ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

ΤΜΗΜΑ ΙΑΤΡΙΚΗΣ

ΤΜΗΜΑ ΦΥΣΙΚΗΣ

ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ
ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΣΤΗΝ ΙΑΤΡΙΚΗ ΦΥΣΙΚΗ

ΑΞΙΟΛΟΓΗΣΗ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ ΓΙΑ ΤΟ
ΧΑΡΑΚΤΗΡΙΣΜΟ ΑΛΛΟΙΩΣΕΩΝ ΣΕ ΣΥΣΤΗΜΑΤΑ
ΥΠΟΒΟΗΘΟΥΜΕΝΗΣ ΔΙΑΓΝΩΣΗΣ ΣΤΗΝ ΙΑΤΡΙΚΗ
ΑΠΕΙΚΟΝΙΣΗ

EMMANOULA POLENES

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΠΑΤΡΑ 2008
UNIVERSITY OF PATRAS

SCHOOL OF MEDICINE    DEPARTMENT OF PHYSICS

INTERDEPARTMENTAL PROGRAMME
OF POSTGRADUATE STUDIES
IN MEDICAL PHYSICS

EVALUATION OF NEURAL NETWORKS FOR
CHARACTERIZATION IN COMPUTER AIDED DIAGNOSIS
IN MEDICAL IMAGING

MANOLIS POLENIS

MASTER THESIS

PATRAS 2008
ACKNOWLEDGMENTS

I would like to express my gratitude to Professor George Panayiotakis for his trust for giving me the opportunity to participate in such knowledgeable process as this postgraduate course.

I would like to thank my official advisor Assistant Professor Lena Costaridou for her support and helpful advises on this work.

Many thanks to my colleagues in the laboratory for their support and guidance through this work. Especially to Anna Karahaliou, Panayiotis Korfiatis, Spyros Skiadopoulos and Nikos Arikidis for trusting me their datasets for evaluation.

I also thank Pantelis Theockarakis for his willingness to helping me in many steps of this master thesis.

Special thanks to Nikos Arikidis and Spyros Skiadopoylos for their scientific support and advises.
# Table of contents

Table of contents

<table>
<thead>
<tr>
<th>Περίληψη</th>
<th>iii</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>v</td>
</tr>
<tr>
<td>Introduction</td>
<td>vi</td>
</tr>
<tr>
<td>Thesis layout</td>
<td>vi</td>
</tr>
<tr>
<td>Chapter 1</td>
<td>1</td>
</tr>
<tr>
<td>Pattern recognition or classification</td>
<td>1</td>
</tr>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Pattern recognition systems</td>
<td>2</td>
</tr>
<tr>
<td>Chapter 2</td>
<td>5</td>
</tr>
<tr>
<td>Neural networks</td>
<td>5</td>
</tr>
<tr>
<td>Introduction</td>
<td>5</td>
</tr>
<tr>
<td>What is neural network [4]</td>
<td>5</td>
</tr>
<tr>
<td>The biological model: The human brain</td>
<td>6</td>
</tr>
<tr>
<td>Models of a neuron</td>
<td>7</td>
</tr>
<tr>
<td>Neuron Model</td>
<td>8</td>
</tr>
<tr>
<td>Transfer Functions</td>
<td>9</td>
</tr>
<tr>
<td>Neuron with vector input</td>
<td>10</td>
</tr>
<tr>
<td>Network Architectures</td>
<td>10</td>
</tr>
<tr>
<td>Knowledge</td>
<td>12</td>
</tr>
<tr>
<td>Learning process [4]</td>
<td>13</td>
</tr>
<tr>
<td>Error-correction learning</td>
<td>14</td>
</tr>
<tr>
<td>Learning rules</td>
<td>14</td>
</tr>
<tr>
<td>Back-propagation algorithm</td>
<td>15</td>
</tr>
<tr>
<td>Understanding back-propagation</td>
<td>15</td>
</tr>
<tr>
<td>Principles of training multi-layer neural network using back-propagation algorithm</td>
<td>16</td>
</tr>
<tr>
<td>Generalization [4]</td>
<td>24</td>
</tr>
<tr>
<td>Early stopping method of training</td>
<td>25</td>
</tr>
<tr>
<td>Chapter 3</td>
<td>26</td>
</tr>
<tr>
<td>Introduction</td>
<td>26</td>
</tr>
<tr>
<td>Components of the GUI</td>
<td>26</td>
</tr>
<tr>
<td>Input component</td>
<td>26</td>
</tr>
<tr>
<td>Pre-processing component</td>
<td>26</td>
</tr>
<tr>
<td>Evaluation component</td>
<td>34</td>
</tr>
<tr>
<td>Network evaluation component (Network parameters)</td>
<td>35</td>
</tr>
<tr>
<td>Network component (network parameters)</td>
<td>37</td>
</tr>
<tr>
<td>Chapter 4</td>
<td>40</td>
</tr>
<tr>
<td>The experiment</td>
<td>40</td>
</tr>
<tr>
<td>Material and methods</td>
<td>40</td>
</tr>
<tr>
<td>Software and Hardware</td>
<td>40</td>
</tr>
<tr>
<td>The Data sets</td>
<td>40</td>
</tr>
<tr>
<td>The classifiers</td>
<td>41</td>
</tr>
<tr>
<td>Evaluation methodology</td>
<td>43</td>
</tr>
<tr>
<td>Statistical validation method</td>
<td>45</td>
</tr>
<tr>
<td>Seeking the best network topology</td>
<td>46</td>
</tr>
<tr>
<td>Best classifier</td>
<td>46</td>
</tr>
<tr>
<td>Preprocessing</td>
<td>47</td>
</tr>
</tbody>
</table>
Περίληψη

Το αντικείμενο αυτής της εργασίας είναι οι ταξινομητές στα συστήματα υποβοηθούμενης διάγνωσης στην ιατρική απεικόνιση. Ειδικότερα, εστιάζει στα τεχνητά νευρονικά δίκτυα καθώς και σε μεθόδους επιλογής χαρακτηριστικών.

Οι στόχοι αυτής της εργασίας είναι:

1. Η αναζήτηση της βέλτιστης τοπολογίας ενός πρόσωπου κατευθυντικού νευρονικού δικτύου, σε τέσσερα (4) προβλήματα ταξινόμησης ιατρικής απεικόνισης (κυτταρολογία, μαγνητική απεικόνιση, αξονική τομογραφία και μαστογραφία).

2. Η μελέτη τριών (3) μεθόδων επιλογής χαρακτηριστικών (μείωσης διαστάσεων) συμπεριλαμβανομένων της ανάλυσης κύριων συνιστώσων, της σταδιακής αναζήτησης και της κατάταξης κατά τ-τέστ για τα τέσσερα (4) προβλήματα ταξινόμησης που είχαμε στη διάθεσή μας.

3. Η σύγκριση της απόδοσης του πρόσωπου κατευθυντικού νευρονικού δικτύου (FFNN) με τους KNN, SVM, PNN και LDA ταξινομητές, στα τέσσερα (4) προαναφερθέντα ιατρικά προβλήματα ταξινόμησης. Για την εκτίμηση της απόδοσης γενίκευσης (σφάλμα γενίκευσης) χρησιμοποιήθηκε η 10-πτυχή διασταυρούμενη επικύρωση.

4. Η στατιστική σημαντικότητα των αποτελεσμάτων ελέγχθηκε με τις δοκιμασίες της ανάλυσης της διακύμανσης κατά ένα παράγοντα (ANOVA) και της δοκιμασίας Duncan.

Για την διευκόλυνση του πειραματικού μέρους αναπτύχθηκε μια φιλική στο χρήστη εφαρμογή που επιτρέπει την αναζήτηση της βέλτιστης τοπολογίας του νευρονικού δικτύου για τα επεξεργασμένα χαρακτηριστικά, και τις προεπεξεργασίες τεχνικές προ-επεξεργασίας, ενώ επιτρέπει και την σύγκριση του με άλλους ταξινομητές.

Τα αποτελέσματα του πειραματικού μέρους αυτής της εργασίας είναι:

1. Δεν αποδεικνύεται στατιστικά ότι η τοπολογία του δικτύου έχει κάποια επίδραση στην απόδοση του στα τέσσερα αυτά προβλήματα που μελετήθηκαν.
2. Η μέθοδος σταδιακής αναζήτησης είναι στατιστικά καλύτερη μέθοδος για επιλογή χαρακτηριστικών (τη μείωση των διαστάσεων), εκτός από το ένα πρόβλημα που αφορούσε στην κυτταρολογία όπου δεν αποδείχθηκε στατιστικά σημαντική διαφορά μεταξύ των μεθόδων. Αυτό οφείλεται στο γεγονός ότι η μέθοδος σταδιακής αναζήτησης έχει την "ενδογενή" ικανότητα να επιλέγει χαρακτηριστικά που είναι ανεξάρτητα μεταξύ τους με αποτέλεσμα την αυξανόμενη διακριτική ικανότητα του τελικού συνόλου (τα προβλήματα που η μέθοδος αυτή επέδειξε καλή συμπεριφορά είχαν χαρακτηριστικά με υψηλό βαθμό συσχέτισης).

3. Δεν αποδεικνύεται στατιστικά καλύτερος ταξινομητής στις περισσότερες περιπτώσεις ενώ ο νευρωνικός ταξινομητής επιδεικνύει πολύ καλή συμπεριφορά σε όλες τις περιπτώσεις. Για το λόγο αυτό, η επιλογή του ταξινομητή δεν φαίνεται να επηρεάζει σε σημαντικό βαθμό την απόδοση του συστήματος στα προβλήματα που έχουν μελετηθεί εδώ. Επιπλέον, η επιλογή του ταξινομητή μπορεί να γίνει με όρους διαφορετικούς από την ταξινομητική ικανότητά τους όπως απλότητα και ευελιξία, χαρακτηριστικά που έχει ο νευρωνικός ταξινομητής.
Abstract

This thesis is dealing with classifiers in Computer Aided Diagnosis in medical imaging. In particular, it focuses on artificial neural networks and feature selection methods.

The specific goals of the thesis are:

1. Search for optimal topology of a feed-forward neural network (FFNN), dealing with four (4) medical imaging classification problems (Cytology, MRI, CT, and Mammography).
2. Study three (3) feature selection (dimensionality reduction) methods including PCA, stepwise analysis and t-test ranking for the FFNN topology defined in the previous step, for the four (4) medical imaging classification problems at hand.
3. Compare performance of the FFNN scheme to KNN, SVM, PNN and LDA classifiers, dealing with the above mentioned four (4) medical imaging classification problems. 10-fold cross validation estimation of generalization performance (generalization error) of the classification schemes was utilized.
4. Statistical significance of the results was validated utilizing ANOVA and Duncan’s test.

To facilitate experimentation, a user-friendly application was developed (Chapter 3) that allows the user to find the best network topology on feature vectors, selected by various pre-processing techniques, and compared with other classifiers.

The results of this are:

1. There is no statistical evidence that the different topology that is tested have any impact on classification performance of FFNN in any of the classification problem that this thesis is dealt off.
2. The stepwise method of dimensionality reduction (feature selection) is statistically significance better method than the other methods, except in the case of one dataset (Cytology) where there are no statistical significant differences. This is because of the inherent ability of stepwise method to
select uncorrelated features unlike the other two methods (the datasets that the stepwise featured better performance had many highly correlated features).

3. There is no statistical significant better classifier in most cases, while neuronal classifier exhibits very good behaviour on all cases. For that reason, the selection of classifier does not seem to affect the classification problems at hand. Furthermore, the choice of classifier could be done based on other criteria than the classification performance, such as, the simplicity and plasticity, features that characterize the FFNN.
Introduction

This thesis is dealing with the pattern recognition and its application in medical imaging in particular, formulated as computer aided diagnosis. Specifically two components of a CAD system favored our attention namely dimensionality reduction method and classification. Although

Thesis layout

This thesis is organized in five chapters as follows

In Chapter 1, the theoretical part of general pattern recognition is presented.

In Chapter 2, the theoretical part of the network architecture as well as some characteristics of it is presented.

In chapter 3, the user-friendly application (GUI) that is developed in this thesis is presented. In this chapter, also the theory behind each component of GUI is presented.

In chapter 4, the classification experiments are contacted using the GUI and the results is presented.

Finally, in chapter 5 there is a discussion on the results of this experiment as well as future steps.
Chapter 1

Pattern recognition or classification

Introduction

The goal of pattern recognition is the classification of objects into a number of categories or classes. Depending on the application, these objects can be images, signal waveforms, or any type of measurements that need to be classified. We will refer to these objects using the generic term patterns. Pattern recognition has a long history, but before the 1960s, it was mostly the output of theoretical research in the area of statistics. As with everything else, the advent of computers increased the demand for practical applications of pattern recognition, which in turn set new demands for further theoretical developments. As our society evolves from the industrial to its postindustrial phase, automation in industrial production and the need for information handling and retrieval are becoming increasingly important. This trend has pushed pattern recognition to the high edge of today’s engineering applications and research. Pattern recognition is an integral part in most machine intelligence systems built for decision-making.

Computer-aided diagnosis is an important application of pattern recognition, aiming at assisting doctors in making diagnostic decisions. The final diagnosis, of course, is made by the doctor. Computer-assisted (or aided) diagnosis has been applied to and is of interest for a variety of medical data, such as X-rays, computed tomographic images, ultrasound images, electrocardiograms (ECGs), and electroencephalograms (EEGs). The need for a computer–aided diagnosis stems from the fact that medical data are often not easily interpretable, and the interpretation can depend very much on the skill of the doctor. Let us take for example X-ray mammography for the detection of breast cancer. Although mammography is currently the best method for detecting breast cancer, 10% - 30% of women who have the disease and undergo mammography have negative mammograms. In approximately two thirds of these cases with false results, the radiologist failed to detect the cancer, which was evident retrospectively. This may be due to poor image quality, eye fatigue of the radiologist, or the subtle nature of the findings. The percentage of correct classifications improves at a second reading by another radiologist. Thus, one can aim to develop a pattern recognition system in order to assist radiologists with a “second”
opinion. Increasing confidence in the diagnosis based on mammograms, in turn, would decrease the number of patients with suspected breast cancer who have to undergo surgical breast biopsy, with its associated complications. [1]

**Pattern recognition systems**

A pattern recognition system is composed of several structural components as shown in the following diagram:

![Figure 1: The basic stages involved in the design of a classification system](image)

The goal of a pattern recognition system is to classify inputs into different classes depending on the characteristics of those inputs.

**Sensor**

The input to a pattern recognition system is often some kind of a transducer. In our situation is the imaging system we used to produce the mammographic images. Such components include the screen-film system used to produce the original image as well as the scanner used to digitize it and recently this modality tent to be replaced by the computed or the digital radiography. In this thesis we do not concerns the limitation of those components.

**Feature generation**

This is a difficult task, and includes the detection of the object under question (mass in our case) and the isolation from the background (segmentation) as well as the feature extraction from those objects. Feature generation is a procedure that computes new variables that in one way or another originate from the stored values of the image array. The goal is to generate features that exhibit high information packing properties, from the class separability point of view. The traditional goal of the feature extractor is to characterize an object to be recognized by measurements whose values are very similar for objects in the same category, and very different for objects in different categories.

The need for feature generation stems from our inability to use the raw data. Even for a small 64 x 64 image the number of pixels is 4096. For most classification tasks, this number is too large, rising computational as well as generalization problems.
Because we cannot use the raw data directly, the features should encode efficiently the relevant information residing in the original data [1].

The feature generation is a problem depended task, that is, the features that are good for a problem may not be the best choice for another one. In this thesis, we do not concern about this task.

**Feature selection**

Because in real life we never know which features are the best ones, we usually have to choose among a plethora of them. In this thesis, we do not develop a feature extractor component, but rather we have a pool of feature to choose. Thus, the feature selection task was an important step in this thesis, and for that reason, we put some effort on this.

**Classifier design**

The task of the classifier component proper of a full system is to use the feature vector provided by the feature selection step to assign the object to a category. The degree of difficulty in the classification task depends on the variability of the feature values for objects in different categories. This task is the major goal of this thesis. The classifier of choice for this thesis is the multi-layer perceptron or the feed-forward error back-propagation neural network.

