FI) from 2 to 80 initial centers.

- Compute and save the pair-wise correlation scores between the features.
- For every correlation interval tested:
  - Create all possible groups of 3 features where:
    - Feature A: Original feature
    - Feature B: Randomly selected feature with correlation with A inside the interval being tested.
    - Feature C: Random selected feature among the ones with correlation between [-0,3 0.3] to A and with a Relief score between [0.01, 0.02] less than B.
  - For m of the groups.
    - Create neural net for set formed by features A and B.
    - Create neural net for set formed by features A and C.

- Compare the Mean-Square Error (MSE) of both nets. Consider a success if MSE of A and C is better than MSE of A and B.
    - Count the number of successes.
- Analyze the results of the intervals.

The results were checked using the binomial test (see section 3.5). Here, the test is whether the new feature, C, will have the same performance as Feature B. Based on the intuitive notion of redundancy, if Features A and B are redundant, they should perform worse than Features A and C. So, this test tries to prove that using a less informative but not redundant feature is a good idea. If B and C can be switched without affecting the quality of the set, then there is a probability of 0.5 that the MSE of $\{A, C\}$ will be better than the MSE of $\{A, B\}$. So, we want to prove that the probability of $\{A, C\}$ yielding a better MSE than $\{A, B\}$ is greater than 0.5. Any statistical variation from 0.5 is assumed to be caused by difference in performance of the new feature (C). This, can not be tested directly. However, we can test if the probability of success, where success is defined as MSE of $\{A, C\}$ is better than the MSE of $\{A, B\}$, is equal to 0.5. The null hypothesis is then: $H_0$: the new feature set $\{A, C\}$ has a probability of 0.5 of success ($\alpha = 0.05$). By rejecting the Null hypothesis we are indirectly showing that to use a less informative but non-redundant feature is better than to use a more informative but redundant feature. The results are shown on Table 4.1.

As can be seen, the probability (1 - P) that Feature C is a better option than Feature B is strong for correlation values from 0.94 to 1, despite the fact that C is a less relevant feature than than B. For most of the intervals, the Null hypothesis is rejected ($H_0 = F$) even using a strict $\alpha$ as 0.005. Another test using a larger interval [0.94,1.0) had 63% of successes and a $P = 3.9e-06$. This indicates that well correlated features are mostly redundant, which agrees with intuition.

| | Correlation Intervals | | | | | | |
|---|---|---|---|---|---|---|---|
| | [0.99,1.0) | [0.98, 0.99) | [0.97, 0.98) | [0.96, 0.97) | [0.95, 0.96) | [0.94, 0.95) | [0.93, 0.94) |
| % S | 66.0 | 62.0 | 57.6 | 56.6 | 61.0 | 58.3 | 56.6 |
| P | 1.5e-08 | 1.9e-05 | 4.6e-03 | 1.2e-02 | 8.2e-05 | 2.2e-03 | 1.2e-02 |
| $H_0$ | F | F | F | F | F | F | F |

Table 4.1: Percentage of successes (% S), probability of observing the given result by chance given that the null hypothesis is true, and test result ($H_0$). Null hypothesis: the new feature set $\{A, C\}$ has a probability of 0.5 of success ($\alpha = 0.05$), meaning that using a less informative but non-redundant feature is equivalent to use a more informative but redundant feature.

## 4.5 Conclusion

In this chapter we addressed the problem of clustering features. Although, it may seem that clustering features is the same as clustering samples, with the exception that the space of features is the transpose of the space of samples, this is not true. The overall goal is the same, to cluster data, but the specific goals are very different. For example, in clustering samples one normally wants to find the natural classes of samples in the data, while when clustering features we want to cluster redundant features. This has consequences for the clustering algorithm.

First, we deal with the modifications on the k-means algorithm to support our goals and the definition of redundance by correlation. Here some theorems were proved that allow us to use correlation as distance measure. Next, some initialization schemes were presented and tested to see which gives a better clustering according with our goals. Finally, we show that it is preferable to include a less relevant but non-redundant feature than a more relevant but redundant one. This leads to the removal of redundant features.

Chapter 5

# SYSTEM VALIDATION

## 5.1 Introduction

There are two distinct problems that make it difficult to test the proposed three-step system for selecting features. The first is the data. There is no publicly available data set in the literature that has the same characteristics as the data used in this work, and most systems described in the literature are not able to handle this kind of data. This limits comparisons between systems proposed in the literature and this one. The second problem is the large number of parameters. The performance of each module depends on many parameters. Testing all combinations of parameters is impractical.

Although the first problem can not be completely solved, we were able to find a data set with some, but not all, of the characteristics assumed in our system. We are aware that tests on this data set can, at best, indicate trends about the comparative performance of different systems, but in our view this is still better than no comparison.

The second problem was handled by assuming that each part is independent and that it can be tested alone. Although this is not always true, the alternative would be to test all combination of parameters, which is impractical for such a big

parameter space. The irrelevance and redundance filters and their parameters were described in the two previous chapters.

Another alternative to validate a system is to test its utility for the data which follows the initial assumptions and to test its harmlessness for data that do not follow those assumptions. Utility is tested by using different versions of the system where one or more components are missing. The best results should be achieved when all components of the system are used, or else one or more components of the system are unnecessary. Harmlessness was tested by showing that the results using our system are no worse than the results using a standard feature selection system, when the data does not follow the initial assumptions.

## 5.2 Implemented system

We implemented a three-step system that performs feature selection on huge feature sets. The first step filters irrelevant features with a modified version of Relief, see chapter 3. The second step filters redundant features using K-means with correlation as the distance measure, see chapter 4. The third step is a standard feature selection algorithm, either SFFS or SFBS were. A graphic description of the system is given at Section 1.3.2, Figure 1.1.

Given a large set of features the system begins by using Relief to rank those features according to their relevance. Then this ordered list of features is cut in two, separating the relevant features from the irrelevant ones. This can be done two ways: by giving the desired number of relevant features; or by giving the minimal Relief score (threshold) to consider a feature as relevant. It is important to limit the number of features at this point because k-means scales poorly for large sets. At the same time, too small a set will remove relevant features which would otherwise end up in the final feature set. Thus, this threshold, however it is specified, is critical to the overall system performance.

K-means receives the list of relevant features and uses correlation to cluster them. The initialization used is the farthest initialization (FI) discussed at 4.3.4. The most significant parameter here is the cluster radius threshold. This value defines if a cluster is too big and must be divided. The value used is 0.03, which is reasonable because section 4.4 shows that if two features correlate above 0.94 , it is preferable to use another non-correlated feature than to use both features. After clustering, one feature from each cluster is selected to represent it. The feature selected is the one with the highest Relief score. At this point the list of representative features may be clipped to a desired length by keeping the ones that have the highest Relief scores.

The final feature selection filter is then applied to the resultant feature set. If the number of features at this moment is 110 or less, either SFFS or SFBS can be used in the selection. If this number is higher than 110, then only SFFS can be used. This limit is a practical one, although it is close to the one proposed in the literature by Ng, et. al. [76] which is 100. Either of the feature selection algorithm, SFFS or SFBS, selects a subset of $n$ features, which is the final feature set.

## 5.3 SFBS and SFFS

The third step in our system is a traditional feature selection algorithm. The algorithms selected were the Sequential Floating Forward Selection (SFFS) and Sequential Floating Backward Selection (SFBS) because their authors claim them to be much faster than Branch and Bound (BB) while achieving comparable results [83].

SFFS and SFBS, proposed in 1994 by Pudil et. al. [83], are modifications of the SFS and SBS algorithms. SBS and SFS algorithms perform Hill-climbing[1] search over the feature space, without backtracking. SFFS and SFBS compensate for the

[1] Can also be described as a heuristic-guided Depth-first search.

lack of backtracking by a two step process. For SFFS the first step is inclusion of features and the second is exclusion of features; SFBS performs similar steps in the reverse order. The idea behind SFBS and SFFS is to allow the number of inclusions and exclusions to vary (float) between steps of the algorithm.

Both algorithms are similar, although SFBS is more effective at finding a good feature subset, because it has more information for making decisions at each step, in terms of having more features in the subset. Nevertheless, this ability comes with a significant cost in processing. In our experience, SFBS becomes infeasible after 110 features. SFFS can efficiently handle much larger features sets but at a price in the quality of the solution.

The high level algorithm of SFFS is shown below.A detailed and formal algorithm can be seen at Appendix B.

- ▷ Create a set with two features using SFS or another feature selection algorithm
- ▷ Do until the selected subset has the desired size
  - ▷ Include
    - ▷ Include the best remaining single feature in the selected subset
    - ▷ For all features in the selected subset, evaluate the selected subset without the feature
    - ▷ If the worst feature is the one just included, then go to Include
  - ▷ Remove
    - ▷ Remove worst feature
    - ▷ If the size of the selected subset equals 2, then go to Include
  - ▷ Next Worst
    - ▷ Find next worst
    - ▷ If criterion of worst is not an improvement, then go to Include
    - ▷ Remove next worst
    - ▷ If the size of the selected subset equals 2, then go to Include
    - ▷ Go to Next Worst

The only algorithm modification is to use the two best features coming from Relief as the initial set, since those are easily available for the system. The SFBS algorithm is a top-down version of SFFS (see Appendix B for the algorithm).

SFFS and SFBS rely on two parameters: a stopping criteria (typicaly the number of features in the selected set); and a distance measure. The size of the selected subset depends on the intrinsic dimensionality of the data and the task. Some studies have been done on finding the optimal number of features for a task and some algorithms proposed [7], but no solution has been proved precise. It is important to note that *"... intrinsic dimensionality is not the same as linear dimensionality which is a global property of the data involving the number of significant eigenvalues of the covariance matrix of the data"* [49]. In this work, the focus is on selecting features from a large feature set with lots of irrelevance and redundancy and not to find the best number of features to select, so no effort was done to survey or test those algorithms. In order to define the number of features to be selected for each dataset, a small sampling of the space of sizes for features sets was done and the best size achieved was subsequently used.

The distance measure used for SFFS and SFBS is the Mahalanobis distance. Shanon's Entropy was also considered, but could not be used because the sample space is too sparse to compute a meaningful a-priori probability. Moreover, Mahalanobis has been successfully used in many feature selection algorithms, for instance, BB [75]; SBS, SFS, SFFS, SFBS, PTA, and Min-Max [47]. An additional advantage is that Mahalanobis is a non monotonic distance measure. For a discussion of how Mahalanobis distance relates to other distance mesures, see section 2.4.4.

The Mahalanobis distance measures the distance between the centroids of the classes weighted by the common covariance matrix $\sum$.

$$J = (\mu_1 - \mu_2)^t \sum^{-1} (\mu_1 - \mu_2)$$

Under Gaussian class-conditional densities, the probability of error is inversely proportional to the Mahalanobis distance [36]. In our system, where many classes are present, we use:

$$J = \sum_{i=1}^{N} \sum_{j=1}^{N} P_i \; P_j \; (\mu_i - \mu_i)^t \; \sum_{i,j}^{-1} \; (\mu_i - \mu_j)$$

where $\sum_{i,j}$ is the pooled covariance matrix for classes i and j.

Another important issue in regard to the Mahalanobis distance is redundancy. Two or more identical features or multiple highly redundant (but not identical) features will generate a singular covariance matrix, which can not be inverted. Consequently, the Mahalanobis distance can not be computed for redundant sets of features. This is more of a problem for SFBS than SFFS, since SFBS begins with a set containing all features except two. As a result, if redundant features are present in the original feature set, the chance of generating a singular matrix is very high. This only happens in SFFS if the two features from the initial set are redundant, or the all features not in the original set are redundant, which is very unlikely. This problem is one more reason to first eliminate redundancy prior to applying the feature selection filter.

## 5.4 Tests on Adore Data

### 5.4.1 Data description

The data set used here is the same as described in section 3.5.1. Nevertheless, the feature set computed over the ADORE data is not exactly the same. In section 3.5, the tests were meant to test the ranking performance of the Relief algorithm. Therefore, 198 synthetic features were created using the Adore task signal and added into the features set. These features would not have any function here and, in fact,

they would be an impediment to the proper evaluation of the system since many of them are too closely correlated with the task signal. In addition to removing the synthetic features, a set of 20 new features were included. These features were the original features used in the ADORE system [35]. The purpose of including these is to check whether the set of features used in ADORE could be improved.

### 5.4.2 Test

We want to test if our three-step system performs better than standard feature selection algorithms for data which follows the basic assumptions stated before, particularly having many irrelevant and redundant features. We use the ADORE data because it has such characteristics. Since the system is composed of filters, any filter can be removed and the system will still work. The exceptions are some combinations which use SFBS, because the number of features would be to large. So, to test the performance of the system, all possible combinations of modules of the system were executed. The idea is that the best performance should be achieved when using all modules, otherwise one or more of the modules are unnecessary.

There are 891 features in the Adore data, from which 10 where selected for the final set. The number of features was defined by sampling the space of feature set sizes and using the best sample size. Each combination of modules selects features in a different way, and consequently the number of features selected at each step may vary. We tried to be consistent, so, where possible Relief selects 300 features, K-means 110 features. SFFS and SFBS selects 10 features. This values where chosen by experimentation. Table 5.1 shows the actual number of features selected at each module for each combination of modules of the system.

