

**Classification of pap-smear data by transductive
neuro-fuzzy methods**
(Master Thesis)

Norup, Jonas c960566

May 2, 2005

¹Supervisor : Jantzen, Jan (jj@oersted.dtu.dk) Ørsted-DTU, Automation Building 326 DK-2800
Kongens Lyngby DENMARK

Abstract

This thesis deals with pattern classification of single pap-smear cells from an existing database developed on Herlev University Hospital. Medical, the method can be used for detecting pre-malignant cells in uterine cervix before the progress into cancer. Available cell features like area, position and brightness of nucleus and cytoplasm are used for the classification into normal and abnormal cells. The classifier performance is measured on the percentage overall error, but also false-negative and false-positive error performance is investigated. The goal is to improve the overall error, compared to previously achieved results.

Different methods were proposed, both global inductive models like linear least square networks(LS) and global transductive models like K-nearest Neighbor(KNN), Weighted K-nearest Neighbor(WKNN) and Nearest NeighborHood(NNH). The applicability of a newly introduced advanced method: Neuro-Fuzzy Inference for Transductive Reasoning(NFI) is also tested.

In the light of previous pap-smear result with a large false-positive error, which probably is descended from a predominance of abnormal samples, a new method is constructed - Nearest Class Center of gravity(NCC). This method actually showed best result of all tested classifiers. An overall error(OE%) of 5.1%, and false-negative/false-positive roughly equally sized.

Results show that advanced NFI performs worse, than the more simple NCC method both applied to the pap-smear data and the Iris benchmark data. The achieved error was 5.1%/5.7% for NFI/NCC on pap-smear and 4.5%/3.6% for the iris data. The simple LS performed with an error of 6.4%

Comparing the simple transductive methods KNN, WKNN and NNH shows almost identical results, an overall error of approximate 6.5%

Through a detailed description of the latest pap-smear data from Herlev University Hospital a new benchmark database *Papsmear* is prepared for publishing on WWW. The *papsmear* database consist of 917 samples distributed unequally on 7 different classes of normal and abnormal cells. Each sample is described by 20 features.

The general performance of the classifiers tested, does not show significant improvement of earlier result, but matching results is achieved using very simple methods.

Keywords

inductive, transductive reasoning, neuro-fuzzy inference, classification, Pap-Smear, simple linear methods.

Table of Contents

1	Introduction	1
1.1	Background	1
1.2	Objective	2
1.3	Literature	2
1.3.1	Transductive vs. Inductive methods	3
1.4	Problem Statement	5
1.5	Work plan	5
1.6	Performance calculation	6
1.6.1	Error measurements	6
1.6.2	k-fold cross-validation	10
1.6.3	Cross-validation rerunning	10
2	The pap-smear Database	11
2.1	The pap-smear screening	11
2.2	The new pap-smear database	15
2.3	Building a classifier	19
3	Inductive classifiers	23
3.1	Least Square Method (LS)	23
3.1.1	Least Square Method Theory	23
3.2	Results	27
3.2.1	Test conditions	27
3.2.2	Least Square Method(LS)	29
3.3	Conclusion	34

4	Transductive classifiers	37
4.1	Methods	37
4.1.1	KNN and WKNN	38
4.1.2	NCC : Nearest Class gravity Center	42
4.1.3	NFI : Neurofuzzy Inference Method for Transductive Reasoning	45
4.1.4	Evolving Clustering Method(ECM)	50
4.2	Results	52
4.2.1	KNN and WKNN	52
4.2.2	NCC - Nearest Class gravity Center	55
4.2.3	NFI results	56
4.3	Conclusion	60
5	Summary of results & discussion	63
6	Conclusion	67
A	CD-rom	71

List of Figures

1.1	Singe pap-smear cell image.	1
1.2	Transductive vs. Inductive models	4
1.3	Confusion matrix	9
1.4	Confusion matrices	9
2.1	Image of single pap-smear image	12
2.2	The uterus in details and the location of <i>a)columnnar</i> cells and <i>b)squamous</i> cells. (Adopted from Dr.Indman, 2005)	13
2.3	Development of the <i>squamous</i> cells through the four layers	13
2.4	Cell type characteristics for single pap-smear cells. (to fit the table, some images are slighty scaled)	14
2.5	Cell picture of a <i>normal superficial</i> cell - class 1	17
2.6	A segmented <i>normal superficial</i> cell - class 1	17
2.7	A binary cell picture with background(white), cytoplasm(black) and nucleus(gray). For the cytoplasm the longest diameter line(L) and shortest diameter(S1 & S2) are shown. Adopted from (Martin, 2003)	18
2.8	Class mean and standard deviation of pap-smear data - Nucleus Area(feature 1) and NC-ratio (feature 3)	21
3.1	Linear network	24
3.2	$g(y)$ - hard limiter activation function - $thr = 1.5$	25
3.3	Simple Matlab script for linear classification.	26
3.4	Least Square Method(LS) - Optimal k for k-fold	28
3.5	Least Square Method(LS) - Optimal number of reruns ($k = 10$)	28
3.6	Overall error OE%, k=2 - Using 10.000 repetitions shows that the error is normal distributed	31

3.7	Overall error OE%, $k=10$ - Using 10.000 repetitions shows that the error is normal distributed	31
3.8	RMSE, $k=2$ - Using 10.000 repetitions to show random distribution	31
3.9	RMSE, $k=10$ - Using 10.000 repetitions to show random distribution	31
3.10	Train error - Threshold dependency on FN%, FP%, OE%	32
3.11	Test error - Threshold dependency on FN,%, FP%, OE%	32
3.12	$g(y)$ - hard limiter activation function 7 classes - thr = [1.5, 2.5, ..., 6.5] . .	32
4.1	Example KNN	41
4.2	Example of NCC: Nearest Class gravity Center	44
4.3	Flowchart of the NFI-algorithm. (Adopted from Kasabov&Song(2005)) . . .	46
4.4	Example ECM	51
4.5	RMSE of WKNN and KNN - scale : range[0;1]	54
4.6	Overall error(OE%) of WKNN and KNN - scale : range[0;1]	54
4.7	RMSE of WKNN and KNN - scale : mean=0/std=1	54
4.8	Overall error(OE%) of WKNN and KNN - scale : mean=0/std=1	54
4.9	RMSE of NNH	55
4.10	Overall error(OE%)of NNH	55
4.11	NCC-2 classes with different scalings - leave-one-out	56
4.12	ECM iris data. Cluster dependency of distance $Dthr$ and number of K nearest neighbors	58
4.13	ECM iris data. Cluster dependency of distance $Dthr$ and number of K nearest neighbors. Zoom of figure	58
4.14	ECM clustering on pap-smear data. Cluster dependency of distance $Dthr$ and number of N_q nearest neighbors.	60
4.15	NFI pap-smear data.	61

List of Tables

1.1	False-positive(FP) and False-negative(FN) definition	3
1.2	Main results by Erik Martin (2003) and Jens Byriel (1999)	3
1.3	<i>Inductive</i> vs. <i>Transductive</i>	5
2.1	The 7 different pap-smear classes	16
2.2	Extracted features for pap-smear data. N and C are abbreviation of the Nucleus and Cytoplasm. The letter in brackets refers to the column in the spreadsheet of the new pap-smear database on the enclosed CD-rom. Column (A) in the spreadsheet is used for images reference.	17
2.3	N/C-ratio class mean values compared to medical observations from figure 2.4	20
2.4	Pap-smear class mean values and standard deviation. Mean values with normal font and standard deviation in italic and brackets.	22
3.1	Test error Least Square method. 2 class problem - threshold=1.5 k=10 rerun=50	29
3.2	Number of test errors - k=10. With $k = 10$ the number of testdata is $N_{test} = N \frac{1}{k} = 917 \cdot \frac{1}{10} \approx 91.7$	29
3.3	Test error Least Square(LS) into 7 classes using scalar class description - k=10 rerun=50	33
3.4	Test error Least Square(LS) into 7 direct classes using scalar class description - k=10 rerun=50	33
3.5	Test error Least Square method - 2 class problem - using all 7 classes for scalar model building - k=10 rerun=50	33
3.6	Test error Least Square(LS) into 7 classes using binary class description - k=10 rerun=50	34

3.7	Test error Least Square(LS) into 7 classes using binary class description - k=10 rerun=50	34
3.8	pap-smear results - test errors	35
4.1	Results WKNN and KNN from example 4.2	42
4.2	WKNN and KNN - scale : $range[0;1]$ - leave-one-out	53
4.3	WKNN and KNN - scale : $mean=0/std=1$ - leave-one-out	53
4.4	NNH - scale : $mean=0/std=1$ - leave-one-out	55
4.5	NCC - 2 classes with different scalings - leave-one-out	55
4.6	NFI tested on Iris data using different combinations of the number of N_q nearest neighbors and the ECM distance threshold $Dthr$. The errors is an average of 10 repetitions, randomly selecting 50% of 150 sample for test data.	59
4.7	Test on <i>iris Data</i>	59
4.8	NFI pap-smear data - leave-one-out	60
5.1	Inductive classification	64
5.2	Summary of classification results	65

Chapter 1

Introduction

1.1 Background

An early and effective treatment is the most important means to prevent pre-cancerous cells to develop. Using an old technique named pap-smear, developed by Georges Papanicolaou, it is possible to stain the cells. And using a microscope enables cyto-technicians to detect pre-cancerous cells in the uterine cervix. Figure 1.1 shows such a single cell. Among other places the method is used in Herlev University Hospital. On the Department of Pathology the classification is done by well trained cyto-technicians specimen by specimen, using a microscope. This method is not that fast and it requires that well skilled cyto-technicians

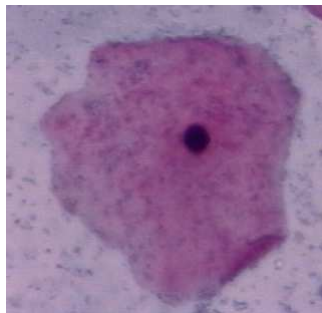


Figure 1.1: Singe pap-smear cell image.

are available. But using a computer for the classification could be a much more optimal solution - but of course having in mind that error-rate is not increased. Some systems are already available, but rather expensive so far¹. In co-operation with Herlev University Hospital several Master's thesis projects has been carried out during the last 10 years, both on Ørsted, DTU and University of Aegean, Chios, Greece. This thesis build on results from

¹<http://www.dimac-imaging.com/sider/projects/euroscreen.htm>

such projects. Besides good basic knowledge and many classifiers based on different methods, this collaboration has also yielded the development of a large database of smear data, carefully examined by cyto-technicians and medical doctors. Publishing of this database, along with a good data description on the World Wide Web, could challenge researchers to apply new methods to great benefit for University Hospital and others working with pap-smear.

Further on the general research into pattern recognition, have recently introduced new methods. The theories behind these methods are not revolutionary new, but more like extensions of existing ones. A method named *NFI: Neuro-fuzzy Inference for Transductive Reasoning* introduced by N. Kasabov & Q. Song (2005) shows quite good results applied on medical data and is an obvious method to try on the pap-smear data.

1.2 Objective

The objective is two-fold

- To publish a good benchmark database of pap-smear data used for testing and comparing various classification methods
- To evaluate the NFI method using these data and compare the results to previous classifiers tested on the data

1.3 Literature

During the research projects on Herlev University Hospital two data sets (*oldData* and *newData*) have been constructed by the cyto-technicians for classification purposes. The latest data set(*newData*) is used by Erik Martin(2003) for both feature extraction and classification. The feature extraction is here handled by Martin(2003) himself, doing image processing in Matlab. Applied to the *oldData* from Herlev University Hospital, his feature extraction showed quite good results. The reference is the features extracted with the CHAMP¹-software from the same set of *oldData* by Byriel(1999). Martin(2003) compares classification results of *oldData* versus *newData*, concluding that a desirable error rate of 5% for the false-negative%(FN%) and for the false-positive%(FP%) as well, are not achieved for the *newData*. The results also show that false-positive error is significantly larger than the false-negative.

In medical terms a bad smear screening is referred to as a *positive* test result. So positive cells mean abnormal cells and negative cells mean normal cells as showed in table 1.1.

¹CHAMP. - Commercial image processing software for medical purpose by dimac-imaging. - <http://www.dimac-imaging.com/sider/products/champ.htm>

		True class	
		normal(÷)	abnormal(+)
Estimated class	normal(÷)	TN	FN
	abnormal(+)	FP	TP

Table 1.1: False-positive(FP) and False-negative(FN) definition

Seen from a medical point of view the false-negative error is the most disastrous to make, because patients with a pre-cancerous indication seem to be fit.

The main methods used by Martin(2003) were Hard C-means(HCM), Fuzzy C-means(FCM) and Gustafson-Kessel clustering(GK). Finding the optimal fuzzy exponent for FCM was one of the investigations. Also direct and hierarchical classifiers were examined, but no linear classifier was tested - either by Martin(2003) or Byriel(1999). The main result of Martin(2003) Byriel(1999) are organized in table 1.2.

Pap-Smear classification (Erik Martin 2003)	Neuro-fuzzy classification of Cells in Cervial Smears (Jens Byriel 1999)
Feature extraction using Matlab Data : both oldData & newData Feature selection Inductive Methods <ul style="list-style-type: none"> ○ HCM, FCM ○ GK Results <ul style="list-style-type: none"> ○ oldData total <5% ○ newData total >5% ○ FP% >> FN% 	Feature extraction using CHAMP Data : only oldData Feature selection Inductive Methods <ul style="list-style-type: none"> ○ FCM — ○ GK ○ ANFIS Transductive Methods <ul style="list-style-type: none"> ○ NNH Results <ul style="list-style-type: none"> ○ oldData total <5%

Table 1.2: Main results by Erik Martin (2003) and Jens Byriel (1999)

All the methods examined by Martin(2003) were global models. A large model is created and then trained to fit the presented training data in an optimal way - minimizing an error function. Afterwards the unseen test data is presented to a global model. And a training error is calculated to show the performance of the model.

1.3.1 Transductive vs. Inductive methods

These global methods are referred to as *inductive* (Kasabov&Song, 2005) showed on the left hand side of figure 1.2. Here the entire set of training data **D** is used for generating the global model **M_g** once for all.

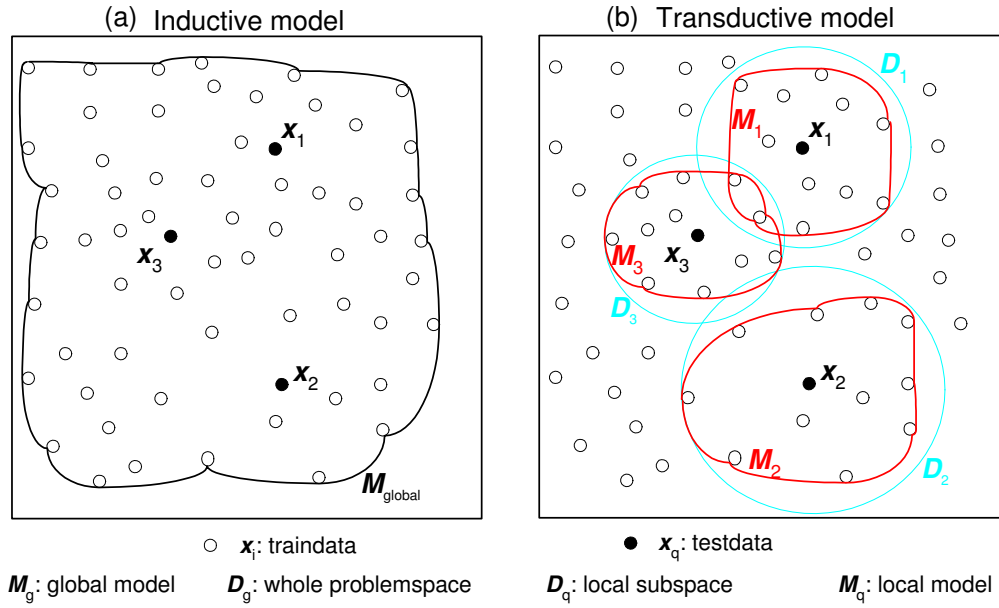


Figure 1.2: Transductive vs. Inductive models

Alternative using small individual model are described as *transductive* also by Kasabov & Song(2005). A subset D_q of K training samples, in the vicinity of a single test point x_q , generates a local model M_q only valid for x_q - right hand side of figure 1.2.

Such method were used by Byriel(1999) eg. NNH - Nearest NeighborHood clustering given quite good results on the previous pap-smear data *oldData*. Using weights to favour samples close the test point x_q . NNH is an extension of common models like KNN (K-nearest neighbor) - see section 4.1.1 for details.

A similar method Weighted K-Nearest Neighbor(WKNN) is also described in Kasabov & Song(2005) and in section 4.1.1 in this thesis. In general the subspace D_q is selected using a distance measurements between x_q and each of the surrounding training points - sketched by the circles in figure 1.2(b).

The method is similar to NNH, but different weights are used. WKNN is presented as an introduction to the NFI - Neuro-Fuzzy Inference Method for Transductive Reasoning (Kasabov&Song, 2005). During an advanced clustering process a fuzzy rule system is updated using Back-Propagation(BP). The method performs quite good results applied on medical data, especially for time series prediction, but also for classification. Table 1.3 shows the differences of transductive vs. inductive methods.

	Inductive	Transductive
Model	global	local
Train data	all	N closest
Ex.	MLP(Multi Layer perceptron) HCM, FCM(Hard- & Fuzzy C-means) LS(Least Square Network)	KNN(K Nearest Neighbor) WKNN(Weighted KNN) NFI

Table 1.3: *Inductive vs. Transductive*

1.4 Problem Statement

To publish the new pap-smear data for benchmark testing, requires a complete description of the data set. The description have to be a stand-alone chapter for easy publishing on World Wide Web. All input features have to be specified individually, to state their physical origin. Images should be included for end-user processing. A complete description also requires a data description, introducing the different cell types used in pap-smear terminology. In this thesis new transductive methods are applied for classification of pap-smear to improve earlier results by Martin(2003)

- False-Negative error $(FN\%) < 3.26\%$
- False-Positive error $(FP\%) < 13.88\%$
- Overall error $(OE\%) < 6.06\%$

Definition of these performance measurements are given in section 1.6. The newly introduced method NFI(Neuro-Fuzzy Inference of Transductive Reasoning), is expected to show the best performance, because it is working on a local subspace, and use Back-Propagation(BP) for training the model. But introducing simple linear methods hopefully shows adequately results, compared to the fact that the methods are easy to implement and performs fast classifications.

1.5 Work plan

The list below shows activities that have to be done:

- Step 1. Prepare benchmark database
 - Introducing the pap-smear method
 - Describe database features
 - Describe database classes

- Step 2. Common Sense analysis (*Inductive*)
 - Analyze database for classification purpose
 - Least Squared Analysis
- Step 3. NFI-algorithm (*Transductive*)
 - Test simple WKNN/KNN methods
 - Performing clustering for NFI
 - Testing NFI

This work plan is only used to list activities, that has to be carried out. And new things will properly pop-up during the working process. The working order will properly differ from the numbering above.

1.6 Performance calculation

To evaluate the performance of the classifiers build and to compare different models, various performance measurements are introduced below. Methods to achieve confident results are also discussed.

1.6.1 Error measurements

To evaluate the performance of the classifiers build, four different performance measurements are introduced: False-Negative(FN) error, False-Positive(FP) error, Overall Error (OE) and Root-Mean-Square Error(RMSE).