**System evaluation**

This is the last goal of this thesis. The evaluation of a classification system is one of the major issues in measuring the system’s performance. [2]

This is the last stage of the design procedure of a classification system. In other words, we assume that an optimal classifier has been designed, based on a selected set of training feature vectors. Our goal now is to evaluate its performance with respect to the probability of classification error associated with the designed system. Once the estimated error is considered satisfactory, full evaluation of the system performance is carried out in the real environment for which the system has been designed, such as a hospital for a medical diagnosis system or a factory for an industrial production-oriented system.

The evaluation of the system’s performance will determine whether the designed system complies with the requirements imposed by the specific application and intended use of the system. If this is not the case, the designer may have to reconsider
and redesign parts of the system. Furthermore, the misclassification probability can also be used as a performance index, in the feature selection stage, to choose the best features associated with a specific classifier [1].

Estimating the accuracy of a classifier induced by a supervised learning algorithm is important not only in order to predict its future performance, but also in order to choose a classifier from a given set (model selection), or in order to combine classifiers. For model selection, it is best to use k-fold cross-validation for low values of k (i.e., \( k \leq 10 \)), even if computation power allows using more folds (e.g., leave-one-out) [3].
Chapter 2

Neural networks

Introduction

The main purpose of this thesis is the evaluation of the neural networks as a classification modality. Thus in this chapter we introduce the design, the capabilities as well as the limitations of such classifiers from a bibliography point of view.

Because, firstly, the application was developed totally in the MATLAB® environment, and secondly there none globally accepted terminology about the neural networks will give some notations that will help to expand the neural network theory.

Here we will also introduce some concepts concerning the network such as training and what a training rule is about, the ability to generalize, the randomness of its nature and for this purpose, we will use as a vehicle the simplest network that the Rosenblatt has created the single layer perceptron. Later on, we expand the theory of the back-propagation algorithm used with the neural network that we concern.

What is neural network [4]

Work on artificial neural networks, commonly referred to as "neural networks," has been motivated right from its inception by the recognition that the human brain computes in an entirely different way from the conventional digital computer. The brain is a highly complex, nonlinear, and parallel computer (information-processing system). It has the capability to organize its structural constituents, known as neurons, to perform certain computations (e.g., pattern recognition, perception, and motor control) many times faster than the fastest digital computer in existence today. Consider, for example, human vision, which is an information-processing task (Marr, 1982; Levine, 1985; Churchland and Sejnowski, 1992). It is the function of the visual system to provide a representation of the environment around us and, more important, to supply the information we need to interact with the environment. To be specific, the brain routinely accomplishes perceptual recognition tasks (e.g., recognizing a familiar face embedded in an unfamiliar scene) in approximately 100-200 ms, whereas tasks of much lesser complexity may take days on a conventional computer.

In its most general form, a neural network is a machine that is designed to model the way in which the brain performs a particular task or function of interest; the
network is usually implemented by using electronic components or is simulated in software on a digital computer.

Neural networks have a history of some six decades but have found solid application only in the past twenty years. The field is still developing rapidly. Thus, it is distinctly different from the fields of control systems or optimization, where the terminology, basic mathematics, and design procedures have been established and applied for many years. [5]

**The biological model: The human brain**

As mentioned above the inspiration of the artificial neural networks, at least in the beginning, were the neurons in the brain.

The human brain consists of a large number (more than a billion) of neural cells that process information. Each cell works like a simple processor and only the massive interaction between all cells and their parallel processing makes the brain's abilities possible.

Below you see a figure of such a neural cell, called a neuron:

![Structure of a Typical Neuron](image)

**Figure 2: Structure of a neural cell in the human brain**

As the figure indicates, a neuron consists of a cell body (core), dendrites for incoming information and an axon with dendrites for outgoing information that is passed to connected neurons.
Information is transported between neurons in form of electrical stimulations along the dendrites.

Incoming information that reaches the neuron's dendrites is added up and then delivered along the neuron's axon to the dendrites at its end, where the information is passed to other neurons if the stimulation has exceeded a certain threshold. In this case, the neuron is said to be activated.

If the incoming stimulation had been too low, the information will not be transported any further. In this case, the neuron is said to be inhibited.

The connections between the neurons are adaptive, what means that the connection structure is changing dynamically. It is commonly acknowledged that the learning ability of the human brain is based on this adaptation [6].

**Models of a neuron**

An artificial neuron is an information-processing unit that is fundamental to the operation of a neural network. The block diagram of Figure 3 shows the model of a neuron, which forms the basis for designing (artificial) neural networks.

![Figure 3: Model of an artificial neuron](image)

Here we identify three basic elements of the neuronal model:

A set of synapses or connecting links, each of which is characterized by a weight or strength of its own. Specifically, a signal $x_i$ at the input of synapse $j$ connected to neuron $k$ is multiplied by a synaptic weight $w_{kj}$.

An adder for summing the input signals, weighted by the respective synapses of the neuron.
An activation function for limiting the amplitude of the output of a neuron. The activation function is also referred to as a squashing function in that it squashes (limits) the permissible amplitude range of the output signal to some finite value. Typically, the normalized amplitude range of the output of a neuron is written as the closed unit interval \([0,1]\) or alternatively \([-1,1]\).

The bias \(b_k\) has the effect of increasing or decreasing the net input of the activation function, depending on whether it is positive or negative respectively.

**Neuron Model**

A neuron with a single scalar input and no bias appears on the left below.

The scalar input \(p\) is transmitted through a connection that multiplies its strength by the scalar weight \(w\) to form the product \(wp\), again a scalar. Here the weighted input \(wp\) is the only argument of the transfer function \(f\), which produces the scalar output \(a\). The neuron on the right has a scalar bias, \(b\). You can view the bias as simply being added to the product \(wp\) as shown by the summing junction or as shifting the function \(f\) to the left by an amount \(b\). The bias is much like a weight, except that it has a constant input of 1.

The transfer function net input \(n\), again a scalar, is the sum of the weighted input \(wp\) and the bias \(b\). This sum is the argument of the transfer function \(f\).

Here \(f\) is a transfer function, typically a step function or a sigmoid function, that takes the argument \(n\) and produces the output \(a\). Examples of various transfer functions are in “Transfer Functions” on page 9. Note that \(w\) and \(b\) are both adjustable scalar parameters of the neuron. The central idea of neural networks is that such parameters can be adjusted so that the network exhibits some desired or interesting behavior. Thus, you can train the network to do a particular job by adjusting the weight or bias parameters, or perhaps the network itself will adjust these parameters to achieve some desired end.
Transfer Functions

The activation function, denoted by $f(n)$, defines the output of a neuron in terms of the induced local field $n$. Here we identify three basic types of activation functions.

\[
a = \text{hardlim}(n) = \begin{cases} 
1 & \text{if } n \geq 0 \\
0 & \text{if } n < 0
\end{cases}
\]

**Figure 5: Hard limit transfer function** $a = \text{hardlim}(n)$

The hard-limit (threshold function) transfer function shown above, limits the output of the neuron to either 0, if the net input argument $n$ is less than 0, or 1, if $n$ is greater than or equal to 0. This function is used in “Perceptrons,” to create neurons that make classification decisions.

The linear transfer function is shown below.

\[
a = \text{purelin}(n)
\]

**Figure 6: Linear transfer function** $a = \text{purelin}(n)$

The sigmoid (logistic) transfer function shown below takes the input, which can have any value between plus and minus infinity, and squashes the output into the range 0 to 1.

\[
a = \logsig(n) = \frac{1}{1 + e^{-n}}
\]

**Figure 7: Sigmoid transfer function** $a = \logsig(n)$

This transfer function is commonly used in back-propagation networks, in part because it is differentiable.

---

1 Or more precisely logistic distribution function
The activation functions defined above, range from 0 to +1. It is sometimes desirable to have the activation function range from -1 to +1, in which case the activation function assumes an anti-symmetric form with respect to the origin; that is, the activation function is an odd function of the induced local field.

The symbol in the square to the right of each transfer function graph shown above represents the associated transfer function. These icons replace the general $f$ in the boxes of network diagrams to show the particular transfer function being used.

**Neuron with vector input**

A neuron with a single R-element input vector is shown below. Here the individual element inputs $p_1, p_2, ..., p_R$ are multiplied by weights $w_{1,1}, w_{1,2}, ..., w_{1,R}$ and the weighted values are fed to the summing junction. Their sum is simply $Wp$, the dot product of the (single row) matrix $W$ and the vector $p$.

![Figure 8: Neuron with vector input](image)

**Figure 8: Neuron with vector input** $a = f(Wp + b)$. **R:** number of elements in input vector

The neuron has a bias $b$, which is summed with the weighted inputs to form the net input $n$. This sum, $n$, is the argument of the transfer function $f$.

$$n = w_{1,1}p_1 + w_{1,2}p_2 + ... + w_{1,R}p_R + b$$

This expression can, of course, be written in MATLAB® code as $n = \mathbf{w} \cdot \mathbf{p} + b$.

**Network Architectures**

The manner in which the neurons of a neural network are structured is intimately linked with the learning algorithm used to train the network. We may therefore speak of learning algorithms (rules) used in the design of neural networks as being *structured*.

To describe networks having multiple layers, the notation must be extended. Specifically, it needs to make a distinction between weight matrices that are connected to inputs and weight matrices that are connected between layers. It also needs to identify the source and destination for the weight matrices.
We will call weight matrices connected to inputs input weights; and we will call weight matrices coming from layer outputs layer weights. Further, superscripts are used to identify the source (second index) and the destination (first index) for the various weights and other elements of the network. To illustrate, the one-layer multiple input network shown earlier is redrawn in abbreviated form below.

As you can see, the weight matrix connected to the input vector $p$ is labeled as an input weight matrix ($IW^{1,1}$) having a source 1 (second index) and a destination 1 (first index). Elements of layer 1, such as its bias, net input, and output have a superscript 1 to say that they are associated with the first layer.

“Multiple Layers of Neurons” uses layer weight ($LW$) matrices as well as input weight ($IW$) matrices.

**Multilayer feed-forward networks**

A class of a feed-forward neural network that we used in this thesis distinguishes itself by the presence of one or more hidden layers, whose computation nodes are correspondingly called hidden neurons or hidden units. The function of hidden neurons is to intervene between the external input and the network output in some useful manner.

By adding one or more hidden layers, the network is enabled to extract higher-order statistics. In a rather loose sense, the network acquires a global perspective despite its local connectivity due to the extra set of synaptic connections and the extra dimension of neural interactions (Churchland and Sejnowski, 1992). The ability of hidden neurons to extract higher-order statistics is particularly valuable when the size of the input layer is large.

The source nodes in the input layer of the network supply respective elements of the activation pattern (input vector), which constitute the input signals applied to the neurons (computation nodes) in the second layer (i.e., the first hidden layer). The output signals of the second layer are used as inputs to the third layer, and so on for the
rest of the network. Typically, the neurons in each layer of the network have as their inputs the output signals of the preceding layer only [4].

A network of this type can have several layers. Each layer has a weight matrix \( W \), a bias vector \( b \), and an output vector \( a \). To distinguish between the weight matrices, output vectors, etc., for each of these layers in the figures, the number of the layer is appended as a superscript to the variable of interest. You can see the use of this layer notation in the three-layer network shown below, and in the equations at the bottom of the figure.

\[
a^1 = f^1(IW^{1,1}p + b^1) \\
a^2 = f^2(LW^{2,1}a^1 + b^2) \\
a^3 = f^3(LW^{3,1}a^2 + b^3)
\]

Figure 10: A three-layer network in abbreviated notation

The layers of a multilayer network play different roles. A layer that produces the network output is called an output layer. All other layers are called hidden layers. The three-layer network shown earlier has one output layer (layer 3) and two hidden layers (layer 1 and layer 2). Some authors refer to the inputs as a fourth layer.

Multiple-layer networks are quite powerful. For instance, a network of two layers, where the first layer is sigmoid and the second layer is linear, can be trained to approximate any function (with a finite number of discontinuities) arbitrarily well. This is the type of network that this thesis is focused.

**Knowledge**

A major task for a neural network is to learn a model of the world (environment) in which it is embedded and to maintain the model sufficiently consistent with the real world to achieve the specified goals of the application of interest. Knowledge of the world consists of two kinds of information:

The known world state, represented by facts about what is and what has been known; this form of knowledge is referred to as prior information.
Appendix: Analysis of variance

Observations (measurements) of the world, obtained by means of sensors designed to probe the environment in which the neural network is supposed to operate. Ordinarily these observations are inherently noisy, being subject to errors due to sensor noise and system imperfections. In any event, the observations so obtained provide the pool of information from which the examples used to train the neural network are drawn.

In a neural network of specified architecture, knowledge representation of the surrounding environment is defined by the values taken on by the free parameters (i.e., synaptic weights and biases) of the network. The form of this knowledge representation constitutes the very design of the neural network, and therefore holds the key to its performance [4].

Learning process [4]

The property that is of primary significance for a neural network is the ability of the network to learn from its environment, and to improve its performance through learning. Mendel and McIrarren (1970) give the following definition of learning: “Learning is a process by which the free parameters of a network are adapted through a process of stimulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameter changes take place”

This definition of the learning process implies the following sequence of events:

1. The neural network is stimulated by an environment
2. The neural network undergoes changes in its free parameters because of this stimulation
3. The neural network responds in a new way to the environment because of the changes that have occurred in its internal structure.

A prescribed set of well-defined rules for the solution of a learning problem is called a learning algorithm\(^2\). There is no unique learning algorithm for the design of neural networks. Rather, we have a "kit of tools" represented by a diverse variety of learning algorithms, each of which offers advantages of its own. Learning algorithms differ from each other in the way in which the adjustment to a synaptic weight of a neuron is formulated. In this thesis, the error correction learning is used.

\(^2\) The word "algorithm" is derived from the name of the Persian mathematician Mohammed al-Kowarisimi, (Algorismus in Latin)
**Error-correction learning**

In this type of learning the output signal of a neural network is compared to a desired response or target output, Figure 11.

![Neural network block diagram](image)

**Figure 11: Neural network block diagram**

Consequently, an error signal is produced. This error signal actuates a control mechanism, the purpose of which is to apply a sequence of corrective adjustments to the synaptic weights of neuron in the network. The corrective adjustments are designed to make the output signal come closer to the desired response in a step-by-step manner, until the system reaches a steady state (i.e., the synaptic weights are essentially stabilized). At that point, the learning process is terminated.

This objective is achieved by minimizing a cost function or index of performance, and for the mean squares error cost function, leads to a learning rule commonly referred to as the delta rule or Widrow-Hoff rule, named in honor of its originators.

This type of learning will be discussed in much detail later as this is the method of learning of the feed-forward error back-propagation neural networks.

**Learning rules**

By learning rule, we mean a procedure for modifying the weights and biases of a network. (This procedure may also be referred to as a training algorithm.) The purpose of the learning rule is to train the network to perform some task. There are many types of neural network learning rules. They fall into three broad categories: supervised learning, unsupervised learning and reinforcement (or graded) learning.

In supervised learning, the learning rule is provided with a set of examples (the training set) of proper network behavior: \( \{p_1, t_1\}, \{p_2, t_2\}, \ldots, \{p_Q, t_Q\} \), where \( p_i \) is an
Appendix: Analysis of variance

input to the network and $t_i$ is the corresponding correct (target) output. As the inputs are applied to the network, the network outputs are compared to targets. The learning rule is then used to adjust the weights and biases of the network in order to move the network outputs closer to the targets.

**Back-propagation algorithm**

The back-propagation algorithm can be summarized in the following five steps [4].

*Initialization:* Assuming that no prior information is available, pick the synaptic weights and thresholds from a uniform distribution whose mean is zero and whose variance is chosen to make the standard deviation of the induced local fields of the neurons lie at the transition between the linear and saturated parts of the sigmoid activation function.

*Presentations of Training Examples:* Present the network with an epoch of training examples. For each example in the set, ordered in some fashion, perform the sequence of forward and backward computations described under points 3 and 4, respectively.

*Forward Computation:* Compute the induced local fields and function signals of the network by proceeding forward through the network, layer by layer and finally compute the error signal as the difference between the output of the network and the target.

*Backward Computation:* Compute the local gradient of each neuron in the network defined by differentiation of the activation function of the neuron. (That is why it is important that the activation function has to be differentiable). Adjust the synaptic weights of the network according to the generalized *delta rule*.

*Iteration:* Iterate the forward and backward computations under points 3 and 4 by presenting new epochs of training examples to the network until the stopping criterion is met.

**Understanding back-propagation**

Beyond the mathematical explanation given in appendix, (see
Mathematical implementation of back propagation algorithm on p.71), there is an intuitive way to understand the back-propagation.