For each combination final set, 100 neural nets were created and executed. For each neural net the data was randomly divided in three sets: the learning set containing 70% of the samples; the validation set containing 15%; and the test set containing

| System | Original | Relief | K-means | Feature Selection | Final set |
|---|---|---|---|---|---|
| Relief | 891 | 10 | — | — | 10 |
| Relief+Kmeans | 891 | 300 | 10 | — | 10 |
| Relief+SFFS | 891 | 110 | — | 10 | 10 |
| Relief+SFBS | 891 | 110 | — | 10 | 10 |
| Relief+Kmeans+SFFS | 891 | 300 | 110 | 10 | 10 |
| Relief+Kmeans+SFBS | 891 | 300 | 110 | 10 | 10 |
| Kmeans+SFFS | 891 | — | 110 | 10 | 10 |
| SFFS | 891 | — | — | 10 | 10 |

Table 5.1: Number of features selected at each module of each tested combination of modules of the system

the remaining 15%. Each net was run for 15000 epochs, saving the net state with the lower MSE for the validation set. Each net is executed over its test set and the MSE saved. Figure 5.1 show the MSE for the validation set of 5 of the 100 nets. The average MSE for the 100 neural nets of each combination is shown in table 5.2. As can be seen, the best performances are given by using all modules of the system. The results are particularly good when the feature selection module used is SFBS. Note that SFBS is only feasible when the number of candidate features has been reduced to 110 or fewer. This threshold was defined empirically, since higher values required to long to run.

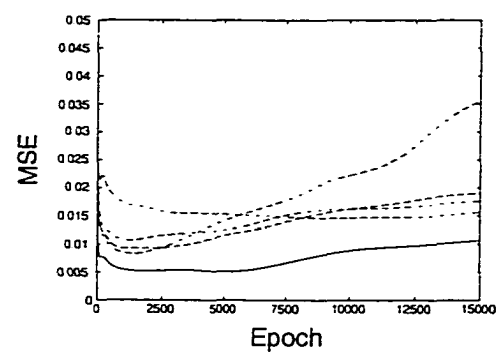



Figure 5.1: MSE for validation sets for 5 example nets over the ADORE data.

The differences between results are relevant. Note that the difference between "Relief+Kmeans+SFBS" and "Relief+Kmeans+SFFS" corresponds to approximately

| Average MSE for 100 nets (15000 epochs each) | |
|---|---|
| System | MSE |
| Relief | 0.0696 |
| Relief+Kmeans | 0.0269 |
| Relief+SFFS | 0.0236 |
| Relief+SFBS | 0.0247 |
| Relief+Kmeans+SFFS | 0.0156 |
| Relief+Kmeans+SFBS | 0.0093 |
| Kmeans+SFFS | 0.0154 |
| SFFS | 0.0649 |

Table 5.2: MSE for each tested combination of modules of the system

8% of the task signal range. Nevertheless, to be precise it is important to check if these results are statistically relevant. We do that by using a T-test. The Null hypothesis was: $H_0$: the two samples have the same average MSE ($\alpha = 0.005$[2]). Table 5.3 shows the results of the T-test for pairs of systems shown on table 5.2. Columns $H_0$ show the result of the T-test (True or False) and columns P show the probability of observing the given result by chance given that the null hypothesis is true.

| System | R | | R+K | | R+F | | K+F | | F | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $H_0$ | P | $H_0$ | P | $H_0$ | P | $H_0$ | P | $H_0$ | P |
| R+K+F | F | $\approx$ zero | F | 6.2e-15 | F | 1.5e-11 | T | 8.1e-01 | F | $\approx$ zero |

| System | R+K+F | | R+B | | K+F | |
|---|---|---|---|---|---|---|
| | $H_0$ | P | $H_0$ | P | $H_0$ | P |
| R+K+B | F | $\approx$ zero | F | $\approx$ zero | F | 2.2e-15 |

Table 5.3: Result of the T-test ($H_0$) and probability of observing the given result by chance given that the null hypothesis is true (P) for pairs of combinations of modules of the system. Values of $P \approx zero$ indicate that the probability is so small that it exceeds the capability of representation used by the statistical package. R, K, F, and B correspond respectively to Relief, K-means, SFFS, and SFBS.

[2]Unless noted, all the tests where the objective is to reject the Null hypothesis an $\alpha$ of 0.005 will be used. On the other hand, if the objective is to accept the Null hypothesis a more common value of 0.05 will be used.

The first line of table 5.3 compares the whole system using SFFS and all other combinations using SFFS or no feature selection. This shows that it is significantly better to run the whole system than parts of the system. Note that Relief and SFFS running alone correspond to two of the best standard feature selection algorithms described on the literature. The only case where the whole system using SFFS can not be shown to run better was when running k-means followed by SFFS only, but the difference was not statistically significant what indicates that no decrease in performance comes from running the whole system. More importantly, when running the whole system using SFBS as the final filter the performance achieved greatly surpasses all other options. This indicates that when most features are irrelevant and/or redundant, the best option is to run the three steps of the system, i.e. Relief, K-means and SFBS.

## 5.5 Tests on Digits Data

### 5.5.1 Data description

To better validate the system, it is useful to test it on publicly available data sets described in the literature. Unfortunately, the are no data sets reported that share all the assumptions used in our system. Important assumptions that are not followed by most data sets are: large features sets; many irrelevant features; many redundant features; continuous data; and a continuous target signal. Nevertheless, we regard it as important to do such comparison. The only dataset with a large number of numeric features in the literature is described by Breuklen et al. [20] and subsequently used by Breuklen et al. [21]; a slightly different version was used by Jain et al. [49]. This data, from now on referred to as the Digits data, was donated by Robert Duin of Delft University of Technology.

The dataset consists of features of handwritten numerals ('0' - '9') extracted from a collection of Dutch utility maps. 200 patterns per class (for a total of 2,000

patterns) have been digitized into binary images. The source image dataset is not available. These digits are represented in terms of the following six feature subsets:

1. 76 Fourier coefficients of the character shapes;
2. 216 profile correlations;
3. 64 Karhunen-Loève coefficients;
4. 240 pixel averages in 2 x 3 windows;
5. 47 Zernike moments;
6. 6 morphological features.

It is important to remind that this data does not follow most of the assumptions made for our system. For example, it does not have many redundant features. Moreover, this data has been mainly used to test combinations of classifiers ([20] and [21]), and so any comparison of results with our system can not be regarded as definitive. However, it can show trends and is the only data set available with a sufficiently large number of features.

For use in our system, the dataset was modified in the following way: 1) all 6 feature sets were merged into one feature set; 2) a target signal was created where each class received a target value. So digit 0 (class 1) received 0, digit 1 (class 2) received 0.11111, and so on until digit 9 (class 10) which received 1; and 3) for some tests, the features were replicated by including noise.

### 5.5.2 Tests

As a first pass, the same tests performed on the ADORE data were performed on the Digits data, albeit with different parameters. The parameter changes were: the use of 30 features in the final selected set instead of 10; and the use of 5000 training

epochs for the neural net, instead of 15000. The result of these tests, however, were very different. The most effective combination was not "Relief+Kmeans+SFBS" or "Relief+Kmeans+SFFS" but "Kmeans+SFFS". Table 5.4 shows the average MSE for 100 nets. The reason for this difference is that most of the Digits data features are relevant and little redundancy is present. In other words, because the original feature set was hand-selected, it does not follow the assumptions used in this work. That is not news, but the consequences were not expected. Contributing to this problem is a poor set of parameters; we restrict the number of features to be applied to the feature selection to 110. Consequently, the system ignores too many useful features. "Kmeans+SFFS" performs a little better than SFFS alone because it eliminates some redundancy and in doing so makes the task easier for SFFS. To solve the performance problem when running the three filters, this option was run again without limiting the number of selected features in either Relief and k-means. This implies that SFBS could not be used, since it can handle at most 110 features. The results are shown on table 5.5.

| System | # Feat | MSE |
|---|---|---|
| Relief+Kmeans+SFFS | 649 | 0.0116 |
| Relief+Kmeans+SFBS | 649 | 0.0132 |
| Kmeans+SFFS | 649 | 0.0046 |
| SFFS | 649 | 0.0059 |

Table 5.4: Average MSE for 100 nets (5000 epochs each) for combinations of the system on the original Digits data.

The results for "Kmeans+SFFS" and "Relief+Kmeans+SFFS" shown on table 5.5 are statistically equivalent (see table 5.6), but these results outperform running only SFFS. The importance of this is that the system does not harm the performance of the feature selection algorithm, even if the feature set does not follow the assumptions. Nevertheless, some gain was achieved even if small. Next it is important to show that the system is useful when the assumptions are followed. To do

| System | # Feat | MSE |
|---|---|---|
| Kmeans+SFFS | 649 | 0.0046 |
| Relief+Kmeans+SFFS | 649 | 0.0049 |
| SFFS | 649 | 0.0059 |

Table 5.5: Average MSE for 100 nets (5000 epochs each) for combinations of the system on the original Digits data. No limits were set on the number of features selected by Relief or k-means.

that, three new tests were performed. For these tests the data set was modified to: 1) include redundant features; 2) include irrelevant features; 3) include various levels of redundant and irrelevant features.

| System | k+F | | F | |
|---|---|---|---|---|
| | $H_0$ | P | $H_0$ | P |
| R+K+F | T | 1.4e-01 | F | 7.8e-05 |

Table 5.6: Result of the T-test ($H_0$) and probability of observing the given result by chance given that the null hypothesis is true (P) for pairs of combinations of modules of the system for original Digits data set. $H_0$: samples have the same average. R, K, and F correspond respectively to Relief, K-means, and SFFS.

### Redundancy and Irrelevancy

The feature set with redundant features was created by duplicating the features and adding 10% Gaussian noise with zero mean and standard deviation 1. This resulted in 1298 features. This data was run through the whole system using SFFS as the feature selection filter. SFBS was not run because in this case we are selecting 600 features in k-means. The parameters for this and the irrelevancy tests on the Digits data is show at Table 5.7. The average MSE achieved to the redundant feature set was 0.0045, see Table 5.8, which is statistically equivalent to the best result for the original feature set (649 features). Table 5.9 show the comparison between this result and the best result for the original feature set. The features selected were not

the same, but only one feature from the synthetic redundant set was selected. This shows that the system was able to remove the redundancy.

The feature set with irrelevant features was created by including 2 sets of features, a set of redundant features and a set of irrelevant features. The redundant features were included as explained above and the set of irrelevant features was included by duplicating the original features and adding 60% Gaussian noise with zero mean and standard deviation 1. Such features are not totally irrelevant, since they are based on the original relevant features, but they are so swamped by noise that they are much less relevant than the original ones. Tables 5.7 and 5.8 show respectively the parameters used to select the features and the average MSE. Although the result achieved by the irrelevant feature set is not the same as the best one over the original set, it is nevertheless statistically equivalent (see Table 5.9). The features selected were, again, not the same, and only one feature from the redundant set was selected. This shows that the system was able to remove irrelevance.

| System | Test | Original | Relief | K-means | Feat. Sel. | Final set |
|---|---|---|---|---|---|---|
| Relief+Kmeans+SFFS | Redundancy | 1298 | 1298 | 600 | 30 | 30 |
| Relief+Kmeans+SFFS | Relevancy | 1946 | 1298 | 600 | 30 | 30 |

Table 5.7: Number of features selected at each module for each of the redundancy and relevance tests

| System | Test | # Feat | MSE |
|---|---|---|---|
| Relief+Kmeans+SFFS | Redundancy | 1298 | 0.0045 |
| Relief+Kmeans+SFFS | Relevancy | 1946 | 0.0046 |

Table 5.8: Average MSE of 100 nets on the Digits data for the redundancy and relevance tests (1298 & 1946 features)

| System | Test | # Feat | K+F (649) | |
|---|---|---|---|---|
| | | | $H_0$ | P |
| R+K+F | Redundancy | 1298 | T | 7.2e-01 |
| R+K+F | Relevance | 1847 | T | 9.2e-01 |

Table 5.9: Result of the T-test ($H_0$) and probability of observing the given result by chance given that the null hypothesis is true (P) for the redundant and irrelevant feature sets and the best result for the original set of the Digits data. R, K, and F correspond respectively to Relief, K-means, SFFS.

### Utility

The most important test, however, was to include various levels of redundancy and/or irrelevance. This was done by duplicating the original features and adding 4 levels of Gaussian noise (10%, 25%, 40%, and 55%) with zero mean and standard deviation 1. The idea here is that the system should be able to remove both redundancy and irrelevancy. In doing that the result should be better than running a standard feature selection algorithm. The combination "K-means+SFFS" was included in the test because it was the one which achieved the best result over the original data. So, this new data, with 3245 features, was submitted for the three combinations of the system mentioned above. Table 5.10 shows the number of features selected at each module for each of the combinations. 600 features were selected in K-means, which is approximately the number of features on the original set. The average MSE for 100 nets is shown on table 5.11.

| System | Original | Relief | K-means | Feat. Sel. | Final set |
|---|---|---|---|---|---|
| Relief+Kmeans+SFFS | 3245 | 1298 | 600 | 30 | 30 |
| Kmeans+SFFS | 3245 | — | 600 | 30 | 30 |
| SFFS | 3245 | — | — | 30 | 30 |

Table 5.10: Number of features selected at each module for the mixed test

Results from table 5.11 show that running the whole system yields a better performance than running only Kmeans and SFFS or running only SFFS, which is

| System | MSE |
|---|---|
| Relief+Kmeans+SFFS | 0.0048 |
| Kmeans+SFFS | 0.0062 |
| SFFS | 0.0086 |

Table 5.11: Average MSE for the mixed test on the Digits data (3245 features)

a standard feature selection. Moreover, these results are statistically relevant. Table 5.12 shows the result of a T-test for the runs. In this case the Null hypothesis was: $H_0$: the two samples have same average ($\alpha = 0.005$). P is the probability of observing the given result by chance given that the null hypothesis is true.

| System | K+F | | F | |
|---|---|---|---|---|
| | $H_0$ | P | $H_0$ | P |
| R+K+F | F | 7.0e-09 | F | $\approx$ zero |

Table 5.12: Result of the T-test ($H_0$) and probability of observing the given result by chance given that the null hypothesis is true (P) for the mixed test on the Digits data. Values of $P \approx zero$ indicate that the probability is so small that it exceeds the capability of representation used by the statistical package. R, K, and F correspond respectively to Relief, K-means, SFFS.

Table 5.11 results show that if the feature set follows the assumptions about the data the system is useful. In particular, running the whole system is statistically better (see Table 5.12) than running only SFFS.