As described a bad smear screening is referred to as a *positive* test result, so positive cells means abnormal cells and negative cells means normal cells in the following definitions. Equation 1.1 and 1.2 defines the FN% and FP%.

$$FN\% = \frac{FN}{TP + FN} \times 100\% \quad (1.1)$$

$$FP\% = \frac{FP}{TN + FP} \times 100\% \quad (1.2)$$

FN and FP indicate the number of cells misclassified as negative and positive cells, respectively. TN and TP are the number of cells truly classified. As described earlier in section 1.3 a FN-error is more disastrous to make than a FP-error, because patients with a

pre-cancerous indication wrongly **seem** to be fit. The total error is defined in equation 1.3 as the number of errors relative to all processed cells

$$OE\% = \frac{FN + FP}{TP + FN + TN + FP} \times 100\% \quad (1.3)$$

All three error measurements introduced above, are classification errors - a sort of integer error-estimates. Since a sample is either right or wrong - and nothing in between - the percentage error is changed in step - like a discrete function. The problem is described in the example below.

Example 1.1 Consider a classification of 100 samples with the following result: $TP = 67$, $TN = 23$, $FN = 5$ and $FP = 5$. Using the definitions above the performance error is:

$$FN\% = \frac{5}{67+5} \times 100\% = 6.9\%$$

$$FP\% = \frac{5}{23+5} \times 100\% = 17.9\%$$

$$OE\% = \frac{5+5}{67+5+23+5} \times 100\% = 10.0\%$$

Applying the smallest possible changes to the previous classification results, shows the minimum step that the errors can change. The sum in the denominator in equation 1.1 and 1.2 are constant. The sums $(TP + FN)$ and $(TN + FP)$ are equal to the number of positive- and negative cells in the data set. Changing 1 sample $TP \Rightarrow FN$ and one sample $FP \Rightarrow TN$ gives $TP = 67 - 1$, $FN = 5 + 1$ $TN = 23 + 1$ and $FP = 5 - 1$

$$FN\% = \frac{6}{66+6} \times 100\% = 8.3\%$$

$$FP\% = \frac{4}{24+4} \times 100\% = 14.86\%$$

$$OE\% = \frac{6+4}{66+6+24+4} \times 100\% = 10\%$$

As expected the $OE\%$ is not changed, but the $FP\%$ is decreased with more than 3% even though just one sample has changed. This mean that error measurements are changed in steps. The step size is strictly depended on the total size of data set and the proportion of the classes.

(end of example)

To minimize the problems with the confidence interval, the *cross-validation* is introduced later on. As an alternative to the classification errors above, the root-mean-square error(RMSE) is introduced.

$$RMSE = \sqrt{\sum_{i=1}^N \frac{(y_i - T_i)^2}{N}} \quad (1.4)$$

RMSE is the mean distance between the classifier model output y_i and the actual class T_i for N samples. If the model output are integers indicating the class, then for a two class problem with the classes 1 and 2, the root-mean-square is equal to the square-root of the overall error - $\sqrt{OE\%/100\%} = RMSE$.

Example 1.2

Consider a simple classification problem with only 2 different classes - class 1 and class 2. Given a classifier with estimated model output $\mathbf{y} = [1 \ 1 \ 2 \ 2 \ 1]$ and true classes $\mathbf{T} = [1 \ 1 \ 1 \ 2 \ 2]$ yielding 2 errors on 5 samples. The overall error(OE%) is simply calculated from equation 1.3.

$$OE\% = \frac{1+1}{5} \times 100\% = 40\%$$

With all model outputs y as integers 1 or 2, the difference $y_i - T_i$ in equation 1.4 are either 1 - if the classification is wrong and 0 - if it is true. Since the numbers 1 and 0 not are affected by squaring the RMSE is:

$$RMSE = \sqrt{\frac{1}{5} (0 + 0 + 1 + 0 + 1)} = \sqrt{\frac{2}{5}} = \sqrt{0.40}$$

Comparing the 2 error measurements shows

$$\sqrt{OE\%/100\%} = \sqrt{40\%/100\%} = \sqrt{0.40} = RMSE$$

(end of example)

The classification errors FN%, FP% and OE% using the definitions above, only makes sense for a two class problem. The error of a M class problem, can be described using an $M \times M$ confusion matrix. The element x_{ij} in row i and column j describes the number of samples, of true class j classified as class i . This means that all truly classified samples are placed in the diagonal and the remaining misclassified samples in the upper- and lower triangular parts. The confusion matrix describes the number of errors, but an error rate is obtained by scaling each column to a norm of 100%. Figure 1.3 shows an example for a 7 class problem with 917 samples identical with pap-smear database. Here the matrix is set up as a table.

		true class						
		1	2	3	4	5	6	7
estimated class	1	58,5	6,5	0	0	0	0	0
	2	15,5	50,2	1,6	0,9	0	0	0
	3	0	13,3	11,7	18,8	2,0	0,7	0
	4	0	0	45,7	78,9	32,2	11,8	3,0
	5	0	0	35,4	75,9	69,7	87,0	19,9
	6	0	0	3,6	7,5	37,7	78,8	81,7
	7	0	0	0	0	4,4	18,7	45,4
	sum	74	70	98	100	146	197	150

		true class						
		1	2	3	4	5	6	7
estimated class	1	79,0	9,2	0	0	0	0	0
	2	21,0	71,7	1,7	0,5	0	0	0
	3	0	19,1	11,9	10,3	1,4	0,3	0
	4	0	0	46,7	43,3	22,0	6,0	2,0
	5	0	0	36,2	41,7	47,7	44,2	13,2
	6	0	0	3,5	4,2	25,9	40,0	54,5
	7	0	0	0	0	3,0	9,5	30,3
	sum	100	100	100	100	100	100	100

Figure 1.3: Confusion matrix

The pap-smear data is actual distributed on 7 classes, but the minimum requirements for the classification is a 2 class problem, distinguishing between normal and abnormal cells. The 2-class errors FN%, FP% and OE% are easily identified from a 7-class problem. As described in chapter 2 class 1,2 and 3 are normal cells and class 4,5,6 and 7 are abnormal. Figure 1.4 shows the relation between a 2-class problem and a 7-class problem.

		true class						
		1	2	3	4	5	6	7
estimated class	1	TN			FN			
	2							
	3							
	4	FP			TP			
	5							
	6							
	7							

Figure 1.4: Confusion matrices

Building and testing of classifiers, requires a division of the given data set into separate sets of training and test data, whether transductive or inductive methods are used. But the optimal ratio between number of training and test examples, respectively, depends on the method.

An inductive model is build from a set of training data with known target values. And then some, so far unseen, test data is evaluated on the model. The performance of a model is divided into train errors and test errors to use for different purposes. The train error is mostly used for training the model, but the test error is the finally performance measurements for new samples applied to the model. Since the model is build to fit the training data only, the

training error is expected to be lower than the test error. To obtain a good estimate of the test error and avoid huge discrete steps as described, a large test set is required. But on the other hand, a large set of training data is also desirable to build a good fitting model. To make the best possible use of the available data set, *cross-validation* is introduced below.

1.6.2 k-fold cross-validation

To make the best possible use of the available data set, *cross-validation* is introduced. The method is described by Stone(1974,1978) and Wabba&Wold in Bishop(1995). Here the whole data set of N samples random, is divided into k segments. Then temporarily hide 1 segment and using the remaining $k - 1$ segments as training data to build the model. The performance is then measured, applying the hidden test segment to the model. This process can be repeated k possible times - because there exist k different possibilities to exclude one of the k segments. The total test error is an average of k tested models.

Using *cross-validation* makes it possible to use a large proportion of the data available to train the model without decreasing the amount of test data validated. The fraction $\frac{k-1}{k}$ describes proportion of training data and total data. When k is large, then the amount of training data is large. The lower bound of k is $k = 2$ - meaning equal size of training and test data. The upper bound is $k = N$ - the total number of samples. This limit is known as *leave-one-out* method (Bishop,1995), and is described below for classification on transductive models.

One thing to be aware of using the k-fold cross-validation, is the computational load because k models have to be build each time.

1.6.3 Cross-validation rerunning

Using the *leave-one-out* method, the partition in k segments are unambiguous, but for all other values of k , the partition is random. And when randomly selected test and training data are used to model building and testing, the error estimates may changes. Therefore a more reliable error estimates could be achieved, if rerunning the k-fold cross-validation process R times and find the mean error of these R repetitions.

Chapter 2

PAPsmearData

This chapter is thought as an independent guide, describing the latest pap-smear data set collected on Herlev University Hospital. Published along with the data on WWW, it should make a benchmark database free to use. The guide describes in detail the 20 different features and 7 different classes that forms the database. A description of purpose and origin of the data is also given.

The pap-smear database consists of 917 samples distributed unequally on 7 different classes. Each sample is described by 20 features extracted from pictures of single human cells. The data class is a number describing cell type. The pap-smear data set is extracted from sample tissues, taken from the uterine cervix as a part of the smear screening. The purpose of smear screening is to diagnose premalignant cell changes before they progress to cancer.

This guide gives a short medical description introducing the Papanicolaou method and the smear screening. The 7 classes and the 20 features are individual described along with the feature extraction process.

2.1 The pap-smear screening

The Papanicolaou Smear method is a medical procedure to detect pre-cancerous cells in the uterine cervix. The medical descriptions below are based on Martin(2003) and (Dr.Indman, 2005).

Using a small brush, cotton-stick or wooden stick, a cytological sample is taken from the cervix and smeared onto a thin glass slide. To clarify the cells characteristics the smear is stained using the Papanicolaou method, so the different components of the cells are emphasized with specific colors - this makes it more clear in a microscope.

In general a stained cell picture contains a nucleus surrounded by cytoplasm on a background as showed in figure 2.1.



Figure 2.1: Image of single pap-smear image

By inspection of the cell characteristics like size, color, shape and texture of nucleus and cytoplasm, cyto-technicians are capable of diagnosing the cells. The job is very demanding and requires well skilled cyto-technicians. Each microscope slide contains up to 300.000 single cells with different orientation and overlap.

In the cervix different kinds of cells exist. They are located in separate areas: (a) *Squamous area* and (b) *Columnar area*.

(a) *The squamous areas* are located in the bottom at the canal of cervix and just outside in the vagina as showed in figure 2.2. The cells here lining the cervix are divided into 4 layers: the basal, the parabasal, the intermediate and the superficial layer. The youngest cells in the basal layer, lie on the basal membrane. When the cells mature they move trough the layers, and finally they get ejected from the surface in the superficial layer. Moving through the layers the cells change shape, color and other characteristics. Cells in the basal layer are small and round, with a large nucleus and a little cytoplasm. Moving through the layers the cytoplasm becomes bigger and the nucleus smaller. The general shape turns more oval, why cells in the superficial layer area are refered to as *flat squamous cells*. The cell shapes and the four layers are showed in figure 2.3.

(b) *The columnar area* is located in the upper part - and specially in the canal of the cervix as illustrated in figure 2.2. The columnar cells exist only in a single layer - the basal layer. Characteristics for these cells are a column-like shape with an oblong cytoplasm and a large nucleus located at one end. Somewhere in between these two areas, the cells meet in the *squamo-columnar junction*. This junction may be located either inside or outside the cervix. The junction is also named the transformation zone, because the tall columnar cells are constantly being transformed into flat squamous cells.

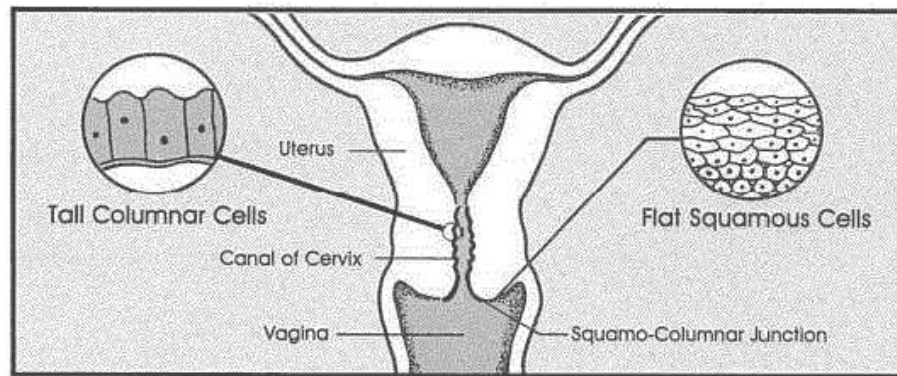


Figure 2.2: The uterus in details and the location of a) *columnar* cells and b) *squamous* cells. (Adopted from Dr.Indman, 2005)

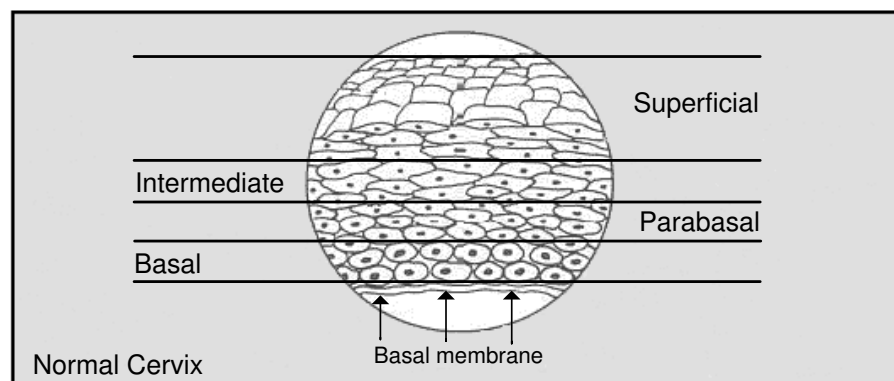


Figure 2.3: Development of the *squamous* cells through the four layers

When the genetic information in a cell somehow is changed, the cell will not divide as it should, it turns into a pre-cancerous cell. In medical terms these are divided into 2 different main diagnoses:

1. *dysplasia*. The term "plasia" means growth, and dysplasia means disordered growth. The cervical dysplasia are normally divided into 3 types: mild, moderate and severe, describing the risk, that the cells turn into malignant cancer cells. *Mild* means of course lowest risk. The characteristics of cells in dysplasia depends on the kind. In the mild dysplasia they have enlarged and light nucleus. For the moderate dysplasia the nucleus is larger and darker. The nucleus has begun to deteriorate, which is seen as a granulation of the nucleus. In severe dysplasia the nucleus is large, dark and often deformed. The cytoplasm is dark

and small, when compared to the nucleus.

2. *carcinoma-in-situ*. Carcinoma-in-situ means "cancer in place" and is characterized of very large nucleus. In the past, there was a tendency to treat "carcinoma-in-situ" as a much more serious problem than severe dysplasia, when in fact they are essentially the same. (Dr.Indman, 2005)

According to the medical description above, properties like size, area, shape and brightness are good descriptive features. Also the relative size between nucleus and cytoplasm seems descriptive, because it grows for the pre-malignant cells. Defining the Nucleus/Cytoplasm-ratio as:

$$N/C - ratio = \frac{NucleusArea}{NucleusArea + CytoplasmArea} \quad (2.1)$$

The characteristics for each class are showed in figure 2.4

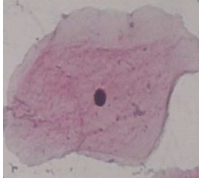


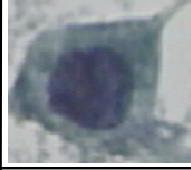
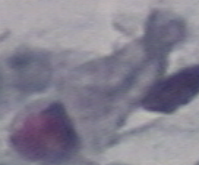

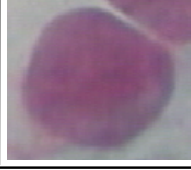
Normal cells		Abnormal cells	
Superficial squamous 1 <ul style="list-style-type: none"> ● Shape: Flat/oval ● Nucleus very small ● N/C very small 		4 Mild dysplasia <ul style="list-style-type: none"> ● Nucleus light/large ● N/C medium 	
Intermediate squamous 2 <ul style="list-style-type: none"> ● Shape: Round ● Nucleus large ● N/C small 		5 Moderate dysplasia <ul style="list-style-type: none"> ● Nucleus large/dark ● Cytoplasm dark ● N/C large 	
Columnar 3 <ul style="list-style-type: none"> ● Shape: Column-like ● Nucleus large ● N/C medium 		6 Severe dysplasia <ul style="list-style-type: none"> ● Nucleus large/dark/deform ● Cytoplasm dark ● N/C very large 	
		7 Carcinoma in situ <ul style="list-style-type: none"> ● Nucleus large/dark/deform ● N/C very large 	

Figure 2.4: Cell type characteristics for single pap-smear cells. (to fit the table, some images are slightly scaled)

2.2 The new pap-smear database

This pap-smear database is the latest one(of two) developed by Herlev University Hospital, the department of Pathology and department of Automation on Technical University of Denmark. The first database was much smaller containing only 500 samples. The set of used features was identical, but the output classes were slightly changed. Classification tasks performed so far, shows more overlap between the classes in the new data set (Martin, 2003). Both data sets are developed for research into automatic classifiers. In this particular data base, the features are extracted by Martin(2003) using Matlab. The single cell pictures analyzed, are prepared by cyto-technicians at Herlev University Hospital using CHAMP¹-software for segmenting the pictures.

The conditions for the development is described below using the terms:

- Data collection
- Data preparation
- Feature extraction

These three terms are very general, but as main topics, they properly are suitable for classification problems in general.

Data collection

Given a raw large set of glass slides, a database of single pap-smear cells pictures is collected at Herlev University Hospital. Skilled cyto-technicians using a microscope with a resolution of $0.201\mu m/pixel$ to grab digital images of the single cells. Each cell picture is afterwards manually classified into the 7 different types of cells described in figure 2.4. For validation the classification is done twice by different cyto-technicians . If the validation is negative the picture is discarded. The 7 diagnostics are identical with the ones used for normal manual classification. Table 2.1 shows the distribution of the data set.

¹CHAMP. - Commercial image processing software for medical purpose by dimac-imaging. - <http://www.dimac-imaging.com/sider/products/champ.htm>

Normal - 242 cells

- . 1 Superficial squamous epithelial, 74 cells.
- . 2 Intermediate squamous epithelial, 70 cells
- . 3 Columnar epithelial, 98 cells.

Abnormal - 675 cells

- . 4 Mild squamous non-keratinizing dysplasia, 182 cells.
- . 5 Moderate squamous non-keratinizing dysplasia, 146 cells.
- . 6 Severe squamous non-keratinizing dysplasia, 197 cells.
- . 7 Squamous cell carcinoma in situ intermediate, 150 cells.

Table 2.1: The 7 different pap-smear classes

Data preparation

A single picture of a normal cell is showed on figure 2.5. Both the nucleus and encircled cytoplasm is possible to identify for the human eye, even that the borders are ambiguous and a second cell is intersecting.

To achieve a solid image, all pictures are segmented into the 3 parts: Background, cytoplasm and nucleus. This segmentation is done one for all and handled by cyto-technicians from Herlev University Hospital using CHAMP. CHAMP is a medical image analysis system based on a colored object recognition algorithm. A segmented picture is illustrated on figure 2.6. Comparing the segmented and non-segmented cell, figures 2.5 and 2.6 show that the single cell is isolated and the intersecting with other cells is moved. Also the segmentation process is validated by cyto-technicians, manually looking through all segmented cells to move potential failed pictures.

