Pap-Smear Classification

Technical University of Denmark - DTU

***Author*:**
Erik Martin
s960863

22nd September 2003

Supervisors:

Jan Jantzen
<jj@oersted.dtu.dk>
Oersted-DTU, Automation

Beth Bjerregaard
Department of Pathology
Herlev University Hospital

Abstract

Classification of a papsmear is a manual and time consuming task. This thesis deals with classification on features extracted from single cell pictures of papsmear's. Classification on an old database with 500 cells and a new database with 917 cells were done. The new database was prepared and features extracted. Classification with Fuzzy C-means and Gustafson-Kessel clustering were applied and errors were measured with K-fold cross-validation.

A supervised clustering procedure was proposed and compared to unsupervised clustering, showing supervised clustering performed better.

Acceptable false-positive and false-negative errors $< 5\%$ was obtained for the old data. Supervised Fuzzy C-means performed the best with a false-positive 2.02% and a false-negative 1.38%. Unacceptable false-positive and false-negative errors $> 5\%$ was obtained for the new data. Supervised Fuzzy C-means and Supervised Gustafson-Kessel clustering performed equal with a false-positive of 13.9% and false-negative of 3.3%.

A robustness analysis was made, showing Fuzzy C-means had a high robustness to noise in the features whereas Gustafson-Kessel showed a much smaller robustness. The conclusion was Fuzzy C-means gave the best classification results and had the best robustness against noise.

Contents

Chapter 1

Introduction

1.1 Background

In the 1930'ties, Georges Papanicolaou found a method to diagnose pre-cancerous cells in the uterine cervix. This method is referred to as pap(anicolaou)-smear diagnosis. The papsmear has made it possible to detect pre-cancerous cells in the uterine cervix, so effective treatment can be given. The importance of papsmear diagnosis became clear with the British Columbia cytology screening program[1] in Canada. It showed that the mortality and morbidity was lowered considerably by offering papsmear screening to the population in British Columbia. As a consequence of the effectiveness of papsmear diagnosing, almost all woman in Denmark are offered papsmear screening every 3 years.

The papsmear screening process needs skilled and trained cyto-technicians, but they are costly and can be hard to find. Several automatic and semi-automatic systems to diagnose papsmear are available, but they are more expensive to use than skilled cyto-technicians. This lead to a project between Herlev University Hospital's department of pathology and the commercial company DIMAC.

Several Master's thesis projects, concerning classification of papsmear cells, has been formed around a database made by Herlev hospital. Herlev hospital has now made a new database, for which classification results are wanted.

1.2 Objective

The objective is to prepare the new database for publication, such that it can be used for benchmarking of classifiers.

[1] See www.chrcrm.org/doc_contrE_boyes.htm

1.3 Literature

The project of Byriel (1999), showed that clustering algorithms could obtain false-positive and false-negative errors $< 5\%$ on single cell papsmear pictures. He obtained the best results with an Adaptive Network based Fuzzy Inference System(ANFIS). The results he obtained for Fuzzy C-means were worse than the Gustafson-Kessel results, indicating Gustafson-Kessel should be a better algorithm. His project used 7 clusters for the clustering algorithms, since there were 7 different diagnoses in the data. This assumption does not necessarily give the best classification error, maybe more than one cluster for each diagnosis would perform better. The fuzziness exponent value used, did not appear in the report. A value of 2 was probably used, since it normally is considered to be acceptable. Therefore the comparison of Fuzzy C-means and Gustafson-Kessel in his project was based on parameters not necessarily giving the best results. This project will try to find a near optimal cluster count and fuzziness exponent for the clustering algorithms.

The clusters found with the clustering algorithms are to be used when classifying on test data. The project of Byriel (1999) seemed to let every cluster represent one diagnose, but did not tell how to assign the diagnoses to the clusters. This project will propose several defuzzification methods, describing how to classify test data on the found clusters.

In the project of Landwehr (2001), classification with the K-nearest neighborhood algorithm on papsmear single cell pictures was performed. His project classified every cell diagnosis with a direct and hierarchical classification. It showed that a hierarchical classification gave better results than a direct classification. In excess of the features used in Byriel (1999), some extra texture features were applied in Landwehr (2001). This project will use the same features as in the project of Byriel (1999), but the features will be extracted from the new database with Matlab code, so others later on can modify and implement other features, like the texture features in Landwehr (2001). In Walker, Jackway & Longstaff (1997), several useful texture features based on the gray value co-occurrence matrix are proven effective, and are therefore a good guess of new features to try out. In the project of Lin & Fu (1983), a hierarchical approach was used with a K-nearest neighborhood to classify on cervical cells. That project used texture, geometrical and optical density features.