In conjunction, the results for the ADORE data set and Digits data set show that if the number of relevant and non redundant features is small (less than 110) then "Relief+Kmeans+ SFBS' should be run; otherwise, "Relief+Kmeans+SFFS' is the best option. Although, no attempts to automate the decision about using SFFS instead of SFBS have been made, it can be easily accomplished by selecting an appropriate threshold on the score of selected features on the Relief module. By experience, this threshold should be approximately 0.025.

### Comparison

So far, we have only compared the system with itself and standard feature selection algorithms. No comparison has been made with systems where the data has been used before. In [21], the authors study the performance of combinations of classifiers. They use four of the subsets of the original data and apply individual sets for different classifiers. The classifiers used are: 1, 2, and 5 k-nearest-neighbors (k-nn); Karhunen-Loève, Gaussian; and Fisher.

To compare our results with theirs, a final test was performed over the data. A K-nearest-neighbors algorithm was implemented. The distance measure used was the same used by Breuklen et. al. [21], which is the posterior probability of a sample belonging to each class. This probability was computed as the sum of the reciprocals of the Euclidean distance of the sample with each neighbor in the class, normalized among the classes.

The 30 features selected by the system on the original data (649 features) was used to train 10 k-nearest-neighbors (k-nn) nets with 5 neighbors (5-nn). The average recognition error achieved by the features selected by our system was 2.1%. This value is the same reported by Breukelen et al. [21], as the best result for a combination of 5-nn classifiers. In their case, the best individual 5-nn performance was 3.7%. Unfortunately, no sequence of results are given and consequently no statistical comparison can be done.

Again it is important to stress that this comparison can not be taken as definitive, because the main objective of both systems are different and the assumptions about the data are also different. Nevertheless, the results suggest that using our system to select features can achieve comparable performance with using a hand-selected set of features and combining the results of the classifiers.

## 5.6 Tests on Cats and Dogs Data

### 5.6.1 Data description

The technique used in this section is not claimed by us to be the best way to do classification or data reduction. However using it allows us to show that our system is able to find relevance and visually show this relevance.

The tests on the ADORE data show that the proposed three-step system performs better than a standard feature selection algorithm for data with lots of redundant and non relevant features. On the other hand, the tests on the Digits data show that the system does not harm the performance if used over a data that does not follows the assumptions. Here, we are interested in show that the system is able to filter relevancy and redundance with very little loss of information and that the interrelation between features of the system is accounted for when the filtering is done. To do that we need results from a task run over a large set of features.

We decided to use a classification task based on an PCA nearest neighbors algorithm over a set of 200 animal faces (100 cats and 100 dogs). Each sample is a 64x64 (4096 pixels) image and shows the frontal view of the face of a cat or dog. The backgrounds vary from image to image and the faces were registered by hand. The data was extracted from Wendy Yambor's Master thesis [112]. Figure 5.2 shows 48 examples (24 dogs and 24 cats) extracted from the Cats and Dogs data set.

It is important to note that pixels are not well correlated to the task signal. The pixel that achieves the best score in Relief, correlates with the learning signal with a score of 0.1849, which is very low. Figure 5.3 shows the graph of three pixels, including the best one, against the learning signal.

There is some information on the pixels, though, not much. So, the only way the system can achieve a good classification is if it is able to detect interrelation between features.

Figure 5.2: Examples of images on the Cats and Dogs data set (24 cats and 24 dogs).

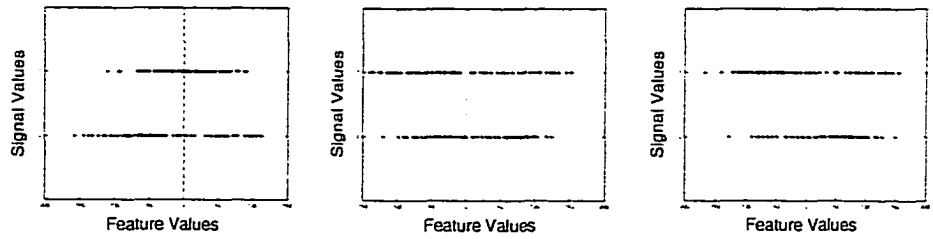



Figure 5.3: Cats and dogs features: a) best feature according to Relief; b) $20^{th}$ best feature; and c) $40^{th}$ best feature.

### 5.6.2 Test

Our purpose in this test is to compare the amount of information lost by filtering irrelevance and redundancy. To do that we run the classification task: first, for the original images (64x64 = 4096 features); and second, to the filtered features. Both results are then compared.

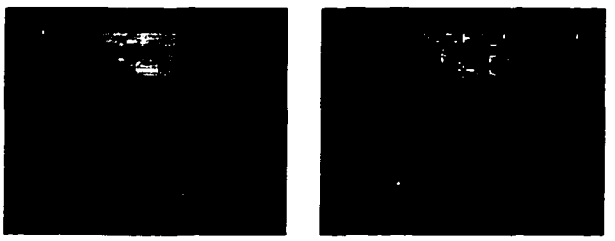
The classifier used was a PCA-based 1-nearest neighbor. This classifier computes all eigenvectors for a learning set and the sample projections over this basis. When a new sample is presented its projection over all eigen-vectors is computed and the result of the classification is the label (class) of the closest neighbor, in terms of Euclidean distance, to the sample projection.

The classification was run five times for each data set (original and filtered). For each run, the data set was divided in two files: learning set and test set. The learning set contains 160 randomly selected samples and the test set the remaining 40 samples. This is done by randomly dividing the data into 10 non overlapping data sets, each with 10 dog images and 10 cat images. These 10 data sets are the same used by Yambor [112]. They are concatenated to form each of the 5 groups of learning and test sets. The result of the five runs were merged to create the final result set with 200 results.

The filtered data set is computed by running the first two filters of the system over the original data set. Here we consider each pixel as being one feature. The first filter, Relief, remove the irrelevant features. Here we selected all features with a Relief score of 0.025 or higher. This threshold was selected by experimenting with the other data sets. The remaining 615 features were clustered by k-means. The result was a feature set with 217 features (pixels). This is approximately 95% reduction.

One advantage of using pixels as features is that pixels can be visualized. Figure 5.4 shows images representing the result of the filters (Relief and K-means). On these

images each pixel is represented by: zero, if it was eliminated during the filter; or the Relief score it it was not eliminated during the filter.



(a) (b)

Figure 5.4: Result of the Relief and Kmeans filters over the Cats and Dogs dataset. The value of each pixel is zero if it passed the filter or the Relief score it it did not passed the filter. a) after Relief filter (615 features/pixels selected); b) after k-means filter (217 features/pixels selected).

Although these images do not prove much, they are in accord with intuition. The features that contain more information are those that differentiate between dogs' and cats' forehead, ears, eyes and mouths. Very few pixels in the faces of the animals and background were selected.

That seems to indicate that most of the information is present after the filtering, but a more formal comparison can be done by comparing the results of the classification over both data sets (original and filtered). For this comparison, each sample result for each data set was counted in one of four sets.

- SS: set of samples were both data sets classified the sample correctly
- SF: set of samples were the original data set classified the sample correctly but the filtered did not.
- FS: set of samples were the filtered data set classified the sample correctly but the original did not.
- FF: set of samples were both data sets classified the sample incorrectly

| SS | SF | FS | FF |
|---|---|---|---|
| 145 | 22 | 16 | 17 |

Table 5.13: Results of comparison between classification test over Cats and Dogs data set

The results are given on table 5.13.

To test if both data sets achieve the same performance we used McNemar's test, as used by Yambor et. al. [111]. This test is similar to the binomial test but it considers only the difference in classification between the data sets (SF and FS). The idea is that if both data sets are equally good, then the probability of the first succeeding while the second fails is the same as the probability of the first failing while the second succeeds, or 0.5. With that the null hypothesis is: $H_0$: P(SF) = P(FS). The NcNemar's test indicates that there is a probability of 0.208 that we would see the result (SS = 26 and FS = 16) if the Null hypothesis is true. Using $\alpha$ = 0.05 the Null hypothesis can not be rejected and we must consider the difference in result for the data sets not statistically significant. In other words the data sets have approximately the same information despite the fact that the filtered version contains only 5% of the pixels in the original one.

## 5.7 Conclusion

Although no other available system in the literature deals with the same kind of data and task as intended in this work, an effort was done to use both an internally created data and data from an external source. Both types of data were tested and we were able to show that if the data follow the assumptions set for this work our system achieves better performance than by using standard feature selection algorithms only. Moreover, an attempt to compare results with other systems on the literature were done and although the comparison can not be taken as conclusive,

because of the different nature of both works, it shows evidence that the system performs well. A final test shows that using the irrelevance and redundancy tests do not degrade the information content of the data set while greatly reducing its size (number of features).

## Chapter 6

# CONCLUSION

The number of features that can be computed from an image is, for practical purposes, infinite. Consequently, it is impossible to compute and use all of them. Nevertheless, a system capable of measuring a few thousand of the most commonly used features would be very useful if a few tens of them could be selected for use in object recognition tasks. The system implemented in this work is a first step in this direction. In particular, this system assumes that the data has the following characteristics: large number of features; many irrelevant features; many redundant features; noisy data; continuous data; small a training set, compared to the number of features. It is also assumed that the task is function approximation, rather than classification.

The proposed algorithm (system) consists of three steps. The first step uses a modified version of the Relief algorithm ([54] and [61]) to filter the irrelevant features. The second step uses a clustering algorithm (k-means [66]) to filter the redundant features. The last step uses a standard feature selection algorithm (SFFS or SFBS [83]) to select the final subset from the remaining features.

The system was tested with three very different data bases. One, ADORE [35], was created by us and has many irrelevant and redundant features. The second, Digits [20] [49], is an external data set used in the literature. It is composed by hand-picked sets of features. This data does not follow the assumptions stated before,

but was used in tests in its original form and transformed to adapt to the same assumptions. The third, Dogs and Cats, also extracted from the literature, was mainly used to visualize the result of the first two filters (Relief and K-means). The results of the tests over ADORE and Digits show that, if the data does not conform to the assumptions, no gain, or a small gain, occurs by using our system, but no loss occurs. On the other hand, when the data follows the assumptions, our systems outperforms standard feature selection algorithms (SFFS or SFBS), which were shown to perform close to optimally for a different data set, where the assumptions here made were not present [83]. The cats and Dogs data show that Relief + Kmeans were able to remove most of the irrelevance and redundancy without significantly reducing the performance of a PCA-based classifier.

## 6.1 Contributions

The main contribution of this work is an algorithm (system) able to effectively perform feature selection in huge feature sets. It is the one of the first systems, as far as we know, that deals with so many features. Exceptions are Automatic Text Classification, and Tieu and Viola [101] work, but in this case the features and the task are very specific and very different from our system. It is also the first feature selection system which explicitly filter irrelevancy and redundancy. The advantage of our system is that by focusing each step of the algorithm over one problem it is able to deal effectively with much bigger data sets. It was shown that in doing so the system achieves a better set of features.

Another contribution was the discovery and subsequent reduction of the bias against non-monotonic features in Relief. Relief performs well on identifying irrelevance, but the original algorithm has a huge bias against functions (features) which have a peak/valley in their range. Our modification of the algorithm allows the

identification of such features and allows dealing with each part of the distribution of the feature separately, eliminating most of the bias against the feature.

In the clustering or redundancy filter, our contribution was the use of correlation as the distance measure. In order to do so, the way the centroid of the cluster is computed had to be modified. Two theorems were proposed and proved showing the right way to do that so the centroid remains in the center of the cluster.

## 6.2 Future work

Many interesting issues could not be pursued in this work. Among then are the following:

1. Elimination of bias for other kinds of non-monotonic features in Relief. Relief is also biased against two peeked features and features with a more complex function like Gaussian features. The modification made by us eliminated most of the bias against Gaussian features, but features with multiple peeks/valleys are still a problem. The main reason for not addressing other kinds of features is that they are not so common as one peaked features, but the issue is worth studying.

2. Study the roll of the distance measure in Relief. Two issues seems to be important here. Is the distance measure necessary in Relief? When searching for the causes of the bias in Relief, our studies indicate that for non peaked features, the Relief score does not vary much when the distance measure is disturbed by including random features. If that is true, there is no need to have a distance measure and to compute hits and misses on Relief. It could be enough to consider all samples in the same class as hits and all samples in different classes as misses. On the other hand, if this is not true, then it is

possible that a multi-step process where the worst features are removed and Relief is executed again, may improve the ordering of the resultant features.

3. Deepen the study of correlation as a distance measure in K-means. In particular, the change in performance if anti-correlated clusters are merged.

4. Study boosting as a way to improve selection. Kohavi and John [57] classify feature selection in Filters and Wrappers. Wrappers use the task for both, performance and evaluation function. This approach yields a better performance but is, in most cases, prohibitive. We propose the use of Boosting as a way to influence the selection of the feature set by the performance function. Boosting can be used to modify the data set and by doing that influence the selection of features toward those which would best approximate the samples that have larger errors. Clearly this influence should be mitigated, compared with wrappers, but the cost would not be so big.

Appendix A

# FEATURE ACQUISITION

## A.1 Features description

The number of features that can be computed from an image is very large. For example Tieu and Viola [101] use 45,000 features in their work. Consequently, it is important to have a way to collect these features. Although the primary goal of this work is feature selection, not feature generation, a feature acquisition module was implemented to generate the data sets. This appendix describes important features used in this work and in ADORE.

### A.1.1 Features

Examples of interesting features computed using the raw value representation of an image are given below. Even if restricted to raw image values, there are nevertheless many options. The most important raw values for our application are: intensity, RGB, hue, saturation and disparity (from stereo or motion data). Yet, the methods used can be applied with minimal or no alteration for other raw values, for example: infrared, LANDSAT [62] and LADAR [99].