Feature extraction

Even though a picture database, containing classified images of single segmented cells, is collected, it is not easy to run it through a classifier algorithm. The cell pictures going to be classified are not stereotype instances of the 7 cell types, but instead pictures of quite different size and orientation

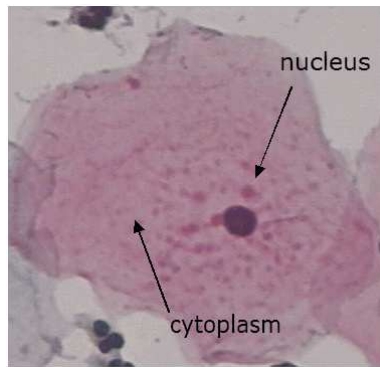


Figure 2.5: Cell picture of a *normal superficial* cell - class 1

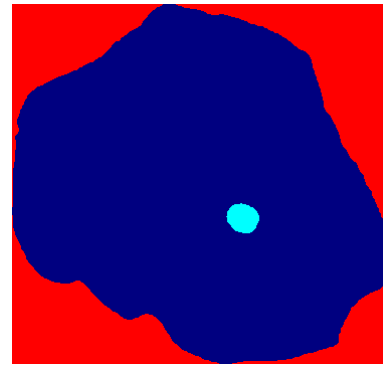


Figure 2.6: A segmented *normal superficial* cell - class 1

As described in section 2.1 and showed in figure 2.4 the 7 cell classes have certain characteristics like size, area, shape and brightness of both nucleus and cytoplasm.

These features - among others - can be extracted from a combination of the segmented and non-segmented cell pictures. Some measurements only uses the segmented picture, and some uses both. A full description of all features is found on page 17. Table 2.2 gives an overview. Some features are selected priority to some expert knowledge. The remaining features is selected, To obtain a complete description of the cell.

1. N area(B)	8. N elongation(I)	15. C perimeter(P)
2. C area(C)	9. N roundness(J)	16. N relative position(Q)
3. N/C ratio(D)	10. C shortest diameter(K)	17. Maxima in N(R)
4. N brightness(E)	11. C longest diameter(L)	18. Minima in N(S)
5. C brightness(F)	12. C elongation(M)	19. Maxima in C(T)
6. N shortest diameter(G)	13. C roundness(N)	20. Minima in C(U)
7. N longest diameter(H)	14. N perimeter(O)	

Table 2.2: Extracted features for pap-smear data. N and C are abbreviation of the Nucleus and Cytoplasm. The letter in brackets refers to the column in the spreadsheet of the new pap-smear database on the enclosed CD-rom. Column (A) in the spreadsheet is used for images reference.

In the following, the 20 features from table 2.2 will be explained.

Nucleus area(1) and Cytoplasm area(2):

Calculated by counting the corresponding pixels of the segmented picture. A pixels area is $(0.201\mu m)^2$.

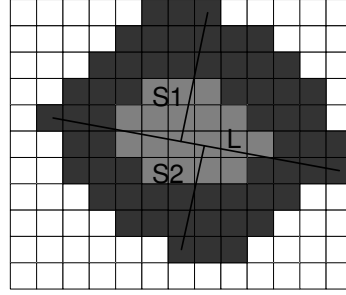


Figure 2.7: A binary cell picture with background(white), cytoplasm(black) and nucleus(gray). For the cytoplasm the longest diameter line(L) and shortest diameter(S1 & S2) are shown. Adopted from (Martin, 2003)

N/C ratio(3): Tells how small the nucleus area is compared to the area of the cytoplasm. It is given by: $N/C = \frac{Nucl_{area}}{Nucl_{area} + Cyto_{area}}$

Nucleus(4) and Cytoplasm(5) brightness:

Nucleus and Cytoplasm brightness is calculated as the average perceived brightness, that is a function of the colors wavelength. In our case it is calculated as $Y = 0.299 \cdot Red_{\mu} + 0.587 \cdot Green_{\mu} + 0.114 \cdot Blue_{\mu}$. Here Red_{μ} , $Green_{\mu}$ and $Blue_{\mu}$ is the average intensity for each of the colors. They are weighted by the perceived brightness of the human eye.

Nucleus(7) and Cytoplasm(11) longest diameter:

This is the shortest diameter a circle can have, when circumscribing the whole object. It is measured as the biggest distance between to pixels on the objects border, and forms a line L as shown in figure 2.7 for the cytoplasm. The names is N_{long} and C_{long} for nucleus and cytoplasm, respectively.

Nucleus(6) and Cytoplasm(10) shortest diameter:

This is the biggest diameter a circle can have, when the circle is totally inscribed in the object. This distance is approximated by the sum of the two lines $S1$ & $S2$ shown in figure 2.7 for the cytoplasm. They are perpendicular to the line L , and is the longest line to each side inside, fitting in the object. The names of the features are N_{short} and C_{short} for nucleus and cytoplasm, respectively.

Nucleus(8) and Cytoplasm(12) elongation:

The elongation is calculated as the ratio between the shortest diameter and the longest diameter of the object.

$$N_{elong} = N_{short} / N_{long}$$

$$C_{elong} = C_{short} / C_{long}$$

Nucleus(9) and Cytoplasm(13) roundness:

The roundness is calculated as the ratio between the actual area and the area inside the circle, given by the longest diameter of the object. They are here named $N_{roundness}$ and $C_{roundness}$ respectively:

$$N_{circle} = \frac{\pi}{4} \cdot N_{long}^2 \Rightarrow N_{roundness} = N_{area}/N_{circle}$$

$$C_{circle} = \frac{\pi}{4} \cdot C_{long}^2 \Rightarrow C_{roundness} = C_{area}/C_{circle}$$

Nucleus(14) and Cytoplasm(15) perimeter:

The length of the perimeter around the object.

Nucleus position(16):

This is a measure of how well the nucleus is centered in the cytoplasm. It is calculated by finding the distance between the nucleus center and the center of the cytoplasm:

$$N_{pos} = \frac{2a \sqrt{(x_N - x_C)^2 + (y_N - y_C)^2}}{C_{long}}$$

Here the position is given by the center $(x_N; y_N)$ and $(x_C; y_C)$ for the nucleus and cytoplasm, respectively.

Nucleus(17;18) and Cytoplasm(19;20) Maxima/Minima:

This is a count of how many pixels is a maximum/minimum value inside of a 3 pixel radius.

2.3 Building a classifier

The idea of this benchmark data is to test and compare different classifiers. As announced, the pap-smear data consists 917 samples distributed on 7 classes - and each sample is described by 20 features. This is a very large data set compared to other benchmark data set, eg. the *iris* data¹. Seen from a medical point of view, it would be desirable to distinguish between diagnoses of mild, moderate and severe dysplasia and also carmino in-situ - class 4, 5, 6 and 7, which are all the abnormal cells. Classifiers tested on the data so far, show that strictly separation between the 7 classes is very difficult (Martin, 2003). But to simplify the classification task, the pap-smear data can be considered as 2 class problem, distinguishing between normal and abnormal classes. But even though that final classification into 2 classes is performed, knowledge about the seven classes can be used for supervise the

¹A data set of 150 samples described by 4 features distributed on 3 classes - see page 57 for further details.

model training

Table 2.4 on page 22 lists the raw mean value of each class. No scaling is performed. The number in brackets is the standard deviation, indicating the dispersion of each feature. Looking at eg. the N/C-ratio(feature 3) shows that the linguistic descriptions almost match the class mean values as showed in table 2.3. (Descriptions like: very small, small medium and so on - from the observations in figure 2.4). The example of the N/C-ratio can also be used to illustrate that prior knowledge of the 7 classes can be valuable for the 2 class problem.

		class						
		1	2	3	4	5	6	7
3	N/C ratio	0.01	0.03	0.35	0.27	0.38	0.49	0.60
		very small	small	medium	medium	large	very large	very large

Table 2.3: N/C-ratio class mean values compared to medical observations from figure 2.4

Even though the distribution is unknown, the standard deviation gives an indication of the class overlap - or missing separation. If the standard deviation is large, compared to the differences between the class means, the separation is poor.

Of course, it is very simplified just looking at one feature at a time, when 20 different features are available. A combination of several features properly improves the separation. But as showed by Martin(2003) and Byriel(1999) the optimal features to use depends on the classifier, and performing feature selection, improves the classification result.

On figure 2.8 Nucleus Area(feature 1) and NC-ratio (feature 3) are plotted against each other, using class mean value and standard deviation from table 2.4. Especially the normal class 1 and 2, seem to be possible to separate, but the last normal class 3 intersects with the remaining abnormal classes.

This mean value analysis is very simplified, but gives an indication of the cluster centers and class properties.

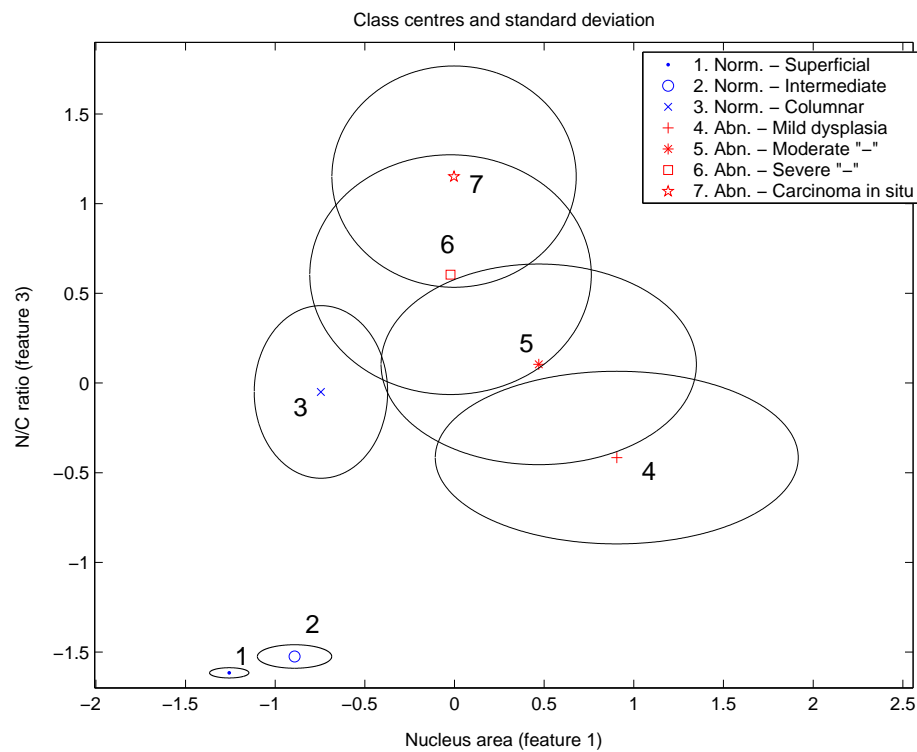


Figure 2.8: Class mean and standard deviation of pap-smear data - Nucleus Area(feature 1) and NC-ratio (feature 3)

		Normal cells			Abnormal cells			
		1	2	3	4	5	6	7
1	N area	631 (206)	1315 (390)	1591 (699)	4690 (1901)	3873 (1651)	2949 (1474)	2986 (1280)
2	C area	61487 (23780)	44961 (15345)	3290 (1829)	15459 (10539)	7288 (5207)	3415 (2276)	2115 (1490)
3	N/C ratio	0.01 (0.01)	0.03 (0.01)	0.35 (0.10)	0.27 (0.10)	0.38 (0.12)	0.49 (0.14)	0.60 (0.13)
4	N bright	66 (17)	67 (19)	94 (25)	98 (17)	92 (15)	94 (22)	97 (18)
5	C bright	134 (23)	131 (22)	138 (36)	142 (19)	135 (18)	143 (29)	142 (22)
6	N short diam	25 (5)	37 (7)	39 (10)	70 (16)	62 (15)	52 (14)	52 (14)
7	N long diam	31 (6)	47 (7)	55 (13)	89 (19)	82 (17)	74 (18)	77 (15)
8	N elongation.	0.82 (0.15)	0.79 (0.15)	0.73 (0.19)	0.79 (0.14)	0.76 (0.13)	0.71 (0.15)	0.69 (0.15)
9	N roundness	0.82 (0.15)	0.77 (0.15)	0.68 (0.17)	0.74 (0.13)	0.72 (0.13)	0.66 (0.14)	0.64 (0.15)
10	C short diam	271 (62)	228 (53)	63 (18)	139 (40)	106 (32)	76 (20)	70 (18)
11	C long diam	338 (68)	303 (53)	117 (36)	212 (73)	154 (46)	119 (36)	102 (25)
12	C elongation	0.81 (0.15)	0.76 (0.14)	0.57 (0.19)	0.69 (0.17)	0.71 (0.17)	0.66 (0.17)	0.71 (0.17)
13	C roundness.	0.67 (0.13)	0.62 (0.12)	0.31 (0.12)	0.42 (0.12)	0.37 (0.12)	0.30 (0.12)	0.25 (0.10)
14	N perimeter	88 (15)	130 (19)	153 (35)	257 (55)	231 (49)	208 (52)	215 (48)
15	C perimeter	1034 (221)	894 (166)	323 (103)	589 (203)	443 (141)	323 (95)	288 (67)
16	N rel. pos	0.17 (0.10)	0.17 (0.09)	0.37 (0.22)	0.25 (0.13)	0.30 (0.17)	0.30 (0.17)	0.28 (0.15)
17	Max N	39 (10)	62 (14)	69 (24)	152 (61)	133 (59)	111 (49)	108 (39)
18	Min N	27 (8)	45 (11)	52 (21)	124 (63)	109 (57)	89 (48)	87 (37)
19	Max C	1881 (808)	1339 (556)	99 (51)	442 (340)	197 (139)	100 (63)	71 (43)
20	Min C	1883 (815)	1258 (537)	101 (50)	464 (343)	213 (141)	116 (65)	85 (46)

Table 2.4: Pap-smear class mean values and standard deviation. Mean values with normal font and standard deviation in italic and brackets.

Chapter 3

Inductive classifiers

This chapter introduces the Least Squared Method - an simple inductive classifier. Using a simple linear network, a set of optimal weights is found. The method is very simple and easy to implement.

The method is tested on the pap-smear data for an optimal classification result, but error distribution is also investigated.

3.1 Least Square Method (LS)

3.1.1 Least Square Method Theory

The most simple model to design, without discard some information, is a network with only one node - the output node. All input coordinates are individually weighted and accumulated in the output node. Consider the network shown in figure 3.1. $\mathbf{x}_i = [x_1, x_2, \dots, x_p]'$ is one data sample with p features presented to model. $\mathbf{w} = [w_1, w_2, \dots, w_p, w_0]$ is the weights and y is the model output representing the class

$$y = w_1 \cdot x_1 + w_2 \cdot x_2 + \dots + w_p \cdot x_p + w_0 = \sum_{j=1}^P x_j w_j + w_0 \quad (3.1)$$

Augmenting the x -vector yield y as vector product

$$y = \begin{bmatrix} x_1 & x_2 & \dots & x_p & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_p \\ w_0 \end{bmatrix} = [\mathbf{x}'_i \ 1] \mathbf{w} \quad (3.2)$$

Given a dataset with N samples equation 3.2 yield $\mathbf{y} = [y_1 \ y_2 \ \dots \ y_N]$ describing the model output for N samples in very simple way, where $\mathbf{F}\mathbf{w}$ is the matrixproduct.

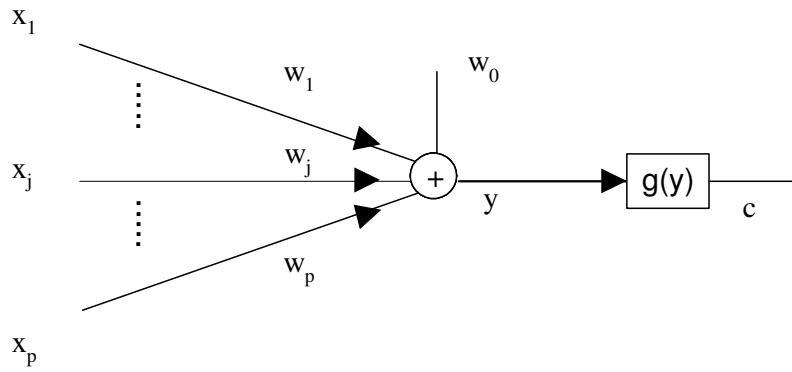


Figure 3.1: Linear network

$$\mathbf{y} = \mathbf{F}\mathbf{w}, \quad \mathbf{F} = \begin{bmatrix} \mathbf{x}'_1 & 1 \\ \mathbf{x}'_2 & 1 \\ \vdots & \vdots \\ \mathbf{x}'_N & 1 \end{bmatrix} \quad (3.3)$$

But how to find the optimal weights?. Using a set of training data with N samples with known classes $\mathbf{C} = [C_1 \ C_2 \ \dots \ C_N]$ assuming the class for a given sample C_i is a scalar that represents the class, e.g. $C_i \in \{1, 2, 3, \dots, 7\}$ for the pap-smear data, equation 3.3 is equal to

$$\mathbf{F}_{N \times P+1} \cdot \mathbf{w}_{P+1 \times 1} = \mathbf{C}_{N \times 1} \quad (3.4)$$

Using basic matrix calculation equation 3.4 have a unique solution \mathbf{w} , if an inverse \mathbf{F}^{-1} exist

$$\mathbf{w} = \mathbf{F}^{-1} \mathbf{C} \quad (3.5)$$

In general terms a matrix \mathbf{A} only has an inverse \mathbf{A}^{-1} if \mathbf{A} is square and non-singular (Jens Eising, 1993).

As described in equation 3.4 \mathbf{F} is of the size $N \times P+1$ and then not squared for most real sets of training data. For most typical sets of training data $N > (P + 1)$ given a over-determined system with full rank, and no exact solution. Working as an engineer, and not as mathematician, an expression like equation 3.4 is possible to solve using Matlab. Using the *backslash*-operator(\backslash) performing a matrix division from left¹ then $\mathbf{w} = \mathbf{F} \backslash \mathbf{C}$ finds a solution \mathbf{w} that minimize the difference between the model output $\mathbf{y} = \mathbf{F}\mathbf{w}$ and the actual class \mathbf{C} using an error function $E(\mathbf{w})$ in 3.6.

¹ $\mathbf{w} = \text{mldivide}(\mathbf{F}, \mathbf{C})$ performs the same calculation as $\mathbf{w} = \mathbf{F} \backslash \mathbf{C}$. Full documentation on <http://www.mathworks.com/access/helpdesk/help/techdoc/ref/mldivide.html>

$$E(\mathbf{w}) = \sum_{i=1}^N (y_i - C_i)^2 = (\mathbf{y} - \mathbf{C})' (\mathbf{y} - \mathbf{C}) \quad (3.6)$$

Since the objective is minimizing the sum of the squared model deviation from each sample \mathbf{w} is said to be the least-square solution to $\mathbf{F}\mathbf{w} = \mathbf{C}$.

The described least square solution can also be achieved using the pseudo-inverse¹ \mathbf{F}^* of \mathbf{F} as described in Bishop(1995).

$$\mathbf{w} = \mathbf{F}^* \mathbf{C} \quad (3.7)$$

But how optimal are these weights actually in the least square sense. The RMSE introduced in equation 1.4 is strictly related to the objective function $E(\mathbf{w})$ from equation 3.6

$$RMSE = \sqrt{\frac{E(W)}{N}} = \sqrt{\frac{(\mathbf{y} - \mathbf{C})' (\mathbf{y} - \mathbf{C})}{N}} \quad (3.8)$$

Which is the mean distance between model output \mathbf{y} and the actual class \mathbf{C} over N samples. If the error $RMSE$ is non-zero, meaning that $y_i - c_i \neq 0$ for $i = 1, 2, \dots, N$, then the solution \mathbf{w} is not exact.

To perform a classification from the model output \mathbf{y} an activation function $g(y)$ is introduced. A hard limiter performing rounding of y . For 2 classes $C = [1, 2]$ the activation function is showed in figure 3.2. The value separating the classes is named the threshold value - here $thr = 1.5$.