The results obtained in Byriel (1999) and in Landwehr (2001), where obtained with a simple division of the data into training and test data on which the classification errors were measured. To maximize the precision of the results compared to their projects, k-fold cross-validation is used in this project.

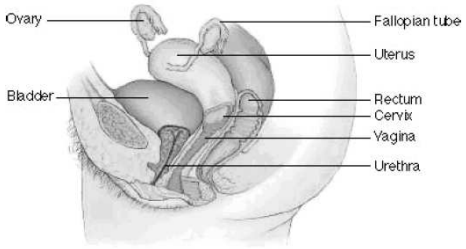

Figure 1.1: The female reproduction system. The figure is copied from Landwehr(2001) with permission.

Figure 1.2: The uterus in detail. The figure is copied from Landwehr(2001) with permission.

1.4 The Papanicolaou Smear

The Papanicolaou smear method, is a medical procedure to find pre-cancerous cells in the uterine cervix.

A small cytological specimen from the uterine cervix(se figure 1.1 and 1.2) is collected with a special cyto-brush and smeared onto a glas slide. Then the slide is stained using the Papanicolaou method, so the different components of the cells are emphasized with specific colors. This glas slide is then viewed under a microscope, so cyto-technicians can diagnose the cells on the glass slide. Cyto-technicians use several different features to get a cell diagnosis. The size, color, shape and the texture of the nucleus and cytoplasm is used. The density of cells in a certain area, can influence the diagnose. It takes a skilled cyto-technician, to be able to differentiate between the different cells. Every glas slide, can contain up to 300.000 cells. Therefore it is a time consuming job viewing the slides.

The papanicolaou cells

Ideally specimens are taken from several areas of the cervix. Depending on the area, the cyto-brush, cotton stick or the wooden stick is used. The specimens most often contain cells from the columnar epithelium and the squamous epithelium. The columnar epithelium is located in the upper part of the cervix, and the squamous epithelium in the lower part. In between these two areas, the metaplastic epithelium is found, also called the transformation zone.

In the squamous epithelium there are 4 layers of cells. The cells start out being formed at the basal layer and while maturing they move out through the parabasal layer, the intermediate layer and at last out through the superficial layer. The cells in the basal layer divide and deliver cells to the layers above it. While the cells mature and move through the layers, they change shape, color and other characteristics. When the cells reach through the superficial layer they are rejected and replaced by the cells coming from below. The basal layer has small round cells with a relative big nucleus and a little cytoplasma. When maturing, the nucleus becomes smaller and the cytoplasma becomes bigger. The shape of the cells become less round the more mature they are.

The columnar epithelium only contains a single layer of cells, the basal layer. The basal layer here contains columnar cells and reserve cells. The reserve cells divides into new reserve cells and into columnar cells.

The metaplastic epithelium consists of reserve cells from the columnar epithelium. When the cells have matured fully in the metaplastic epithelium, they look like the cells found in the squamous epithelium.

When the genetic information in a cell somehow has changed, the cell will not divide as it should. This is a pre-cancerous cell. Depending on which kind of cell that is dividing incorrectly, it is given diagnoses like dysplasia and carcinoma in situ. The dysplastic cells are divided up into mild, moderate and severe dysplastic. The graduation into different degrees of dysplasia are determined from the probability of the cells later on turning into malignant cancer cells. A high amount of the mild dysplastic cells will disappear without becoming malignant, whereas severe dysplastic cells quite likely will turn into malignant cells. The pre-malignant cells are especially characterized by a larger nucleus and a bigger N/C ratio. The N/C ratio is given by:

$$N/C - ratio = \frac{NucleusArea}{NucleusArea + CytoplasmaArea} \tag{1.1}$$

Characteristics like these are called features later on. Many other characteristics are also used. In general the pre-malignant cells are characterized by having an appearance deviating from the appearance of the normal cells.

1.5 Problem Statement

Classification of cells from the uterine cervix papsmear, is a problem even for skilled cyto-technicians. A newly created database and an older, made by the cyto-technicians at Herlev hospital, containing images and diagnoses of papsmear single cells, is the foundation on which automated classification shall be performed on.

The classification shall use features extracted from images of the cells. To extract features, a segmentation of nucleus, cytoplasm and background is nec-

essary. The so-called CHAMP software, developed by DIMAC Imaging[2], is used for the segmentation by the cyto-technicians. Feature extraction shall be implemented in Matlab[3] and compared with feature extraction by CHAMP.