In fact, the whole back-propagation process is intuitively very clear. When a learning pattern is presented to the network, the activation values are propagated to the output units, and the actual network output is compared with the desired output values, we usually end up with an error in each of the output units. The goal is to reduce that error to zero. The simplest method to do this is to change the connections in the neural network in such a way that, next time around, the error will be zero for this particular pattern. We know from the delta rule that, in order to reduce an error, we have to adapt its incoming weights according to that error.

That is step one. However, it alone is not enough: when we only apply this rule, the weights from input to hidden units are never changed, and we do not have the full representational power of the feed-forward network as promised by the universal approximation theorem (that is a single hidden unit network can approximate any function with finitely many discontinuities to arbitrary precision). In order to adapt the weights from input to hidden units, we again want to apply the delta rule. In this case, however, we do not have a value for deltas for the hidden units. This is solved by the chain rule, which does the following: distribute the error of an output unit to all the hidden units that is it connected to, weighted by this connection. Differently put, a hidden unit receives a delta from each output unit equal to the delta of that output unit weighted with (= multiplied by) the weight of the connection between those units.

The application of the generalized delta rule thus involves two phases: During the first phase, each input is presented and propagated forward through the network to compute the output values for each output unit. This output is compared with its desired value, resulting in an error signal for each output unit. The second phase involves a backward pass through the network during which the error signal is passed to each unit in the network and appropriate weight changes are calculated [7].

Below is given an intuitive example of teaching process of a neural network

**Principles of training multi-layer neural network using back-propagation algorithm**

To illustrate this process the three layer neural network with two inputs and one output, which is shown in the picture below, is used:
Chapter 2: Neural networks

Each neuron is composed of two units. First unit adds products of weights coefficients and input signals. The second unit realizes nonlinear function, called neuron activation function. Signal $e$ is adder output signal, and $y = f(e)$ is output signal of nonlinear element. Signal $y$ is also output signal of neuron.

To teach the neural network we need training data set. The training data set consists of input signals ($x_1$ and $x_2$) assigned with corresponding target (desired output) $z$. The network training is an iterative process. In each iteration weights coefficients of nodes are modified using new data from training data set. Modification is calculated using algorithm described below: Each teaching step starts with forcing both input signals from training set. After this stage we can determine output signals values for each neuron in each network layer. Pictures below illustrate how signal is propagating through the network, Symbols $w_{(m)n}$ represent weights of connections between network input $x_m$ and neuron $n$ in input layer. Symbols $y_n$ represents output signal of neuron $n$. 
Chapter 2: Neural networks

Propagation of signals through the hidden layer. Symbols $w_{mn}$ represent weights of connections between output of neuron $m$ and input of neuron $n$ in the next layer.
Chapter 2: Neural networks

Propagation of signals through the output layer.

In the next algorithm step the output signal of the network $y$ is compared with the desired output value (the target), which is found in training data set. The difference is called error signal $\delta$ of output layer neuron.

It is impossible to compute error signal for internal neurons directly, because output values of these neurons are unknown. For many years, the effective method for training multiplayer networks has been unknown. Only in the middle eighties, the back-propagation algorithm has been worked out. The idea is to propagate error signal
\( \delta \) (computed in single teaching step) back to all neurons, which output signals were input for discussed neuron.

The weights' coefficients \( w_{mn} \) used to propagate errors back are equal to this used during computing output value. Only the direction of data flow is changed (signals are propagated from output to inputs one after the other). This technique is used for all network layers. If propagated errors came from few neurons, they are added. The illustration is below:
Chapter 2: Neural networks

When the error signal for each neuron is computed, the weights coefficients of each neuron input node may be modified. In formulas below $df(e)/de$ represents derivative of neuron activation function (which weights are modified).

$$\delta_2 = w_{24}\delta_4 + w_{25}\delta_5$$

$$\delta_3 = w_{34}\delta_4 + w_{35}\delta_5$$

$$W'_{(x1)1} = W_{(x1)1} + \eta \delta_1 \frac{df(e)}{de} x_1$$

$$W'_{(x2)1} = W_{(x2)1} + \eta \delta_1 \frac{df(e)}{de} x_2$$

$$y$$
Chapter 2: Neural networks

\[ w'_{(x_1)2} = w_{(x_1)2} + \eta \delta_2 \frac{df_2(e)}{de} x_1 \]

\[ w'_{(x_2)2} = w_{(x_2)2} + \eta \delta_2 \frac{df_2(e)}{de} x_2 \]

\[ w'_{(x_1)3} = w_{(x_1)3} + \eta \delta_3 \frac{df_3(e)}{de} x_1 \]

\[ w'_{(x_2)3} = w_{(x_2)3} + \eta \delta_3 \frac{df_3(e)}{de} x_2 \]
\[ w'_{14} = w_{14} + \eta \delta_4 \frac{df_4(e)}{de} y_1 \]
\[ w'_{24} = w_{24} + \eta \delta_4 \frac{df_4(e)}{de} y_2 \]
\[ w'_{34} = w_{34} + \eta \delta_4 \frac{df_4(e)}{de} y_3 \]
\[ w'_{15} = w_{15} + \eta \delta_5 \frac{df_5(e)}{de} y_1 \]
\[ w'_{25} = w_{25} + \eta \delta_5 \frac{df_5(e)}{de} y_2 \]
\[ w'_{35} = w_{35} + \eta \delta_5 \frac{df_5(e)}{de} y_3 \]
Coefficient $\eta$ affects network-teaching speed. There are a few techniques to select this parameter. The first method is to start teaching process with large value of the parameter. While weights coefficients are being established, the parameter is being decreased gradually. The second, more complicated, method starts teaching with small parameter value. During the teaching process, the parameter is being increased when the teaching is advanced and then decreased again in the final stage. Starting teaching process with low parameter value enables to determine weights coefficients signs.

**Generalization [4]**

In back-propagation learning, we typically start with a training sample and use the back-propagation algorithm to compute the synaptic weights of a multilayer perceptron by loading (encoding) as many of the training examples as possible into the network. The hope is that the neural network so designed will generalize. A network is said to generalize well when the input-output mapping computed by the network is correct (or nearly so) for test data never used in creating or training the network, the term generalization is borrowed form psychology. Here it is assumed that the test data are drawn from the same population used to generate the training data.

The learning process may be viewed as a “curve-fitting” problem. The network itself may be considered simply as a non-linear input-output mapping. Such a viewpoint then permits us to look on generalization not as a mystical property of neural networks but rather simply as the effect of a good nonlinear interpolation of the input data.

A neural network that is designed to generalize well will produce a correct input-output mapping even when the input is slightly different from the examples used to
train the network. When, however, a neural network learns too many input-outputs examples, the network may end up memorizing the training data. It may do so by finding a feature that is present in the training data but not true of the underlying function that is to be modeled. Such a phenomenon is referred to as over-fitting or overtraining. When the network is over-trained, it loses the ability to generalize between similar input-output patterns.

**Early stopping method of training**

We may identify the onset of over-fitting using cross-validation, for which the training data split into an estimation subset and a validation subset. The estimation subset of examples is used to train the network in the usual way, except for a minor modification: the training session is stopped periodically, and the network is tested on the validation subset after each period of training. After a period of estimation (training), the synaptic weights and bias levels of the multilayer perceptron are all fixed, and the network is operated in its forward mode (simulation). The validation error is thus measured for each example in the validation subset. When the validation phase is completed, the estimation (training) is resumed for another period, and the process is repeated. This procedure is referred as the early stopping method of training [4].
Introduction
In order to investigate the use of neural network to a decision-making system, a graphical user interface (GUI) was developed in MATLAB®. As the component of the application is presented, the associated theory is developed too.

Components of the GUI
The GUI is composed of four main components.

Input component
In this area, the user can insert a formatted text file or a Excel® type file, containing the feature vectors that represents the patterns (examples) for the network training and evaluation.

After inserting the file, the user has the option to select the classes, the wavelet level of the features, if any, as well as the features themselves to be evaluated by the network.

Pre-processing component
In this area, the user can choose from several pre-processing techniques in order to evaluate the impact on the performance of the neural network.
Chapter 3: Graphical User Interface

We can see the so-called preprocessing step as two separate stages both in implementation and in function.

The theory

According to Bishop [8]: The distinction between the pre-processing stage and the neural network is not always clear cut, but often the pre-processing can be regarded as a fixed transformation of variables, while the network itself contains adaptive parameters whose values are set as part of the training process. Since neural networks can perform essentially arbitrary non-linear functional mappings between sets of variables, in principle a single neural network could be used to map the raw input data directly onto the required final output values. In practice, for all but the simplest problems, such an approach will generally give poor results.

For most applications, it is necessary first to transform the data into some new representation before training the neural network. To some extent, the general-purpose nature of a neural network mapping degrades the importance of careful optimization of this pre-processing, but in many practical applications, the choice of pre-processing will be one of the most significant factors in determining the performance of the final system especially on generalization performance. In the simplest case, pre-processing may take the form of a linear transformation of the input data, and possibly also of the output (where it is sometimes termed post-processing). More complex pre-processing may involve reduction of the dimensionality of the data. The fact that such dimensionality reduction can lead to improved performance at first may appear somewhat paradoxical, since it cannot increase the information content of the input data, and in most cases will reduce it. The resolution is related to the curse of dimensionality [8].

One of the most important forms of pre-processing involves a reduction in the dimensionality of the input data. A network with fewer inputs has fewer adaptive parameters to be determined and these are more likely to be properly constrained by a data set of limited size, leading to a network with better generalization properties. In addition, a network with fewer weights may be faster to train. Clearly, in most situations a reduction in the dimensionality of the input vector will result in loss of information. One of the main goals in designing a good pre-processing strategy is to ensure that as much of the relevant information as possible is retained. If too much information is lost in the pre-processing stage then the resulting reduction in
performance more than offsets any improvement arising from a reduction of dimensionality [1].

**Feature selection**

In theory, more features should provide more discriminating power, but in practice, with a limited amount of training data, excessive features will not only significantly slow down the learning process, but also cause the classifier to over-fit the training data as irrelevant or redundant features may confuse the learning algorithm. It has proven in both theory and practice effective in enhancing learning efficiency, increasing predictive accuracy, and reducing complexity of learned results [9].

There is more than one reason for the necessity to reduce the number of features to a sufficient minimum. Computational complexity is the obvious one. A related reason is that although two features may carry good classification information when treated separately, there is little gain if they are combined together in a feature vector, because of a high mutual correlation. Thus, complexity increases without much gain. Another major reason is that imposed by the required generalization properties of the classifier. According to the discussion there and as we will state more formally at the end of this chapter, the higher the ratio of the number of training patterns N to the number of free classifier parameters, the better the generalization properties of the resulting classifier. A large number of features are directly translated into a large number of classifier parameters (e.g., synaptic weights in a neural network, weights in a linear classifier). Thus, for a finite and usually limited number N of training patterns, keeping the number of features as small as possible is in line with our desire to design classifiers with good generalization capabilities [1].

One of the simplest techniques for dimensionality reduction is to select a subset of the inputs, and to discard the remainder. This approach can be useful if there are inputs which carry little useful information for the solution if the problem, or if there are very strong correlations between sets of inputs so that the same information is repeated in several variables.

Any procedure for feature selection must be based on two components. First, a criterion must be defined by which it is possible to judge whether one subset of features is better. Second, a systematic procedure must be found for searching through
candidate subsets of features. In practice, the selection criterion should be the same as will be used to assess the complete system (such as misclassification rate for a classification problem or sum-of-squares error for regression problem). Similarly, the search procedure could simply consist of an exhaustive search of all possible subsets of features since this is in general the only approach that is guaranteed to find the optimal subset. In a practical application, however, we are often forced to consider simplified selection criteria as well as non-exhaustive search procedures in order to limit the computational complexity of the search process.

It is clear that the optimal subset of features selected from a given starting set will depend, among other things, on the particular form of the model (neural network or otherwise) with which they are to be used. Ideally, the selection criterion would be obtained by training the network on the given subset of features, and then evaluating its performance on an independent set of test data. If the network training procedure involves non-linear optimization, such an approach is likely to be impractical since the training and testing process would have to be repeated for each new choice of feature subset, and the computational requirements would become too great. It is therefore common to use a simpler model, such as a linear mapping, in order to select the features, and then use these features with the more sophisticated non-linear model. The simplified model is chosen so that it can be trained relatively quickly (using linear matrix methods for instance) thereby permitting a relatively large number of feature combination to be explored. It should be emphasized, however, that the feature selection and the classification (or regression) stages should be ideally be optimized together, and that practical constrains we are often forced to treat them independently [8], [10].

An optimal feature subset need not be unique because it may be possible to achieve the same accuracy using different sets of features (e.g., when two features are perfectly correlated, one can be replaced by the other). By definition, to get the highest possible accuracy, the best subset that a feature subset selection algorithm can select is an optimal feature subset. The main problem with using this definition in practical learning scenarios is that one does not have access to the underlying distribution and must estimate the classifier’s accuracy from the data [11].

There are a number of different approaches to subset selection. The filter approach, selects features using a pre-processing step. The main disadvantage of the
filter approach is that it totally ignores the effects of the selected feature subset on the performance of the induction algorithm. A second approach more familiar among researchers is the wrapper approach in which the subset feature selection algorithm exists as a wrapper around the induction algorithm (classifier). The feature selection algorithm conducts a search for a good subset using the induction algorithm (classifier) itself as a part of evaluation function [12]. After all, the optimal features depend on the specific biases and heuristics of the algorithm, and hence the wrapper approach naturally fits with this definition [11]. Sophisticated wrapper or embedded methods improve predictor performance compared to simpler variable ranking methods like correlation methods, but the improvements are not always significant [13].

Most feature selection methods mainly focus on finding relevant features. Feature relevance alone is insufficient for efficient feature selection of high-dimensional data. Notions of feature redundancy are normally in terms of feature correlation. It is widely accepted that two features are redundant to each other if their values are completely correlated. In reality, it may not be so straightforward to determine feature redundancy when a feature is correlated (perhaps partially) with a set of features. We now formally define feature redundancy in order to devise an approach to explicitly identify and eliminate redundant features.

As mentioned earlier, there exist two major approaches in feature selection: individual evaluation and subset evaluation. Individual evaluation, also known as feature weighting/ranking (Blum and Langley, 1997; Guyon and Elisseeff, 2003), assesses individual features and assigns them weights according to their degrees of relevance. A subset of features is often selected from the top of a ranking list, which approximates the set of relevant features (II, III, and IV in Figure 1). With its linear time complexity in terms of dimensionality N, this approach is efficient for high-dimensional data. However, it is incapable of removing redundant features because redundant features likely have similar rankings. As long as features are deemed relevant to the class, they will all be selected even though many of them are highly correlated to each other. For high-dimensional data, which may contain a large number of redundant features, this approach may produce results far from optimal. [9]

**Our implementation**

The major task of this component (pre-processing) is that given a number of features to reduce them as much as possible and in the same time to retain as much
class discriminatory information as possible. This procedure is known as feature selection or reduction and this is a very crucial as a preprocessing step. Another task is to prepare the feature vectors by applying some linear transformation techniques such as PCA or normalization.

Although this is not the main goal of this thesis, we give special attention to feature selection as it is of such importance. We avoid doing an exhaustive search even though this would be the best method of feature selection because of its computational burden. For this, we implement three different feature selection techniques, which include filters such as ranking, stepwise regression, or wrappers with forward search or backward elimination.

The pre-processing techniques included in this thesis may divided in two distinct components. The first has to do with dimensionality reduction which can be achieved either by feature transformation via (PCA) or via filter feature selection and wrapper feature selection. The other is the preparation of data via normalization of input vectors.

**Principal components analysis.** The PCA of the input vector is a linear transformation and/or a dimensionality reduction. Linear techniques perform dimensionality reduction by embedding the data into a linear subspace of lower dimensionality. Although there are various techniques to do so, PCA is by far the most popular (unsupervised) linear technique [14].

In general, a reduction in the dimensionality of the input space will be accompanied by loss of some of the information that discriminates between different classes (or, more generally, which determines the target values). The goal in dimensionality reduction is therefore to preserve as much of the relevant information as possible. This procedure relies entirely on the input data itself, without reference to the corresponding target data, and can be regarded as a form of unsupervised learning. (For more details, see Appendix page 64).