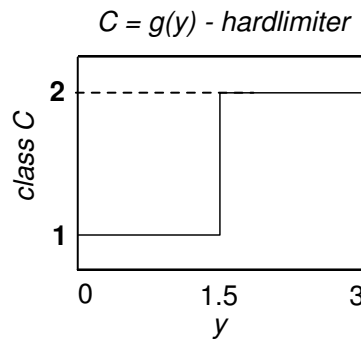


Figure 3.2: $g(y)$ - hard limiter activation function - $thr = 1.5$

As described earlier, this model is very simple and also very fast, eg. using Matlab. The matlab script in figure 3.3 shows how simple a linear classification can be carried out, if a data set available. To simplify the program it is assumed that the data set is already

¹The pseudo-inverse \mathbf{F}^* is a matrix of same size as \mathbf{F} where $\mathbf{F}\mathbf{F}^*\mathbf{F} = \mathbf{F}$, \mathbf{F}^* only exist if \mathbf{F} has Singular-Value decomposition (Eising, 1993)

separated in training- and testdata. The function `mldivide` is a standard Matlab function and not from a specialized toolbox.

The matrices `trainFeatures` and `testFeatures` having samples along the *rows* and features along the *columns*. The classes is given by the *column*-vectors `trainClass` and `testClass`. Using least-square techniques as `mldivide` no pre-scaling is required. Different means and deviations of the features has showed to be phased out by the weights.

Having 2 sets of some training data $\mathbf{X1}(\mu_1, \sigma_1)$ and $\mathbf{X2}(\mu_2, \sigma_2)$ with different mean μ_1, μ_2 and deviation σ_1, σ_2 . Given a sets of optimal weights $\mathbf{w1}$ fitting the $\mathbf{X1}$ – *dataset* in a least square, the optimal weights $\mathbf{w2}$ fitting the $\mathbf{X2}$ – *dataset* is related $\mathbf{w1}$ in this way

Having P features the vector $\mathbf{w2} = [w2_1, w2_2, \dots, w2_P, w2_0]'$ is given as

$$w2_i = \frac{\sigma_{1i}}{\sigma_{2i}} w1_i \quad \text{for } i = 1, 2, \dots, P \quad (3.9)$$

$$w2_{P+1} = \sum_{i=1}^P (\mu_{1i} - \mu_{2i}) \frac{\sigma_{1i}}{\sigma_{2i}} w1_i + w1_{1+P} \quad (3.10)$$

```

Ntrain = size(TrainFeatures,1) %No. of trainingdata
Ntest  = size(TestFeatures,1)  %No. of testdata
Ftrain = [TrainFeatures ones(Ntrain,1)]; % size Ntrain x features+1
Ftest  = [TestFeatures  ones(Ntest,1)]; % size Ntest  x features+1
Ctrain = TrainClass;      % C in [1,2] - size Ntrain x 1
Ctest  = TestClass;       % C in [1,2] - size Ntest  x 1

w = mldivide(Ftrain,Ctrain) % calc. the weights
yTrain = Ftrain*w;         % calc. model output training data
yTest  = Ftest*w;         % calc. model output test data

% performing classification
%if output y <= 1.5, then class 1, otherwise class 2
CyTrain = (yTrain >= 1.5) + 1 ;
CyTest  = (yTest  >= 1.5) + 1 ;

EtrainRMS = sqrt( sum( (yTrain-Ctrain).^2 ) / Ntrain );
EtestRMS  = sqrt( sum( (yTest -Ctest ).^2 ) / Ntest );

```

Figure 3.3: Simple Matlab script for linear classification.

Binary class vector

In the least square modeling above the sample class C_i was described by a single scalar to keep the implementation and solution simple. In this section the scalar class description is extended to a binary vector \mathbf{CM}_i . The length of the vector is equal to the number of

K different classes. For a given sample with scalar class C_i the elements in CM_{il} for $l = 1, 2, \dots, M$ is

$$CM_{il} = \begin{cases} 0 & \text{if } l \neq C_i \\ 1 & \text{if } l = C_i \end{cases} \quad (3.11)$$

A short example. Having a simple $M = 2$ class problem and a sample with class $C_i = 2$ the binary class vector turns into 2-element vector $\mathbf{CM}_i = [0 \ 1]$ and for a 7 class problem \mathbf{CM}_i is extended to $[0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0]$. Consider a whole data set N samples the right hand side of equation 3.4 turns into a matrix $CM_{N \times M}$. And since F isn't changed, the weight matrix \mathbf{w} is correspondingly changed to $\mathbf{w}_{P+1 \times K}$. This gives M weights for each feature.

$$\mathbf{F}_{N \times P+1} \cdot \mathbf{w}_{P+1 \times K} = \mathbf{C}_{N \times K} \quad (3.12)$$

3.2 Results

Using these simply least square methods, the results are presented below. As described above, the classification is straight forward, except for selecting the hard limiter threshold to separate the classes. This tuning parameter is of course investigated.

No feature selecting is done - meaning that all features are used.

For validation all results are based on k-fold crossvalidation running 50 repetitions. Using $k=10$ gives a large set of training data based on 90% of whole dataset. Since none of the models are based on some random initial guess, the large error deviation is only related to the random process in selecting test- and trainingdata.

Two types of classification are performed using 2 classes and 7 classes respectively. Using two classes only, the pap-smear data is divided into normal and abnormal cells. The purpose of 7 classes is twofold. For one thing, a classification result, where the diagnose of mild, moderate and severe dysplasia is known, is more suitable for medical applications. For another, using a least square classifier with 7 binary classes instead of 2 increases the number of decision-boundaries as showed in section 3.1.1. And subsequent the results can be considered as 2 class problem.

3.2.1 Test conditions

This section researches the test conditions for the optimal value of k to divide into training and- test data as described in 1.6.2. The number of necessary crossvalidation is also investigated.

Using the simple Least mean Square classifier distinguishing between 2 classes of normal and abnormal cells, figure 3.4 shows the test- and training-RMSE for different values of k . The number cross-validation repetitions is 100.

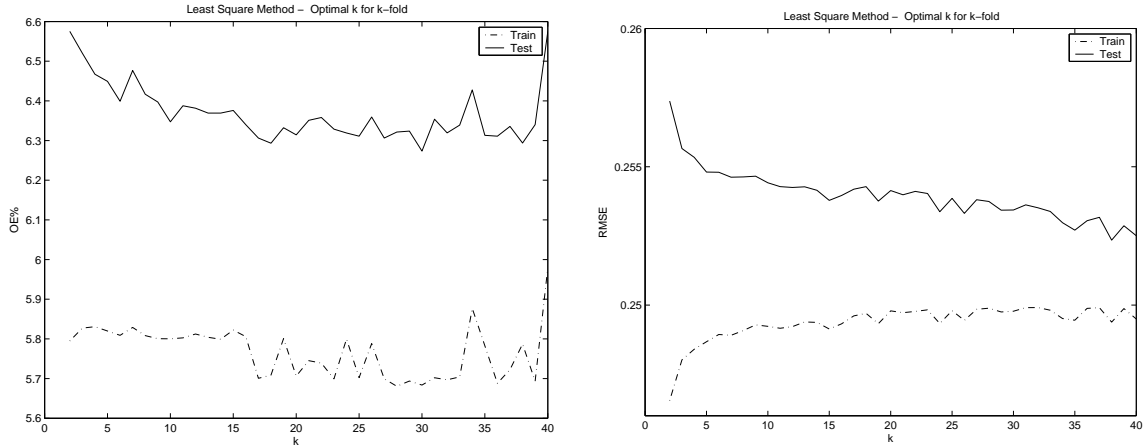


Figure 3.4: Least Square Method(LS) - Optimal k for k-fold

The test error is clearly decreased with k , but the improvement from $k = 10$ to $k = 40$ is minor. The size of the test set N_{test} is naturally decreased with k . If $k = 40$ then $N_{test} = N \frac{1}{K} = 917 \cdot \frac{1}{40} \approx 23$. Even though that the RMSE is decreased the overall error is more or less independent of k . For further experiments $k = 10$ is used.

As described in section 1.6.3 the crossvalidation ensures, that a steady error is achieved. Using approximately 50 repetitions gives a reliable error as showed in figure 3.5 and will be used further on in this project. But using a smaller number of repetitions is also decent, the overall error changes only 0.2 percentage point from $k = 10$ to $k = 70$.

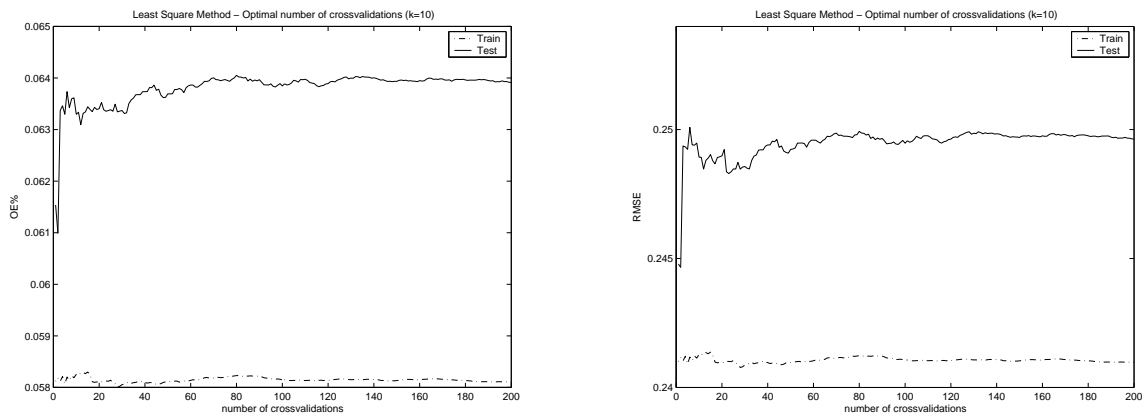


Figure 3.5: Least Square Method(LS) - Optimal number of reruns ($k = 10$)

3.2.2 Least Square Method(LS)

2 class problem - scalar class

Performing classification into two classes of normal and abnormal cells, using a scalar class, shows quite optimistic results - table 3.1 shows the test error. The threshold value thr for separating the class is initial 1.5.

	mean	std	min	max
FN%	1.23%	1.34%	0.00%	7.25%
FP%	20.66%	6.26%	4.00%	36.00%
Overall error	6.40%	1.93%	1.06%	11.70%
RMSE	0.25	0.02	0.21	0.30

Table 3.1: Test error Least Square method. 2 class problem - threshold=1.5 k=10 rerun=50

The most interesting performance parameter, is the *mean* value showing an overall error on 6.4%. A quite big difference between the FP% and FN% is observed. Remembering the ratio of about 1 normal to 3 abnormal cells in the entire database, this maybe seems reasonable. A small TN squeeze down the FP% and a large TP squeeze up the FN%. But this is not the only reason. Looking at the actual number of test errors in table 3.2, shows that most errors are FP-errors - normal classified as abnormal. Indicating that the model fits closest to the most represented class - the abnormal cells. And this definitely makes sense, using a least square algorithm. Fortunately the lowest error rate is the FN%, meaning that the classifier mostly diagnoses normal cells as abnormal.

FN	FP	TN	TP	FN+FP+TN+TP
0.81	5.01	19.19	66.16	91.7
0.89%	5.45%	21.12%	72.54%	100%

Table 3.2: Number of test errors - k=10. With $k = 10$ the number of testdata is $N_{test} = N \frac{1}{k} = 917 \cdot \frac{1}{10} \approx 91.7$

Since the errors described above, are mean values, also the standard deviations *std* are quite interesting. In general the standard deviations expresses the average deviation from the mean value, but if the distribution is unknown, the probability to be in certain error interval, is also unknown.

Performing a linear classification with 10000 repetitions shows that the overall mean error can be approximated to the normal distribution - figure 3.6 and 3.7. The mean is the average of the 2-fold and 10-fold cross-validation, respectively. The RMSE error is also roughly

normal distributed as showed on figure 3.8 and 3.9, but appears to be more "compact" for values below the mean value. Assuming normal distribution ensures that about 70% are in range ± 1 standard deviation from the mean value (Shanfugan&Breipohl, 1988). That gives overall errors in the range [4.4%; 8.4%] in approximate 70% of the classifications.

The classifications above use a threshold value of 1.5 for separating the classes - an excellent initial choice to separate the classes 1 and 2. Figure 3.10 shows the train error of the threshold dependency as a tuning parameter. The Overall error is almost constant with a threshold in the range [1.5; 1.65], but the FP% and FN% changes a lot in this interval. Evaluating the test error using a threshold of 1.65 gives the result below.

threshold(<i>thr</i>)	FN%	FP%	OE %
1.50 %	1.23%	20.66 %	6.40%
1.65 %	6.04%	7.82 %	6.52%

As expected the RMSE is not affected by threshold values as described in section 1.6. So tuning the threshold parameter mostly effect the ratio between FN% and FP%, and not the OE%.

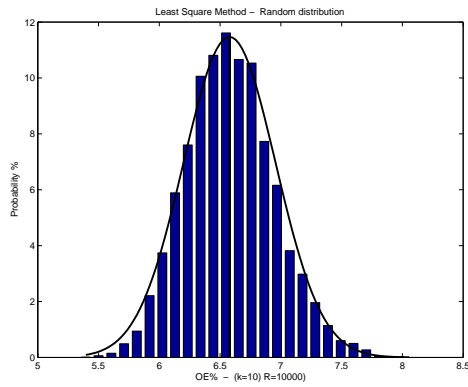


Figure 3.6: Overall error OE%, $k=2$ - Using 10.000 repetitions shows that the error is normal distributed

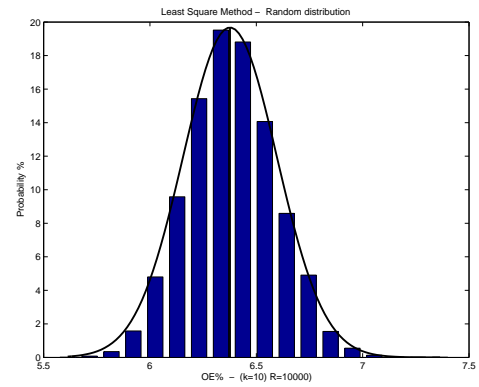


Figure 3.7: Overall error OE%, $k=10$ - Using 10.000 repetitions shows that the error is normal distributed

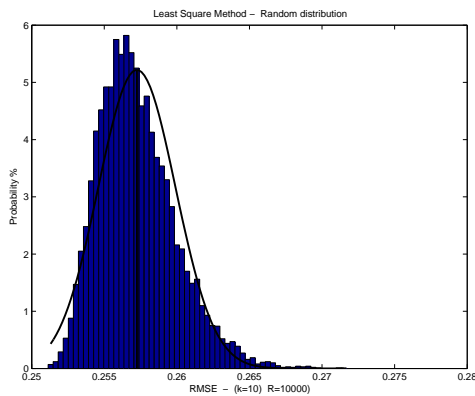


Figure 3.8: RMSE, $k=2$ - Using 10.000 repetitions to show random distribution

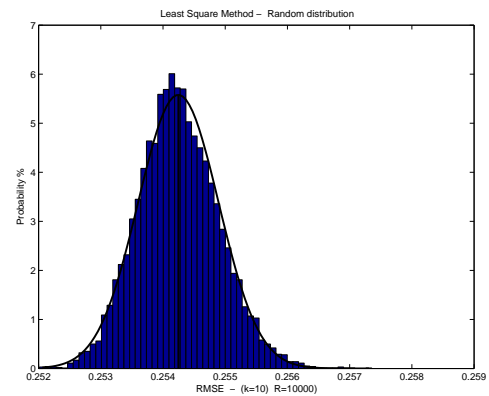


Figure 3.9: RMSE, $k=10$ - Using 10.000 repetitions to show random distribution

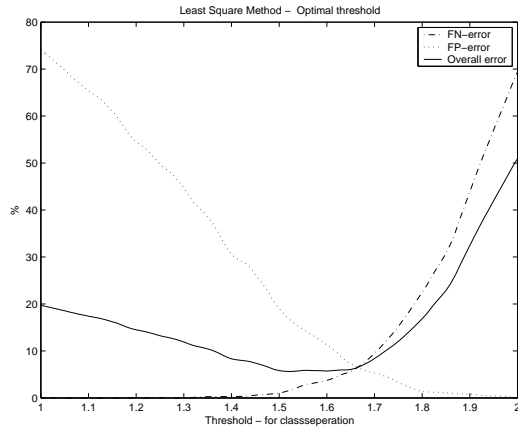


Figure 3.10: Train error - Threshold dependency on FN%, FP%, OE%

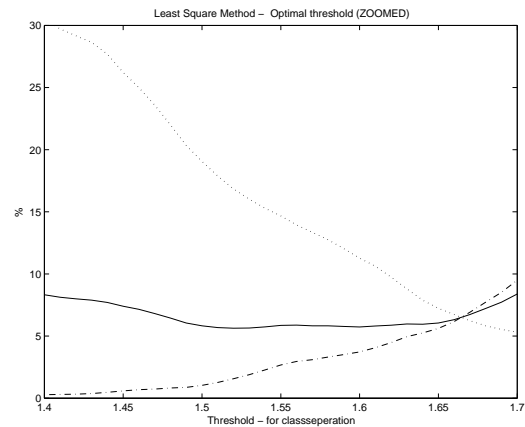


Figure 3.11: Test error - Threshold dependency on FN%, FP%, OE% (ZOOM of figure 3.10)

7 class problem - scalar class

Instead of performing classification into only 2 groups of normal and abnormal cells, the classification task is extended to include all 7 original classes from the papsmear database. Still using a scalar describing the class, gives $C_i \in [1, 2, 3, 4, 5, 6, 7]$ for all $i = 1, 2, \dots, N$. To perform classification into 7 classes the activation function $C = g(y)$ from 3.2 is described by seven regions using six thresholds levels, $\mathbf{thr} = [1.5 \ 2.5 \ 3.5 \ 4.5 \ 5.5 \ 6.5]$ as showed on figure 3.12.

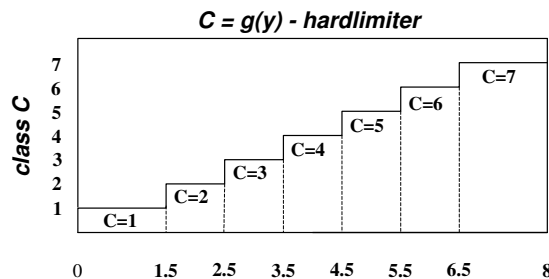


Figure 3.12: $g(y)$ - hard limiter activation function 7 classes - $\mathbf{thr} = [1.5, 2.5, \dots, 6.5]$

Using 10-fold cross-validation with 50 repetitions, the confusion matrices in table 3.3 shows the classification results.