The newly created database, made by Herlev hospital, containing images and diagnoses of pap-smear cells are to be used. But also an older database made by Herlev hospital shall be used.

The new database shall first be prepared. Segmentation and feature extraction shall be performed on it. The features of the old database was extracted in the project of Byriel (1999) with CHAMP, and therefore are ready for use.

The following classification methods, are to be used and compared:

- C-means clustering
- Gustafson-kessel clustering

The classification shall first be between normal and abnormal cells, afterwards between all the diagnoses. Classification on the exact diagnose shall be done with direct and hierarchical classification. Results shall be obtained for both databases separately, without mixing their data.

Performance shall be measured by the False-positive, False-negative and Overall correct classification error.
The goal is classifying the cells with a less than 5% false positive and false negative rate, and therefore also a less than 5% in overall error.

Work plan

In the following, a list of activities are shown:

- Preparation of the new database.
- Feature extraction on the new database with Matlab and CHAMP software. Comparison of features from Matlab and CHAMP.
- Classification on the old database between normal/abnormal cells.
- Classification on the new database between normal/abnormal cells.
- Classification on all cells diagnoses in the old and in the new database by using direct and hierarchical classification.

This is a short list, where every activity can have many sub-activities underway. The order of the activities is not necessarily in which they are carried out. Some of the activities can be carried out in parallel.

[2] see www.dimac-imaging.com
[3] See www.mathworks.com

1.6 Performance calculation

When a classifier is built, the succes of it has to be measured. False-positive error, false-negative error and overall error will here be used. False-positive and false-negative error is defined in equation 1.2 and 1.3:

$$FN\% = 100\% \times \frac{FN}{TP + FN} \tag{1.2}$$

$$FP\% = 100\% \times \frac{FP}{TN + FP} \tag{1.3}$$

FN and FP indicates the number of cells wrongly classified as normal and abnormal, respectively. TN and TP is the number of cells truly classified as normal and abnormal, respectively. The overall error rate is given by equation 1.4:

$$OE\% = 100\% \times \frac{FN + FP}{ALL} \tag{1.4}$$

All is defined as $All = TP + FN + TN + FP$, and is the total number of cells. Normally the dataset is divided into a training and test set. The training set is used to build the classifier, and will therefore give a wrong-low error estimate. Instead the test set is used to estimate the error of the classifier. We then get a measure of how well the classifier has adapted to the underlying structure of the data, and not a measure of how well the data in the training set is modelled. But a problem arises, if the available dataset is limited. On one hand we want as much test data as possible to produce a good error estimate of the classifier, on the other hand we want as much training data as possible to build a good model. A solution for this could be K-fold cross-validation, described latter on.

Confidence interval

When estimating the average error of a classifier, the size of the test data is important. For example, if we obtain an error of zero wrong classified out of 50 samples of test data, we would calculate an error of 0%. But if instead just 1 sample had been classified as wrong, the error estimate would have been 2%.

Another example is in the project of Byriel (1999), where genfis2 classification results was obtained as shown in table 1.1. Standard, swap and random were different ways he had split the data up into training and test data. The question could arise, how precise the error estimates are. For false positive%, the results are 1%, 3% and 4%. From 1% to 4% is a big difference in error relative to the size of the error. It is obvious that the standard deviation of the errors is big in comparison to the size of the errors. The confidence in the results therefore is not very big.

	Standard	Swap	Random
False-negative%	0%	1%	1%
False-positive%	1%	3%	4%

Table 1.1: Classification errors for genfis2 from table 4.7 in Byriel (1999).

Therefore we have to take the amount of test data into consideration, when estimating how precise the calculated mean error is. This requires us to know the distribution of the test results. But the distribution of the error estimates is unknown. Maximizing the amount of test data therefore is a way to predict the mean error with a bigger confidence, even though the confidence level is unknown.

K-fold cross-validation

If the amount of data is small, we have to utilize it best possible. In the first place because the more test data available the more reliable an error estimate is obtained, secondly because the more training data used to build the classifier, the bigger the chance is that we have data representing all aspects of the model to build. k-fold cross-validation is a way to utilize the data best possible. The algorithm contains 2 steps:

1. Model building and testing
2. Cross-validation

Model building and testing

The idea is to partition the data in k groups of data. Then one of the groups is temporary hidden. The rest are used to build the classifier. The hidden data group is used as test data to validate the built classifier.

Cross-validation

Choosing one of k groups of data to leave out, can be done in k possible ways. Cross-validation runs the model building and testing on all k possible groups. The cross-validation error is the average of the k test error estimates.