**Normalization of the input vectors.** One of the most common forms of preprocessing consists of rescaling of the input variables. This is often useful if different variables have typical values that differ significantly. Thus, features with large values may have a larger influence in the cost function than feature with small values, although this is not necessarily reflect their respective significance in the design of the
classifier. The problem is overcome by normalizing the features so that their values lie within similar ranges. A straightforward technique is normalization via the respective estimates of the mean and variance. This is obviously a linear method. Other linear techniques limit the feature values in the range of [0, 1] or [-1, 1] by proper scaling [1].

Although this is a linear transformation and theoretically a redundant procedure for a multilayer perceptron, in practice, ensure that all of the input and target variables are of order unity, in which case we expect that the network weights should also be of order unity. The weights can then be given a suitable random initialization prior to network training. Without the linear rescaling, we would need to find a solution for the weights in which some weigh values had markedly different values from others. (For more details, see Appendix page Σφάλμα! Δεν έχει οριστεί σελιδοδείκτης).

Unbalanced compensate. This option allows the user to equalize the examples for each category. This is useful in case of neural network especially when the error function that tries to minimize is the mean squares error. By unbalanced data set, we mean a data set with much more cases (examples) from one class than the others do. This is a common problem and is related to the error function. In the case that 99% of a data set belongs to a specific class, then the classifier can easily achieve an overall accuracy of 99% by always predicting that specific class. If this is not satisfactory, then it is important to define precisely what sort of error measure is desired.

The easiest solution, where available, is to have the learning algorithm weight the cases to compensate appropriately (examples from under-represented classes are weight more). Many learning algorithms (at least as commonly implemented) do not provide this capability, so manipulation of the data is ordinarily used instead.

Cases in under-represented classes may be duplicated, but (among other things) this has the undesirable effect of fooling some algorithms into believing that discovered patterns have greater statistical significance than they really do.

Cases in over-represented classes may be discarded, although this throws away potentially valuable information. If data sets are large enough, this may be perfectly acceptable.

According to ‘Greg Heath’: The best training is accomplished when the effective training set class ratios are balanced. Operational prior probabilities and misclassification costs can be implemented post-training.
Chapter 3: Graphical User Interface

The "balancing" can be accomplished in a number of ways:

4. Higher presentation frequencies for sparsely represented classes’
5. creating additional noisy versions of sparse class vectors
6. reducing the number of abundant class vectors to a smaller, but adequate subset
7. a weighted objective function
8. a combination of the above.

Rank features: we use the rankfeature function of Matlab® witch performs a feature selection based on class separability criteria (see below).

The feature selection is done by statistical methods and is independent of the features.

ttest Absolute value two-sample T-test with pooled variance estimate,
entropy Relative entropy, also known as Kullback-Lieber distance or divergence,
brattacharyya Minimum attainable classification error or Chernoff bound,
roc Area under the empirical receiver operating characteristic (ROC) curve,
wilcoxon Absolute value of the u-statistic of a two-sample unpaired Wilcoxon test, also known as Mann-Whitney [15].

Stepwise regression: we use the function stepwisefit of Matlab® to perform a stepwise regression. Stepwise regression is a systematic method for adding and removing terms from a multi-linear model based on their statistical significance in a regression. The method begins with an initial model and then compares the explanatory power of incrementally larger and smaller models. At each step, the p-value of an F-statistic is computed to test models with and without a potential term. If a term is not currently in the model, the null hypothesis is that the term would have a zero coefficient if added to the model. If there is sufficient evidence to reject the null hypothesis, the term is added to the model. Conversely, if a term is currently in the model, the null hypothesis is that the term has a zero coefficient. If there is insufficient
evidence to reject the null hypothesis, the term is removed from the model. The method proceeds as follows:

Fit the initial model.

If any terms not in the model have p-values less than an entrance tolerance (that is, if it is unlikely that they would have zero coefficient if added to the model), add the one with the smallest p-value and repeat this step; otherwise, go to step three.

If any terms in the model have p-values greater than an exit tolerance (that is, if it is unlikely that the hypothesis of a zero coefficient can be rejected), remove the one with the largest p-value and go to step 2; otherwise, end.

Depending on the terms included in the initial model and the order in which terms are moved in and out, the method may build different models from the same set of potential terms. The method terminates when no single step improves the model. There is no guarantee, however, that a different initial model or a different sequence of steps will not lead to a better fit. In this sense, stepwise models are locally optimal, but may not be globally optimal [16].

**Sequential feature selection:** This is a wrapper approach on feature selection. We use the function `sequentialfs` of Matlab® combined with the `classify (discriminant analysis)` function as criterion of classification error, both forward selection and backward elimination are considered [16].

**Evaluation component**

This component includes some very important option as it is the judge of the classifier performance. Here the user can choose the method of generalization method estimation for each classifier. The options currently are the k-fold cross-validation and leave one out methods. There is also the option of stratification.

In stratified cross-validation, the folds are stratified so that they contain approximately the same proportions of labels as the original dataset. If background knowledge indicates that the dataset was stratified, i.e., each class was sampled in proportion to its probability in the domain (these probabilities should be known based on background knowledge), then stratified cross-validation is clearly the correct choice, as we condition on something that is true. However, if no such knowledge exists, it is not clear that stratification is the correct thing to do [3].
Finally, there is the option of repetition the procedure in order to achieve better statistics especially in cases where the variance between the cross-validation runs is high. The bias of cross-validation can be reduced by increasing the number of folds, but increasing it too much may increase the variance. Even for the optimum number of folds chosen to reduce variance, the variance may be much too high to be useful.

The ultimate cross-validation would be the complete cross-validation where the procedure repeated until all possibilities of choices the k folds are exhausted.

Beyond that, executing cross-validation multiple times, each time with a different split into the k folds, can viewed as a Monte-Carlo estimation (Binder & Heerman 1988) to complete k-fold cross-validation, which is usually too expensive to run. Repeating cross-validation multiple times will not change the bias inherent in the method but it might change the variance of the estimates [3].

**Network evaluation component (Network parameters)**

**The theory**

The estimation of the classification error probability presupposes that one has decided upon the data set to which the error counting will be applied. This is not a straightforward task. The set of samples that we have at our disposal is finite and it has to be utilized for both training and testing. For that reason the following methods have been suggested,

**Re-substitution Method**

The same data set is used, first for training and then for testing. One need not go into mathematical details in order to see that such a procedure is not very fair. This method provides an optimistic estimate of the true error probability. The amount of bias of the re-substitution estimate is a function of the ratio of the data set size and the dimension of the feature space. Furthermore, the variance of the estimate is inversely proportional to the data set size N. In words, in order to obtain a reasonably good estimate, N as well as the above ratio must be large enough [1].

**Holdout Method**

The available data set is divided into two subsets, one for training and one for testing. The major drawback of this technique is that it reduces the size for both the training and the testing data. Another problem is to decide how many of the N
available data will be allocated to the training set and how many to the test set. This is an important issue. The classification error probability of a classifier, designed using a finite training data set, is always higher than the corresponding asymptotic error probability. This excess error decreases as the data set increases.

On the other hand, the variance of the error counting depends on the size of the test set, and for small test data sets, the estimates can be unreliable. Efforts made to optimize the respective sizes of the two sets have not yet led to practical results.

**Cross-validation method**

Cross-validation is a method for estimating the generalization error and for that reason can be used to compare two or more learning models to estimate which model will perform the best on the problem at hand. With n-fold Cross-validation, the available training data is partitioned into n disjoint subsets, the union of which is equal to the original training set.

Each learning model is trained on n-1 of the available subsets, and then tested on the one subset that was not used during training. This process is repeated n times, each time using a different test set chosen from the n available partitions of the training data, until all possible choices for the test set have been exhausted. The n test set scores for each learning model are then averaged (or summed), and the model with the highest average test set score is chosen as the most likely to perform well on unseen data [12, 17].

An extreme case of Cross-Validation method where the k equals the cases size is the Leave-One-Out Cross-validation or Leave-One-Out.

**Leave-One-Out Method**

This method alleviates the lack of independence between the training and test sets in the re-substitution method and at the same time frees itself from the dilemma associated with the holdout method. The training is performed using all but one sample, and the test is carried out using the excluded sample. If this is misclassified, an error is counted. This is repeated as many times as all data set, each time excluding a different sample.
The total number of errors leads to the estimation of the classification error probability. Thus, training is achieved using, basically, all samples, and at the same time independence between training and test sets is maintained.

The major drawback of the technique is its high computational complexity.

**Bootstrap Method**

Another set of techniques have been developed around the bootstrap method. According to the “bootstrap” philosophy, new data sets are artificially generated. This is a way to overcome the limited number of available data and create more data in order to better assess the statistical properties of an estimator. Let \( X \) be the set of the available data of size \( N \). A bootstrap design sample set of size \( N \), \( X^* \), is formed by random sampling with replacement of the set \( X \). Replacement means that when a sample, say \( x_i \), is “copied” to the set \( X^* \), it is not removed from \( X \) but is reconsidered in the next sampling. A number of variants have been built upon the bootstrap method. A straightforward one is to design the classifier using a bootstrap sample set and count the errors using the samples from \( X \) that do not appear in this bootstrap sample set. This is repeated for different bootstrap sample sets. The error rate estimate is computed by counting all the errors and dividing the sum by the total number of test samples used.

**Our Implementation**

In our application, the user can only choose between cross-validation and its special form the leave-one-out. Though in cross validation mode it can choose also the number of folds.

**Network component (network parameters)**

This component of the application is divided into training parameters and the layer parameters sub-component.

In the training parameter subcomponent, the user can choose the training function used to train the network, the number of epochs as well as the goal in order to define an upper limit where the training of the network will stop. In particular, by choosing the epochs and the goal, the network will train until one of the two will first satisfied.

Another limitation of network training is the early stopping method, which is discussed earlier. Finally, the user can choose the number of repetition that the
algorithm will run in order to improve the statistics for the network evaluation given that each time the network is trained the results will vary depended on initial conditions of the network.

In the layer parameters subcomponent, the user can choose to automatic evaluation of the network by checking the “automatic” checkbox and choosing a maximum number of layer and neurons to be evaluated or manual evaluate a particular network topology by choosing the number of layer up to 10 and the number of neurons up to 100 for each layer.

There are many variations of the back-propagation algorithm. The simplest implementation of back-propagation learning updates the network weights and biases in the direction in which the performance function decreases most rapidly, the negative of the gradient.

**Our implementation**

Although Matlab® has others training algorithms this thesis investigates only theScaled conjugate gradient back-propagation (trainscg), for the type of classification problem at hand, because of its advantages over other algorithms, as discussed in Neural Network Toolbox 6.0™ documentation [18].

On the other hand, during the automatic procedure, the epochs is set to 300, which is far enough for the kind of problem that this thesis is dealing. This is because there is no improvement after 150 to 200 epochs in all configurations.

**Scaled conjugate gradient back-propagation (trainscg)**

trainscg can train any network as long as its weight, net input, and transfer functions have derivative functions. Back-propagation is used to calculate derivatives of performance with respect to the weight and bias variables X.

The scaled conjugate gradient algorithm is based on conjugate directions, as in traincgp, traincfg, and traincgb, but this algorithm does not perform a line search at each iteration. See Moller (Neural Networks, Vol. 6, 1993, pp. 525 to 533) for a more detailed discussion of the scaled conjugate gradient algorithm. Training stops when any of these conditions occurs:

1. The maximum number of epochs (repetitions) is reached.
2. The maximum amount of time is exceeded.
3. Performance is minimized to the goal.
4. The performance gradient falls below min_grad.
5. Validation performance has increased more than max_fail times since the last time it decreased (when using validation).
Chapter 4

The experiment

In this chapter, we will try to put all that we have discussed together in a real life application. In particular, we will test the application that we have developed in a classification task evolving some different classification problems.

Material and methods

Software and Hardware

We use the MathWorks™ Matlab® (R2008a) software along with Neural Network Toolbox 6.0™, Statistics Toolbox™ 6.2 for the development of the GUI (see Chapter 3) we use as well as the evaluation purposes of 5 classifiers (see below).

For some statistical analysis of the result as well as presentation purposes, we use the standard tools that offer the Microsoft Excel® (see below).

Finally the statistical application SPSS® is used for a more explanatory statistical approach of the results.

All calculations are done on a Toshiba™ notebook based on Intel® Pentium® M 1.8 GHz processor and this was the main reason for limitation of the experiment.

The Data sets

In order to perform the experiment we used four different data sets. Three of them come from the work of three doctoral student of Medical Physics department of University of Patras and one from the University of Wisconsin Hospitals.

Biopsy characteristics – Cytology dataset

The data set consists of 699 samples with 9 features that represent biopsy characteristics. Data obtained from the University of Wisconsin Hospitals, Madison, from Dr. William H. Wolberg [18]

The next three dataset are obtained from Medical Physics Department of University of Patras (see acknowledgments).

Breast mass characterization – DCE MRI dataset

The data set consists of 83 samples with 19 features. The features were extracted from the kinetic curves of DCE MRI images.
Lung parenchyma segmentation - HRCT dataset

The data set consists of 295 samples with 76 features. Features were used to segment the lung parenchyma from surrounding tissue.

Micro-calcification cluster characterization – X-ray Mammography dataset

The data set consists of 148 samples with 40 features that represent metrics from digitized film mammographies. Cluster features extracted to analyse the morphology of individual micro-calcifications within the cluster to decide between benignity/malignancy.

The classifiers

There are many different classifiers we can use for classification purposes. Below there is a brief description of those we use in this thesis.

FFNN

There is an extensive description of this classifier in Chapter 2.

KNN

The k-nearest neighbours’ algorithm is amongst the simplest of all machine learning algorithms. An object is classified by a majority vote of its neighbours, with the object being assigned to the class most common amongst its k nearest neighbours. k is a positive integer, typically small. If k = 1, then the object is simply assigned to the class of its nearest neighbour. In binary (two class) classification problems, it is helpful to choose k to be an odd number as this avoids tied votes.

The neighbours are taken from a set of objects for which the correct classification is known. This can be thought of as the training set for the algorithm, though no explicit training step is required. In order to identify neighbours, the objects are represented by position vectors in a multidimensional feature space. It is usual to use the Euclidean distance, though other distance measures, such as the Manhattan distance could in principle be used instead. The k-nearest neighbour algorithm is sensitive to the local structure of the data.

The best choice of k depends upon the data; generally, larger values of k reduce the effect of noise on the classification, but make boundaries between classes less distinct. A good k can be selected by various heuristic techniques, for example, cross –
validation. The special case where the class is predicted to be the class of the closest training sample (i.e. when \( k = 1 \)) is called the nearest neighbour algorithm.

The accuracy of the k-NN algorithm can be severely degraded by the presence of noisy or irrelevant features, or if the feature scales are not consistent with their importance.

**SVM**

Support vector machines (SVM) are a group of supervised learning methods that can be applied to classification or regression. Support vector machines represent an extension to nonlinear models of the generalized portrait algorithm developed by Vladimir Vapnik. The SVM algorithm is based on the statistical learning theory and the Vapnik-Chervonenkis (VC) dimension introduced by Vladimir Vapnik and Alexey Chervonenkis.

Support vector machines represent the most important development in chemometrics after (chronologically) partial least squares and artificial neural networks.

**PNN**

Probabilistic neural networks can be used for classification problems. When an input is presented, the first layer computes distances from the input vector to the training input vectors and produces a vector whose elements indicate how close the input is to a training input. The second layer sums these contributions for each class of inputs to produce as its net output a vector of probabilities. Finally, a compete transfer function on the output of the second layer picks the maximum of these probabilities, and produces a 1 for that class and a 0 for the other classes [18], [19].

Probabilistic networks perform classification where the target variable is categorical.

Although the implementation is very different, probabilistic neural networks are conceptually similar to k-Nearest Neighbours (k-NN) models. The basic idea is that a predicted target value of an item is likely to be about the same as other items that have close values of the predictor variables.
Chapter 4: The experiment

LDA

Linear discriminant analysis (LDA) is a method used in statistics and machine learning to find the linear combination of features which best separate two or more classes of objects or events. The resulting combination may be used as a linear classifier or, more commonly, for dimensionality reduction before later classification. This is the method we choose to implement in our application for one type (wrapper feature search) of dimensionality reduction (feature selection) see Chapter 3.

Evaluation methodology

In this thesis, we accessed three different problems:

1. The selection of best parameters within the classifier (namely the number of hidden units in the FFNN)
2. The selection of best classifier in each one of the different datasets
3. The selection of best preprocessing technique (mainly feature selection method) for given classifier and dataset

In order to solve the above “problems” we need

1. A metric for the evaluation of each classifier
2. A method (algorithm) to apply this metric on all classifiers
3. A statistical test to check whether our results is statistical significant

First we will analyze the “tools” that we will use and then we will forward to problems more analytically.