The truly classified samples, are found along the diagonal. The mean test error is calculated to be more than 42% for classification into 7 classes. Very large compared to the one found for the 2 class problem. The percentage errors for each class in also shown in table 3.3. Looking at eg. class 3, the table shows that only approximately 12% of the columnar

		true class						
		1	2	3	4	5	6	7
estimated class	1	58,5	6,5	0	0	0	0	0
	2	15,5	50,2	1,6	0,9	0	0	0
	3	0	13,3	11,7	18,8	2,0	0,7	0
	4	0	0	45,7	78,9	32,2	11,8	3,0
	5	0	0	35,4	75,9	69,7	87,0	19,9
	6	0	0	3,6	7,5	37,7	78,8	81,7
	7	0	0	0	0	4,4	18,7	45,4
	sum	74	70	98	100	146	197	150

		true class						
		1	2	3	4	5	6	7
estimated class	1	79,0	9,2	0	0	0	0	0
	2	21,0	71,7	1,7	0,5	0	0	0
	3	0	19,1	11,9	10,3	1,4	0,3	0
	4	0	0	46,7	43,3	22,0	6,0	2,0
	5	0	0	36,2	41,7	47,7	44,2	13,2
	6	0	0	3,5	4,2	25,9	40,0	54,5
	7	0	0	0	0	3,0	9,5	30,3
	sum	100	100	100	100	100	100	100

Table 3.3: Test error Least Square(LS) into 7 classes using scalar class description - k=10 rerun=50

cells are truly classified. Best classification is achieved for class 1 and 2 - normal superficial and intermediate squamous cells. Even though that classification is performed into 7 classes, the task can still be considered as 2 class problem as described on page 9 adding up the normal(1,2&3) and abnormal(4,5,6&7) classes. Table 3.5 shows the results, including false-negative and false-positive error. The mean error of 11.67% is large, compared to results achieved for the 2 class problem above.

	mean	std	min	max
FN%	N/A			
FP%	N/A			
Overall error	42.86%	4.67%	26.09%	56.04%
RMSE	0.91%	0.06%	0.69%	1.11%

Table 3.4: Test error Least Square(LS) into 7 direct classes using scalar class description - k=10 rerun=50

	mean	std	min	max
FN%	3.31%	2.10%	0.00%	10.45%
FP%	34.99%	4.47%	20.83%	41.67%
Overall error	11.67%	1.85%	6.52%	17.39%
RMSE	0.91	0.06	0.69	1.11

Table 3.5: Test error Least Square method - 2 class problem - using all 7 classes for scalar model building - k=10 rerun=50

		true class						
		1	2	3	4	5	6	7
estimated class	1	56,3	16,8	0	0	0	0	0
	2	15,8	48,3	0	0	0,9	0,8	0
	3	1,9	1,0	73,2	1,0	15,1	18,3	7,4
	4	0	3,9	0,9	16,0	75,7	27,9	7,5
	5	0	0	0	5,6	19,5	8,1	3,5
	6	0	0	18,6	13,2	23,7	85,5	37,0
	7	0	0	5,3	2,2	11,1	56,4	94,6
	sum	74	70	98	182	146	197	150

		true class						
		1	2	3	4	5	6	7
estimated class	1	76,1	23,9	0	0	0	0	0
	2	21,4	69,0	0	0	0,6	0,4	0
	3	2,5	1,4	74,7	0,6	10,3	9,3	4,9
	4	0	5,6	0,9	88,0	51,8	14,1	5,0
	5	0	0	0	3,0	13,4	4,2	2,3
	6	0	0,1	19,0	7,2	16,3	43,4	24,7
	7	0	0	5,4	1,2	7,6	28,6	63,1
	sum	100	100	100	100	100	100	100

Table 3.6: Test error Least Square(LS) into 7 classes using binary class description - k=10 rerun=50

	mean	std	min	max
FN%	6.46%	2.87%	0.00%	16.18%
FP%	11.86%	5.99%	0.00%	37.50%
Overall error	7.88%	2.67%	1.10%	18.48%
RMSE	1.13	0.13	0.76	1.57

Table 3.7: Test error Least Square(LS) into 7 classes using binary class description - k=10 rerun=50

3.3 Conclusion

Looking at the achieved error rates, the result is not magnificent. But what is expected and what is realistic - since the properties of data set, are of great central importance. Martin(2003) have tested different classifiers on exactly the same data. And compared to his results, it is not that bad, even though the error rates are not improved.

Some of Martins's(2003) results, along with the Linear results achieved above, are presented in table 3.8. All results are using 10-fold cross validation, but Martin(2003) used 20 repetitions compared to 50 the above. But otherwise the results are comparable. The unsupervised HCM(Hard C-means) is the most simple of his methods, obtaining a overall error at 8.28%, which is improved using the simple linear least squared method. The best result is achieved using the Supervised GK(Gustafson-Kessel), with an overall error of 6.06%. All results below are achieved with feature selection, but Martin(2003) also performed classification without feature selection with less optimal results.

So generally the Least Square Method performs better than the unsupervised clustering algorithms like HCM, FCM and GK, but the supervised FCM(Fuzzy C-means) and GK are superior. Compared to the found standard deviations a disparity of 0.34% on the classification error are minor. But even though the results is almost identical, the Least Squared Method is characterized by being very simple for execution and implementation.

		OE%	FP%	FN%	Feature Selection	
Erik Martin (2003)	Unsupervised HCM	8.28%	21.81%	3.41%	yes	
	Unsupervised FCM	8.08%	20.97%	3.45%	yes	
	Unsupervised GK	7.68%	22.66%	2.31%	yes	
	Supervised HCM	7.86%	17.12%	4.55%	yes	
	Supervised FCM	6.10%	13.89%	3.29%	yes	
	Supervised GK	6.06%	13.88%	3.26%	yes	
Jonas	2 class LS	6.40%	20.66%	1.23%	no	thr=1.50
Norup	2 class LS	6.52%	7.82 %	6.04%	no	thr=1.65
(2005)	7 direct classes LS	42.86% %	N/A	N/A	no	
	7 class(scalar) LS	11.67%	34.99%	3.31%	no	
	7 class(binary) LS	7.88%	11.86%	6.46%	no	

Table 3.8: pap-smear results - test errors

A considerable difference between the false-negative and false-positive, is also characteristic of the Least Square Method, just as it was observed by Martin(2003) using other methods. But it was showed, that using the threshold value of the activation function it is possible to squeeze the difference without considerably increase the overall error.

An extension from a simple 2 class classification between normal and abnormal cells. to a sophisticated classification distinguishing between all 7 cell types, dramatically increases the classification error to more than 40%. Considering the large intersecting between the classes found in section 2.3, this result is not unexpected.

But using the knowledge of the distribution into 7 classes for the 2 class problem, doesn't improved the result, as it could be expected.

Chapter 4

Transductive classifiers

This chapter describes transductive methods for classification, both the most simple ones and some more complex. The main issue is the NFI algorithm - *Neuro-Fuzzy Inference Method for Transductive Reasoning* (Kasabov&Song, 2005). This method is mostly used for time series prediction, but good results are also achieved for classification problems. Tested on eg. the Iris Data¹ the method gives quite good results (Kasabov& Song, 2005).

4.1 Methods

As described in section 1.3.1 using transductive methods, only a local model is developed from a limited number of training data.

The most simple models calculate a weighted average of the classes from the training samples in the neighborhood. The actual weighting depends on the method.

Also another simple method is introduced: Nearest Class gravity Center(NCC), that calculates a center of gravity of each class among the K nearest neighbors. The method is introduced, because it is not sensitive to pre-dominance of one class.

NFI performs an advanced clustering process, where cluster centers and radii are calculated. When using these cluster properties, a fuzzy inference system is then optimized to fit the training data during Back-Propagation(BP). And finally the test point is presented to the model.

¹An often used benchmark dataset for classification - see section 4.6 for details

4.1.1 KNN and WKNN

The methods described in this section are 2 very simple transductive methods for classification - probably the most simple ones at all.

- KNN - **K** Nearest Neighbor
- WKNN - **W**eighted **K** Nearest Neighbor

Having a set of training data with known classes, KNN simply selects the K nearest samples to a single test point. The most frequent class between these K sample is then assigned to the test point. A simple example is given in example 4.2.

The WKNN method is almost identical to KNN, but the class of the K nearest samples is weighted in the output. The weighting of each sample is inversely proportional with the distance between training and test samples. In this way the training samples most far away from the test point, is weighted as a minimum. The KNN method is widespread in the classification literature - and adding on the weights is an obvious extension (Kasabov&Song, 2005), (Byriel, 1999).

The methods are, as described, simple for implementation and calculation. Both show the basics of transductive methods: Developing a local model for each test point using a limited number of training points.

To select the K nearest points from the training data, the *Normalized Euclidean Distance* d_i is introduced as the distance between the training sample $\mathbf{x}_i = \{x_{i1}, x_{i2}, \dots, x_{iP}\}$ and test point $\mathbf{x}_q = \{x_{q1}, x_{q2}, \dots, x_{qP}\}$. The number of features/coordinates describes each sample is P .

$$d_i = \|\mathbf{x}_q - \mathbf{x}_i\| = \left[\sum_{j=1}^P \frac{|x_{qj} - x_{ij}|^2}{P} \right]^{\frac{1}{2}} \quad (4.1)$$

Compared to the traditional definition of Euclidean distance the definition in equation 4.1 is basically an average of P distances. The definition above is identically to statistic definition of RMS (Shanmugan&Breiphof, 1988).

Assuming all features are scaled between 0 and 1, the normalization factor $\frac{1}{P}$ gives distances in the range $[0; 1]$. Using the WKNN method, the weighting is essential to ensure a gradually influence of each training point in the local model - just like the grade of membership in a fuzzy inference system. In general the weighting is inversely proportional to the distance d_i . The method name "WKNN" and the weight definition below, are adopted from (Kasabov&Song, 2005)

$$w_i = \frac{\max(D) - [d_i - \min(D)]}{\max(D)} \quad \text{for } i = 1, 2, \dots, K \quad (4.2)$$

where $\max(D)$ and $\min(D)$ are the maximum and minimum distance from the testpoint to the K training points. All weights are between 0 and 1, according to the definition below.

$$d_{close} = \min(D) \quad d_{far} = \max(D) \quad \implies \quad w_{close} = 1, \quad w_{far} = \frac{\min(D)}{\max(D)} < 1$$

But also other weight definitions can be applied. Byriel(1999) uses the definition $w_i = 1 - \frac{d_i}{\max(D)}$ - always weighting the sample most far away with zero.

As described the KNN method is the most straight forward because it is independent of the weights. The model output is simply the most frequent class among the K nearest samples. Eg. given $K=5$ and the classes $[C_1, C_2, C_3, C_4, C_5] = [2, 2, 1, 2, 1]$ the most frequent class is 2. Using a mathematical expression the model output y_{q-KNN} is

$$y_{q-KNN} = \frac{\sum_{i=1}^K C_i}{K} \quad (4.3)$$

To perform a classification C_q of the test point \mathbf{x}_q , an activation function $g(y_q)$ is applied to the model output y_{q-KNN} - just like Least Square case in section 3.1.1. The purpose of the activation function, is to perform a rounding of the model output to distinguish between the classes.

The WKNN and KNN algorithm is presented step-by-step below. Comparing the methods shows that KNN is identical with WKNN using unity weights, $w_i = 1$ for $i = 1, 2, \dots, K$.

$$y_{q-KNN} = y_{q-WKNN} = \frac{\sum_{i=1}^K w_i C_i}{\sum_{i=1}^K w_i} \quad w_i = 1 \quad \text{for } i = 1, 2, \dots, K$$

WKNN algorithm step-by-step

Given

- A set of training data x with N samples each of P features and known class C_i

$$x = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1P} \\ x_{21} & x_{22} & \dots & x_{2P} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{NP} \end{bmatrix} \quad C = \begin{bmatrix} C_1 \\ C_1 \\ \vdots \\ C_N \end{bmatrix}$$

- One test point \mathbf{x}_q of P features with unknown class C_q

$$\mathbf{x}_q = [x_1 \ x_2 \ \dots \ x_P] \quad C_q$$

- A proper value of K specifying the number of closest training points. $K \leq N$
 - An activation function $C_q = g(y_q)$ with threshold $thr = 1.5$
1. Calculate all distances $D_{global} = [d_1, d_2, \dots, d_N]$ from test point \mathbf{x}_q to all training points \mathbf{x}_i for $i = 1, 2, \dots, N$

$$d_i = \|\mathbf{x}_q - \mathbf{x}_i\| = \left[\sum_{j=1}^P \frac{|x_{qj} - x_{ij}|^2}{P} \right]^{\frac{1}{2}}$$

2. Sort all distances in D in ascending order. And pick out the K nearest neighbours

$$S = \begin{bmatrix} d_1 & C_1 \\ \vdots & \vdots \\ d_K & C_k \\ \vdots & \vdots \\ d_N & C_N \end{bmatrix}, \quad D = [d_1, d_2, \dots, d_K] \quad d_1 < d_2 < \dots < d_K < \dots < d_N \quad S_K = \begin{bmatrix} d_1 & C_1 \\ \vdots & \vdots \\ d_K & C_k \end{bmatrix}$$

3. Calculate the weights w_i for $i = 1, 2, \dots, K$

$$w_i = \frac{\max(D) - [d_i - \min(D)]}{\max(D)} \quad \min(D) = d_1 \quad \max(D) = d_K$$

4. Calculate the model output y_q for the test point x_q using KNN or WKNN

$$y_{q-KNN} = \frac{\sum_{i=1}^K C_i}{K} \quad y_{q-WKNN} = \frac{\sum_{i=1}^K w_i C_i}{\sum_{i=1}^K w_i}$$

5. Perform classification of x_q finding C_q

$$C_q = g(y_q) = \begin{cases} 1 & \text{if } y_q < 1.5 \\ 2 & \text{if } y_q \geq 1.5 \end{cases}$$

Example 4.2

Considering a simple set of data, this example gives a numeric illustration using both weighted and un-weighted methods.

Having a set of $N=4$ training points x_1, x_2, x_3, x_4 with known class C_1, C_2, C_3, C_4 . Each point is described by $P=2$ features - their Cartesian coordinates. The test point is \mathbf{x}_q with unknown class C_q . Figure 4.1 shows both training and test data.

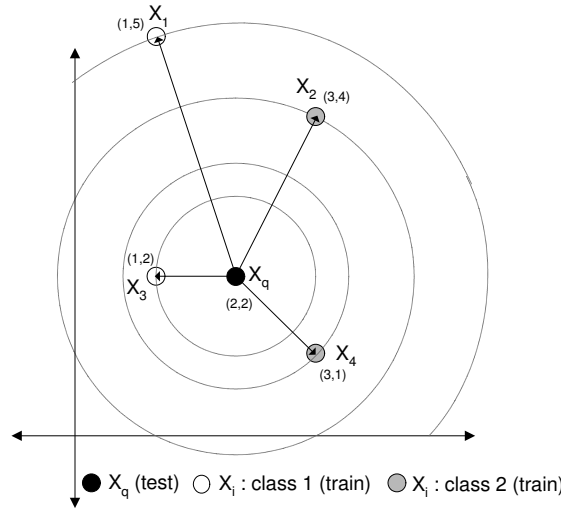


Figure 4.1: Example KNN

Selecting $K=3$. The distances $D = [d_1, d_2, d_3, d_4]$ from testpoint x_q to all training points x_1, x_2, x_3, x_4 is calculated to select the $K=3$ nearest neighbors.

$$\begin{aligned}
 d_1 &= \|x_q - x_1\| = \frac{1}{2} \left[|2-1|^2 + |2-5|^2 \right]^{\frac{1}{2}} = \frac{1}{2} [1+9]^{\frac{1}{2}} = \frac{1}{2} \sqrt{10} = 1.58 \\
 d_2 &= \|x_q - x_2\| = \frac{1}{2} \left[|2-3|^2 + |2-4|^2 \right]^{\frac{1}{2}} = \frac{1}{2} [1+4]^{\frac{1}{2}} = \frac{1}{2} \sqrt{5} = 1.11 \\
 d_3 &= \|x_q - x_3\| = \frac{1}{2} \left[|2-1|^2 + |2-2|^2 \right]^{\frac{1}{2}} = \frac{1}{2} [1+0]^{\frac{1}{2}} = \frac{1}{2} = 0.5 \\
 d_4 &= \|x_q - x_4\| = \frac{1}{2} \left[|2-3|^2 + |2-1|^2 \right]^{\frac{1}{2}} = \frac{1}{2} [1+1]^{\frac{1}{2}} = \frac{1}{2} \sqrt{2} = 0.707
 \end{aligned}$$

The distances are sorted and inserted in S as pair of (d_i, C_i) and the $K=3$ samples closest to x_q is kept for the local model. The rest - here one sample - is discarded.

$$S = \begin{bmatrix} d_3 & C_3 \\ d_4 & C_4 \\ d_2 & C_2 \\ d_1 & C_1 \end{bmatrix} = \begin{bmatrix} 0.5 & 1 \\ 0.71 & 2 \\ 1.11 & 2 \\ 1.58 & 1 \end{bmatrix} \Rightarrow S_3 = \begin{bmatrix} d_3 & C_3 \\ d_4 & C_4 \\ d_2 & C_2 \end{bmatrix} = \begin{bmatrix} 0.5 & 1 \\ 0.71 & 2 \\ 1.11 & 2 \end{bmatrix}$$

It is observed that $[d_1; d_2] > 1$, because the inputs aren't scaled. Using the simple KNN method gives the model output y_{q-KNN} and the classification C_q using $g(y)$

$$y_{q-KNN} = \frac{\sum_{i=1}^K C_i}{K} = \frac{1+2+2}{3} = 1.667$$

$$\Rightarrow C_q = g(1.667) = 2$$

For the WKNN method the weights are calculated. The minimum and maximum distances is given from S_3

$$\max(D) = 1.11 \quad \min(D) = 0.5$$

$$W_{S1} = 1$$

$$W_{S2} = \frac{\max(D) - [d_i - \min(D)]}{\max(D)} = \frac{1.11 - [0.71 - 0.5]}{1.11} = 0.487$$

$$W_{S3} = \frac{\min(D)}{\max(D)} = \frac{0.5}{1.11} = 0.454$$

Using the calculated weights gives

$$y_{q-WKNN} = \frac{\sum_{i=1}^K w_i C_i}{\sum_{i=1}^K w_i} = \frac{1 \cdot 1 + 0.487 \cdot 2 + 0.454 \cdot 2}{1 + 0.487 + 0.454} = 1.485$$

$$\Rightarrow C_q = g(1.485) = 1$$

Comparing the two methods KNN and WKNN, it is interesting to evaluate the different results - table 4.1. The class yield from the weighted method is, in this example, based only on one sample - the one closest to the test point. In this case the WKNN method can be inappropriate. If the closest training point is a single outlier from another class lying very close to the test point, then the $\min(D)$ distance is very small giving small weights according to (4.2). And since w_{close} always is equal to 1, this sample dictates the class. But the selection of a model generally depends on the dataset and the size of K .

KNN	WKNN
$y_q = 1.667$	$y_q = 1.485$
$y_q < 1.5$	$y_q \geq 1.5$
Class 1	Class 2

Table 4.1: Results WKNN and KNN from example 4.2

(end of example)

4.1.2 NCC : Nearest Class gravity Center

The composition of the training data is very important for the performance of the WKNN and KNN methods. Since no clustering is performed on the K picked out training data, a large presence of a certain class, will favour this class in the model output.

Consider figure 4.1 from example 4.2, if 2 samples \mathbf{x}_5 and \mathbf{x}_6 equivalent to \mathbf{x}_2 and \mathbf{x}_4 were

added.

$$y_{q-WKNN} = \frac{w_1 C_1 + 2(w_2 C_2 + w_3 C_3)}{w_1 + 2(w_2 + w_3)} = \frac{1 \cdot 1 + 2(0.487 \cdot 2 + 0.454 \cdot 2)}{1 + 2(0.487 + 0.454)} = 1.653$$

$$\Rightarrow C_q = g(1.653) = 2$$

Seen from a classification point of view, nothing has changed, but the WKNN model output is changed and tipped to class 2.

As described on page 16 the incidence of abnormal classes is 3 times larger than for normal ones in the pap-smear database. This will give preferential treatment to abnormal classes and cause a large FP-error, because many normal cells are misclassified as abnormal.