The idea is that their should be as much training data as possible. The fraction $\frac{k-1}{k}$ of the total dataset is the amount of training data used to build the models. The bigger k is, the more data is used as training data. The upper limit for k is equal to the size of our data, and ends up being LOOCV (**L**eave **O**ne **O**ut **C**ross-**V**alidation) on the upper border. The lower limit is $k = 2$, because the data at least has to be split in two, a test set and a training set.

At the same time, we have a fraction of the data for testing given by $\frac{1}{k}$ for each built classifier. And since we build k classifiers, we end up having a test set of the same size as our total data set. As it can be seen, the size of k does not influence on the total test data size, but only on how much training data we have for each classifier. K-fold cross-validation is proportional computationally more expensive with $O(k)$, because there has to be build k classifier models.

Partitioning of data

When dividing the data up into k groups of data, this is normally done in total random. If this is not done in total random, but by dividing the data, so all kinds of data is split most possible over all k groups of data, better error estimates will quite likely be obtained.

For example, consider 4 classes: {boots}, {sandals}, {shoes} and {slippers}. If a 2-fold cross-validation is used, where one of the groups unfortunately contains all data samples belonging to the class {boots} and some data from the other classes. Then building the classifier on the group of data not containing the class {boots}, will make it hard for the classifier making a correct class prediction on the data containing the class {boots}, since no {boots} was used in the training data. The error estimate will therefore be to big, in comparison with the error a classifier build on all data can obtain. Therefore data should be spread as evenly as possible over all k groupings of data with k-fold cross-validation.

K-fold cross-validation rerunning

If the model building proces has some random steps, building a new model with the same data, will not necessarily give the same model, *i.e.* the error estimates will be different. Furthermore if k-fold cross-validation is not used as LOOCV, two consequent runs of the k-fold cross validation will divide the data differently into the k groups of data, *i.e.* the test data and training data will possibly give rise to different error estimates. For this reason, it could be a good idea rerunning the k-fold cross-validation several times.

This project will use k-fold cross-validation to estimate the error, and will try to distribute the different kinds of data as evenly as possible over all k groupings of data used with k-fold cross-validation. Furthermore, rerunning k-fold cross-validation several times is applied. Averaging of the errors is done for all results, to get an estimate of the testing error with a good confidence level.

1.7 The Papsmear Data

This thesis is based on single cell pictures of papsmear cells from the uterine cervix. Data collected in the thesis of Byriel (1999) is used, and is called the old data. Furthermore, a new single cell papsmear database created by cyto-technicians at Herlev Hospital, will here be presented, and referred to as the new data.

Data collection

The pictures are obtained by skilled cyto-technicians using a microscope connected to a frame grabber. The pictures are taken with a resolution of $0.201 \mu m/pixel$. The cyto-technicians have classified all the pictures in diagnostic categories. Every picture is classified by two different cyto-technicians, to be sure of the actual class. The pictures on which the cyto-technicians agree, are used.

Segmentation

Every picture was segmented into background, cytoplasm and nucleus, to make the later feature extraction possible. This was done by cyto-technicians using the CHAMP software. It uses a patented method where a 3D color space is build. This color space contains clusters belonging to nucleus, cytoplasm and background.

Data preparation

The work carried out by the cyto-technicians is kept in a paradox database from which the relevant data was extracted. The pictures were one by one examined to see if the segmentation was ok, and corrected if necessary and possible. A few pictures were removed from the new database in co-operation with a cyto-technician, because the removed cells had some inconsistencies.

Old data

The old database was prepared and made ready in the thesis of Byriel (1999). Therefore the previous mentioned steps was taken care of in that project. The old data contains 500 cells with the following distribution:

1. Normal - Columnar epithelial, 50 cells.
2. Normal - Parabasal squamous epithelial, 50 cells.
3. Normal - Intermediate squamous epithelial, 50 cells.
4. Normal - Superficial squamous epithelial, 50 cells.
5. Abnormal - Mild squamous non-keratinizing dysplasia, 100 cells.
6. Abnormal - Moderate squamous non-keratinizing dysplasia, 100 cells.
7. Abnormal - Severe squamous non-keratinizing dysplasia, 100 cells.

New data

The newly made database is prepared as mentioned previously. The new data contains 917 cells with the following distribution:

1. Normal - Superficial squamous epithelial, 74 cells.
2. Normal - Intermediate squamous epithelial, 70 cells
3. Normal - Columnar epithelial, 98 cells.
4. Abnormal - Mild squamous non-keratinizing dysplasia, 182 cells.
5. Abnormal - Moderate squamous non-keratinizing dysplasia, 146 cells.
6. Abnormal - Severe squamous non-keratinizing dysplasia, 197 cells.
7. Abnormal - Squamous cell carcinoma in situ intermediate, 150 cells.