Many of the features used in this work, for example standard deviation, are not scale invariant. Consequently, it is important that some scaling factor be applied to the data. This has a multiplicative effect on the number of raw value images, because now each representation will be scaled one or more times generating new data. It is important to note that scaling up the images (i.e. increasing the resolution) has little meaning because it only duplicates pixels. On the other hand, scaling down has the effect of averaging the image; effectively smoothing it and reducing high frequency elements while expanding the spatial extent of local operators (eg. derivatives). Similar effect could be achieved by using smoothing filters or band filters.

In addition, the rate of variation (derivatives) across the x and y axes of images can be computed. Five types of derivatives are used in this work, the first derivatives of the image for each axis, the second derivatives of the images for each axis, and the second derivative of the image for axis $x$ and $y$. In other words, $\frac{\partial f(s)}{\partial x}$, $\frac{\partial f(s)}{\partial y}$, $\frac{\partial f(s)}{\partial x^2}$,

$\frac{\partial f(s)}{\partial y^2}$, and $\frac{\partial f(s)}{\partial xy}$. These derivatives can be applied for all raw values at all scales to further increase the number of representations.

Although mathematically well defined, the derivative of combinations of raw values, for example derivatives of the red component in relation to the blue component in RGB, have not been widely used in the reviewed literature (although see Adelson and Bergen [1]). Therefore they are not used in this work. More meaningful derivatives are the derivatives of the result of the statistics over the raw values. Despite that, these are also not computed because it is equivalent, in most cases, to compute the statistic over the derivative of the raw value, since $g(s)^* \frac{\partial f(s)}{\partial s} = \frac{\partial (f(s)g(s)^*)}{\partial s}$. In other words, the result of a convolution over the derivative of a function is equal to the result of the derivative of the convolution over the function. Because many of the statistics computed in this work, for example average, can be seen as convolutions, the derivatives of these statistics do not need to be computed; it is enough to compute the statistics over the derivative. Adelson and Bergen [1] give an extended explanation of this property.

### Statistical

For each value image, a set of statistical features can be computed. While the number of such statistical features is almost limitless, this work restricts them to the most important ones (see Figure A.1). For example, the proposed standard library Vsipl (Vector, Signal and Image Processing Library) [107], in its Image Processing Operations manual [108], defines: area, mean, variance, standard deviation, skewness, kurtosis, root mean square, median, sum, positive sum, negative sum, maximum, minimum, number of positive values, number of negative values and number of zeros.

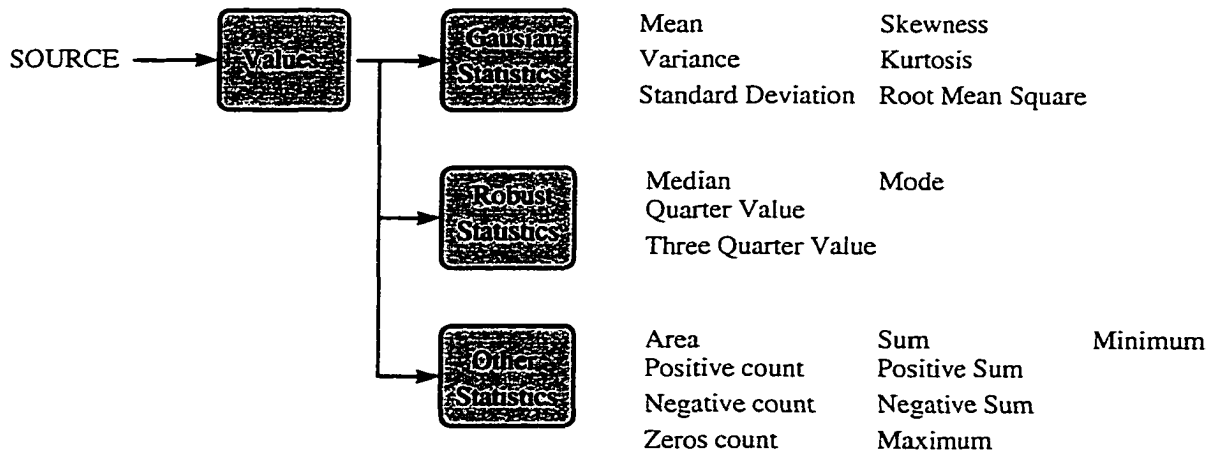



Figure A.1: Statistical features.

Robust statistics, such as the median, are important because they are less sensitive to noise. Croxton [29] suggests other robust statistics such as mode, quartiles, quintiles and percentiles. We use mode and quartiles (see Figure A.1). Robust statistics has motivated other important works, for example the Hausdorff Distance Measures used in Sim et al. [97].

### Texture

Another important kind of feature are texture features. Haralick and Shapiro [44] define texture as "concerned" with the spatial distribution of image intensities and discrete tonal features. Texture features are an area of active research, including Markov Random Fields [12] [52] and Wavelets [81]. Right now, the texture features used in the system are entropy and correlogram features. Haralarick and Shapiro [44] list 9 measures that can be computed over a co-ocurrence matrix (correlogram). All these measures are used in this work.

It is important to note that both statistical and texture features can be computed either over the whole image (set of pixels) or locally over a window of the image. For some features, the effect of using windows can also be achieved by scaling the image. Nevertheless, that is not true for all features, for example the median.

### Model Matching

When a prototypical model or template is available, any of the previous features can be computed for models compared with the same feature computed for the samples. However, if the difference is a meaningful feature, the function approximator can infer that from the original features. Moreover, for most features, the difference is linearly dependent of the sample feature, since the model features are constant for all samples, therefore the model features do not need to be computed at all. Nevertheless, there are some direct comparison measures that are very important. One of the most important of such comparisons is correlation. In this work, a rotation-free correlation based on the work of Ravela et al. [88] and Freeman and Adelson [37] is used. Mutual information computes a measure similar to correlation which establishes the degree of matching between sets of values from model and sample. Mutual Information may consequently be less sensitive to changes in sensor, illumination, normalization, equalization, etc, than correlation. Other similar measures are the dependence measures in Table 2.2.

Another class of comparison techniques are histogram matching algorithms. Schiele and Crowley show that histogram comparison is a powerful, although pose sensitive, method for recognizing objects, especially for color images under constant illumination [94]. In this work, four methods of comparing model and sample histograms are used: intersection measurement [100]; $\chi^2$-test, assuming exact knowledge of the model; $\chi^2$-test, when no knowledge of the model is assumed [82] and the well known Kolmogorov-Smirnov test[82]. Multiple measures are used because they rely

on different underlying assumptions. For example, Schiele and Crowley hypothesize that the intersection method works better when objects are occluded while the $\chi^2$-test method works better for distortions due to appearance change or Gaussian noise.

### Registration Measures

When the samples are in registration with the models, and/or with each other, other classes of features can be computed. One important set of possible features are probes. Der and Chellappa [32] define probes as simple mathematical functions which operates locally on image grey levels and produce outputs that are more directly usable by an algorithm. A directional probe image is calculated by taking the difference in grey levels between pixels a set distance apart in a given direction, centered on the probe image pixel. Probes are common in Automatic Target Recognition system (ATR) like the ones described in [13] and [32]. In those systems, the probe values are used to compute a function; common functions are number of probes over a threshold and the probability of the probes being drawn from the target given the probe values. In this work, the probe values are features used by the approximator to learn a function, if they are selected in the feature selection process. Figure A.2 shows example of potentially useful probes on the ADORE data.

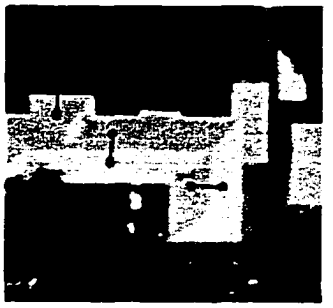


Figure A.2: Example of probes.

Another class of powerful features are principal components. If the samples are registered to each other, a basis for the space of samples (M) can be computed. The best basis is given by the left singular vectors of M, which equal the eigenvectors of $MM^t$. Given that, the projection of a sample over an eigenvector is a feature. A set of $n$ features can be created this way for each sample by using the first $n$ eigenvectors. The first eigenvectors are the directions of greatest variance among the data. As mentioned earlier, this variance may provide information for the function approximation, or it may be random background noise. Because of that, in some applications it is necessary to reduce the influence of the background. This can be done

by segmentation [74] or by applying a Gaussian filter centered on the object [104]. It is important to note that we are not proposing to implement any of the techniques used to recognize objects with eigenvectors. In these techniques it is assumed that the first eigenvectors provides the best features for the object recognition. Here, the first eigenvectors are only used to create features that are part of the initial feature set.

The same technique can be used with the models. In this case the eigenspace of the models can be used to project the samples and these projections used as features. The eigenspace computed over the models gives the difference between models. This difference may be more accurate than the ones computed using the samples because the models have no background to affect the process and the models can be precisely centered.

### A.1.2 ADORE features

A complete description of the ADORE data features would be to big to show here. Instead a brief description of the representations used is shown below. Each line is composed by operations executed over the data until the final representation. As an example, line 3 generates features 171 to 220 below. Here the initial image (SUBIMAGE) was reduced by cutting a sub-window (WINDOW), this window was centered (CENTERORIGIN), then normalized, then projected over the first 50 eigenvectors of the data (DATAEIGEN). The resultant 50 values were used as features 171 to 220. If two lines have the same representation, that indicates that the same program with different options was run over the file. For example DIFFHISTOGRAM is the result of computing a difference histogram for the image for different directions and with different step sizes. Features from 388 up to the end are statistical features computed over the representation.

```
>>> SUBIMAGE ADORE  20:(1 - 20)
>>> SUBIMAGE PROBE  150:(21 - 170)
>>> SUBIMAGE WINDOW CENTERORIGIN NORMALIZATION DATAEIGEN  50:(171 - 220)
>>> SUBIMAGE WINDOW CENTERORIGIN NORMALIZATION TEMPLATEEIGEN  5:(221 - 225)
>>> SUBIMAGE TPLTHISTOGRAM  15:(226 - 240)
>>> SUBIMAGE MASK TPLTHISTOGRAM  15:(241 - 255)
>>> SUBIMAGE HISTOGRAM  6:(256 - 261)
>>> SUBIMAGE MASK HISTOGRAM  6:(262 - 267)
>>> SUBIMAGE DIFFHISTOGRAM  6:(268 - 273)
>>> SUBIMAGE DIFFHISTOGRAM  6:(274 - 279)
>>> SUBIMAGE DIFFHISTOGRAM  6:(280 - 285)
>>> SUBIMAGE DIFFHISTOGRAM  6:(286 - 291)
>>> SUBIMAGE DIFFHISTOGRAM  6:(292 - 297)
>>> SUBIMAGE DIFFHISTOGRAM  6:(298 - 303)
>>> SUBIMAGE DIFFHISTOGRAM  6:(304 - 309)
```

```
>>> SUBIMAGE DIFFHISTOGRAM  6:(310 - 315)
>>> SUBIMAGE CORRELOGRAM  9:(316 - 324)
>>> SUBIMAGE CORRELOGRAM  9:(325 - 333)
>>> SUBIMAGE CORRELOGRAM  9:(334 - 342)
>>> SUBIMAGE CORRELOGRAM  9:(343 - 351)
>>> SUBIMAGE CORRELOGRAM  9:(352 - 360)
>>> SUBIMAGE CORRELOGRAM  9:(361 - 369)
>>> SUBIMAGE CORRELOGRAM  9:(370 - 378)
>>> SUBIMAGE CORRELOGRAM  9:(379 - 387)
>>> SUBIMAGE  15:(388 - 402)
>>> SUBIMAGE MASK  15:(403 - 417)
>>> SUBIMAGE DERIVATIVEX  18:(418 - 435)
>>> SUBIMAGE DERIVATIVEX MASK  18:(436 - 453)
>>> SUBIMAGE DERIVATIVEY  18:(454 - 471)
>>> SUBIMAGE DERIVATIVEY MASK  18:(472 - 489)
>>> SUBIMAGE DERIVATIVEX DERIVATIVEXX  18:(490 - 507)
>>> SUBIMAGE DERIVATIVEX DERIVATIVEXX MASK  18:(508 - 525)
>>> SUBIMAGE DERIVATIVEY DERIVATIVEYY  18:(526 - 543)
>>> SUBIMAGE DERIVATIVEY DERIVATIVEYY MASK  18:(544 - 561)
>>> SUBIMAGE DERIVATIVEX DERIVATIVEXY  18:(562 - 579)
>>> SUBIMAGE DERIVATIVEX DERIVATIVEXY MASK  18:(580 - 597)
>>> SUBIMAGE SCALE1  14:(598 - 611)
>>> SUBIMAGE SCALE1 MASK  14:(612 - 625)
>>> SUBIMAGE SCALE1 DERIVATIVEX  17:(626 - 642)
>>> SUBIMAGE SCALE1 DERIVATIVEX MASK  17:(643 - 659)
>>> SUBIMAGE SCALE1 DERIVATIVEY  17:(660 - 676)
>>> SUBIMAGE SCALE1 DERIVATIVEY MASK  17:(677 - 693)
>>> SUBIMAGE SCALE1 DERIVATIVEX DERIVATIVEXX  17:(694 - 710)
>>> SUBIMAGE SCALE1 DERIVATIVEY DERIVATIVEYY  17:(711 - 727)
>>> SUBIMAGE SCALE1 DERIVATIVEX DERIVATIVEXY  17:(728 - 744)
>>> SUBIMAGE SCALE1 SCALE2  14:(745 - 758)
>>> SUBIMAGE SCALE1 SCALE2 MASK  14:(759 - 772)
>>> SUBIMAGE SCALE1 SCALE2 DERIVATIVEX  17:(773 - 789)
>>> SUBIMAGE SCALE1 SCALE2 DERIVATIVEX MASK  17:(790 - 806)
>>> SUBIMAGE SCALE1 SCALE2 DERIVATIVEY  17:(807 - 823)
>>> SUBIMAGE SCALE1 SCALE2 DERIVATIVEY MASK  17:(824 - 840)
>>> SUBIMAGE SCALE1 SCALE2 DERIVATIVEX DERIVATIVEXX  17:(841 - 857)
>>> SUBIMAGE SCALE1 SCALE2 DERIVATIVEY DERIVATIVEYY  17:(858 - 874)
>>> SUBIMAGE SCALE1 SCALE2 DERIVATIVEX DERIVATIVEXY  17:(875 - 891)
```