The NFI method described in section 4.1.3 applies clustering to a local model using ECM.¹ This very advanced clustering method is also introduced in section 4.1.3. The most simple way to cluster the K nearest data, is to group them by their respective classes - one cluster for every single class. And then assigning the cluster center to the mean of the samples that belongs to that class. Using this method the K nearest samples is mapped down to M centers of gravity.

NCC algorithm step-by-step

The premise are equal to the ones used for describing the WKNN method on page 45.

1. Calculate all distances $D_{global} = [d_1, d_2, \dots, d_N]$ from test point \mathbf{x}_q to all training points \mathbf{x}_i for $i = 1, 2, \dots, N$

$$d_i = \|\mathbf{x}_q - \mathbf{x}_i\| = \left[\sum_{j=1}^P \frac{|x_{qj} - x_{ij}|^2}{P} \right]^{\frac{1}{2}}$$

2. Sort all distances in D in ascending order. And pick out the K nearest neighbors. Differently from WKNN S and S_k have to be extended with the training points \mathbf{x}_i to calculate the centers of gravity in step 3.

$$S = \begin{bmatrix} d_1 & C_1 & \mathbf{x}_1 \\ \vdots & \vdots & \vdots \\ d_K & C_K & \mathbf{x}_K \\ \vdots & \vdots & \vdots \\ d_N & C_N & \mathbf{x}_N \end{bmatrix}, \quad d_1 < d_2 < \dots < d_K < \dots < d_N \quad S_K = \begin{bmatrix} d_1 & C_1 & \mathbf{x}_1 \\ \vdots & \vdots & \vdots \\ d_K & C_k & \mathbf{x}_K \end{bmatrix}$$

3. Calculate the class centers $\Phi_l = [\Phi_1 \ \Phi_2 \ \dots \ \Phi_P]$ for possible classes $l = 1, 2, \dots, M$ of the K nearest neighbors. The number of training points with class C_i is K_i , so

¹ECM - Evolving Clustering Method

$$\sum_{l=1}^M K_l = K.$$

$$\Phi_{lj} = \frac{1}{K_l} \sum_{i=1}^{K_l} x_{ij} \quad \text{for } C_i = l \quad i = 1, 2, \dots, K$$

4. Calculate distances $\sigma = [\sigma_1, \sigma_2, \dots, \sigma_M]$ from test point x_q to the M centers of gravity

$$\sigma_l = \|\mathbf{x}_q - \Phi_l\| = \left[\sum_{j=1}^P \frac{|x_{qj} - \Phi_{lj}|^2}{P} \right]^{\frac{1}{2}}$$

5. Sort the distances $\sigma = [\sigma_1, \sigma_2, \dots, \sigma_M]$ in ascending order

$$S_\sigma = \begin{bmatrix} \sigma_1 & C_1 \\ \vdots & \vdots \\ \sigma_M & C_M \end{bmatrix}, \quad \sigma_1 < \sigma_2 < \dots < \sigma_M$$

6. Pick the closest class center in $S(1)$, and assign the output class C_q to the class of this center.

Reconsider the example on figure 4.1. Using the $K = 5$ nearest neighbors x_1 is discarded. The centre of mass of class Φ_1 is equal to x_3 of course, since only one class-1 sample is among the K nearest neighbors. Centre of mass of class 2 is located between x_2, x_5 and x_4, x_6 at $\Phi_2 = (3, 2\frac{1}{2})$ as showed on figure 4.2.

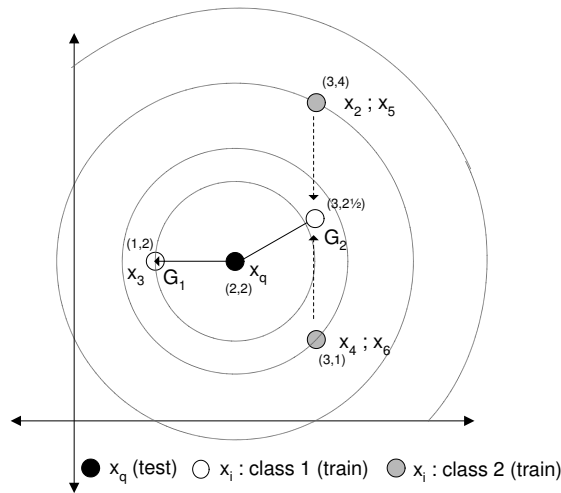


Figure 4.2: Example of NCC: Nearest Class gravity Center

4.1.3 NFI : Neurofuzzy Inference Method for Transductive Reasoning

NFI is a local dynamic neural-fuzzy inference system using a Back-Propagation(BP) learning algorithm for optimizing the parameters. The fuzzy inference engine applied, is either Takagi-Sugeno or Zadeh-Mamdani type - but only the Takagi-Sugeno type is used in this current project. As a transductive method NFI generates local models only, valid for a subspace of the entire problem space, just like WKNN and KNN as described in the previous section. To match the names of the parameters used by (Kasabov&Song) number of nearest neighbors are following designated N_q instead of K . Gaussian membership functions are applied only to the fuzzy rule input (the antecedent part), because the piecewise linear Takagi-Sugeno controller type is used. To partition the local input subspace for creating fuzzy rules, a cluster algorithm is applied. ECM(Evolving Clustering Method) has a relatively small numbers of clusters covering the subspace (Kasabov&Song, 2002). The cluster centers and radii found by ECM, are then used to form the rules of the fuzzy membership functions. Figure 4.3 shows a flowchart of the NFI algorithm.

NFI algorithm step-by-step

The NFI method is described step-by-step below. Since the method is transductive, only one new data point \mathbf{x}_q is presented at a time to build a unique local model M_q . And finally the model output for that point is calculated $y_q = M_q(\mathbf{x}_q)$. Assuming that all data are normalized(every feature is scaled between 0 and 1). Refer to figure 1.2 on page 4 for refreshing on transductive models.

1. Calculate the distances d_i , $i = 1, 2, \dots, N$, between the new data vector \mathbf{x}_q and each of the all N training samples in the entire training set.
2. Pick out the N_q nearest training examples with lowest distance d_i to form the subset \mathbf{D}_q . The value of N_q can be pre-defined based on experience, or optimized through the application of an optimization procedure.
3. Calculate the weights w_i of the N_q nearest samples $w_i = 1 - (d_i - \min(\mathbf{d}))$, $i = 1, 2, \dots, N_q$, $\min(\mathbf{d})$ is minimum distance of all distances $\mathbf{d} = [d_1, d_2, \dots, d_{N_q}]$ in the subspace \mathbf{D}_q .
4. Use the ECM clustering algorithm to cluster and partition the input subspace \mathbf{D}_q of the N_q selected training samples.
5. Create the fuzzy rules and set their initial parameters according to results from the ECM clustering. Each cluster corresponds to one rule. The cluster centers and radii are used to form center and width of the Gaussian membership functions.

6. Apply the steepest descent method (back-propagation) to optimize the fuzzy parameters in the local model M_q by minimizing the error function. The equations are given in (4.10) - (4.19)
7. Calculate the model output y_q for the test point \mathbf{x}_q using the optimized local model M_q given a set of fuzzy rules.

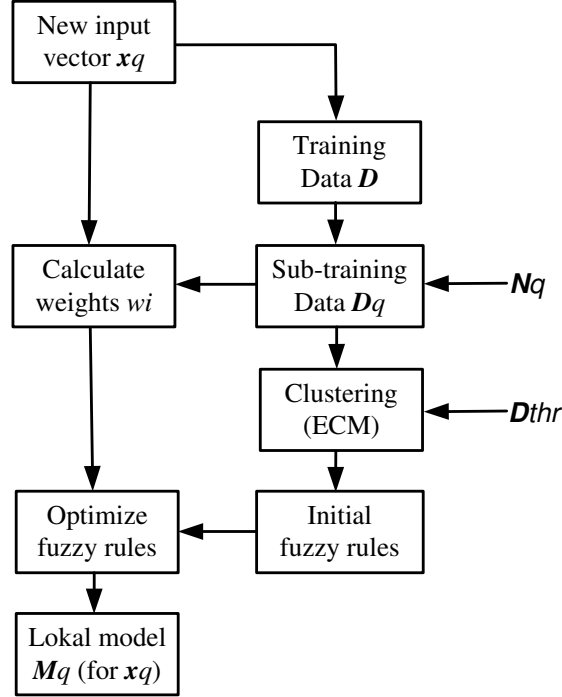


Figure 4.3: Flowchart of the NFI-algorithm. (Adopted from Kasabov&Song(2005))

More details about the weighting(step 3), the fuzzy inference system(step 5), and the back-propagation(step 6) is given below. In section 4.1.4 a complete description of the ECM clustering(step 4) is found.

step 3 - Calculating the weights

In NFI the N_q training points are weighted in the error function. The weights w_i are a little bit different from the weights used in the WKNN in section 4.2.

$$w_i = 1 - [d_i - \min(D)] \quad (4.4)$$

where $\max(D)$ and $\min(D)$ are the maximum and minimum distance of the N_q training points. All weights are between 0 and 1.

$$d_{close} = \min(D) \quad d_{far} = \max(D) \quad \implies \quad w_{close} = 1, \quad w_{far} = 1 - [\max(D) - \min(D)] < 1$$

step 4 - ECM : Evolving Clustering Method

The used ECM algorithm for data partition of the input space is a distance-based clustering method. Every single cluster C_l is described by its center C_{cl} and elongation (or radius) C_{rl} . The largest possible elongation of a clusters is specified by a threshold value, $Dthr$. This radius is the maximum distance from a cluster center to an example point, that belongs to that cluster. This means that the actual number of clusters M is determined by this threshold $Dthr$, and not specified by the user. The distance measurement method used is the *normalized Euclidian distance* as defined in equation 4.1.

Starting with an empty space of clusters, the points to be clustered is passed one by one to the ECM algorithm. The first point always gives a cluster with center on top of this point and radius 0. When more points are presented, some created clusters will be updated through new centers and increased radii. Furthermore new clusters will be created - always with initial radius 0. The updating of the clusters is based only on the distances between the current point \mathbf{x}_i and actually clusters centers and radii, meaning that a point simply passes through the model - which is suitable for on-line learning and time series prediction.

The model is described step-by-step in section 4.1.4 showing equations for the cluster updates. An simple 2-D example constructed by Nikola Kasabov and Qun Song (Kasabov & Song, 2002) is also given.

step 5 - The fuzzy inference system

With the reduced local input space divided into M clusters, the fuzzy inference model of M rules is sat up.

$$\begin{array}{lll}
 \text{cluster } C_1 & R_1 & \text{If } x_1 \text{ is } F_{11} \text{ and } x_2 \text{ is } F_{12} \text{ and } \dots x_P \text{ is } F_{1P}, \text{ then } y \text{ is } n_1 \\
 \text{cluster } C_2 & R_2 & \text{If } x_1 \text{ is } F_{21} \text{ and } x_2 \text{ is } F_{22} \text{ and } \dots x_P \text{ is } F_{2P}, \text{ then } y \text{ is } n_2 \\
 & \dots & \\
 \text{cluster } C_M & R_M & \text{If } x_1 \text{ is } F_{M1} \text{ and } x_2 \text{ is } F_{M2} \text{ and } \dots x_P \text{ is } F_{MP}, \text{ then } y \text{ is } n_M
 \end{array} \tag{4.5}$$

The type selected is the *Takagi-Sugeno* type. The rule output is expressed by piecewise linear functions keeping the model simple. The output of the l' th rule n_l is

$$n_l = b_{l0} + b_{l2}x_2 + \dots + b_{lP}x_P \quad l = 1, 2, \dots, M \tag{4.6}$$

The membership functions defining the fuzzy sets F of \mathbf{x}_i are Gaussian. For the l' th rule and j 'th input

$$GaussianMF_{lj}(\mathbf{x}_i) = \alpha \exp \left[-\frac{(x_{ij} - m_{lj})^2}{2\sigma_{lj}^2} \right] \tag{4.7}$$

The parameters m and σ is center and width of this "bell-shape" membership function. Using fuzzy inference all N_q training points have a membership to each cluster - even though the distance between them is larger than the cluster radius. Having a point $\mathbf{x}_i = [x_{i1}, x_{i2}, \dots, x_{iP}]$, this point has $P \times M$ memberships. One for each of the P features/coordinates relative to each of the M clusters. Is the point \mathbf{x}_i located on top on the cluster center m the membership is α . If it is placed on the border of the cluster so $x = m + \sigma$ then $(x - m)^2 / \sigma^2 = 1$ and the membership is $\alpha \exp(-1/2) \approx 0.6 \alpha$.

Described by the *and*-aggregation in the rules from equation 4.5 the activation μ_l of rule number l is a product of membership functions

$$\mu_l = \prod_{j=1}^P \alpha_{lj} \exp \left[-\frac{(x_{ij} - m_{lj})^2}{2\sigma_{lj}^2} \right] \quad (4.8)$$

Using the *Modified Centre Average defuzzification* procedure described by Bezdek(1981) in (Kasabov&Song, 2005) the model output $f(\mathbf{x})$ is calculated. This method is identical with the *Centre Of Gravity method for Singletons(GOGS)* described by Jantzen(1998). For one input vector $\mathbf{x}_i = [x_1, x_2, \dots, x_P]$ the model output $f(\mathbf{x}_i)$ is an weighted average over the contribution from the M rules. P is the number of features.

$$f(\mathbf{x}_i) = \frac{\sum_{l=1}^M n_l \prod_{j=1}^P \alpha_{lj} \exp \left[-\frac{(x_{ij} - m_{lj})^2}{2\sigma_{lj}^2} \right]}{\sum_{l=1}^M \prod_{j=1}^P \alpha_{lj} \exp \left[-\frac{(x_{ij} - m_{lj})^2}{2\sigma_{lj}^2} \right]} \quad (4.9)$$

For NFI classification purpose with N_q training samples \mathbf{x}_i with known class t_i , $i = 1, 2, \dots, N_q$ the objective is to minimizing the weighted error function for each i

$$\begin{aligned} E_i &= \frac{1}{2} w_i [f(\mathbf{x}_i) - t_i]^2 \\ E &= \sum_{i=1}^{N_q} E_i \end{aligned} \quad (4.10)$$

step 6 - Back propagation

To optimize the model parameters \mathbf{b} , α , \mathbf{m} and σ , the NFI-algorithm applies back-propagation using the *delta rule*. Through a backward pass calculating of local gradients the sensitivity of each parameter is calculated. Consider the parameter b_{l0} - the Takagi offset for cluster l using the *delta rule* stated below.

$$b_{l0} = b_{l0} + \eta \frac{\delta E_i}{\delta b_{l0}} \quad (4.11)$$

The learning rate η , controlling the step size of the updates. When η is small the update is small and the error decrease in small steps and the learning process is slow. On the other hand a large η speeds up learning, but may not ensure a global minimum to be found.

Performing the differentiation of 4.11 according to error function in 4.10 gives the parameter update equation of b_{l0}

$$b_{l0}(k+1) = b_{l0}(k) - \eta_b w_i \Phi(\mathbf{x}_i) [f^{(k)}(\mathbf{x}_i) - t_i] \quad (4.12)$$

where $\Phi(\mathbf{x}_i)$ is

$$\Phi(\mathbf{x}_i) = \frac{\prod_{j=1}^P \alpha_{lj} \exp\left[-\frac{(x_{ij}-m_{lj})^2}{2\sigma_{lj}^2}\right]}{\sum_{l=1}^M \prod_{j=1}^P \alpha_{lj} \exp\left[-\frac{(x_{ij}-m_{lj})^2}{2\sigma_{lj}^2}\right]} \quad (4.13)$$

Optimizing the local NFI-model having training samples $[\mathbf{x}_i, t_i]$, for $i = 1, 2, \dots, N_q$ gives N_q contribution to the parameter update.

$$b_{l0}(k+1) = b_{l0}(k) - \sum_{i=1}^{N_q} \eta_b w_i \Phi(\mathbf{x}_i) [f^{(k)}(\mathbf{x}_i) - t_i] \quad (4.14)$$

The update equations of b_{lj} , α_{lj} , m_{lj} and σ_{lj} is given in (4.15) - (4.18)

$$b_{lj}(k+1) = b_{lj}(k) - \eta_b w_i x_{ij} \Phi(\mathbf{x}_i) [f^{(k)}(\mathbf{x}_i) - t_i] \quad (4.15)$$

$$\alpha_{lj}(k+1) = \alpha_{lj}(k) - \frac{\eta_\alpha}{\alpha_{lj}(k)} w_i \Phi(\mathbf{x}_i) [f^{(k)}(\mathbf{x}_i) - t_i] [n_l(k) - (\mathbf{x}_i)] \quad (4.16)$$

$$m_{lj}(k+1) = m_{lj}(k) - \frac{\eta_m}{\alpha_{lj}^2(k)} w_i \Phi(\mathbf{x}_i) [f^{(k)}(\mathbf{x}_i) - t_i] [n_l(k) - f(\mathbf{x}_i)] [x_{ij}(k) - m_{lj}(k)] \quad (4.17)$$

$$\sigma_{lj}(k+1) = \sigma_{lj}(k) - \frac{\eta_\sigma}{\alpha_{lj}^3(k)} w_i \Phi(\mathbf{x}_i) [f^{(k)}(\mathbf{x}_i) - t_i] [n_l(k) - f(\mathbf{x}_i)] [x_{ij}(k) - m_{lj}(k)]^2 \quad (4.18)$$

$$\Phi(\mathbf{x}_i) = \frac{\sum_{l=1}^M \prod_{j=1}^P \alpha_{lj} \exp\left[-\frac{(x_{ij}-m_{lj})^2}{2\sigma_{lj}^2}\right]}{\sum_{l=1}^M \prod_{j=1}^P \alpha_{lj} \exp\left[-\frac{(x_{ij}-m_{lj})^2}{2\sigma_{lj}^2}\right]} \quad (4.19)$$

4.1.4 Evolving Clustering Method(ECM)

This section describes the individual steps of the ECM algorithm along with an example by Kasabov&Song(2002). The only parameter controlling the clustering process, is the maximum cluster radius specified by the threshold $Dthr$

Figure 4.4 shows a brief ECM clustering process in a 2-D space. The ECM algorithm is described below:

Step 0 Create the first cluster C_1 by simply taking the position of the first example from the input data stream as the first cluster centre C_{c1} , and setting a value 0 for its cluster radius Ru_1 (Figure 4.4a.)

Step 1 If all examples of the data stream have been processed, the algorithm is finished. Else, the current example, \mathbf{x}_i , is taken and the distances, between this example and all the m already created cluster centres C_{cl} , $D_{il} = \|\mathbf{x}_i - C_{cl}\|$, $l = 1, 2, \dots, m$, are calculated.

Step 2 If there is a cluster center (centers) C_{cl} , for $l = 1, 2, \dots, m$, so that the distance value, $D_{il} = \|\mathbf{x}_i - C_{cl}\|$ is equal to, or less than, the radius Ru_l , it is assumed that the current example \mathbf{x}_i belongs to a cluster C_m with the minimum of these distances: $D_{im} = \|\mathbf{x}_i - C_{cm}\| = \min(\|\mathbf{x}_i - C_{cl}\|)$, where: $D_{il} \leq Ru_l$, $l = 1, 2, \dots, m$

In this case, neither a new cluster is created, nor any existing cluster is updated (the cases of x_4 and x_6 in figure 4.4 and the algorithm returns to Step 1, else it goes to the next step.

Step 3 Find a cluster C_a (with a centre C_{ca} and a cluster radius Ru_a) from all m existing cluster centres through calculating the values $S_{il} = D_{il} + Ru_l$, $l = 1, 2, \dots, m$, and then select the cluster centre C_{ca} with the minimum value S_{ia} : $S_{ia} = D_{ia} + Ru_a = \min(S_{il})$, $l = 1, 2, \dots, m$.