Metric

The metric we choose to use for this experiment is the misclassification error or the performance of classifier in order to estimate generalization error. This is not the best choice but it is the easiest one to implement on all classifiers (see discussion for more details on the subject).

Estimating the generalization error is one of the key ingredients of supervised learning since a good generalization error estimator can be used for model selection.

The goal of supervised learning is to estimate an unknown input-output relation from samples, which is mathematically formulated as a function approximation problem. If the learning target function is accurately learned, the output values for
unlearned input points can be estimated. This is called the generalization capability. The level of generalization capability is evaluated by the “closeness” between the learned function and the true function, i.e., the generalization error [20].

**Algorithm**

Each one of the three above “problems” defines a “collection” of models. We need an “algorithm” to “choose” a model from the selection (called induction algorithm in the literature). It is obvious that the “best” model is not the one that performs well on the limited dataset at hand. In the contrary, we need to know what will be its performance on unknown cases. Therefore, we need an independent set (namely test set) that it has not participated in any way in the training procedure. In that set we will measure the error rate of a classification scheme performance and this will be the estimator we looking for. It is known that single train and test partitions (namely hold out method) are not reliable estimators of the true error rate of a classification scheme on a limited dataset as it gives unacceptable high bias. Therefore, it was decided that a random sub-sampling scheme should be used in this experiment to minimize any estimation bias.

The two main candidates for the evaluation purposes in this thesis are the leave one out cross validation and the 10-fold cross-validation (see below).

A leave-one-out classification scheme was thought computationally too expensive and so 10-fold cross-validation was used on all of the datasets. In addition, leave one out cross validation has the reputation of being very variable, and 10-fold cross-validation is usually a better guide. [21]

**Cross-Validation**

For details, see chapter 3 p. 36.

The standard practice for FFNN model selection is to use 10-fold Cross-validation and this is the type of Cross-validation that is tested in this thesis. The 10-fold cross-validation scheme has been extensively tested and has been shown to provide an adequate and accurate estimate of the true error rate. The cross-validation sampling

---

3 Here is worth mention that there is another famous statistic technique the bootstrap and its variation (see p. 37). Though it is not included in this thesis is widely approved even though, bootstrap can sometimes create a false sense of good performance when used with an ANN. Since the training and testing sets overlap, it is possible for the network to have a low prediction error just because it has already seen (and memorized) some of the test patterns [21]
technique used was random but ensured that the approximate proportions of examples
of each class remain 90% in the training set and 10% in the test set [12, 17].

The conduct of the experiment takes the form of three runs of 10 folds of stratified
cross validation to reduce any possible random effects due to performing only a single
run. Although more run is advisable our resources (see Software and Hardware on p.
40) does not allow them [22].

For consistency, exactly the same data were used to train and test all of the five
classification schemes (as well as any of its variations), this is often called a paired
experimental design (see below on Statistical validation method p. 45) [12].

**Statistical validation method**

In this thesis, we will use Analysis of Variance (ANOVA) techniques to test the
hypothesis of equal means over a number of models (classifiers) and feature selection
methods simultaneously. The experimental design allows us to compare, on each data
set, the mean performance for each model and for each train and test partition.

In this context, there are two hypotheses that arise:

$H_{01}$: that all the means are equal due to the different train and test partitions

$H_{02}$: that all the means are equal due to the different learning algorithms

In order to make the statistical task simpler and because we are actually interested
only for the second one hypothesis $H_{02}$, we choose to contact a paired (or matched
pairs) experimental design in our test. That is all of the models are checked on the
same partition of the 10-fold cross validation. In this way, the performance of all
models is evaluated using the same random samples. This design actually invalidates
the first null hypothesis because all of the models are trained and tested on the same
cross validation partition [23].

When the analysis of variance test on an accuracy measure produces evidence to
reject the null hypothesis, we can accept the alternative hypothesis that all of the mean
accuracies are not equal. However, we still do not know which of the means are
significantly different from which other means, so we will use Duncan's multiple range
test to separate significantly different means into subsets of homogeneous means [24].
Unfortunately, ANOVA is based on assumptions which are most probably violated when analyzing the performance of machine learning algorithms as in our case (see chapter 5). First, ANOVA assumes that the samples are drawn from normal distributions. In general, there is no guarantee for normality of classification accuracy distributions across a set of problems. Admittedly, even if distributions are abnormal this is a minor problem and many statisticians would not object to using ANOVA unless the distributions were, for instance, clearly bi-modal (Hamilton, 1990). The second and more important assumption is the homogeneity of variance, which requires that the random variables have equal variance. In our case in the vast majority of the cases this assumption is not violated. To check for homogeneity of variance we do not contact a sophisticated test, but rather use a simple but effective rule. We assume that if the ratio of maximum to minimum variance is smaller than two we do not have enough evidence to reject the homogeneity of variance.

**Seeking the best network topology**

As discussed in theory the seeking of the best network topology is problem depended and there is no some standard procedure that can we follow in order make easier this step. One of the major difficulties with FFNN lies in the selection of the optimal network architecture for a given problem. FFNN architecture selection is concerned with the number of layers in the network and the number of nodes in each layer. For any given learning problem, there are an essentially infinite number of possible FFNN network architectures [17].

In our seeking process we adopt a rather naïve approach of adding one hidden neuron at a time and checking the generalization error with the method described above (see Cross-Validation on p. 44). The statistical significance of the results is checked with one – way ANOVA described above (p. 45).

**Best classifier**

Many researchers have done extensive work on seeking the best overall classifier among many either artificial or real life datasets. In general, there is an agreement among them that there is no overall best classifier and choose of best is more problem depended. In this thesis, we are more interested finding if there is a statistical significant better classifier on a dataset basis and not over all dataset in hand. For that
reason, we compare the classification performance of each classifier with other in the same dataset only, and again check its statistical significance with ANOVA test.

**Preprocessing**

Details on preprocessing can be found on chapter 3 (p. 26). In this thesis, from all preprocessing techniques we are more interested in dimensionality reduction and even more in feature selection part. Feature selection is one of the most difficult problems to solve in machine learning and major effort has been done to access the problem.

For fairness comparison of different dimensionality reduction techniques (namely PCA, stepwise and t-test ranking), we try to test them all in the same dimensionality. This dimensionality is suggested by stepwise search (function stepwisefit in Matlab®) using its default values (penter 0.05, premove 0.1).

**Considerations**

As pointed out by Friedman, no classification method is universally better than any other, each method having a class of target functions for which it is best suited. These experiments then, are an attempt to investigate which combination of classifier, classifier parameter and feature selection method should be used on a particular subset of problems.

Our conclusions are therefore targeted towards this subset of problems and should not be extrapolated beyond the scope of this class of problem.

Except for specific parameters values that we want to study like number of hidden units in FFNN or the value of k for the KNN algorithm all other values are kept at their default values on all classifiers. This actually caused a bias on some of the classifiers as discussed on next chapter.

First thing we interested of is the tracking of the best network topology for a given task. In order to achieve the goal of this thesis, we have to make some trade offs based on some rules of thumbs mainly because an exhaustive search would be very computer intuitive if not feasible at all. These rules were widely accepted and applicable in industry for many years. Here are some of them that we accept as a start to our study.

**Activation function**: A sigmoid activation function is used because of faster training that it offers [4]. (Although, others have been tried, with similar results).
Number of hidden layers and units (in each layer): The best number of hidden units depends in a complex way on:

1. the numbers of input and output units
2. the number of training cases
3. the amount of noise in the targets
4. the complexity of the function or classification to be learned
5. the architecture
6. the type of hidden unit activation function
7. the training algorithm
8. regularization

In most situations, there is no way to determine the best number of hidden units without training several networks and estimating the generalization error of each. If you have too few hidden units, you will get high training error and high generalization error due to under-fitting and high statistical bias. If you have too many hidden units, you may get low training error but still have high generalization error due to overfitting and high variance. Geman, Bienenstock, and Doursat (1992) discuss how the number of hidden units affects the bias/variance trade-off. We only use one hidden layer with increasing number of hidden units in it. This is because we have a small dataset and in this case, even if the underlying function is more complicated than the one that only one hidden layer can estimate, we can achieve better generalization performance with only one hidden layer [25].

Training functions. Matlab® Neural Network Toolbox 6.0™ offers over nine (9) different learning algorithms. Although, in this thesis we used the default training algorithm, scaled conjugate gradient back-propagation (trainscg), for the type of classification problem at hand, because of its advantages over other algorithms, as discussed in Neural Network Toolbox 6.0™ documentation [18].
Chapter 4: The experiment

Results

The output of our application is saved in Excel files in order to undertake the ANOVA test for statistical significance of the results and for representation purposes. Below we choose to present them in the form of charts, as it would be very difficult to interpret otherwise.

There is three graphs for each data set (we do not conduct comparisons between datasets), each one access one of the “problems” described above (see p. 43).

The significance tests are done in Excel using its standard tool and in SPSS® in order to run the Duncan’s test, where there is evidence to reject the null hypothesis about equality of means.

**FFNN topology**

In the following figures, four (one for each dataset) graphs are displayed representing the results from the FFNN topology experiment for the three methods of dimensionality reduction that we have study.

![Figure 13: Performance vs. Number of Hidden Units (0 indicates no hidden layer) for three dimensionality reduction techniques on X-ray Mammography dataset for the FFNN classifier.](image-url)
Chapter 4: The experiment

Figure 14: Performance vs. Number of Hidden Units (0 indicates no hidden layer) for three dimensionality reduction techniques on DCE MRI dataset for the FFNN classifier.

Figure 15: Performance vs. Number of Hidden Units (0 indicates no hidden layer) for three dimensionality reduction techniques on HRCT dataset for the FFNN classifier.
Figure 16: Performance vs. Number of Hidden Units (0 indicates no hidden layer) for three dimensionality reduction techniques on Cytology dataset for the FFNN classifier.
Chapter 4: The experiment

Feature selection

X-ray Mammography

Classifier performance on Feature selection method

Figure 17: Performance vs. Classifier for three dimensionality reduction methods on X-ray Mammography dataset.

DCE MRI

Classifier performance on Feature selection method

Figure 18: Performance vs. Classifier for three dimensionality reduction methods on DCE MRI dataset
Chapter 4: The experiment

HRCT
Classifier performance on Feature selection method

<table>
<thead>
<tr>
<th>Method</th>
<th>Performance 1</th>
<th>Performance 2</th>
<th>Performance 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>0.968237548</td>
<td>0.954712644</td>
<td>0.949233716</td>
</tr>
<tr>
<td>KNN</td>
<td>0.960091954</td>
<td>0.951340996</td>
<td>0.942337165</td>
</tr>
<tr>
<td>SVM</td>
<td>0.963831418</td>
<td>0.926828352</td>
<td>0.920957854</td>
</tr>
<tr>
<td>PNN</td>
<td>0.964980843</td>
<td>0.943524904</td>
<td>0.925479927</td>
</tr>
<tr>
<td>LDA</td>
<td>0.935517241</td>
<td>0.93789272</td>
<td>0.931034483</td>
</tr>
</tbody>
</table>

Figure 19: Performance vs. Classifier for three dimensionality reduction methods on HRCT dataset

Cytology
Classifier performance on Feature selection method

<table>
<thead>
<tr>
<th>Method</th>
<th>Performance 1</th>
<th>Performance 2</th>
<th>Performance 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>0.956611456</td>
<td>0.956611456</td>
<td>0.962332643</td>
</tr>
<tr>
<td>KNN</td>
<td>0.970427881</td>
<td>0.970427881</td>
<td>0.96757764</td>
</tr>
<tr>
<td>SVM</td>
<td>0.948992409</td>
<td>0.948992409</td>
<td>0.948992409</td>
</tr>
<tr>
<td>PNN</td>
<td>0.960883368</td>
<td>0.960883368</td>
<td>0.967584541</td>
</tr>
<tr>
<td>LDA</td>
<td>0.96615597</td>
<td>0.96615597</td>
<td>0.946100759</td>
</tr>
</tbody>
</table>

Figure 20: Performance vs. Classifier for three (stepwise and ttest have the same results) dimensionality reduction methods on Cytology dataset
Chapter 4: The experiment

**Classifier selection**

**Figure 21:** Performance vs. Dimensionality Reduction Method for the classifiers on X-ray Mammography dataset

**Figure 22:** Performance vs. Dimensionality Reduction Method for the classifiers on DCE MRI dataset
Chapter 4: The experiment

**Figure 23:** Performance vs. Dimensionality Reduction Method for the classifiers on HRCT dataset

**Figure 24:** Performance vs. Dimensionality Reduction Method for the classifiers on Cytology dataset
Chapter 5: Conclusions

In this thesis, we study the effect of different parameters on performance of five classifiers (that is ANN, KNN, PNN, SVM and LDA) on four different data sets. For that we develop a graphical user interface (see chapter 3), to make easy and fast evaluation of these parameters.

There are two distinct evaluations we performed in this thesis. The first one concerns the selection of best model parameters (same classifier, we may assume that every parameter define a different model). The second one concerns the selection between different models (different classifiers, as ANN, KNN ...).

Because our concern is, whether or not there is a difference on performances of classifier over different circumstances we adopt an experimental design in a way that for each evaluation keeps constant as many parameters as possible, in order to study the condition under investigation only. As estimator of generalization error we use 10-fold cross-validation measured at misclassification error.

Next, we present the conclusions of those experiments.

Preprocessing

We may see the preprocessing as any manipulation of the data just before entrance into classifier. In this context, there is a dimensionality reduction part and a linear transformation part (there is also non – linear transformation we do not study). The techniques for linear transformation include Normalization and PCA (under some assumptions). The techniques for dimensionality reduction includes again PCA, and feature selection techniques as wrapper feature selection, and filter feature selection like stepwise and ranking of feature based on some criteria (see Figure 25 bellow).

![Figure 25: Preprocessing techniques. The techniques in dashed lines are not evaluated in this thesis](image-url)
The preprocessing techniques that we study in this thesis include:

1. Normalization (that is normalization of feature vectors in interval [0,1])
2. PCA (that is dimensionality reduction with Principal Component Analysis)
3. Stepwise search feature selection
4. Feature ranking on discriminate ability with t-test

We check the impact of preprocessing on ANN model parameter selection (number of hidden layer units)

**Normalization**

The question is how the normalization affects the performances of classifiers.

In order to check if normalization of feature vectors has any effect on classifier performance we do a single factor ANOVA test comparing normalized versus non normalized feature vectors on classification performance. The null hypothesis is that there is no effect of normalization on classifier performance. We choose an ANOVA test versus a simple ttest (which is identical with ANOVA test in this case) only for consistency, (all other statistics are done with ANOVA). In Table 1 we can see in bold where we can reject the null hypothesis of equal means.

Only on HRCT dataset the normalization of feature vectors had proved (with high statistical confidence) that improved the efficiency of all classifiers. On the other hand, PNN seems to be affected the most from normalization.

### Table 1: Statistical significance (p value) on differences between Normalized and Non-Normalized feature vectors measured on best performance (- indicates failure of LDA on NonNormalized feature vector)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ANN</th>
<th>KNN</th>
<th>SVM</th>
<th>PNN</th>
<th>LDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-ray Mammography</td>
<td>0.9054</td>
<td>0.7926</td>
<td>0.7564</td>
<td>0.0019</td>
<td>-</td>
</tr>
<tr>
<td>DCE MRI</td>
<td>0.0028</td>
<td>0.5551</td>
<td>0.0025</td>
<td>0.0026</td>
<td>0.7859</td>
</tr>
<tr>
<td>HRCT</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>Cytology</td>
<td>0.7873</td>
<td>0.9970</td>
<td>0.4811</td>
<td>0.8816</td>
<td>0.9353</td>
</tr>
</tbody>
</table>

**Dimensionality reduction**

The question is whether the method of dimensionality reduction affects the performance of each classifier. For the sake of comparison, we choose for each dataset...
a different dimensionality that it aroused from the stepwise feature selection method (worked at its defaults). ANOVA is chosen for statistical significance tests despite its drawbacks (see Statistical validation method p. 45).

When there is evidence to reject the null hypothesis of equality of means (no effect) a Duncan’s test is contacted in order to find significant group of equal means. We choose the Duncan’s test because is simple and easy to interpret. A graphical representation and the results of tests are shown in Chapter 4 (p. 74). The Table 2 bellow summarizes the results.