Appendix B

# ALGORITHM DESCRIPTIONS

## B.1 Definitions

Set of features: $F = \{f_1, f_2, ..., f_p\}$
Subset of features: $F' = \{f'_1, f'_2, ..., f'_d\}$
Subset of features of size d: $F'_d = \{f'_1, f'_2, ..., f'_d\}$
Set of Samples: $S = \{s_1, s_2, ..., s_n\}$
Sample $i$: $s_i = \{x_1^i, x_2^i, ..., x_p^i\}$
Feature $i$: $f_i = \{x_i^1, x_i^2, ..., x_p^n\}$
Set of Positive Instances: $S^+$
Set of Negative Instances: $S^-$
Set of Classes: $C = \{c_1, c_2, ..., c_q\}$
Criterion : J(F,S)
Cardinality of X: $|X|$ is the number of elements in set X.
p: Number of features
n: Number of instances
d: Number of features selected or to select
k: Number of Classes
q: Number of disjoint subsets of S or number of classes

## B.2 Heuristic search Algorithms

### B.2.1 SFS:Sequential Forward Selection

**Algorithm:**

▷ $F' = \phi$

▷ For $i = 1$ to $d$

▷ Find $f_j \in F - F'$, where :

$$J(F' \bigcup \{f_j\}, S) = \max_{\forall f \in F - F'} J(F' \bigcup \{f\}, S)$$

[▷] $F' = F' \bigcup \{f_j\}$

▷ Return F'

### B.2.2 SBS:Sequential Backward Selection

**Algorithm:**

▷ $F' = F$

▷ For $i = 1$ to $(p - d)$

▷ Find $f_j \in F'$, where :

$$J(F' - \{f_j\}, S) = \min_{\forall f \in F'} J(F' - \{f\}, S)$$

▷ $F' = F' - \{f_j\}$

▷ Return F'

### B.2.3 Max-Min

**Algorithm:**

▷ Compute $J(f_i)$ and $J(f_i, f_j)$ $\forall i, j$ with $i \neq j$.

▷ Create $F_2'$ ($|F_i'| = i$) using any algorithm

▷ For $i = 3$ to $d$

▷ Include the new feature $f_i$ that satisfies

$$J(f_i) = \max_{\forall\ f_j \in F - F'_{i-1}} \quad \min_{\forall\ f_k \in F'_{i-1}} \Delta\ J(f_j, f_k)$$

where

$$\Delta\ J(f_j, f_k) = J(f_j, f_k) - J(f_k)$$

### B.2.4 SFFS:Sequential Float Forward Selection

**Algorithm:**

- $z = 2$
- Create $F_z$ such that $|F'_z| = 2$ using SFS
- While $z \neq d$:
  - Inclusion
    - $F'_{z+1} = f_i + F'_z$ such that:

$$J(f_i + F'_z) = \max_{\forall\, f \in F - F'_z} J(f + F'_z)$$

    - Find $f_j$ such that:

$$J(F'_z - f_j) = \min_{\forall\, f \in F'_z} J(F'_z - f)$$

    - If $i = j$ then $z = z + 1$ and go to Inclusion
  - Remotion
    - $F'_z = F'_{z+1} - f_j$
    - $z = z - 1$
    - If $|F'_z| = 2$ go to Inclusion
  - Next worst
    - Find $f_j$ such that:

$$J(F'_z - f_j) = \min_{\forall\, f \in F'_z} J(F'_z - f_j)$$

    - If $J(F'_z - f_j) \leq J(F'_{z-1})$ then go to Inclusion
    - $F'_z = F'_{z+1} - f_j$
    - $z = z - 1$
    - If $|F'_z| = 2$ go to Inclusion
    - Go to Next Worst

### B.2.5 SFBS:Sequential Float Backward Selection

**Algorithm:**

- $z = p - 2$
- Create $F_z$ such that $|F'_z| = p - 2$ using SBS
- While $z \neq d$:
  - Exclusion
    - $F'_{z-1} = F'_z - f_i$ such that:

$$J(F'_z - f_i) = \min_{\forall\, f \in F'_z} J(F'_z - f)$$

    - Find $f_j$ such that:

$$J(f_j + F'_z) = \max_{\forall\, f \in F - F'_z} J(f + F'_z)$$

    - If $i = j$ then $z = z - 1$ and go to Exclusion
  - Inclusion
    - $F'_{z+1} = F'_z + f_j$
    - $z = z + 1$
    - If $|F'_z| = p - 2$ go to Exclusion
  - Next Best
    - Find $f_j$ such that:

$$J(f_j + F'_z) = \max_{\forall\, f \in F - F'_z} J(f + F'_z)$$

    - If $i = j$ then $z = z - 1$ and go to Exclusion
  - Remotion
    - $F'_{z+1} = F'_z + f_j$
    - $z = z + 1$
    - If $|F'_z| = p - 2$ go to Exclusion
    - Go to Next Best

### B.2.6 Relief

**Definitions:**

$\tau$: relevance threshold (value picked should be less than $\frac{1}{\sqrt{\alpha n}}$, $\alpha$ is the desired Probability of Type I error).

Difference between two instances of features:

For nominal features:

$$diff(x, y) = \begin{cases} 0 & if\ x\ and\ y\ are\ the\ same \\ 1 & if\ x\ and\ y\ are\ different \end{cases}$$

For numerical features:

$$diff(x, y) = \frac{(x - y)}{NU} \quad where\ NU\ is\ the\ normalization\ unit$$

**Algorithm:**

- ▷ Separate S in $S^+$ and $S^-$
- ▷ W = (0, 0, ...., 0)
- ▷ For i = 1 to n
  - ▷ Pick at random an instance X ∈ S
  - ▷ Pick at random one of the positive instances ($X^+$) closest to X
  - ▷ Pick at random one of the negative instances ($X^-$) closest to X
  - ▷ If (X is a positive instance)
    then Near-hit = $X^+$ and Near-miss = $X^-$
    else Near-hit = $X^-$ and Near-miss = $X^+$
  - ▷ For j = 1 to p
    - ▷ $W_j = W_j - diff(x_j, Near - hit_j)^2 + diff(x_j, Near - miss_j)^2$
  - ▷ Relevance = $\frac{1}{n}W$
- ▷ For i = 1 to p
  - ▷ If ($Relevance_i < \tau$)
    then $f_i$ is a relevant feature
    else $f_i$ is a irrelevant feature

### B.2.7 EUBAFES: Euclidean Based Feature Selection

**Algorithm:**
Apply gradient descent for:

$$\frac{\partial J}{\partial W_q} \approx l * (\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \delta_{ij}^{kNN} (\delta_{ij} \frac{\rho(x_q^i, x_q^j)^2}{N_s} - (1 - \delta_{ij}) \frac{\rho(x_q^i, x_q^j)^2}{N_v}))$$

where :

- ▷ N: is the number of instances
- ▷ q: is the number of the current feature
- ▷ i: is the number of the current instance
- ▷ $\delta_{ij}$ : 1 if $x^i$ and $x^j$ are in the same class, 0 otherwise
- ▷ $\delta_{ij}^{kNN}$: 1 if $x^i$ is one of k nearest neighbors of $x^j$, 0 otherwise
- ▷ $\rho(x_q^i, x_q^j) = \begin{cases} |x_q^i - x_q^j| & if\ feature\ q\ is\ continuous \\ 1 & if\ feature\ q\ is\ discrete\ and\ x_q^i \neq x_q^j \\ 0 & if\ feature\ q\ is\ discrete\ and\ x_q^i = x_q^j \end{cases}$
- ▷ $N_s = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \delta_{ij}$
- ▷ $N_v = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} (1 - \delta_{ij})$
- ▷ l depends of $\tau$ by means of :

$$\sqrt{\sum_{q=1}^{Q} w_q \rho(x_q^i, x_q^j)^2 + \tau} \approx (\sum_{q=1}^{Q} w_q \rho(x_q^i, x_q^j)^2) * l + t$$

### B.2.8 LVF: Las Vegas Filter

**Definitions:** Max-Tries: maximum number of sets tested $\gamma$: maximum allowed inconsistency rate

**Algorithm:**

- ▷ Best = F
- ▷ p = number of features of Best

▷ For i = 1 to Max-Tries

▷ F' = random set of F

▷ d = number of features of f'

▷ if ($d < p$ and inconsistency(F') $< \gamma$)

▷ Best = F'

▷ p = d

▷ Return best

## B.3 Optimal Algorithms

### B.3.1 Focus

**Algorithm:**

▷ For i = 0, 1, ...

▷ For all $F' \subset F$ of size i

▷ If all pair of instances in S that agree on all features in F' do agree on the class then

return F'

exit

### B.3.2 Focus2

**Definitions:**

Conflict: a conflict is a binary vector for every pair of features that predict different classes with 1 where the feature values differ and 0 anywhere else.

$M_{A,B}$: is the space of all feature subsets that include all features of A and none of B.

Explained: a is explained by $x_i$ if $a_i$ is 1.

**Algorithm:**

▷ G = set of all conflits generated from S

▷ create queue $Q = \{M_{\phi,\phi}\}$

▷ Repeat

▷ Pop the first element in Q

▷ Let O = B

▷ Let **a** be the conflict in G not explained by any of the features in A such that $|Z_a - B|$ is minimized , where $Z_a$ is the set of features explaining a.

▷ For each $x \in Z_a - B$

▷ If Sufficient($A \bigcup \{x\}$), return ($A \bigcup \{x\}$)

▷ Insert $M_{A \bigcup \{x\},O}$ in Q

▷ $O = O \bigcup \{x\}$

### B.3.3 BB:Branch and Bound

**Conditions:** monotonicity of the criterion.
**Algorithm:**

▷ Bound = 0

▷ Avail = F

▷ List(0) = $\phi$

▷ Ns(0,1) = d + 1 (Ns = Number of successors)

▷ Pointer(0) = 1

▷ i = 1

▷ Initialize List

▷ Node = Pointer(i-1)

▷ Compute the criterion for all available features united with the subset $Z_1, Z_2, ..., Z_{i-1}$.

▷ Order the features

▷ s = Ns(i-1,Node)

▷ List(i) = s first features by the criterion

▷ Ns(i,j) = s + 1 - j, for j = 1,2, ..., s

▷ Remove List(i) from Avail

- ▷ Select Node
  - ▷ If List(i) = $\phi$: Go to Backtrack
  - ▷ $Z_i$ = last element of List(i)
  - ▷ Pointer(i) = number of elements of List(i)
  - ▷ Delete $Z_i$ of List(i)
- ▷ Check Bound
  - ▷ If J($Z_1, Z_2, ..., Z_i$) < Bound: Include $Z_i$ in Avail and Go to Backtrack
  - ▷ If i = d: Go to Update Bound
  - ▷ i = i + 1
  - ▷ Go to Initialize List
- ▷ Backtrack
  - ▷ i = i - 1
  - ▷ If i = 0: end (Desired set is the last saved)
  - ▷ Include $Z_i$ in Avail
  - ▷ Go to Select Node
- ▷ Update Bound
  - ▷ Bound = $J(Z_1, Z_2, ..., Z_i)$
  - ▷ Save $\{Z_1, Z_2, ..., Z_i\}$
  - ▷ Include $Z_i$ in Avail
  - ▷ Go to Backtrack

## Appendix C

# QUADRATIC FUNCTION

In creating quadratic functions it is important that the function is different from a linear function that passes through the same extremes (see Figure C.1). To guarantee that, only quadratic functions with difference from the linear function bigger than some threshold are accepted. The threshold is given in relation to the size of the triangle (area in light color at figure C.1) formed by $(x_{min}, y_{min})$, $(x_{max}, y_{max})$ and $(x_{max}, y_{min})$, assuming a positive slope for the linear function . The threshold used is 0.25. That means that the difference between the quadratic function and the linear function (dark color at figure C.1) has to be at least 25% of the area of the triangle. This threshold was selected because it gives some flexibility in selecting functions, given that the maximum value for the relation between the two areas is 0.33.

**Proof for maximum threshold**

Theorem: Given $y = ax^2 + bx + c$, a quadratic function; $x = \frac{-b+\sqrt{b^2-4\ a\ (c-y)}}{2\ a}$, the function to be computed; and (0,0) and (1,1) as extremes of $x$, the maximum area difference between the plot of $x$ and the plot of a line between (0,0) and (1,1) is 0.3333.

**Proof:**

The area under $x$ is:

$$\int_0^1 \frac{-b+\sqrt{b^2-4\ a\ (c-y)}}{2\ a}\,\partial y = \int_0^1 \frac{-b+\sqrt{b^2-4\ a\ c+4\ a\ y}}{2\ a}\partial y =$$

$$\left(\frac{-by}{2\ a} + \frac{2}{12\ a}\frac{(\sqrt{b^2-4\ a\ c+4\ a\ y})^3}{2\ a}\right)\Bigg]_0^1 = \frac{-b}{2\ a} + \frac{(\sqrt{b^2-4\ a\ c+4\ a})^3}{12\ a^2} - \frac{(\sqrt{b^2-4\ a\ c})^3}{12\ a^2} \tag{C.1}$$

The function that has the biggest difference with the linear function, is $y = ax^2$, where b and c are zero. Substituting the values in equation C.1 we have:

$$\frac{(\sqrt{4\ a})^3}{12\ a^2} = \frac{8\ a\ \sqrt{a}}{12\ a^2} = \frac{8\sqrt{a}}{12\ a} \tag{C.2}$$

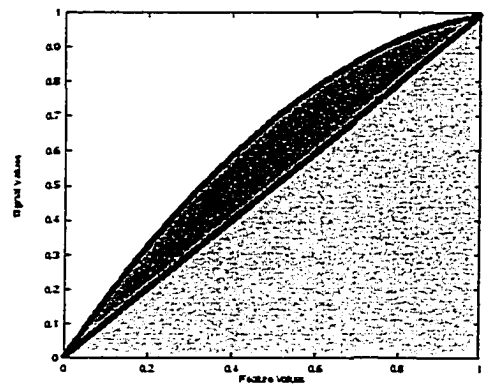


Figure C.1: Linear and quadratic function with same extremes.