Step 4 If S_{ia} is greater than $2 \times Dthr$, the example \mathbf{x}_i does not belong to any existing clusters. A new cluster is created in the same way as described in Step 0 (the cases of x_3 and x_8 in figure 4.4, and the algorithm returns to Step 1.

Step 5 If S_{ia} is not greater than $2 \times Dthr$, the cluster C_a is updated by moving its centre, C_{ca} , and increasing the value of its radius, Ru_a . The updated radius Ru_a new is set to be equal to $S_{ia}/2$ and the new centre C_{ca} new is located on the line connecting the new input vector \mathbf{x}_i and the cluster centre C_{ca} , so that the distance from the new centre C_{ca} new to the point \mathbf{x}_i is equal to Ru_a new (the cases of x_2 , x_5 , x_7 and x_9 in 4.4. The algorithm returns to Step 1. In this way, the maximum distance from any cluster centre to the farthest example that belongs to this cluster, is kept less than the threshold value, $Dthr$ though the algorithm does not keep any information of passed examples.

This example shows the principles of ECM. The example itself and figure 4.4 is adopted from Kasabov&Song(2002), but some small changes are applied.

Given a set of data \mathbf{x}_1 to \mathbf{x}_9 in a 2-D space. When the points are processed by ECM one by one, new clusters are created or existing clusters are updated. Cluster number j is described by C_j^k , and when a cluster is updated k is increased. Cc_j^k is the cluster center and Ru_j^k is the radius. For a color version of this thesis the evolving of a cluster is described by colors.

- (a) \mathbf{x}_1 : create new cluster C_1^0
- (b) \mathbf{x}_2 : update cluster $C_1^0 \rightarrow C_1^1$
 \mathbf{x}_3 : create a new cluster C_2^0
 \mathbf{x}_4 : do nothing
- (c) \mathbf{x}_5 : update cluster $C_1^1 \rightarrow C_1^2$
 \mathbf{x}_6 : do nothing
 \mathbf{x}_7 : update cluster $C_2^0 \rightarrow C_2^1$
 \mathbf{x}_8 : create a new cluster C_3^0
- (d) \mathbf{x}_9 : update cluster $C_1^2 \rightarrow C_1^3$

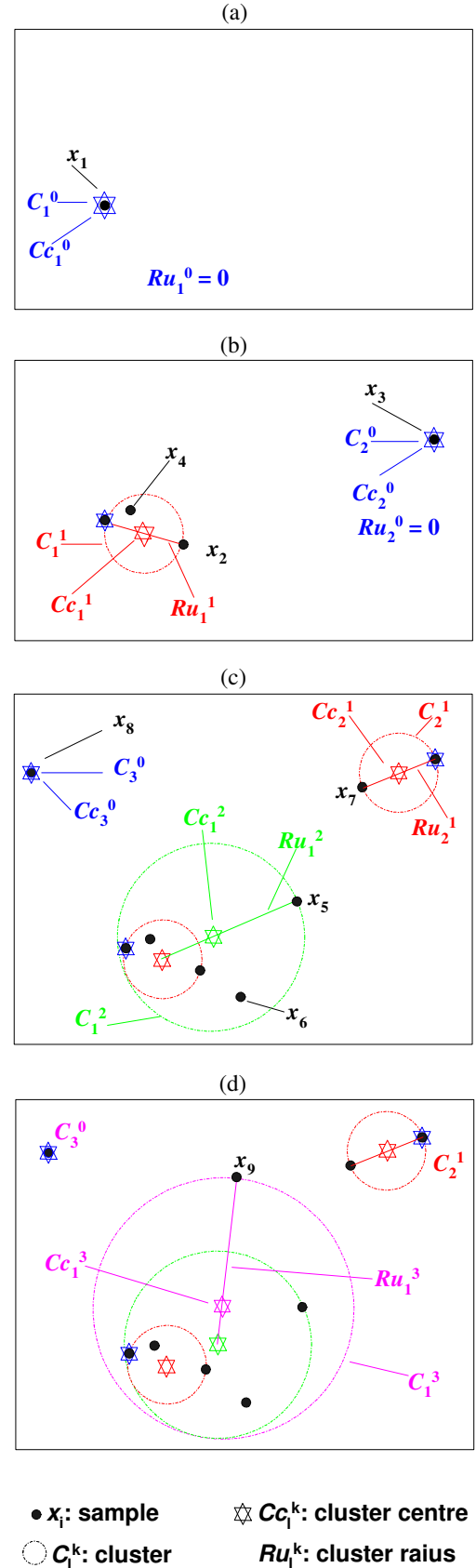


Figure 4.4: Example ECM

4.2 Results

The performance of various transductive classifiers is tested in this section, hopefully improving the results achieved with the inductive Least Square classifier tested in chapter 3. Common to all the transductive classifier is, that they are build on a number of limited training samples in neighborhood of the test point. The size of this neighborhood is expected to influence the classification error considerably and is therefore investigated.

For the simple models like: WKNN, KNN and NCC the influence of scaling in the input data is also investigated. The initial scaling normalize each feature in the closed range $[0; 1]$ (Kasabov& Song, 2005)

All methods are tested on the new pap-smear data, but only for separation into 2 classes of normal and abnormal cells.

In Kasabov&Song(2005) the NFI method is tested on the *iris* data, therefore the NFI-algorithm implemented in this thesis, also is applied to this set of benchmark data for comparison.

4.2.1 KNN and WKNN

Performing classification using WKNN and KNN is simply a weighting of the K nearest sample. Since no parameters are optimized during training, there is no actual training error and the only performance parameter is the test error.

As described in the introduction in section 1.6.2, the *leave-one-out* method is most suitable for transductive models. So this method is the one used in the following experiments. Using the leave-one-out cross-validation - the model building and testing - are performed for every sample in the entire database, this means that the classification result is unambiguous. And with no need for re-running, no statistical deviations are achieved.

Introducing the WKNN method in Kasabov&Song(2005) a scaling of the input data in the range $[0;1]$ for each feature is assumed. Using this scaling, the number-of-nearest-neighbors dependency is tested. Figure 4.5 shows, that the RMSE performance of WKNN and KNN varying the number of nearest neighbors used for the classification. Both methods perform almost the same way, but the WKNN is a bit better. The likeness between the methods shows that the benefit of the applied weighting using the WKNN method, is increased with K . The optimal value of K seems to be $K \approx 10$. Figure 4.6 shows the corresponding classification error(OE%). Apparently KNN performs slightly better for some values of K . In general the overall error changes a lot with K , indicating that many samples changes class. The lowest overall error is achieved for $K = 8$. For WKNN the error is 7.09% and 6.76% for KNN, showing that KNN actually performs better. The results are listed in table 4.2, showing a very large false-positive error.

method	Overall error	FN%	FP%	RMSE	N	scaling
WKNN	7.09%	1.78 %	21.90%	0.2290	8	range[0;1]
KNN	6.76%	1.78 %	22.64%	0.2264	8	range[0;1]

Table 4.2: WKNN and KNN - scale : $range[0;1]$ - leave-one-out

To improve the results, a different scaling is applied. Instead of a normalization of each feature in the range $[0;1]$, a standardization is applied with zero mean. This scaling is from now on named $mean=0/std=1$. Figure 4.7 and 4.8 show the performance error. The overall error is slightly improved to $\approx 6.5\%$ for both WKNN and KNN, and the number of nearest neighbors is reduced to only $K = 3$. In general WKNN is not much different from KNN, indicating that the results above more or less remain unaffected by the applied weights.

As described on page 39 Byriel(1999) used an alternative weighting. Here the closest

method	Overall error	FN%	FP%	RMSE	N	scaling
WKNN	6.54%	2.52 %	17.77%	0.2326	3	mean=0/std=1
KNN	6.65%	2.52 %	18.18%	0.2348	3	mean=0/std=1

Table 4.3: WKNN and KNN - scale : $mean=0/std=1$ - leave-one-out

sample with distance d_i was weighted with $w_i = 1 - \frac{d_i}{\max(D)}$. And the sample most far away was weighted with zero. Only if the nearest neighbor was located on top of the test point($d_i = 0$), it was weighted with 1.

$$d_{close} = \min(D) \quad d_{far} = \max(D) \quad \implies \quad w_{close} = 1 - \frac{d_i}{\max(D)}, \quad w_{far} = 0$$

The method using this weighting, was named NNH(Nearest NeighborHood)¹ by Byriel(1999). The K dependency for NNH for both types of scaling is showed in figure 4.9 and 4.10. The standardize scaling $mean=0/std=1$ shows the best results, an overall error of 6.43% using the $K = 5$ nearest neighbors. But compared the original weighting using WKNN and KNN, the improvement is minor. Table 4.4 shows the false-positive and false-negative error as well.

The achieved overall error is not impressive, compared to linear results from chapter 3. The huge variation of the overall error indicates, that many samples changes class when K is changed. A large overlap of some classes is indicated by this large error. All methods are sensitive to the distribution of the data set, because predominance of one class will favour this class in the model output as described in section 4.1.2. To improve the error the NCC method is tested below.

¹Applied on the old pap-smear, the result show good appliance (Byriel, 1999)

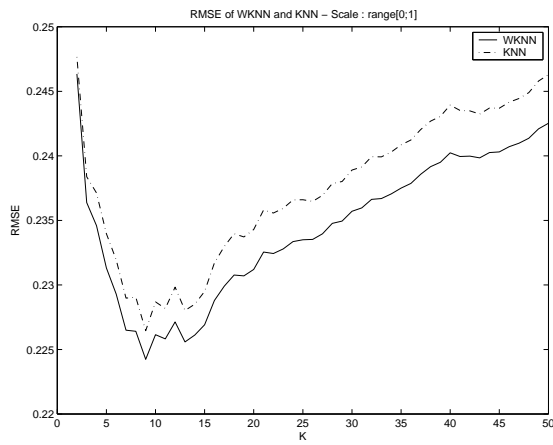


Figure 4.5: RMSE of WKNN and KNN - scale : range[0;1]

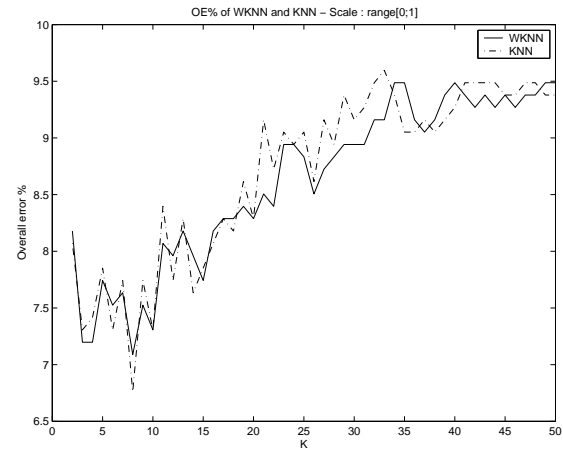


Figure 4.6: Overall error(OE%) of WKNN and KNN - scale : range[0;1]

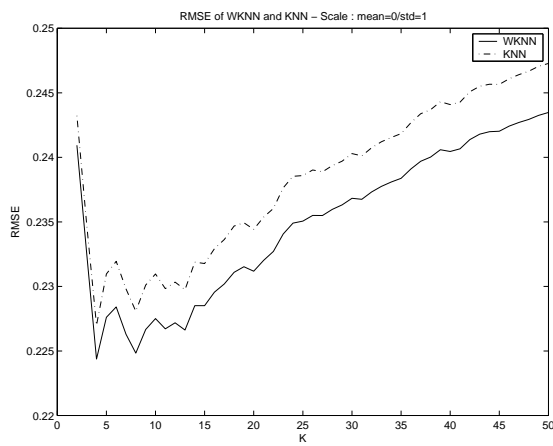


Figure 4.7: RMSE of WKNN and KNN - scale : mean=0/std=1

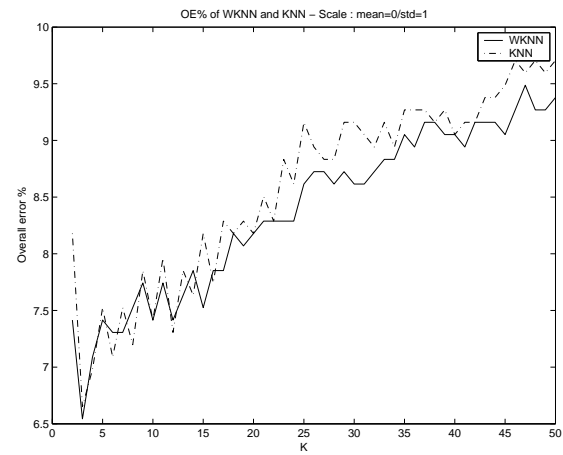


Figure 4.8: Overall error(OE%) of WKNN and KNN - scale : mean=0/std=1

method	Overall error	FN%	FP%	RMSE	N	scaling
NNH	6.43%	2.96 %	16.12%	0.2261	5	mean=0/std=1

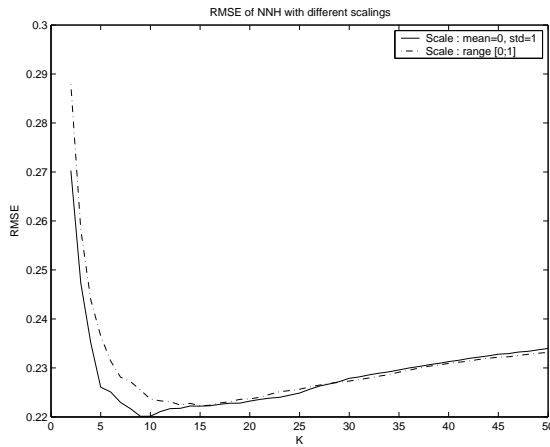
Table 4.4: NNH - scale : $mean=0/std=1$ - leave-one-out

Figure 4.9: RMSE of NNH

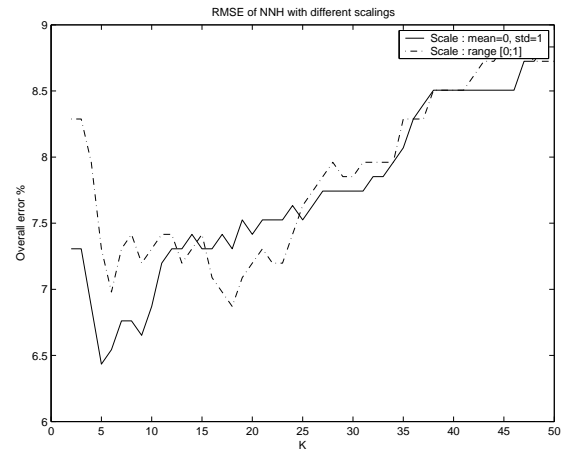


Figure 4.10: Overall error(OE%) of NNH

4.2.2 NCC - Nearest Class gravity Center

Since the NCC classifier adopt the class of closest center of gravity, the model output are scalar class values. A system with the classes 1 & 2 gives $RMSE = \sqrt{OE/100\%}$ as showed in example 1.2 in section 1.6.1. Testing the performance of the NCC method only, the overall error is showed on figure 4.11. In the light of the results from the WKNN and KNN classification, both scaling methods are applied.

The best result is achieved using the $K = 21$ nearest neighbors and a scaling with zero mean. An overall error of 5.13% is superior compared to all models previously tested. The matching false-negative and false-positive error is given in table 4.5, showing that the difference between them are decreased - FN%=4.30 and FP = 7.44%. In general the standardize scaling of $mean=0/std=1$ performs better than $range [0;1]$, just as it was observed for WKNN and KKN methods.

method	Overall error	FN%	FP%	RMSE	N	scaling
NCC	6.22%	5.04 %	9.50%	0.2493	21	range[0;1]
NCC	5.13%	4.30 %	7.44%	0.2264	21	mean=0/std=1

Table 4.5: NCC - 2 classes with different scalings - leave-one-out

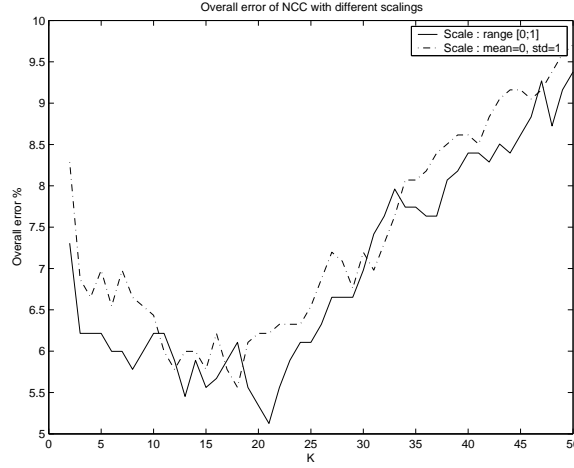


Figure 4.11: NCC-2 classes with different scalings - leave-one-out

4.2.3 NFI results

The implemented NFI algorithm is tested on both the *iris* data and the new pap-smear data.

As described the cluster radii and centers found by ECM are used for initial values of the fuzzy memberships functions in equation 4.7. Remembering, that if only a single point is assigned to a cluster, the radius is zero. Using a zero radius as initial width σ of a cluster, performs a division by zero in the Gaussian Membership definition, so instead a of zero a small number $\sigma_0 = 0.0001$ is used.

The number of epochs used for updating the parameters is determined by a stop criteria. The criteria is two fold. Either the maximum number $Epoch_{max} = 500$ is reached or the error function is minimized below a certain threshold value. The total error E in equation 4.10 is a sum of contributions from each of the nearest neighbors used to build the model. And is therefore depended of N_q . Defining an error threshold $Epoch_{error}$ the back-propagation stops when $\frac{E}{N_q} \leq Epoch_{error}$. The $Epoch_{error}$ can be considered as the mean error of one training sample. The implementation below use $Epoch_{error} = 0.001$.

Testing NFI on iris Data

For validation of the implemented NFI algorithm, it is tested on another set of benchmark data. Since the originators to the NFI method in (Kasabov&Song, 2005) testing the performance on the *iris data*, these are chosen.

The iris data contains 150 samples separated on three classes of 50 instances each. Each sample is described by four features:

1. sepal length
2. sepal width

3. petal length
4. petal width

Each Class refers to a type of iris plant with the following medical descriptions:

1. Iris Setosa
2. Iris Versicolour
3. Iris Virginica

The class Iris Setosa(1) is linearly separable from the other two classes, Iris Versicolour(2) and Iris Virginica(3). The latter are not linearly separable from each other.

As described earlier the only tuning parameters for NFI-method is the ECM threshold distance $Dthr$ and the number of N_q nearest neighbors used to build the model.

The result achieved by (Kasabov&Song, 2005) showed an average error of 3.3 samples or 4.4% with 6.3 clusters found by ECM. The average was calculated over 10 repetitions, using 50% of data for testing and 50% for training randomly selected. It is worthy of note, that the method is not a 2-fold cross validation, since training and test data probably are not swapped. Unfortunately only a few test conditions and parameters are stated in (Kasabov&Song, 2005), which makes it difficult to perform a matching test. The number of clusters found by the ECM clustering process, depends on the threshold distance $Dthr$ and the number of N_q nearest neighbors used to build the model. The relation is investigated running ECM for different combinations of these parameters as showed on figure 4.12. The number of clusters is an average of 20 repetitions of ECM. The horizontal line at 6.3 clusters indicates the number of clusters used by Kasabov&Song(2005). It noticed that the curve has a bend at $N_q \approx 25$, which probably is related to contribution of the iris data.