**Table 2**: Statistical significance (p value) on differences between dimensionality reduction methods measured on best classifier performance for all classifiers we study. S, T, P are abbreviations of Stepwise, t-test and PCA dimensionality reduction methods respectively. The dash “–” separates statistical significant groups (see Duncan’s tables for details in Chapter 4)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ANN</th>
<th>KNN</th>
<th>SVM</th>
<th>PNN</th>
<th>LDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-ray Mammography</td>
<td>S, T – P (0.002)</td>
<td>S, T – P (0.000)</td>
<td>S, T – P (0.000)</td>
<td>S – T – P (0.000)</td>
<td>(0.061)</td>
</tr>
<tr>
<td>DCE MRI</td>
<td>S – P, T (0.005)</td>
<td>(0.506)</td>
<td>(0.243)</td>
<td>(0.409)</td>
<td>(0.079)</td>
</tr>
<tr>
<td>HRCT</td>
<td>(0.149)</td>
<td>(0.066)</td>
<td>S – T, P (0.000)</td>
<td>S – T, P (0.002)</td>
<td>(0.848)</td>
</tr>
<tr>
<td>Cytology</td>
<td>(0.256)</td>
<td>(0.496)</td>
<td>(1.000)</td>
<td>(0.176)</td>
<td>ST – P (0.000)</td>
</tr>
</tbody>
</table>

From the table above we can see that stepwise in all the cases exhibits statistical significant differences among the others techniques (see discussion for more details)

**Classifier choice**

The question is whether the choice of classifier is critical for each classification problem. In order to check this, we measure the best performance of each classifier with different dimensionality reduction methods. We again conduct ANOVA and Duncan’s test if there is statistical evidence that there is a difference in performance. The Table 3 below summarizes the results that could be found on Chapter 4.

**Table 3**: Statistical significance (p value) on differences between Classifiers measured on its best performances for three dimensionality reduction methods. A, K, S, P, L are abbreviations for ANN, KNN, SVM, PNN, LDA respectively. The dash “–” separates statistical significant groups (see Duncan’s tables for details in Chapter 4)
Chapter 5: Conclusions

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Stepwise</th>
<th>t-test</th>
<th>PCA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(0.002)</td>
<td>(0.032)</td>
<td>(0.001)</td>
</tr>
<tr>
<td>DCE MRI</td>
<td>A, L, K, P – S</td>
<td>(0.258)</td>
<td>A, K, P, L – S</td>
</tr>
<tr>
<td></td>
<td>(0.002)</td>
<td></td>
<td>(0.024)</td>
</tr>
<tr>
<td>HRCT</td>
<td>A, K, P, S – L</td>
<td>(0.120)</td>
<td>(0.075)</td>
</tr>
<tr>
<td></td>
<td>(0.004)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A – A, S</td>
<td></td>
<td>(0.000)</td>
</tr>
</tbody>
</table>

From the table above the only useful conclusion that we can make is that the SVM has surprisingly bad performance (see discussion below). On the other hand, the ANN seems to perform very good on almost all the cases.

**FFNN number of hidden units**

Again, we conduct an ANOVA test on all different network architectures (number of hidden units). The null hypothesis is again that there are no differences in performance through all network architectures.

From the experiment we conduct, we may conclude that the selection of different number of hidden unit in general does not statistically significant affect the performance of FFNN classifier.

**Discussion**

In this thesis, we deal with the pattern classification discipline and more specifically with its three last components namely preprocessing, classification and evaluation.

These are three different identities even though they work together for the purpose of pattern classification. For that are three questions comes in mind:

1. Is there a best preprocessing tactic?
2. Is there a best classifier?
3. Is there a best evaluation method?

Many researchers tried to answer each one of these questions. Most of the time, the answers were controversial. To what, all seem to agree is that the answer of any of these questions is not only one, but it depends on the problem at hand.

To this in mind, we try to set up an experiment with four classification problems represented by four datasets (see The Data sets p. 40).
Chapter 5: Conclusions

The results as presented above in this chapter are not clear. This is not a surprise but it erases some questions that need some answers. In the next of the discussion we will try to give some explanation on our results and maybe to give a useful conclusion.

**Normalization**

One of the most prominent effect of normalization occurred in HRCT dataset where the benefits on normalization features does not even required a statistical test to validate.

However, on a closer look on the data we can see why this was so prominent. The HRCT dataset includes feature vectors with very large differences between them (the range is in order of $1.2 \times 10^{17}$). In other dataset, this is much smaller with Cytology, which is the smallest (it was already normalized) as well as normalization effect (in some cases the effect was reversed as non-normalized vectors perform better, even though, there are not statistical significant differences).

**Feature selection method**

An interest remark here is that only in the Cytology dataset the two feature reduction methods (stepwise and t-test) agree on best features. We also see that the stepwise method performs better (even though without confirmation from statistical point of view) as the comparison was made on same dimensionality.

In the next figures, we see a very informative scatter plot of the nine features that the stepwise selects as well as the first nine features in t-test rank.

We can clearly see that the t-test ranked the features based on discriminant ability though the first four of them is actually the same feature (highly correlated and it does not contribute to the final classification performance or else have highly redundant information). On the other hand, in the second scatter plot, the stepwise managed well not to include the redundant features as t-test does (and it supposes to do). Therefore, we may see a reason why the stepwise performs better than t-test ranking. Of course, this is not the only one as the main drawback of t-test ranking (and all ranking methods) is that it checks the discriminant ability one by one and not combined as stepwise does (this could be ok for linear separable cases though). On the other end, the Cytology dataset does not include any redundant feature to confuse the t-test ranking.
Chapter 5: Conclusions

Figure 26: Scatter plot of t-test ranking features. The main diagonal represents the histogram of feature. The binominal distribution of feature 10 (F10) implies the great discriminant ability of this feature (comes first in the ranking list).

Figure 27: Scatter plot of stepwise choice of features. The main diagonal represents the histogram of feature. The binominal distribution of feature 10 (F10) implies the great discriminant ability of this feature.

Classifiers

There are two comments that we can make from our results. First the SVM classifier does not performed as well as literature implies that it could. Second, the ANN in most cases ranked among the classifier for the best performances (see Table 3 on p. 58).
To comment the first we can say that this behavior of SVM does not reflect its popularity and definitely requires more research from our part as it was used as black box with its default only parameters. Additionally there is a methodological error we made and maybe lead to this situation. Other classifiers are evaluated repeatedly because we try to find the best parameters for them. However, this leads to a bias toward those classifiers as they have more chance to perform better by chance. The remedy for that would be a second independent test set on which all classifiers would evaluated. However, this would reduce even more the available data for training.

Evaluation methods

No useful conclusion could ever be made if the evaluation method does not allow it. Many researchers had pointed out the importance of such a method and there is none a universal better one.

We may divide the evaluation procedure into parts by asking some questions. What are we evaluating? With what do we evaluate it? What is the evaluation procedure?

The answers to this question are not easy and straightforward and it is beyond the scope of this thesis. Nevertheless, we can make some comments on that just to point out how difficult it could be.

Because at the end, in order to compare some systems, we have to evaluate a whole system (with many components) with a single measurement (whatever what this would be), we have to make sure that we keep all other parameters of the system constant while evaluating only one if possible. However, this does not guaranteed that we are not biased to a method. For example we may measure the misclassification error on a trained FFNN while the objective function (error function) to learn was the mean square error (mse is the default for Matlab®). Additionally, from the theory, we know that the dimensionality does not affect the classifier in the same manner but in order to measure the effect of dimensionality reduction we have to keep it constant based on assumptions (see Preprocessing on p. 47) while our results based on classification performance of those classifiers.
Chapter 5: Conclusions

Future work

This thesis was an attempt to verge on some pattern recognition components. Many things have to be done before we reach some useful and safe conclusion on the subjects we dealt with.

1. Finalize the application that we have developed in order to contact any test we wish without the need other programs as Excel® and SPSS®.

2. Evaluate other dimensionality reduction techniques especially wrappers.

3. Improve the generalization error estimation techniques of 10 fold cross validation by repeat it ten times instead of three improving statistics. Use other estimators such bootstrap or 0.63 bootstrap.

4. Optimize the search procedure of best network topology from the naïve search (current approach see p. 46) to a more pragmatic search adopted by many researchers and specialists on the field. This approach is more based on practice than in theory. This is why many researchers claim that seeking of the best network topology is rather art than a science.

5. Search a better model for SVM classifier as well as LDA.

6. Use a second test set, independent from that is was used for parameter estimation, in order to compare different classifiers. The failure to do this in this thesis could be a major drawback, as it gives biased results toward classifier with many parameters to evaluate and against especially SVM and LDA.
Appendixes

Principal Components Analysis

Introduction
PCA is a useful statistical technique that has found application in fields such as face recognition and image compression, and is a common technique for finding patterns in data of high dimension. It is a way of identifying patterns in data, and expressing the data in such a way as to highlight their similarities and differences. Since patterns in data can be hard to find in data of high dimension, where the luxury of graphical representation is not available, PCA is a powerful tool for analysing data.

The method

Step 1: Get some data
In this simple example, only two dimensions are chosen as this could provide provide plots of the data to show what the PCA analysis is doing at each step. The data is found in Figure 28, along with a plot of that data.

Step 2: Subtract the mean
For PCA to work properly, the data have to standardized meaning to subtract the mean from each of the data dimensions. The mean subtracted is the average across each dimension. Therefore, all the $x$ values have $\bar{x}$ (the mean of the $x$ values of all the data points) subtracted, and all the $y$ values have $\bar{y}$ subtracted from them. This produces a data set whose mean is zero.

![Figure 28: PCA example data, original data on the left, data with the means subtracted on the centre and a plot of the data](image)

Step 3: Calculate the covariance matrix
Since the data is 2 dimensional, the covariance matrix will be 2x2.
Appendix: Principal Component Analysis

\[
\text{cov} = \begin{pmatrix}
0.61655556 & 0.61544444 \\
0.61544444 & 0.71655556
\end{pmatrix}
\]

Therefore, since the non-diagonal elements in this covariance matrix are positive, we should expect that both \( x \) and \( y \) variable increase together.

**Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix**

Since the covariance matrix is square, we can calculate the eigenvectors and eigenvalues for this matrix. These are rather important, as they tell us useful information about our data. Here are the eigenvectors and eigenvalues:

\[
eigenvalues = \begin{pmatrix}
0.0490833989 \\
1.28402771
\end{pmatrix},
eigenvectors = \begin{pmatrix}
-0.735178656 & -0.677873399 \\
0.677873399 & -0.735178656
\end{pmatrix}
\]

It is important to notice that these eigenvectors are both unit eigenvectors i.e. their lengths are both 1. This is very important for PCA, but luckily, most mathematics packages, when asked for eigenvectors, will give you unit eigenvectors.

Looking at the plot of the data in Figure 29 one can see how the data has quite a strong pattern. As expected from the covariance matrix, they two variables do indeed increase together. On top of the data is plotted the eigenvectors as well. They appear as diagonal dotted lines on the plot. As stated in the eigenvector section, they are perpendicular to each other. However, more importantly, they provide with information about the patterns in the data. It is clearly shown that one of the eigenvectors goes through the middle of the points, like drawing a line of best fit. That eigenvector is showing us how these two data sets are related along that line. The second eigenvector gives us the other, less important, pattern in the data, that all the points follow the main line, but are off to the side of the main line by some amount.

Therefore, by this process of taking the eigenvectors of the covariance matrix, we have been able to extract lines that characterize the data. The rest of the steps involve transforming the data so that it is expressed in terms of them lines.

![Figure 29: A plot of the normalized data (mean subtracted) with the eigenvectors of the covariance matrix overlayed on top.](image)
Step 5: Choosing components and forming a feature vector

Here is where the notion of the data compression and reduced dimensionality comes into it. If you look at the eigenvectors and Eigen values from the previous section, you will notice that the Eigen values are quite different values. In fact, it turns out that the eigenvector with the highest Eigen value is the *principal component* of the data set.

In our example, the eigenvector with the largest Eigen value was the one that pointed down the middle of the data. It is the most significant relationship between the data dimensions.

In general, once eigenvectors are found from the covariance matrix, the next step is to order them by Eigen value, highest to lowest. This ranks the components in order of significance. Now, the components of lesser significance can be ignored. Some information is lost, however if the Eigen values are small, the loss is least. Leaving out some components, the final data set will have fewer dimensions than the original.

What needs to be done now is you need to form a *feature vector*. This is constructed by taking the eigenvectors that you want to keep from the list of eigenvectors, and forming a matrix with these eigenvectors in the columns.

*FeatureVector* = \((eig_1 \ eig_2 \ eig_3 \ ... \ eign)\)

Given this example set of data and the fact that two eigenvectors, are computed there are two choices. Either forms a feature vector with both of the eigenvectors:

\[
\begin{pmatrix}
-.677873399 & -.735178656 \\
-.735178656 & .677873399
\end{pmatrix}
\]

Alternatively, choose to leave out the smaller, less significant component and only have a single column:

\[
\begin{pmatrix}
-.677873399 \\
-.735178656
\end{pmatrix}
\]

The results of each of these are shown in the next section.

Step 6: Deriving the new data set

This is the final step in PCA, and is the easiest. Once the components (eigenvectors) to be kept have been chosen, they formed a feature vector.

In the case of keeping both eigenvectors for the transformation, the data and the plot is presented in Figure 30. This plot is the original data, rotated so that the eigenvectors are the axes. This is understandable since no information in this decomposition is been lost.
Appendix: Principal Component Analysis

Figure 30: The table of data by applying the PCA analysis using both eigenvectors, and a plot of the new data points.

The other transformation is by taking only the eigenvector with the largest Eigen value. The table of data resulting from that is found in Figure 31. As expected, it only has a single dimension. Comparing this data set with the one resulting from using both eigenvectors, one will notice that this data set is exactly the first column of the other. Therefore, plotting this data, it would be one dimensional, and would be points on a line in exactly the $x$ positions of the points in the plot in Figure 30. That way effectively the whole other axis is thrown away, which is the other eigenvector.

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-.827970186</td>
<td>-.175115307</td>
</tr>
<tr>
<td>1.77758033</td>
<td>.142857227</td>
</tr>
<tr>
<td>-.992197494</td>
<td>.384374989</td>
</tr>
<tr>
<td>-.274210416</td>
<td>130417207</td>
</tr>
</tbody>
</table>

Transformed Data=

| -.673580142 | -.209498461 |
| -.912949103 | 175282444 |
| .0991094375 | -.349824698 |
| 1.14457216 | .0464172582 |
| .438046137 | .0177646297 |
| 1.22382056 | -.152675287 |

Data transformed with 2 eigenvectors
Appendix: Principal Component Analysis

Transformed Data (Single eigenvector)

\[ x \]

-0.827970186
1.77758033
-0.92197464
-0.274210416
-1.67580142
-0.912949103
0.0991094375
1.14457216
0.438046137
1.22382056

**Figure 31:** The data after transforming using only the most significant eigenvector.
Analysis of Variance (ANOVA)

Purpose
The reason for doing an ANOVA is to see if there is any difference between groups on some variable.
For example, you might have data on student performance in non-assessed tutorial exercises as well as their final grading. You are interested in seeing if tutorial performance is related to final grade. ANOVA allows you to break up the group according to the grade and then see if performance is different across these grades.
ANOVA is available for both parametric (score data) and non-parametric (ranking/ordering) data.

Types of ANOVA

One-way between groups
The example given above is called a one-way between groups model.
You are looking at the differences between the groups.
There is only one grouping (final grade) which you are using to define the groups.
This is the simplest version of ANOVA.
This type of ANOVA can also be used to compare variables between different groups - tutorial performance from different intakes.

One-way repeated measures
A one way repeated measures ANOVA is used when you have a single group on which you have measured something a few times.
For example, you may have a test of understanding of Classes. You give this test at the beginning of the topic, at the end of the topic and then at the end of the subject.
You would use one-way repeated measures ANOVA to see if student performance on the test changed over time.

Two-way between groups
A two-way between groups ANOVA is used to look at complex groupings.
For example, the grades by tutorial analysis could be extended to see if overseas students performed differently to local students. What you would have from this form of ANOVA is:
The effect of final grade
The effect of overseas versus local
The interaction between final grade and overseas/local
Each of the main effects is one-way tests. The interaction effect is simply asking "is there any significant difference in performance when you take final grade and overseas/local acting together".