For $y = ax^2$ the extremes (y = 0 and y = 1) correspond to $x = \sqrt{\frac{y}{a}} = 0$ and $\frac{1}{\sqrt{a}}$. So, the area of the triangle is:

$$\frac{(\frac{1}{\sqrt{a}} - 0)(1 - 0)}{2} = \frac{1}{2\ \sqrt{a}} \tag{C.3}$$

To find the relation between the quadratic function and the triangle it is necessary to divide equation C.2 by equation C.3:

$$\frac{\frac{8\ \sqrt{a}}{12\ a}}{\frac{1}{2\ \sqrt{a}}} = \frac{16\ a}{12\ a} = \frac{16}{12} = 1.33$$

The difference is then 0.33.

It is straight forward to prove the same to the second root and/or for negative slope of the linear function.

## Appendix D

# CORRELATION PROOFS

## D.1 Preliminaries

### D.1.1 Definitions [63]

Let A and B be random variables with mean $\mu_A$ and $\mu_B$ respectively.

The sample variance of $A$ is:

$$Var(A) = \frac{1}{n-1}\sum_{i=0}^{N}(A_i - \mu_A)^2 = \frac{n\sum_{i=0}^{N}A_i^2 - \left(\sum_{i=0}^{N}A_i\right)^2}{n\,(n-1)} \tag{D.1}$$

The standard deviation of $A$ is:

$$\sigma(A) = \sqrt{Var(A)} \tag{D.2}$$

The covariance between $A$ and $B$ is:

$$Cov(A,B) = \frac{1}{n-1}\sum_{i=0}^{N}(A_i - \mu_A)\,(B_i - \mu_B) = \frac{n\sum_{i=0}^{N}A_iB_i - \sum_{i=0}^{N}A_i\sum_{i=0}^{N}B_i}{n\,(n-1)} \tag{D.3}$$

The correlation between $A$ and $B$ is:

$$Corr(A,B) = \frac{Cov(A,B)}{\sigma(A)\,\sigma(B)} \tag{D.4}$$

### D.1.2 Important Properties [63]

Given A and B random variables and x and y scalar values.

P1:

$$Var(xA + y) = x^2 Var(A)$$

P2:

$$Cov(A, B) = Cov(B, A)$$

P3:

$$Cov(xA + y, B) = xCov(A, B)$$

### D.1.3 Basic Theorems

In proving the correlation theorems we will need two more simple theorems.

B1:

$$Cov(A + B, C) = Cov(A, C) + Cov(B, C)$$

**Proof:**

$$\begin{aligned}
Cov(A + B, C) &= \frac{n \sum_{i=0}^{N}(A_i + B_i)(C_i) - \sum_{i=0}^{N}(A_i + B_i) \sum_{i=0}^{N} C_i}{n\,(n-1)} \\
&= \frac{n \sum_{i=0}^{N}(A_i C_i) + n \sum_{i=0}^{N}(B_i C_i) - \left(\sum_{i=0}^{N} A_i + \sum_{i=0}^{N} + B_i\right) \sum_{i=0}^{N} C_i}{n\,(n-1)} \\
&= \frac{n \sum_{i=0}^{N}(A_i C_i) + n \sum_{i=0}^{N}(B_i C_i) - \sum_{i=0}^{N} A_i \sum_{i=0}^{N} C_i - \sum_{i=0}^{N} B_i \sum_{i=0}^{N} C_i}{n\,(n-1)} \\
&= \frac{n \sum_{i=0}^{N}(A_i C_i) - \sum_{i=0}^{N} A_i \sum_{i=0}^{N} C_i}{n\,(n-1)} + \frac{n \sum_{i=0}^{N}(B_i C_i) - \sum_{i=0}^{N} B_i \sum_{i=0}^{N} C_i}{n\,(n-1)} \\
&= Cov(A, C) + Cov(B, C)
\end{aligned}$$

B2:

$$Corr(xA, B) = Corr(A, B) \quad \forall x \neq 0$$

**Proof:**

$$\begin{aligned} Corr(xA, B) &= \frac{Cov(xA, B)}{\sigma(xA)\ \sigma(B)} \\ &= \frac{x\ Cov(A, B)}{\sqrt{Var(xA)}\ \sigma(B)} \\ &= \frac{x\ Cov(A, B)}{\sqrt{x^2 Var(A)}\ \sigma(B)} \\ &= \frac{x\ Cov(A, B)}{x\ \sqrt{Var(A)}\ \sigma(B)} \\ &= \frac{x\ Cov(A, B)}{x\ \sigma(A)\ \sigma(B)} \\ &= \frac{Cov(A, B)}{\sigma(A)\ \sigma(B)} \\ &= Corr(A, B) \end{aligned}$$

## D.2 Correlation Theorems

Theorem 1: Given two random variables $A$ and $B$, $C = \sigma(A)\ B + \sigma(B)\ A$ has the property that correlation between A and C is equal to correlation between B and C.

**Proof:**

$$\begin{aligned} Corr(A, C) - Corr(B, C) &= \frac{Cov(A, C)}{\sigma(A)\ \sigma(C)} - \frac{Cov(B, C)}{\sigma(B)\ \sigma(C)} \\ &= \frac{\sigma(B)\ Cov(A, C) - \sigma(A)\ Cov(B, C)}{\sigma(A)\ \sigma(B)\ \sigma(C)} \end{aligned}$$

$$
\begin{aligned}
&= \frac{\sigma(B)\ Cov(A,\ \sigma(B)\ A+\sigma(A)\ B)-\sigma(A)\ Cov(B,\ \sigma(B)\ A+\sigma(A)\ B)}{\sigma(A)\ \sigma(B)\ \sigma(C)} \\
&= \frac{\sigma(B)\ [Cov(A,\sigma(B)A)+Cov(A,\sigma(A)B)]-\sigma(A)\ [Cov(B,\sigma(B)A)+Cov(B,\sigma(A)B)]}{\sigma(A)\sigma(B)\sigma(C)} \\
&= \frac{\sigma(B)[\sigma(B)Cov(A,A)+\sigma(A)Cov(A,B)]-\sigma(A)[\sigma(B)Cov(B,A)+\sigma(A)Cov(B,B)]}{\sigma(A)\sigma(B)\sigma(C)} \\
&= \frac{\sigma(B)\ [\sigma(B)\ Var(A)+\sigma(A)\ Cov(A,\ B)]-\sigma(A)\ [\sigma(B)\ Cov(A,\ B)+\sigma(A)\ Var(B)]}{\sigma(A)\ \sigma(B)\ \sigma(C)} \\
&= \frac{Var(B)\ Var(A)+\sigma(B)\ \sigma(A)\ Cov(A,\ B)-\sigma(B)\ \sigma(A)\ Cov(A,B)-Var(A)\ Var(B)}{\sigma(A)\ \sigma(B)\ \sigma(C)} \\
&= \frac{0}{\sigma(A)\ \sigma(B)\ \sigma(C)} \\
&= 0
\end{aligned}
$$

Lemma 1: Given two random variables A and B and scalar x, any D of the form $D = x\ \sigma(A)\ B + x\ \sigma(B)\ A$ has the same correlation with A and B as $C = \sigma(A)\ B + \sigma(B)\ A$. Or $Corr(A,C) = Corr(B,C) = Corr(A,D) = Corr(B,D)$.

**Proof:**

Lemma 1 comes directly from B2. So,

$$
\begin{aligned}
Corr(A,\ D) &= Corr(A, x\ \sigma(A)\ B + x\ \sigma(B)\ A) \\
&= Corr(A, x\ [\sigma(A)\ B + \sigma(B)\ A]) \\
&= Corr(A, x\ C) \\
&= Corr(A,\ C)
\end{aligned}
$$

The same prove holds for $Corr(B,\ D) = Corr(B,\ C)$. and the theorem is proved.

Theorem 2: Given two random variables A and B and scalars x and y ($x \neq 0$ and $y \neq 0$), any D of the form $D = x\ \ \sigma(A)\ B + y\ \sigma(B)\ A$ has a smaller or equal average correlation with A and B than $C = \sigma(A)\ B + \sigma(B)\ A$.

**Proof:**

$$
\begin{aligned}
\frac{Corr(A,\ C)+Corr(B,\ C)}{2} &\geq \frac{Corr(A,\ D)+Corr(B,\ D)}{2} \Leftrightarrow \\
Corr(A,\ C)+Corr(B,\ C) &\geq Corr(A,\ D)+Corr(B,\ D) \Leftrightarrow \\
Corr(A,\ C)+Corr(B,\ C)-Corr(A,\ D)-Corr(B,\ D) &\geq 0
\end{aligned}
$$

$$
\begin{aligned}
&Corr(A,\ C) + Corr(B,\ C) - Corr(A,\ D) - Corr(B,\ D) \\
&= \frac{Cov(A,\ C)}{\sigma(A)\ \sigma(C)} + \frac{Cov(B,\ C)}{\sigma(B)\ \sigma(C)} - \frac{Cov(A,\ D)}{\sigma(A)\ \sigma(D)} - \frac{Cov(B,\ D)}{\sigma(B)\ \sigma(D)} \\
&= \frac{\sigma(B)\ Cov(A,\ C) + \sigma(A)\ Cov(B,\ C)}{\sigma(A)\ \sigma(B)\ \sigma(C)} - \frac{\sigma(B)\ Cov(A,\ D) + \sigma(A)\ Cov(B,\ D)}{\sigma(A)\ \sigma(B)\ \sigma(D)} \\
&= \frac{\sigma(B)\ \sigma(D)\ Cov(A,\ C) + \sigma(A)\ \sigma(D)\ Cov(B,\ C)}{\sigma(A)\ \sigma(B)\ \sigma(C)\ \sigma(D)} \\
&\quad - \frac{\sigma(B)\ \sigma(C)\ Cov(A,\ D) + \sigma(A)\ \sigma(C)\ Cov(B,\ D)}{\sigma(A)\ \sigma(B)\ \sigma(C)\ \sigma(D)} \\
&= \frac{\sigma(B)\ \sigma(D)\ [\sigma(A)\ Cov(A,\ B) + \sigma(B)\ Cov(A,\ A)]}{\sigma(A)\ \sigma(B)\ \sigma(C)\ \sigma(D)} \\
&\quad + \frac{\sigma(A)\ \sigma(D)\ [\sigma(A)\ Cov(B,\ B) + \sigma(B) Cov(B,\ A)]}{\sigma(A)\ \sigma(B)\ \sigma(C)\ \sigma(D)} \\
&\quad - \frac{\sigma(B)\ \sigma(C)\ [x\ \sigma(A)\ Cov(A,\ B) + y\ \sigma(B)\ Cov(A,\ A)]}{\sigma(A)\ \sigma(B)\ \sigma(C)\ \sigma(D)} \\
&\quad - \frac{\sigma(A)\ \sigma(C)\ [x\ \sigma(A)\ Cov(B,\ B) + y\ \sigma(B) Cov(B,\ A)]}{\sigma(A)\ \sigma(B)\ \sigma(C)\ \sigma(D)} \\
&= \frac{\sigma(A)\ \sigma(B)\ \sigma(D)\ Cov(A,\ B) + \sigma(D)\ Var(B)\ Var(A)}{\sigma(A)\ \sigma(B)\ \sigma(C)\ \sigma(D)} \\
&\quad + \frac{\sigma(D)\ Var(A)\ Var(B) + \sigma(A)\ \sigma(B)\ \sigma(D)\ Cov(A,\ B)}{\sigma(A)\ \sigma(B)\ \sigma(C)\ \sigma(D)} \\
&\quad - \frac{x\ \sigma(A)\ \sigma(B)\ \sigma(C)\ Cov(A,\ B) + y\ \sigma(C)\ Var(B)\ Var(A)}{\sigma(A)\ \sigma(B)\ \sigma(C)\ \sigma(D)} \\
&\quad - \frac{x\ \sigma(C)\ Var(A)\ Var(B) + y\ \sigma(A)\ \sigma(B)\ \sigma(C)\ Cov(A,\ B)}{\sigma(A)\ \sigma(B)\ \sigma(C)\ \sigma(D)} \\
&= \frac{2\ \sigma(A)\ \sigma(B)\ \sigma(D)\ Cov(A,\ B) + 2\ \sigma(D)\ Var(A)\ Var(B)}{\sigma(A)\ \sigma(B)\ \sigma(C)\ \sigma(D)} \\
&\quad - \frac{(x+y)\ \sigma(A)\ \sigma(B)\ \sigma(C)\ Cov(A,\ B) + (x+y)\ \sigma(C)\ Var(B)\ Var(A)}{\sigma(A)\ \sigma(B)\ \sigma(C)\ \sigma(D)} \\
&= \frac{[2\ \sigma(D) - (x+y)\ \sigma(C)]\ [\sigma(A)\ \sigma(B)\ Cov(A,\ B) + Var(A)\ Var(B)]}{\sigma(A)\ \sigma(B)\ \sigma(C)\ \sigma(D)} \\
&= \frac{[2\ \sigma(D) - (x+y)\ \sigma(C)]\ [\frac{\sigma(A)\ \sigma(B)}{\sigma(A)\ \sigma(B)}\sigma(A)\ \sigma(B)\ Cov(A,\ B) + Var(A)\ Var(B)]}{\sigma(A)\ \sigma(B)\ \sigma(C)\ \sigma(D)} \\
&= \frac{[2\ \sigma(D) - (x+y)\ \sigma(C)]\ [Var(A)\ Var(B)\ Corr(A,\ B) + Var(A)\ Var(B)]}{\sigma(A)\ \sigma(B)\ \sigma(C)\ \sigma(D)} \\
&= \frac{[2\ \sigma(D) - (x+y)\ \sigma(C)]\ Var(A)\ Var(B)\ [1 - Corr(A,\ B)]}{\sigma(A)\ \sigma(B)\ \sigma(C)\ \sigma(D)}
\end{aligned}
$$