To achieve a NFI classification matching earlier results, various combinations of $Dthr$ and N_q matching 6.3 clusters are picked out for a full classification. Figure 4.13 shows a zoom of figure 4.12 pointing out results in the vicinity of 6.3 clusters. Points encircled by (O) are picked out.

Table 4.6 shows the resulting classification of the NFI method using these different combinations. An overall error in the range from 4.4 % to 6.0% is achieved, matching results of 4.4% achieved by Kasabov&Song(2005).

According to Kasabov&Song(2005) the NFI method is superior to eg. the inductive Multi-layer perceptron(MLP) - table 4.7. But applying other methods on the iris data eg. the methods introduced earlier in the thesis, the NFI results are not that exceptional.

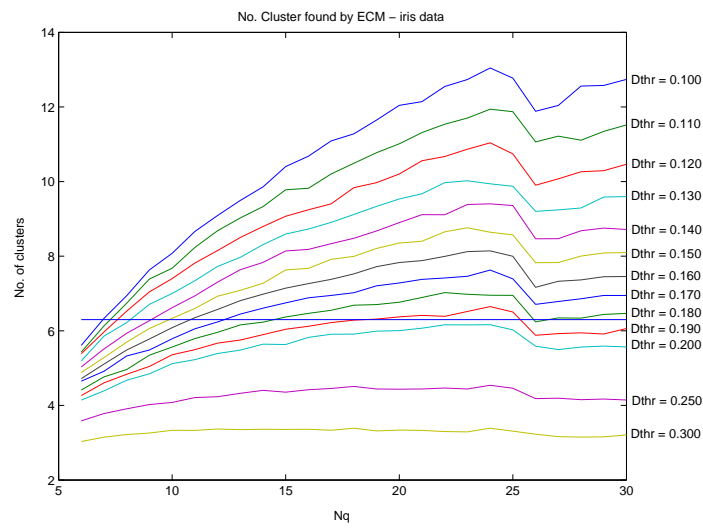


Figure 4.12: ECM iris data. Cluster dependency of distance $Dthr$ and number of K nearest neighbors

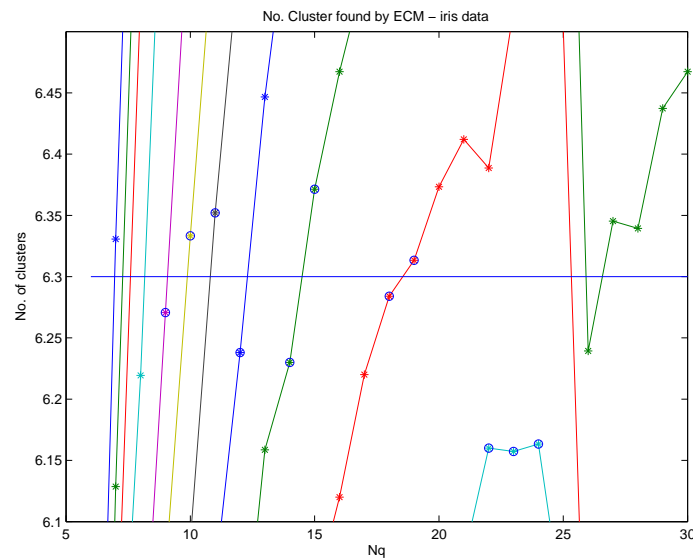


Figure 4.13: ECM iris data. Cluster dependency of distance $Dthr$ and number of K nearest neighbors. Zoom of figure

no. cluster	no. errors	Overall error	N_q	$Dthr$
6.2	4.3	5.7 %	9	0.140
6.4	4.0	5.3 %	10	0.150
6.1	4.2	5.6 %	10	0.160
6.3	3.8	5.1 %	11	0.160
6.0	3.9	5.2 %	11	0.170
6.2	4.4	5.9 %	12	0.170
6.3	3.9	5.2 %	14	0.180
6.3	3.3	4.4 %	15	0.180
6.3	4.5	6.0 %	18	0.190
6.4	3.8	5.1 %	19	0.190
6.1	3.7	4.9 %	22	0.200
6.2	3.9	5.2 %	23	0.200
6.1	3.7	4.9 %	24	0.200

Table 4.6: NFI tested on Iris data using different combinations of the number of N_q nearest neighbors and the ECM distance threshold $Dthr$. The errors is an average of 10 repetitions, randomly selecting 50% of 150 sample for test data.

	method	Cluster	Overall error	N_q	$Dthr$
Kasabov&Song(2005)	MLP	12.0	6.13 %	N/A	N/A
Kasabov&Song(2005)	NFI	6.3	4.4 %	Unknown	Unknown
Norup(2005)	NFI	6.3	4.4 %	15	0.180
Norup(2005)	Least Square	-	3.6 %	-	-
Norup(2005)	NCC	-	3.6 %	24	-

Table 4.7: Test on *iris Data*

Testing NFI on pap-smear data

The NFI method is also tested on the new pap-smear data. As described earlier the number of clusters used, depends on the ECM threshold radius $Dthr$ and the number of nearest neighbors N_q . Remembering that the method is transductive the clustering is performed only on a subspace of nearest neighbors.

Selecting the threshold and number of nearest neighbors is, of course, essential for the performance of the NFI-algorithm. Figure 4.14 shows the ECM cluster dependency of N_q and $Dthr$.

For small thresholds the number of clusters is increased with N_q . But using a large threshold, the number of clusters is more or less independent of N . In general the number of clusters found by ECM is, of course, smaller or equal to numbers of nearest neighbors used.

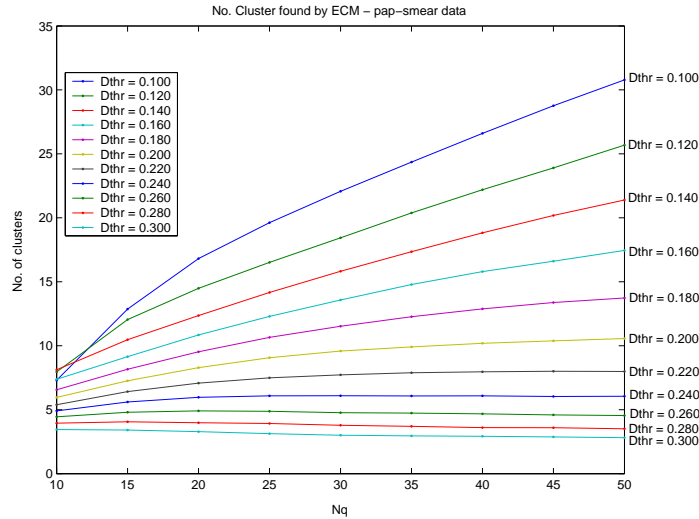


Figure 4.14: ECM clustering on pap-smear data. Cluster dependency of distance $Dthr$ and number of N_q nearest neighbors.

Using the ECM algorithm the clusters are formed from the data points and an empty cluster can't exist. For optimal performance of the NFI method, the number of nearest neighbors have to be much larger than the number of clusters. Figure 4.15 shows the classification error dependency of N_q for various ECM thresholds $Dthr$. An optimal classification error is achieved for a large threshold of $Dthr = 0.300$. In general the overall error(OE%) seems to be improved with the threshold.

method	Overall error	FN%	FP%	RMSE	N_q	Dthr
NFI	5.67%	3.85 %	10.74%	0.2214	35	0.300

Table 4.8: NFI pap-smear data - leave-one-out

4.3 Conclusion

In this chapter different transductive classifiers have been tested. The main issue was the NFI algorithm. A fuzzy inference model based on ECM clustering of a local input space. The method have been tested for different combinations of threshold radius for the clustering algorithm and number of training samples forming the subspace. An overall error of 5.67% was achieved. The method uses back-propagation for training the local model, but performs slowly compared to the other transductive methods tested.

The NCC method performed actually better, and much faster. An overall error 5.13% was

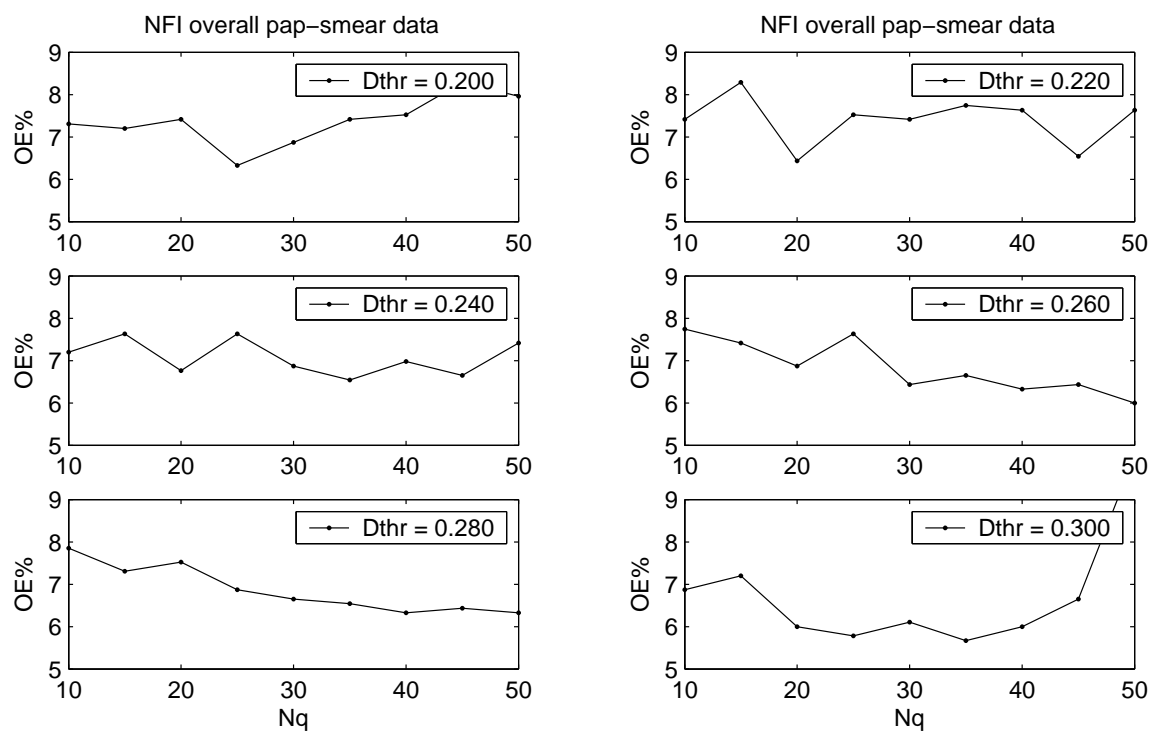


Figure 4.15: NFI pap-smear data.

achieved. The remaining transductive methods KNN, WKNN and NNH all performed unsatisfactory, because the classification result was beaten by the simple Least Squared method tested in chapter 3.

The idea of using transductive methods is definitely not exploded, but may be the applied models can be improved in some way. Further on traditional global models can be applied directly or adjust to local modeling.

Chapter 5

Summary of results & discussion

Benchmark database

The latest data set of pap-smear data from Herlev University Hospital has been described in a small reference guide in chapter 2. This guide includes an individual description of all input features and the 7 different classes.

The guide is intended as a supplement to the existing pap-smear database of extracted features found on the enclosed CD-rom in Appendix A. The idea is to publish both on WWW as a benchmark data free to use. It is possible to test a classifier simply downloading the Excel-datasheet, but descriptions of classes and features could be valuable for eg. reduction into a more simple 2 class problem or removing features. The pap-smear data base is very complex with many samples, many features and many different classes compared to other benchmark data sets. But further simplification is of course possible. From the Least Squared classification into 7 direct classes, the confusion matrix in table 3.3 on page 33 shows, that moving all class 3 samples probably would make a separation between normal and normal cells much easier. Looking at the class 3 row and class 3 column, shows that many class 3 samples is misclassified as abnormal(class 4,5&6&7) and many abnormal cells are misclassified as class 3 samples.

The objective for publishing the pap-smear on World Wide Web is to extend the knowledge of the database, and in the long term hopefully achieve improved classifiers. This would of course be at benefit for Herlev University Hospital and the pap-smear research area in general. To increase the interest of the database and get more hits on WWW a web-based documentation could with benefit be developed. Also maybe with some simple Matlab scripts for downloading. Another well presented cancerdemo is already available on the Internet¹ as a demo of the Matlab Bioinformatics Toolbox 2.0.1.

¹<http://www.mathworks.com/products/demos/bioinfo/cancerdetectdemo/cancerdetectdemo.html>

Classification results

To improve earlier results, various types of classifiers have been tested on the pap-smear data. Fundamental they have been grouped into

- Inductive classifiers
- Transductive classifiers

Where the inductive classifiers perform global model building on the entire training set, and the transductive classifiers build a local model optimized individually for each test point.

The only inductive method tested was the Least Squared Method(LS). One of its strong points is, that it is very simple and fast. Using both 2-fold and 10-fold cross-validation the classification error was showed to be normal distributed. And the RMSE error was roughly normal distributed as well. The best classification error achieved was an overall error of 6.40%, but with a consequently large difference in the false-positive and false-negative error. But squeezing the activation threshold, this difference could be minimized without increasing the overall error considerably. The inductive results is showed in table 5.1.

method	Overall error	FN%	FP%	
LS - 2 classes - $thr = 1.50$	6.40%	1.23 %	20.66 %	
LS - 2 classes - $thr = 1.65$	6.52%	6.04 %	7.82 %	

Table 5.1: Inductive classification

A direct classification into the 7 original classes in pap-smear database was almost impossible, given an useless classification error of more than 40%. But in general the investigation showed, that for classification with more than 2 classes, a scalar class description was not adequate. But it has to be added, that no optimization of the 6 thresholds levels was applied. And most likely the classification error could be improved adjusting these thresholds. Using least square fitting for optimization of the 6 threshold level could be applied. This would introduce new weights and the optimal results is probably identical with binary class solution. Using the knowledge of all 7 classes for building the Least Square model, and then perform a classification into 2 classes did not improved the results. The method can be considered as a sort of supervised model building, because prior knowledge of the classes was build into the model.

The Least square Method was introduced because it was inductive and simple, but its performance was expected to be improved, using both more advanced transductive methods and transductive methods in general.

Unfortunately it turned out that none of the most simple transductive classifiers like WKNN, KNN and NNH improved the Least Square classification error. The influence of

the different weighting was minor. Comparing the weight functions W_{i-NNH} and W_{i-WKNN} shows that they almost are identical with same slope, but different off-set.

$$w_{i-WKNN} = 1 + \frac{\min(D)}{\max(D)} - \frac{1}{\max(D)}d_i \quad w_{i-NNH} = 1 - \frac{1}{\max(D)}d_i$$

In general the three methods KNN, WKNN and NNH can be considered as identical methods, but with different weighting applied. And for the KNN methods all weights are 1. Also different scalings applied on the input data showed improvements. The normalization in the range[0;1] introduced by Kasabov&Song(2005) does not seem to be optimal for any of the nearest neighbors classifiers tested. But a standardization with zero mean was more useful.

The main issue for testing classifiers was the Neuro-Fuzzy Inference Method for Transductive Reasoning(NFI). The method is the most advanced of all tested methods and with a large computational load. An overall error of 5.7% improved some of the simple transductive classifiers. But maybe NFI-algorithm can be further improved. Using the ECM clustering the clusters have same elongation in all directions, given by the cluster radius. The use of hyper-spheres for the fuzzy memberships is maybe not optimal. Kasabov&Song(2003) introduce ECMC : Evolving Clustering Method for Classification applying different shapes for each cluster. The intention is identical with the one used in the Gustafson-Kessel(GK)-algorithm tested by Martin(2003). The achieved error was 6.06%, applied for inductive modeling, as mentioned in Chapter 1. To verify the NFI implementation is was tested on the *iris* data as well, and then compared to previously results by (Kasabov&Song), it showed matching results. Applying of simply methods like NNC and LS showed surprising results improving the error from 4.4% til 3.6% on the *iris* data.

The classification error of all tested transductive classifiers is showed in table 5.2.

method	Overall error	FN%	FP%	RMSE	N_q/K	
NCC	5.13%	4.30 %	7.44%	0.2264	21	mean=0/std=1
NFI	5.67%	3.85 %	10.74%	0.2214	35	Dthr = 0.300
NNH	6.43%	2.96 %	16.12%	0.2261	5	mean=0/std=1

Table 5.2: Summary of classification results

Compared to previously results achieved on the *pap-smear* data by Martin(2003) the overall classification error has been improved from 6.06% to 5.13% with NCC as the most superior method.

Chapter 6

Conclusion

From an existing database of single pap-smear cells with 20 already extracted features, a detailed stand-alone description has been given. Using the database, a simple worksheet with features and class description along the columns and samples along the rows, and the developed guide it is possible to test own classifiers on the data.

The introduction of local transductive methods, applied on the pap-smear data, has improved the overall classification error for some methods compared to previous results. But not all transductive methods improved the results. Simple transductive methods like K Nearest Neighbors(KNN) actually performed worse than the simple inductive Least Square Method(LS). Adding on different weights to KNN method did not improve the results considerably. The optimal classifier tested was Nearest Class gravity Center(NCC) achieving an overall error of 5.1%. A simple method introduced to compensate for pre-dominance of single classes in the data-set.

The newly introduced Neuro-Fuzzy Inference Method for Transductive Reasoning(NFI) was applied and showed partly satisfactory results. With an achieved error of 5.67% existing classification results was improved, but on the other hand the results was beaten by the much more simple NCC method. Both NCC and NFI improved earlier optimal classification results of 6.06% for the overall error.

Regrettably, no feature selection was applied in this current project. But previous pap-smear projects have showed large improvement using feature selection. Extending the tested methods with feature selection the satisfactory results above could improved.

Jonas Norup
(norup@teknik.dk)
Lyngby, 2nd of May, 2005

Bibliography

- Bezdek, J. (1981). *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York.
- Bishop, C. M. (1995). *Neural Network and Pattern Recognitions*. Oxford University Press. ISBN 0-19-853864-2.
- Byriel, J. (1999). Neuro-fuzzy classification of cells in cervial smears. Master's thesis, Technical University of Denmark, Oersted. Dept. of Automation.
- Eising, J. (1993). *Linær Algebra*, volume ISBN 87-88764-39-7. Dept. of Mathematics, Technical University of Denmark (DTU), 1. edition, 2. impression edition.
- Indman, D. P. (2005). M.d. <http://www.gynalternatives.com/abnpap.htm>.
- Jantzen, J. (1998). Design of fuzzy controllers. Technical report, Technical University of Denmark, Dept. of Automation, 98-E-864.
- Kasabov, N. and Song, Q. (2002). Denfis : Dynamic, evolving neaural fuzzy inference system and its application for time-series prediction. *IEEE Transaction on Fuzzy System*, 10(2):144–154.
- Kasabov, N. and Song, Q. (2003). Weighted data normalizations and feature selection for evolving connectionist systems proceedings. pages 285–290. Sydney, Australia,.
- Kasabov, N. and Song, Q. (2005). Nfi: A neuro-fuzzy inference method for transductive reasoning. *IEEE Transactions on Fuzzy System*, NOT PUBLISHED YET.
- Martin, E. (2003). Pap-smear classification. Master's thesis, Technical University of Denmark(DTU), Oersted. Dept. of Automation.
- Shanmugan, K. S. and Breipohl, A. (1988). *Random signals Detection, Estimation and Dataanalysis*, volume ISBN-0-471-81555-1. John Wiley and Sons, Inc.

Appendix A

CD-rom

The enclosed CD-rom contains :

- This master thesis : `Norup2005.pdf` and `Norup2005.ps`
- The new pap-smear database (Excel-sheet) : `new_database_results.xls`
- Directory of Matlab classification files : `matlab`