Two-way repeated measures
This version of ANOVA simply uses the repeated measures structure and includes an interaction effect.
In the example given for one-way between groups, you could add Gender and see if there was any joint effect of gender and time of testing - i.e. do males and females differ in the amount they remember/absorb over time.
Non-parametric and Parametric

ANOVA is available for score or interval data as parametric ANOVA. The non-parametric version is usually found under the heading "Nonparametric test". It is used when you have rank or ordered data.

You cannot use parametric ANOVA when your data is below interval measurement. Where you have categorical data you do not have an ANOVA method - you would have to use Chi-square, which is about interaction rather than about differences between groups.

How its done

What ANOVA looks at is the way groups differ internally versus what the difference is between them. To take the above example:

ANOVA calculates the mean for each of the final grading groups (HD, D, Cr, P, N) on the tutorial exercise figure - the Group Means.

It calculates the mean for all the groups combined - the Overall Mean.

Then it calculates, within each group, the total deviation of each individual's score from the Group Mean - Within Group Variation.

Next, it calculates the deviation of each Group Mean from the Overall Mean - Between Group Variation.

Finally, ANOVA produces the F statistic which is the ratio Between Group Variation to the Within Group Variation.

If the Between Group Variation is significantly greater than the Within Group Variation, then it is likely that there is a statistically significant difference between the groups.

The statistical package will tell you if the F ratio is significant or not.

All versions of ANOVA follow these basic principles but the sources of Variation get more complex as the number of groups and the interaction effects increase. [26]
Mathematical implementation of back propagation algorithm

For the development of the back-propagation algorithm, we will use the tree layer network introduced earlier

\[ a_1 = f_1(IW_{1,1}^1 p + b_1^1) \]
\[ a_2 = f_2(LW_{2,1}^1 a_1^1 + b_2^2) \]
\[ a_3 = f_3(LW_{3,2}^2 a_2^2 + b_3^3) \]

**Figure 32:** A three-layer network in abbreviated notation

As we discussed earlier, for multilayer networks the output of one layer becomes the input to the following layer. The equations that describe this operation are

\[ a_{m+1} = f_{m+1}(W_{m+1}^m a_m + b_{m+1}) \]

for \( m = 0, 2, \ldots, M-1 \), where \( M \) is the number of layers in the network. The neurons in the first layer receive external inputs:

\[ a_0^0 = p. \]

The outputs of the neurons in the last layer are considered the network outputs:

\[ a = a_M. \]

The algorithm is provided with a set of examples of proper network behavior:

\{ \{p_1, t_1\}, \{p_2, t_2\}, \ldots, \{p_Q, t_Q\} \},

where \( p_q \) is an input to the network, and \( t_q \) is the corresponding target output. As each input is applied to the network, the network output is compared to the target.

Here we will adapt the mean squares error as the performance index (cost function).

The algorithm should adjust the network parameters in order to minimize the mean square error:

\[ F(x) = E \left[ e^2 \right] = E \left[ (t - a)^2 \right], \]

where \( x \) is the vector of network weights and biases. If the network has multiple outputs this generalizes to:

\[ F(x) = E \left[ e^T e \right] = E \left[ (t - a)^T (t - a) \right]. \]

We will approximate the mean squared error by

\[ \hat{F}(x) = (t(k) - a(k))^T (t(k) - a(k)) = e^T (k) e(k), \]

where the expectation of the squared error has been replaced by the squared error at iteration \( k \).

The steepest descent algorithm for the approximate mean square error is

\[ w_{ij}^m(k+1) = w_{ij}^m(k) - a \frac{\partial \hat{F}}{\partial w_{ij}^m}, \]
\[ b_i^m(k+1) = b_i^m(k) - a \frac{\partial \hat{F}}{\partial b_i^m}, \]

where \( a \) is the learning rate.

Now we come to the difficult part of computation of the partial derivatives. For the multilayer network the error is not an explicit function of the weights in the hidden layers, therefore these derivatives are not computed so easily.
Appendix: Back propagation algorithm

Because the error is an indirect function of the weights in the hidden layers, we will use the chain rule of calculus to calculate the derivatives.

Thus for the previous equation the chain rule gives:

\[
\frac{\partial \hat{F}}{\partial w_{i,j}^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial w_{i,j}^m} \quad \text{and} \quad \frac{\partial \hat{F}}{\partial b_j^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial b_j^m}.
\]

The second term in each of these equations can be easily computed, since the net input to layer \( m \) is an explicit function of the weights and bias in that layer:

\[
n_i^m = \sum_{j=1}^{S_{m-1}} w_{i,j}^m a_j^{m-1} + b_j^m \quad \text{and} \quad \frac{\partial n_i^m}{\partial w_{i,j}^m} = a_j^{m-1}, \quad \frac{\partial \hat{F}}{\partial b_j^m} = 1.
\]

For \( s_i^m = \frac{\partial \hat{F}}{\partial n_i^m} \) we have:

\[
\frac{\partial n_i^m}{\partial w_{i,j}^m} = s_j^m a_j^{m-1}, \quad \frac{\partial \hat{F}}{\partial b_i^m} = s_i^m.
\]

We can now express the approximate steepest descent algorithm as

\[
W_i^m(k+1) = W_i^m(k) - \alpha s_i^m a_j^{m-1} \quad \text{and} \quad b_i^m(k+1) = b_i^m(k) - \alpha s_i^m,
\]

In the matrix form this becomes:

\[
W^{m}(k+1) = W^{m}(k) - \alpha \mathbf{s}^{(m)} (\mathbf{a}^{m-1})^{T} \quad \text{and} \quad \mathbf{b}^{m}(k+1) = \mathbf{b}^{m}(k) - \alpha \mathbf{s}^{m},
\]

where \( s \equiv \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \begin{bmatrix} \frac{\partial \hat{F}}{\partial n_1^m} \\ \frac{\partial \hat{F}}{\partial n_2^m} \\ \vdots \\ \frac{\partial \hat{F}}{\partial n_{S_{m}}^m} \end{bmatrix} \).

For the computation of the \( \mathbf{s}^m \) (or sensitivities) we again apply the chain rule. It is this process that gives us the term back-propagation, because it describes a recurrence relationship in which the sensitivity at layer \( m \) is computed from the sensitivity at layer \( m + 1 \).

To derive the recurrence relationship for the sensitivities, we will use the following

\[
\mathbf{j}^{m+1} \equiv \left[ \begin{array}{c} \frac{\partial n_1^{m+1}}{\partial n_1^m} \\ \frac{\partial n_1^{m+1}}{\partial n_2^m} \\ \vdots \\ \frac{\partial n_1^{m+1}}{\partial n_{S_{m+1}}^m} \\ \frac{\partial n_2^{m+1}}{\partial n_1^m} \\ \frac{\partial n_2^{m+1}}{\partial n_2^m} \\ \vdots \\ \frac{\partial n_2^{m+1}}{\partial n_{S_{m+1}}^m} \\ \vdots \\ \frac{\partial n_{S_{m+1}}^{m+1}}{\partial n_1^m} \\ \frac{\partial n_{S_{m+1}}^{m+1}}{\partial n_2^m} \\ \vdots \\ \frac{\partial n_{S_{m+1}}^{m+1}}{\partial n_{S_{m+1}}^m} \end{array} \right].
\]

For each element of this matrix we have:

\[
\frac{\partial n_{j}^{m+1}}{\partial n_{j}^m} = \frac{\partial}{\partial \left( \sum_{k=1}^{S_{m}} w_{j,k}^m a_k^m + b_j^m \right)} \left( \sum_{k=1}^{S_{m}} w_{j,k}^m a_k^m + b_j^m \right) = w_{j,k}^m \frac{\partial a_k^m}{\partial n_{j}^m} = w_{j,k}^m \frac{\partial f^m(n_k^m)}{\partial n_{j}^m} = w_{j,k}^m \hat{f}^m(n_k^m).
\]
Therefore the Jacobian matrix can be written
\[
\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^{m}} = \mathbf{W}^{m+1} \hat{\mathbf{F}}^{m} (\mathbf{n}^{m}),
\]

where \( \hat{\mathbf{F}}^{m} (\mathbf{n}^{m}) = \begin{bmatrix} \hat{f}^{m}(n_1^m) & 0 & \ldots & 0 \\ 0 & \hat{f}^{m}(n_2^m) & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \hat{f}^{m}(n_s^m) \end{bmatrix} \).

We can now write out the recurrence relation for the sensitivity by using the chain rule in matrix form:
\[
\mathbf{s}^{m} = \frac{\partial \hat{\mathbf{F}}}{\partial \mathbf{n}^{m}} = \left( \frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^{m}} \right)^T \frac{\partial \hat{\mathbf{F}}}{\partial \mathbf{n}^{m+1}} = \hat{\mathbf{F}}^{m} (\mathbf{n}^{m}) (\mathbf{W}^{m+1})^T \frac{\partial \hat{\mathbf{F}}}{\partial \mathbf{n}^{m+1}} = \hat{\mathbf{F}}^{m} (\mathbf{n}^{m}) (\mathbf{W}^{m+1})^T \mathbf{s}^{m+1}.
\]

So far we evaluated the sensitivity \( \mathbf{s}^{m} \) for each hidden layer \( m \). The calculation for the output layer sensitivity \( \mathbf{s}^{M} \) (starting point for backward computation) is as follows:

\[
s_{i}^{M} = \frac{\partial \hat{\mathbf{F}}}{\partial n_{i}^{M}} = \frac{\partial (\mathbf{t} - \mathbf{a})^{T} (\mathbf{t} - \mathbf{a})}{\partial n_{i}^{M}} = \frac{\partial \sum_{j=1}^{n_{M}} (t_{j} - a_{j})^{2}}{\partial n_{i}^{M}} = -2(t_{i} - a_{i}) \frac{\partial a_{i}}{\partial n_{i}^{M}}.
\]

Now, since \( \frac{\partial a_{i}}{\partial n_{i}^{M}} = \frac{\partial f^{M}(n_{i}^{M})}{\partial n_{i}^{M}} = f^{M}(n_{i}^{M}) \), we can write \( s_{i}^{M} = -2(t_{i} - a_{i}) f^{M}(n_{i}^{M}) \).

In the matrix form: \( \mathbf{s}^{M} = -2 \hat{\mathbf{F}}^{M} (\mathbf{n}^{M}) (\mathbf{t} - \mathbf{a}) \).
## Results

**Table 4:** ANOVA and Duncan’s test for Classifier performance over Dimensionality Reduction Methods on X-ray Mammography dataset

### ANOVA

<table>
<thead>
<tr>
<th>Method</th>
<th>Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stepwise(3)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Between Groups</td>
<td>.234</td>
<td>4</td>
<td>.059</td>
<td>4.377</td>
<td>.002</td>
</tr>
<tr>
<td>Within Groups</td>
<td>1.941</td>
<td>145</td>
<td>.013</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>2.176</td>
<td>149</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ttest(3)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Between Groups</td>
<td>.119</td>
<td>4</td>
<td>.030</td>
<td>2.717</td>
<td>.032</td>
</tr>
<tr>
<td>Within Groups</td>
<td>1.583</td>
<td>145</td>
<td>.011</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>1.702</td>
<td>149</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>PCA(3)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Between Groups</td>
<td>.280</td>
<td>4</td>
<td>.070</td>
<td>4.862</td>
<td>.001</td>
</tr>
<tr>
<td>Within Groups</td>
<td>2.086</td>
<td>145</td>
<td>.014</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>2.366</td>
<td>149</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Stepwise(3)

<table>
<thead>
<tr>
<th>Classifier</th>
<th>N</th>
<th>Subset for alpha = 0.05</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>30</td>
<td>.63048</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA</td>
<td>30</td>
<td>.64429</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KNN</td>
<td>30</td>
<td>.70683</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANN</td>
<td>30</td>
<td>.71873</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PNN</td>
<td>30</td>
<td>.72460</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sig.</td>
<td></td>
<td>.645</td>
<td>.579</td>
<td></td>
</tr>
</tbody>
</table>

Means for groups in homogeneous subsets are displayed.

- a. Uses Harmonic Mean Sample Size = 30.000.

#### ttest(3)

<table>
<thead>
<tr>
<th>Classifier</th>
<th>N</th>
<th>Subset for alpha = 0.05</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>30</td>
<td>.59714</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PNN</td>
<td>30</td>
<td>.63952</td>
<td>.63952</td>
<td></td>
</tr>
<tr>
<td>KNN</td>
<td>30</td>
<td>.65540</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA</td>
<td>30</td>
<td>.67095</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANN</td>
<td>30</td>
<td>.67476</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sig.</td>
<td></td>
<td>.118</td>
<td>.240</td>
<td></td>
</tr>
</tbody>
</table>

Means for groups in homogeneous subsets are displayed.

- a. Uses Harmonic Mean Sample Size = 30.000.

#### PCA(3)

<table>
<thead>
<tr>
<th>Classifier</th>
<th>N</th>
<th>Subset for alpha = 0.05</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>PNN</td>
<td>30</td>
<td>.50460</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>30</td>
<td>.51143</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KNN</td>
<td>30</td>
<td>.58968</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA</td>
<td>30</td>
<td>.59556</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANN</td>
<td>30</td>
<td>.60190</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sig.</td>
<td></td>
<td>.826</td>
<td>.713</td>
<td></td>
</tr>
</tbody>
</table>

Means for groups in homogeneous subsets are displayed.

- a. Uses Harmonic Mean Sample Size = 30.000.
Table 5: ANOVA and Duncan’s test for Classifier performance over Dimensionality Reduction Methods on DCE MRI dataset

### ANOVA

<table>
<thead>
<tr>
<th></th>
<th>Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stepwise3</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Between Groups</td>
<td>246</td>
<td>4</td>
<td>.061</td>
<td>4.608</td>
<td>.002</td>
</tr>
<tr>
<td>Within Groups</td>
<td>1,935</td>
<td>145</td>
<td>.013</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>2,181</td>
<td>149</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ttest3</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Between Groups</td>
<td>.109</td>
<td>4</td>
<td>.027</td>
<td>1.339</td>
<td>.258</td>
</tr>
<tr>
<td>Within Groups</td>
<td>2,958</td>
<td>145</td>
<td>.020</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>3,067</td>
<td>149</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>PCA3</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Between Groups</td>
<td>.205</td>
<td>4</td>
<td>.051</td>
<td>2.914</td>
<td>.024</td>
</tr>
<tr>
<td>Within Groups</td>
<td>2,554</td>
<td>145</td>
<td>.018</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>2,759</td>
<td>149</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Means for groups in homogeneous subsets are displayed.
a. Uses Harmonic Mean Sample Size = 30.000.

### Stepwise3

<table>
<thead>
<tr>
<th>Classifier</th>
<th>N</th>
<th>Subset for alpha = 0.05</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>30</td>
<td>.73981</td>
</tr>
<tr>
<td>PNN</td>
<td>30</td>
<td>.80463</td>
</tr>
<tr>
<td>KNN</td>
<td>30</td>
<td>.80556</td>
</tr>
<tr>
<td>LDA</td>
<td>30</td>
<td>.82546</td>
</tr>
<tr>
<td>ANN</td>
<td>30</td>
<td>.86481</td>
</tr>
<tr>
<td>Sig.</td>
<td></td>
<td>1.000 .066</td>
</tr>
</tbody>
</table>

### ttest3

<table>
<thead>
<tr>
<th>Classifier</th>
<th>N</th>
<th>Subset for alpha = 0.05</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>30</td>
<td>.72269</td>
</tr>
<tr>
<td>KNN</td>
<td>30</td>
<td>.76667 .76667</td>
</tr>
<tr>
<td>ANN</td>
<td>30</td>
<td>.76667 .76667</td>
</tr>
<tr>
<td>PNN</td>
<td>30</td>
<td>.77824 .77824</td>
</tr>
<tr>
<td>LDA</td>
<td>30</td>
<td>.80648</td>
</tr>
<tr>
<td>Sig.</td>
<td></td>
<td>.174 .332</td>
</tr>
</tbody>
</table>

Means for groups in homogeneous subsets are displayed.
a. Uses Harmonic Mean Sample Size = 30.000.