$\sigma(A)\ \sigma(B)\ \sigma(C)\ \sigma(D)$ is always greater than zero, unless one of the variables is constant. $Var(A)\ Var(B)\ [1 - Corr(A,\ B)]$ is always greater or equal zero because variances are always positives or zero and Corr(A, B) varies between one and minus one. So we only have to prove that $2\ \sigma(D) - (x + y)\ \sigma(C)$ is greater than zero to prove the theorem.

$$
\begin{aligned}
& 2\ \sigma(D) - (x+y)\ \sigma(C) \\
&= 2\ \sqrt{Var(D)} - (x+y)\ \sqrt{Var(C)} \\
&= 2\ \sqrt{Var(x\ \sigma(B)\ A + y\ \sigma(A)\ B)} - (x+y)\sqrt{Var(\sigma(B)\ A + \sigma(A)\ B)} \\
&= 2\ \sqrt{Var(x\ \sigma(B)\ A) + Var(y\ \sigma(A)\ B) + 2\ Cov(x\ \sigma(B)\ A,\ y\ \sigma(A)\ B)} \\
&\quad -(x+y)\ \sqrt{Var(\sigma(B)\ A) + Var(\sigma(A)\ B) + 2\ Cov(\sigma(B)\ A,\ \sigma(A)\ B)} \\
&= 2\ \sqrt{x^2\ Var(B)\ Var(A) + y^2\ Var(A)\ Var(B) + 2\ xy\ \sigma(A)\ \sigma(B)\ Cov(A, B)} \\
&\quad -(x+y)\ \sqrt{Var(B)\ Var(A) + Var(A)\ Var(B) + 2\ \sigma(A)\ \sigma(B)\ Cov(A, B)} \\
&= 2\ \sqrt{(x^2+y^2)\ Var(A)\ Var(B) + 2\ xy\ Var(A)\ Var(B)\ Corr(A, B)} \\
&\quad -(x+y)\ \sqrt{2\ Var(B)\ Var(A) + 2\ Var(A)\ Var(B)\ Corr(A, B)} \\
&= \sqrt{4\ [x^2 + y^2 + 2xy\ Corr(A,\ B)]\ Var(A)\ Var(B)} \\
&\quad -\sqrt{2[x^2 + y^2 + 2xy]\ Var(B)\ Var(A)\ [1 + Corr(A,\ B)]}
\end{aligned}
$$

Because $\sigma$ and variance are always greater or equal zero we need only to prove that $4\ [x^2+y^2+2xy\ Corr(A,\ B)]$ is greater than or equal to $2[x^2+y^2+2xy][1+Corr(A,\ B)]$ to prove that $2\ \sigma(D) - (x + y)\ \sigma(C)$ is greater than or equal to zero.

$$
\begin{aligned}
& 4\ [x^2 + y^2 + 2xy\ Corr(A,\ B)] - 2[x^2 + y^2 + 2xy]\ [1 + Corr(A,\ B)] \\
&= 4x^2 + 4y^2 + 8xy\ Corr(A,\ B) - 2x^2 - 2y^2 - 4xy - 2x^2\ Corr(A, B) \\
&\quad -2y^2\ Corr(A,\ B) - 4xy\ Corr(A,\ B) \\
&= 2x^2 + 2y^2 - 4xy - 2x^2\ Corr(A,\ B) - 2y^2\ Corr(A,\ B) + 4xy\ Corr(A,\ B) \\
&= 2[x^2 + y^2 - 2xy]\ [1 - Corr(A,\ B)] \\
&= 2(x-y)^2\ [1 - Corr(A,\ B)]
\end{aligned}
$$

$1 - Corr(A,\ B)$ is always greater than or equal to zero. At the same time for any value of $x$ and $y$, $(x - y)^2$ is also greater than or equal to zero. So the equation is always greater or equal zero and the theorem is proved.

## Bibliography


[1] Edward H. Adelson and James R. Bergen, *"The Plenoptic Function and the Elements of Early Vision"*,In Computational Models of Visual Processing, M. Landy and J. Moyshon eds., Cambridge, MA, MIT Press, pp. 3-20, 1991.

[2] David W. Aha and Richard L. Bankert, *"A Comparative Evaluation of Sequential Feature Selection"*, Learning from Data: AI and Statistics V, Edited by D. Fisher and H.J. Lenz, Springer-Verlag, 1996, pp. 200-206.

[3] David W. Aha *"Tolerating Noisy, Irrelevant and Novel Attributes in Instance-based Learning Algorithms"*, International Journal of Man-machine Studies, V. 36, N. 1, 1992, pp. 267-287.

[4] Hussein Almuallim and Thomas G. Dietterich, *"Efficient Algorithms for Identifying Relevant Features"*, Proceedings of the $9^{th}$ Canadian Conference on Artificial Intelligence, Vancouver, May 1992.

[5] Hussein Almuallim and Thomas G. Dietrterich, *"Learning with Many Irrelevant Features"*, Proceedings of the $9^{th}$ National Conference on Artificial Intelligence (AAAI-91), 1991.

[6] E. Backer and J. A. De Shipper, *"On the Max-Min Approach for Feature Ordering and Selection"*, Proceedings of the Seminar on Pattern Recognition, Liege, Nov, 1977.

[7] E. Backer, *"Computer-assisted Reasoning in Cluster Analysis"*, Prentice-Hall, 1995.

[8] J. Bala, K. De Jong, J. Huang, H. Vafaie and H. Weschler, *"Using Learning to Facilitate the Evolution of Features for Recognizing Visual Concepts"*, Evolutionary Computation (Special Issue: The Baldwin Effect), V. 4, N. 3, pp. 297-311, 1996.

[9] J. Bala, K. De Jong, J. Huang, H. Vafaie and H. Weschler, *"Hybrid Learning using Genetic Algorithms and Decision Trees for Pattern Classification"*, Proceedings of the $14^{th}$ International Joint Conference on Artificial Intelligence (IJCAI), 1995.

[10] J. Bala, K. De Jong, J. Huang, H. Vafaie and H. Weschler, *"Visual Routine for Eye Detection using Hybrid Genetic Architectures"*, Proceedings of the $14^th$ International Joint Conference on Artificial Intelligence (IJCAI), 1996.

[11] Moshe Ben-Bassat, *"Use of Distance Measures, Information Measures and Error Bounds in Feature Evaluation"*, Handbook of Statistics, V. 2, P.R. Krishnaiah & L.N. Kanal eds., North Holland Publishing Company, 1982.

[12] Jesse Bennett and Alireza Khotanzad, *"Multi-spectral Random Field Models for Synthesis and Analysis of Color Images"*, IEEE Transactions on Pattern Analysis and Machine Intelligence , V. 20, n. 3, pp. 327-332, 1998.

[13] Elie Bienstock, Donald Geman, Stuart Geman and Donald E. McClure, *"Development of Laser Radar ATR Algorithms: Phase II - military objects"*, Technical Report, Mathematical Technologies Inc, Providence, Rhode Island, oct. 1990.

[14] Jeff Bilmes, *"A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models"*, Technical Report ICSI-TR-97-021, U.C Berkeley, 1997.

[15] Lisa Belue and Keneth Bauer, *" Determining input features for multilayer perceptrons"*, Neurocomputing, V. 7, pp. 111-121, 1995.

[16] J. Ross Beveridge, Joey Griffith, Ralf R. Kohler, Allen R. Hanson, and Edward M. Riseman, *"Segmenting Images using Localized Histograms and Regions Merging"*, International Journal of Computer Vision, N 2, pp. 311-347, 1989.

[17] A. L. Blum and R. L. Rivest, *"Training a 3-node Neural Network is NP-complete"*, Neural Networks, V. 5, pp. 117-127, 1992.

[18] P. S. Bradley and Usama M Fayyad, *"Refining Initial Points for K-means Clustering"*, To appear in: Proceedings of the Fiftennth International Conference on Machine Learning, Morgan Kaufmann, July 1998.

[19] L. Breiman, J.H. Friedman, R.A. Olshen and C.J. Stone, *"Classification and Regression Trees"*, Wadsworth & Brooks, California, 1984.

[20] M. van Breukelen, R.P.W. Duin, D.M.J. Tax and J.E. den Hartog, *"Combining Classifiers for the Recognition of Handwritten digits"*, Kybernetika, V. 34, N. 4, 1998, pp. 381-386.

[21] M. van Breukelen, R.P.W. Duin, D.M.J. Tax and J.E. den Hartog, *"Handwritten Digit Recognition by Combined Classifiers"*, In: Proceedings of the $1^{st}$ IAPR TC1 Workshop on Statistical Techniques in Pattern Recognition, Prage, 1997, pp. 13-18.

[22] T. Calinski and J. Harabasz, *"A Dendrite Method for Cluster Analysis, " Communications in Statistics"*, V. 3, 1974, pp.1-27

[23] Rich Caruana and Dayne Freitag, *"Greedy Attribute Selection"*, Proceedings of the $11^{th}$ International Conference on Machine Learning, 1994.

[24] Giovanna Castelano, Anna Maria Fanelli and Marcelo Pelillo, *"An Iterative Prunning Algorithm for Feedforward Neural Networks"*, IEEE Transactions on Neural Networks, V. 8, N. 3, pp. 519-531, 1997.

[25] Ives Chauvin, *"A Back-propagation Algorithm With Optimal Use of Hidden Units"*, Advances in Neural Information Processing Systems, V. 1, Denver, Co, Morgan Kaufmann Publishers, pp. 519-526, 1988.

[26] Kevin J. Cherkauer, Jude W. Shavlik, *"Rapidly Estimating the Quality of Input Representations for Neural Networks"*, IJCAI-95 Workshop on Data Engineering for Inductive Learning.

[27] T. M. Cover, *"The Best Two Independent Measurements Are Not the Two Best"*, IEEE Transaction on Systems, Man and Cybernetics, SMC-4, pp. 116-117, 1974.

[28] T. M. Cover and J.A.Tomas, *"Elements of Information Theory"*, Wiley-Interscience, 1991.

[29] Frederick E. Croxton, *"Elementary Statistics with Applications in Medicine and the Biological Sciences"*, Dover Publications Inc., New York, 1953.

[30] Yann Le Cun, John S. Denker and Sara A. Solla, *"Optimal Brain Damage"*, Advances in Neural Information Processing Systems, San Mateo, Ca, Morgan Kaufmann Publishers, pp. 598-605, 1989.

[31] A.P.Dempster, N.M. Laird, and D.B. Rubin, *"Maximun-Likelyhood from Incomplete Data via the EM Algorithm"*, Journal of the Royal Statistical Society B, V. 39, pp. 1-39, 1991.

[32] Sandor Z. Der and Rama Chellappa, *"Probe Based Recognition of Targets in Infrared Images"*, Technical Report CS-TR-3174/CAR-TR-693, University of Maryland, Nov. 1993.

[33] K. Diamantaras and S. Y. Kung, *"Principal Components Neural Networks: Theory ans Applications"*, Wiley, New York, 1996.

[34] J. Doak, *"An Evaluation of Feature Selection Methods an Their Application to Computer Security"*, Technical Report CSE-92-18, Davis, Ca: University of California, Department of Computer Science, 1992.

[35] Bruce A. Draper, Jose Bins, Kyungim Baek. *"ADORE: Adaptive Object Recognition"*, International Conference on Vision Systems ( ICVS ), Los Palmas, Gran Canaria (Spain), pp. 522-537, Jan. 13-15, 1999.

[36] R.O.Duda and P.E.Hart, —emph"Pattern Classification and Scene Analysis", John Wiley and Sons, 1973.

[37] Freeman and Adelson, *" The Design and Use of Steerable Filters"*, IEEE TRans. on PAttern Analysis and Machine Intelligence, V. 13, N. 9, pp. 891-906, Sept. 1991

[38] Yoav Freund and Robert E. Shapire, *"A Decision-theoretic Generalization of On-line Learning and an Appication to Boosting"*, Proceedings of the Second European Conference on Computational Learning Theory, pp. 23-37, Springer-Verlag, 1995.

[39] Cesar Guerra-Salcedo and Darrel Whitley, *"Genetic Approach to Feature Selection for Ensemble Creation"*, 1999.

[40] Xuan Guorong, Chai Peiqi and Wu Munhui, *"Bhattacharyya Distance Feature Selection"*, Proceedings of International Conference on Pattern Recognition (ICPR), pp. 195-199, 1996.

[41] Mark A. Hall *"Correlation-based Feature Selection for Discrete and Numeric Class Machine Learning"*, In proceedings odf the $17_th$ Conference in Machine Leaning (ICML2000), June 29-July 2, 2000, pp. 359-65.

[42] Yoshihiko Hamamoto, Shunji Uchimura, Yutaka Matsura, Taiho Kanaoka and Shingo Tomita, *"Evaluation of the Branch and Bound Algorithm for Feature Selection"*, Pattern Recognition Letters, N. 11, 1990, pp. 453-456.

[43] A.R. Hanson, E.R. Riseman and P. Nagin, *"Region Growing in Textured Outdoors Scenes"*, Tech. Report TR 75C-3, University of Massachusetts, Amherst, 1975.

[44] Robert M. Haralick and Linda G. Shapiro, *"Computer and Robot Vision"*, V. 1 & 2, Addison Wesley, Reading, MA, 1992.