### PCA3

<table>
<thead>
<tr>
<th>Classifier</th>
<th>N</th>
<th>Subset for alpha = 0.05</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>30</td>
<td>.68009</td>
</tr>
<tr>
<td>LDA</td>
<td>30</td>
<td>.75417</td>
</tr>
<tr>
<td>PNN</td>
<td>30</td>
<td>.75833</td>
</tr>
<tr>
<td>KNN</td>
<td>30</td>
<td>.77917</td>
</tr>
<tr>
<td>ANN</td>
<td>30</td>
<td>.78194</td>
</tr>
<tr>
<td>Sig.</td>
<td></td>
<td>1.000 .468</td>
</tr>
</tbody>
</table>

Means for groups in homogeneous subsets are displayed.
a. Uses Harmonic Mean Sample Size = 30.000.
### Results

**Table 6:** ANOVA and Duncan’s test for Classifier performance over Dimensionality Reduction Methods on HRCT dataset

#### ANOVA

<table>
<thead>
<tr>
<th>Method</th>
<th>Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Between Groups</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stepwise9</td>
<td>.022</td>
<td>4</td>
<td>.006</td>
<td>4.083</td>
<td>.004</td>
</tr>
<tr>
<td></td>
<td>Within Groups</td>
<td>.196</td>
<td>145</td>
<td>.001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>.220</td>
<td>149</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ttest9</td>
<td>Between Groups</td>
<td>.015</td>
<td>4</td>
<td>.004</td>
<td>1.861</td>
</tr>
<tr>
<td></td>
<td>Within Groups</td>
<td>.293</td>
<td>145</td>
<td>.002</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>.308</td>
<td>149</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCA9</td>
<td>Between Groups</td>
<td>.017</td>
<td>4</td>
<td>.004</td>
<td>2.192</td>
</tr>
<tr>
<td></td>
<td>Within Groups</td>
<td>.274</td>
<td>145</td>
<td>.002</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>.291</td>
<td>149</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Stepwise9 Duncan

<table>
<thead>
<tr>
<th>Classifier</th>
<th>N</th>
<th>Subset for alpha = 0.05</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>LDA</td>
<td>30</td>
<td>.93552</td>
</tr>
<tr>
<td>SVM</td>
<td>30</td>
<td>.96383</td>
</tr>
<tr>
<td>PNN</td>
<td>30</td>
<td>.96498</td>
</tr>
<tr>
<td>KNN</td>
<td>30</td>
<td>.96609</td>
</tr>
<tr>
<td>ANN</td>
<td>30</td>
<td>.98824</td>
</tr>
<tr>
<td>Sig.</td>
<td></td>
<td>1.000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Means for groups in homogeneous subsets are displayed.

#### ttest9 Duncan

<table>
<thead>
<tr>
<th>Classifier</th>
<th>N</th>
<th>Subset for alpha = 0.05</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>SVM</td>
<td>30</td>
<td>.92663</td>
</tr>
<tr>
<td>LDA</td>
<td>30</td>
<td>.93789</td>
</tr>
<tr>
<td>PNN</td>
<td>30</td>
<td>.94352</td>
</tr>
<tr>
<td>KNN</td>
<td>30</td>
<td>.95134</td>
</tr>
<tr>
<td>ANN</td>
<td>30</td>
<td>.95471</td>
</tr>
<tr>
<td>Sig.</td>
<td></td>
<td>.052</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Means for groups in homogeneous subsets are displayed.

#### PCA9 Duncan

<table>
<thead>
<tr>
<th>Classifier</th>
<th>N</th>
<th>Subset for alpha = 0.05</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>SVM</td>
<td>30</td>
<td>.92096</td>
</tr>
<tr>
<td>PNN</td>
<td>30</td>
<td>.92548</td>
</tr>
<tr>
<td>LDA</td>
<td>30</td>
<td>.93103</td>
</tr>
<tr>
<td>KNN</td>
<td>30</td>
<td>.94234</td>
</tr>
<tr>
<td>ANN</td>
<td>30</td>
<td>.94923</td>
</tr>
<tr>
<td>Sig.</td>
<td></td>
<td>.084</td>
</tr>
</tbody>
</table>

Means for groups in homogeneous subsets are displayed.
Table 7: ANOVA and Duncan’s test for Classifier performance over Dimensionality Reduction Methods on Cytology dataset

### ANOVA

<table>
<thead>
<tr>
<th></th>
<th>Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stepwisettest3</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Between Groups</td>
<td>.008</td>
<td>4</td>
<td>.002</td>
<td>5.766</td>
<td>.000</td>
</tr>
<tr>
<td>Within Groups</td>
<td>.052</td>
<td>145</td>
<td>.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>.061</td>
<td>149</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>PCA3</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Between Groups</td>
<td>.013</td>
<td>4</td>
<td>.003</td>
<td>8.313</td>
<td>.000</td>
</tr>
<tr>
<td>Within Groups</td>
<td>.055</td>
<td>145</td>
<td>.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>.068</td>
<td>149</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Stepwisettest3

<table>
<thead>
<tr>
<th>Classifier</th>
<th>N</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>30</td>
<td>.94899</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANN</td>
<td>30</td>
<td>.95661</td>
<td>.95661</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PNN</td>
<td>30</td>
<td>.96088</td>
<td>.96088</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA</td>
<td>30</td>
<td>.96616</td>
<td>.96616</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KNN</td>
<td>30</td>
<td></td>
<td></td>
<td>.97043</td>
<td></td>
</tr>
<tr>
<td>Sig.</td>
<td></td>
<td>.123</td>
<td>.067</td>
<td>.067</td>
<td></td>
</tr>
</tbody>
</table>

Means for groups in homogeneous subsets are displayed.

- Uses Harmonic Mean Sample Size = 30.000.

### PCA3

<table>
<thead>
<tr>
<th>Classifier</th>
<th>N</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>30</td>
<td>.94899</td>
<td></td>
</tr>
<tr>
<td>ANN</td>
<td>30</td>
<td>.96233</td>
<td></td>
</tr>
<tr>
<td>KNN</td>
<td>30</td>
<td>.96758</td>
<td></td>
</tr>
<tr>
<td>PNN</td>
<td>30</td>
<td>.96758</td>
<td></td>
</tr>
<tr>
<td>Sig.</td>
<td></td>
<td>.568</td>
<td>.331</td>
</tr>
</tbody>
</table>

Means for groups in homogeneous subsets are displayed.

- Uses Harmonic Mean Sample Size = 30.000.
Table 8: ANOVA and Duncan’s test for Dimensionality Reduction Methods over Classifier performance on X-ray Mammography dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Feature</th>
<th>Subset for alpha = 0.05</th>
<th>N</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>PCA</td>
<td></td>
<td>30</td>
<td>.60190</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Duncan*</td>
<td></td>
<td>30</td>
<td>.67476</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>stepwise</td>
<td></td>
<td>30</td>
<td>.71873</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td></td>
<td></td>
<td>1.000</td>
<td>.214</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KNN</td>
<td>PCA</td>
<td></td>
<td>30</td>
<td>.51143</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Duncan*</td>
<td></td>
<td>30</td>
<td>.59714</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>stepwise</td>
<td></td>
<td>30</td>
<td>.63048</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td></td>
<td></td>
<td>1.000</td>
<td>.241</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>PCA</td>
<td></td>
<td>30</td>
<td>.51143</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Duncan*</td>
<td></td>
<td>30</td>
<td>.59714</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>stepwise</td>
<td></td>
<td>30</td>
<td>.63048</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td></td>
<td></td>
<td>1.000</td>
<td>.241</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PNN</td>
<td>PCA</td>
<td></td>
<td>30</td>
<td>.59460</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Duncan*</td>
<td></td>
<td>30</td>
<td>.63952</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>stepwise</td>
<td></td>
<td>30</td>
<td>.72460</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td></td>
<td></td>
<td>1.000</td>
<td>1.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA</td>
<td>PCA</td>
<td></td>
<td>30</td>
<td>.59556</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Duncan*</td>
<td></td>
<td>30</td>
<td>.64429</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>stepwise</td>
<td></td>
<td>30</td>
<td>.67095</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td></td>
<td></td>
<td>1.29</td>
<td>.404</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 30.000.
Table 9: ANOVA and Duncan’s test for Dimensionality Reduction Methods over Classifier performance on DCE MRI dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>Between Groups</td>
<td>.167</td>
<td>2</td>
<td>.084</td>
<td>5.553</td>
</tr>
<tr>
<td></td>
<td>Within Groups</td>
<td>1.311</td>
<td>87</td>
<td>.015</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>1.478</td>
<td>89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KNN</td>
<td>Between Groups</td>
<td>.024</td>
<td>2</td>
<td>.012</td>
<td>.687</td>
</tr>
<tr>
<td></td>
<td>Within Groups</td>
<td>1.498</td>
<td>87</td>
<td>.017</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>1.522</td>
<td>89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>Between Groups</td>
<td>.057</td>
<td>2</td>
<td>.028</td>
<td>1.437</td>
</tr>
<tr>
<td></td>
<td>Within Groups</td>
<td>1.718</td>
<td>87</td>
<td>.020</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>1.775</td>
<td>89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PNN</td>
<td>Between Groups</td>
<td>.032</td>
<td>2</td>
<td>.016</td>
<td>.902</td>
</tr>
<tr>
<td></td>
<td>Within Groups</td>
<td>1.560</td>
<td>87</td>
<td>.018</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>1.592</td>
<td>89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA</td>
<td>Between Groups</td>
<td>.062</td>
<td>2</td>
<td>.041</td>
<td>2.615</td>
</tr>
<tr>
<td></td>
<td>Within Groups</td>
<td>1.361</td>
<td>87</td>
<td>.016</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>1.442</td>
<td>89</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For ANOVA:
- **ANN**
  - Feature: ttest, PCA, stepwise
  - Means for groups in homogeneous subsets are displayed.
- **KNN**
  - Feature: ttest, pca, stepwise
  - Means for groups in homogeneous subsets are displayed.
- **SVM**
  - Feature: ttest, pca, stepwise
  - Means for groups in homogeneous subsets are displayed.
- **PNN**
  - Feature: ttest, pca, stepwise
  - Means for groups in homogeneous subsets are displayed.
- **LDA**
  - Feature: ttest, pca, stepwise
  - Means for groups in homogeneous subsets are displayed.

*Means for groups in homogeneous subsets are displayed. a. Uses Harmonic Mean Sample Size = 30.000.*
## Table 10: ANOVA and Duncan’s test for Dimensionality Reduction Methods over Classifier performance on HRCT dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ANN</strong></td>
<td>Between Groups</td>
<td>.006</td>
<td>2</td>
<td>.003</td>
<td>1.948</td>
</tr>
<tr>
<td></td>
<td>Within Groups</td>
<td>.128</td>
<td>87</td>
<td>.001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>.134</td>
<td>89</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>KNN</strong></td>
<td>Between Groups</td>
<td>.009</td>
<td>2</td>
<td>.004</td>
<td>2.798</td>
</tr>
<tr>
<td></td>
<td>Within Groups</td>
<td>.134</td>
<td>87</td>
<td>.002</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>.143</td>
<td>89</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SVM</strong></td>
<td>Between Groups</td>
<td>.033</td>
<td>2</td>
<td>.016</td>
<td>8.864</td>
</tr>
<tr>
<td></td>
<td>Within Groups</td>
<td>.160</td>
<td>87</td>
<td>.002</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>.192</td>
<td>89</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>PNN</strong></td>
<td>Between Groups</td>
<td>.023</td>
<td>2</td>
<td>.012</td>
<td>6.739</td>
</tr>
<tr>
<td></td>
<td>Within Groups</td>
<td>.151</td>
<td>87</td>
<td>.002</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>.175</td>
<td>89</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LDA</strong></td>
<td>Between Groups</td>
<td>.001</td>
<td>2</td>
<td>.000</td>
<td>.165</td>
</tr>
<tr>
<td></td>
<td>Within Groups</td>
<td>.191</td>
<td>87</td>
<td>.002</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>.192</td>
<td>89</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### ANN

<table>
<thead>
<tr>
<th>Feature</th>
<th>N</th>
<th>Duncan&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td>30</td>
<td>.94923</td>
</tr>
<tr>
<td>ttest</td>
<td>30</td>
<td>.95471</td>
</tr>
<tr>
<td>stepwise</td>
<td>30</td>
<td>.96824</td>
</tr>
<tr>
<td>Sig.</td>
<td></td>
<td>.073</td>
</tr>
</tbody>
</table>

### KNN

<table>
<thead>
<tr>
<th>Feature</th>
<th>N</th>
<th>Duncan&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td>30</td>
<td>.94234</td>
</tr>
<tr>
<td>ttest</td>
<td>30</td>
<td>.95134</td>
</tr>
<tr>
<td>stepwise</td>
<td>30</td>
<td>.96609</td>
</tr>
<tr>
<td>Sig.</td>
<td></td>
<td>.377</td>
</tr>
</tbody>
</table>

### SVM

<table>
<thead>
<tr>
<th>Feature</th>
<th>N</th>
<th>Duncan&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td>30</td>
<td>.92096</td>
</tr>
<tr>
<td>ttest</td>
<td>30</td>
<td>.92663</td>
</tr>
<tr>
<td>stepwise</td>
<td>30</td>
<td>.96383</td>
</tr>
<tr>
<td>Sig.</td>
<td></td>
<td>.610</td>
</tr>
</tbody>
</table>

### PNN

<table>
<thead>
<tr>
<th>Feature</th>
<th>N</th>
<th>Duncan&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td>30</td>
<td>.92548</td>
</tr>
<tr>
<td>ttest</td>
<td>30</td>
<td>.94352</td>
</tr>
<tr>
<td>stepwise</td>
<td>30</td>
<td>.96498</td>
</tr>
<tr>
<td>Sig.</td>
<td></td>
<td>.098</td>
</tr>
</tbody>
</table>

### LDA

<table>
<thead>
<tr>
<th>Feature</th>
<th>N</th>
<th>Duncan&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td>30</td>
<td>.93103</td>
</tr>
<tr>
<td>ttest</td>
<td>30</td>
<td>.93552</td>
</tr>
<tr>
<td>stepwise</td>
<td>30</td>
<td>.93789</td>
</tr>
<tr>
<td>Sig.</td>
<td></td>
<td>.598</td>
</tr>
</tbody>
</table>

Means for groups in homogeneous subsets are displayed.
a. Uses Harmonic Mean Sample Size = 30.000.

### ANOVA

- **Mean Square**: The mean square is calculated by dividing the sum of squares by its degrees of freedom (df).
- **F**: The F-statistic is used to test the null hypothesis that the means of the groups are equal.
- **Sig.**: The significance level (p-value) indicates the probability of observing the F-statistic if the null hypothesis is true.

### Duncan’s Test

Duncan’s test is a multiple comparison procedure that identifies subsets of groups that do not differ significantly from each other. The subset values are calculated based on the Harmonic Mean Sample Size, which is used to adjust for unequal sample sizes across groups.

- **Harmonic Mean Sample Size**: 30.000

---

80
Table 11: ANOVA for Dimensionality Reduction Methods over Classifier performance on Cytology dataset

<table>
<thead>
<tr>
<th></th>
<th>Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>Between Groups</td>
<td>.000</td>
<td>1</td>
<td>.000</td>
<td>1.317</td>
</tr>
<tr>
<td></td>
<td>Within Groups</td>
<td>.022</td>
<td>58</td>
<td>.000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>.022</td>
<td>59</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KNN</td>
<td>Between Groups</td>
<td>.000</td>
<td>1</td>
<td>.000</td>
<td>.469</td>
</tr>
<tr>
<td></td>
<td>Within Groups</td>
<td>.015</td>
<td>58</td>
<td>.000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>.015</td>
<td>59</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>Between Groups</td>
<td>.000</td>
<td>1</td>
<td>.000</td>
<td>.000</td>
</tr>
<tr>
<td></td>
<td>Within Groups</td>
<td>.029</td>
<td>58</td>
<td>.001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>.029</td>
<td>59</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PNN</td>
<td>Between Groups</td>
<td>.001</td>
<td>1</td>
<td>.001</td>
<td>1.878</td>
</tr>
<tr>
<td></td>
<td>Within Groups</td>
<td>.021</td>
<td>58</td>
<td>.000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>.021</td>
<td>59</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA</td>
<td>Between Groups</td>
<td>.006</td>
<td>1</td>
<td>.006</td>
<td>16.390</td>
</tr>
<tr>
<td></td>
<td>Within Groups</td>
<td>.021</td>
<td>58</td>
<td>.000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>.027</td>
<td>59</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Reference List

17. Andersen, T. and T. Martinez, *Cross Validation and MLP Architecture Selection*.