[45] Mohamad H. Hassoun, *"Fundamentals of Artificial Neural Networks"*, Mit Press, Cambridge, 1995.

[46] L. Hyafil and R. L. Rivest, *"Constructing Optimal Binary Decision Trees is NP-complete"*, Information Processing Letters, V. 5, N. 1, pp. 15-17, 1976.

[47] Anil Jain and Douglas Zongker, *"Feature Selection: Evaluation, Application, and Small Sample Performance"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, V. 19, N. 2, Feb. 1997.

[48] Anil Jain and B. Chandrasekaran, *"Dimensionality and Sample Size Considerations"*, In P.R. Krishnaiah and I.N. Kanal editors, Pattern Recognition Practice, V. 2, pp. 835-855, North-Holland, 1982.

[49] Anil K. Jain, Robert P.W. Duin, and Jianchang Mao, *"Statistical Pattern Recognition: A review"*, IEEE Transactions on Pattern Recognition and Machine Intelligence (PAMI), V. 22, N. 1, Jan. 2000.

[50] Byung Hwan Jun, Chang Soo Km and Jaihie Kim, *"A New Criterion for Selection and Discretization of Attributes for the Generation of Decision Trees"*, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), V. 19, N. 12, Dec. 1997.

[51] Michael Kearns, Yishay Mansour and Andrew Y. Ng, *"An Information-Theoretic Analysis of hard and Soft Assignment methods for Clustering"*, Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence, 1997.

[52] Charles Kervrann and Fabrice Heitz, *"A Markov Random Field Model-Based Approach to Unsupervised Texture Segmentation using Local and Global Spatial Statistics"*, TR: 752, Institut de Recherxhe en Informatique et Systemes Aleatories (IRISA), France, 1993.

[53] M. Kirby and L Sirovich, *"Applications of the Karhunen-Loève Procedure for the characterization of Human Faces"*, IEEE Transaction on Pattern Analisys and Machine Inteligence, 12(1):103-108, 1990.

[54] Kenji Kira and Larry A. Rendell, *"The Feature Selection Problem: Traditional Methods and a New Algorithm"*, Proceedings of the $10^{th}$ National Conference on Machine Intelligence (AAAI-92), 1992, pp. 129-134.

[55] J. Kittler, *"Feature Set search Algorithms"*, Pattern Recognition and Signal Processing, C.H. Chen Editor, Sijthoff & Noordhoff, Alphen aan den Rijn, The Netherlands, 1978, pp. 41-60.

[56] Ron Kohavi, *"Feature Subset Selection as Search with Probabilistic Estimates"*, AAAI Fall Symposium on Relevance, 1994.

[57] Ron Kohavi and George H. John, *"Wrappers for Feature Subset Selection"*, Technical Report, Computer Science Department, Stanford University, 1995, published to in: Artificial Intelligence Journal (Special Issue on Relevance), V. 97, N. 1-2, pp. 273-324, 1997.

[58] R.R. Kohler, *"Integrating Non-Semantic Knowledge into Image Segmentation Processes"*, Ph.D. thesis, COINS, University of Massachussetts, Amherst, 1984.

[59] T. Kohonen, *"Self-Organizing Maps"*, Springer-Varlag, Berlin, 1995.

[60] Daphne Koller and Mahran Sahami, *"Toward Optimal Feature Selection"*, Proceedings of the $13^{th}$ International Conference on Machine Learning, Bari, Italy, Jul. 1996.

[61] I. Kononenko, *"Estimation attributes: Analysis and Extensions of RELIEF"*,in Proceedings of the European Conference on Machine Learning, Catana, Italy, pp. 171-182, Springer Verlag, 1994.

[62] LANDSAT-1, European Space Agency, http://earth1.esrin.esa.it:81/spglandsat1.

[63] Richard Larsen and Morris Marx, *"An Introduction to Mathematical Statistics and its Applications"*, Prentice-Hall, Englewood Cliffs, NJ, 1986.

[64] P. M. Lewis, *"The Characteristic Selection Problem in Recognition Systems"*, IEEE Transactions on Information Theory, IT-8, pp. 171-178, 1962.

[65] Huan Liu and Rudy Setiono, *"Incremental Feature Selection"*, Applied Intelligence, V. 9, N. 3, pp. 217-230, Kluwer Academic Publishers, 1998.

[66] J. MacQueen, *"Some Methods for Classification and Analysys of Multivariate Observations"*, Proceedings of the Fifth Berkley Symposium on Mathematics, Statistics and Probability, L. M. LeCam and J. Neyman eds., pp. 281-297, University of California Press, Berkeley, 1967.

[67] M. Maloof, P. Langley, S. Sage and T. Binford. *"Learning to Detect Rooftops in Aerial Images"*, IUW, 835-846, 1997.

[68] Deba Prasad Mandal, *"Partitioning of Feature Space For Pattern Classification* Pattern Recognition, V. 30, N. 12, pp. 1971-1990, 1997.

[69] Jianchang Mao, K. Mohiuddin, and Anil K. Jain, *"Parsimonious Network Design and Feature Selection Through Node Pruning"*, Proceedings of the $12^{th}$ International Conference on Pattern Recognition, Jerusalen, pp. 622-624, 1994.

[70] T. Marill and D. M. Green, *"On the Effectiveness of Receptors in Recognition Systems"*, IEEE transactions on Information Theory, V. 9, 1963, pp. 11-17.

[71] BogdamMatei, Peter Meer and David Tyler, *"Performance Assesment by Resampling: Rigid Motion Estimators"*, Empirical Evaluation Techniques in Computer Vision, pp. 72-95, Kevin W Bowyer and P. Jonathon Phillips eds., IEEE Computer Society Press, Washington, 1998.

[72] D. McKeown, W. Harvey and J. McDermott. *"Rule-Based Interpretation of Aerial Imagery"*, Pattern ANalysis and Machine Intelligence (PAMI), 7(5), pp. 570-585, 1985.

[73] Tom Mitchell *"Machine Learning"*, McGraw-Hill, Boston, 1997.

[74] Hiroshi Murase and Shree K. Nayar, *"Visual Learning and Recognition of 3-D Objects from Appearance"*, International Journal of Computer Vision, V. 14, pp. 5-24, 1995.

[75] Patrenahalli Narendra and Keinosure Fukunaga, *"A Branch and Bound Algorithm for Feature Subset Selection"*, IEEE Transactions on Computers, V. 26, N. 9, sept. 1977, pp. 917-922.

[76] Hwee Tou Ng, Wei Boon Goh and Kok Leong Low, *"Feature Selection, Perceptron Learning, and a Usability Case Study for text Categorization"*, SIGIR '97. Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval, July 27-31, Philadelphia, pp. 67-73, 1997.

[77] E. Oja, J. Karhunen, L. Wang and R. Vigario, *"Principal and Independent Components in Neural Networks - Recent Developments"*, Proceedings VII Italian Workshop on Neural Nets (WIRN'95), May 18 - 20, Vietri sul Mare, Italy, 1995.

[78] G. Pagallo and D. Haussler, *"Boolean Feature discovery in empirical learning"*, Machine learning, V. 5, N. 1, pp. 71-100, 1990.

[79] Karl Pearson, *"Notes on the History of Correlation"*, Biometrica, V. 13, 1920-1921.

[80] Dan Pelleg and Andrew Moore, *"X-means: Extending K-means with Efficient Estimation of the Number of Clusters"*, In proceedings of the $17^{th}$ Conference in Machine Leaning (ICML2000), June 29-July 2, 2000, pp. 727-34.

[81] Jing Peng, Bir Bhanu and Shan Qing, *"Probabilistic Feature Relevance Learning for Content-Based Image Retrieval*, Image Understanding, (to appear).

[82] W. H. Press, S. A. Teukolsky, W. T. Veterling and B. P. Flanerry, *"Numerical Recipes in C"*, Cambridge University Press, 2nd edition, 1992.

[83] P. Pudil, J. Novovicova and J. Kittler, *"Floating Search Methods in Feature Selection"*, Pattern Recognition Letters, V. 15, 1994, pp. 1119-1125.

[84] W.F. Punch, E.D. Goodman, Min Pei, Lai Chia-Shun, P. Hovland, R. Enbody, *"Further Research on Feature Selection and Classification using Genetic Algorithms"*, In Stephanie Forrest, ed., Proceedings of the $5^{th}$ International Conference in Genetic Algorithms, pp. 557-564, Morgan Kaufman, 1993.

[85] J.R. Quinlan, *"Induction of Decision Trees"*, Machine Learning, V. 1, pp. 81-106, 1986.

[86] J.R. Quinlan, *"C4.5: Programs for Machine Learning"*, San Mateo, Calif, Morgan Kaufman, 1993.

[87] J.R. Quinlan, *"Decision Trees and Multivalued Attributes"*, J. Richards, ed., Machine Intelligence, V. 11, Oxford, England, Oxford Univ. Press, pp. 305-318, 1988.

[88] S. Ravela, B. Draper, J. Lim and R. Weiss, *"Adaptative Tracking and Model Registration Across Distinct Aspects"*, International Conference on Intelligent Robots and Systems, Pittsburgh, PA, V. 1, pp. 174-180, Aug. 1995.

[89] Reuters. Reuters collection available via anonymous ftp. ftp://ciir-ftp.cs.umass.edu.pub/reuters1, 1995.

[90] W. Richards and A. Jepson, *"What Makes a Good Feature"*, TR-MIT-AI-1356, Apr. 1992.

[91] Robert E. Shapire, *"The Streanth of Week Learnability"*, Machine Learning, V. 5, N. 2, pp. 197-227, 1990.

[92] Robert E. Shapire, Yoav Freund, Peter Bartlett and Wee Sun Lee, *"Boosting the Margin: A new Explanation for the Effectiveness of Voting Methods"*,Proceedings of the Fourteenth International Conference on Machine Learning, 1997.

[93] M. Scherf and W. Brauer, *"Improving RBF-Networks by the Feature Selection Approach EUBAFES"*, In Proceedings of the $7^{th}$ International Conference on Artificial Neural Networks, ICANN97, pp. 391-396. Lausanne, Switzerland, 1997.

[94] Bernt Schiele and James L. Crowley, *"Recognition without Correspondence using Multidimensional Receptive Field Histograms"*, M.I.T. Media Laboratory, Perceptual Computing Section Technical Report No. 453, Dec. 1997, submitted to International Journal on Computer Vision (IJCV).

[95] Rudy Setiono and Huan Liu, *"Neural-Network Feature Selector"*, IEEE Trans. on Neural Networks, V. 8, N. 3, pp. 654-662, May 1997.

[96] W. Siedlecki and J. Sklansky, *"On Automatic Feature Selection"*, International Journal of Pattern Recognition and Artificial Intelligence, V. 2, N. 2, pp. 197-220, 1988.

[97] D. G. Sim, O.K. Kwon and R.H. Park, *"Object Matching Algorithms using Robust Hausdorff Distance Measures"*, IEEE Transactions on Image Processing, V. 8, N. 3, pp. 425-429, Mar. 1999.

[98] S. D. Stearns, *"On Selecting Features for Pattern Classifiers"*, Third International Conference on Pattern Recognition, Coronado, CA, pp. 71-75, 1976.

[99] Mark R. Stevens, J. Ross Beveridge and Michael E. Goss, *"Visualizing Multisensor Model-Based Object Recognition"*, Machine Graphics and Vision, V. 6, N. 3, pp. 279-303, 1998. also in : Reconnaissance, Surveillance and Target Acquisition (RSTA) for the Unmanned Ground Vehicle: Providing Eyes for an Autonomous Ground Vehicle. Morgan Kaufmann Publishers, 1997.

[100] M. Swain and D. Ballard, *"Color Indexing"*, International Journal of Computer Vision, V. 7, N. 11, pp. 11-32, 1991.

[101] Kinh Tieu and Paul Viola. *"Boosting Image Retrieval"*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2000), 2000, pp. 228-235.

[102] G. T. Toussaint, *"Note on the Optimal Selection of Independent Binary Features for Pattern Recognition"*, IEEE transactions on information theory. IT-17, pp. 618, 1971.

[103] Ruck Thawonmas and Shigeo Abe, *"A Novel Approach to Feature Selection Based on Analysis of Class Regions* , IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, V. 27, N. 2, Apr. 1997.

[104] Mathew Turk and Alex Petland, *"Eigenfaces for Recognition"*, Journal of Cognitive Neuroscience, V. 3, N. 1, 1991.

[105] Haleh Vafaie and Kenneth De Jong, *"Robust Feature Selection Algorithms"*, Proceedings of the IEEE International Conference on Tools with Artificial Intelligence, Boston, Mass., pp. 356-363, Nov. 1993.

[106] Haleh Vafaie and Ibrahim Imam, *"Feature selection Methods: Genetic Algorithm vs. Greedy Like Search"*, Proceedings of the IEEE International Conference on Fuzzy and Intelligent Control Systems, 1994.

[107] Vector Signal Image Processing Library (VSIPL) Home page: http://www.vsipl.org/

[108] Vector Signal Image Processing Library (VSIPL) Image Processing operations manual: http://www.vsipl.org/PubInfo/Imageman.pdf

[109] http://www.etl.go.jp/etl/divisions/ vaie/neuroFAQ/FAQ.html#A_Kohonen

[110] A. Wayne Whitney, *"A Direct Method of Non-parametric Feature Selection"*, IEEE Transactions on Computers, V. 20, Sept. 1971, pp. 1100-1103.

[111] Wendy S. Yambor, Bruce A. Draper and J. Ross Beveridge, *"Analyzing PCA-based Face Recognition Algorithms: Eigenvector Selection and Distance Measures"*, $2^{nd}$ Workshop on Empirical Evaluation in Computer Vision, Dublin, Ireland, July 1, 2000.

[112] Wendy S. Yambor, *"Analysis of PCA-Based and Fisher Discriminant-Based Image Recognition Algorithms."*, M.S. Thesis, Colorado State University, 2000.