Chapter 2

Pre-classification tasks

The features used in this project are presented and explained in the following. Feature extraction on these features is implemented in Matlab and compared to feature extraction with the commercial CHAMP software.

The features extracted, does not necessarily have to be good features, since feature selection is used to select the subsets of features performing the best. A short intro to feature selection and the method chosen is presented.

The clustering fuzzy C-means and Gustafson-kessel algorithms used in this project defines the clusters in the data. 4 Different defuzzification methods defining how to use the clusters when classification is wanted, are proposed.

Lastly, a simple algorithm for supervised clustering is proposed.

2.1 Feature extraction

Feature extraction deals with converting information to a format that is usable for the classifier algorithms. For ex. pictures cannot easily be feed directly into the classifier algorithms. Instead special characteristics are extracted from the pictures. This fase is quite crucial for the succes of the classification algorithm. Therefore when possible, experts in the actual field are used to identify the features.

Feature extraction is used to find features that possibly can help in the classification. It does not deal with removal of features that are unhelpful, this task is taking care of by feature selection, see section 2.2.

Chosen Features

In our case, we have to extract features from pictures. The same kind of features extracted in the project of Byriel (1999), are extracted here. The

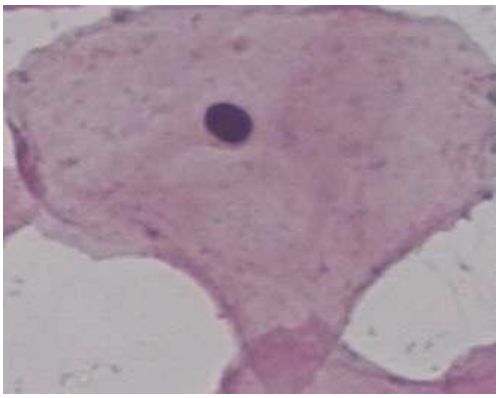

Figure 2.1: A normal superficial cell.

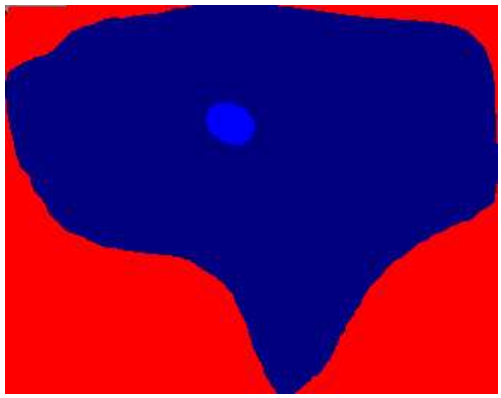

Figure 2.2: A normal superficial cell, segmented.

features are shown in table 2.1.

Nucleus area	Cytoplasma area	N/C ratio
Nucl. brightness	Cytopl. brightness	Nucl. shortest diam.
Nucl. longest diam.	Nucl. elongation	Nucl. roundness
Cytopl. shortest diam.	Cytopl. longest diam.	Cytopl. elongation
Cytopl. roundness	Nucl. perimeter	Cytopl. perimeter
Nucl. relative position	Maxima in Nucl.	Minima in Nucl.
Maxima in Cytopl.	Minima in Cytopl.	

Table 2.1: Features used in this project. Nucl., cytopl. and diam. are abbreviations of nucleus, cytoplasm and diameter, respectively.

In the following, the features from table 2.1 will be explained. The features are extracted by analyzing the original and segmented pictures. In figure 2.1 and 2.2, a normal superficial cell and the same cell segmented is shown. Some measurements, only use the segmented picture, and some use both. In the following, every feature is associated with a number in brackets, this number will later be used as an abbreviation instead of its name. There are 20 features in all:

Nucleus area(1) and Cytoplasm area(2):
Calculated by counting the corresponding pixels of the segmented picture. A pixels area is $(0.201\mu m)^2$.

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

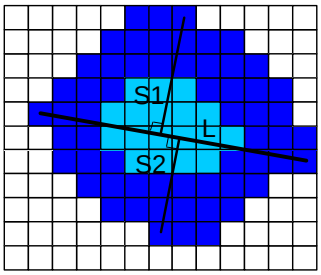

Figure 2.3: A binary picture of a cell with background(white), cytoplasm(dark blue) and nucleus(light blue). Longest diameter line(L) and shortest diameter lines(s1 & s2) is shown.

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 surrounding 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 2.3 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 encircled of the object. This distance is approximated by the sum of the two lines s1 & s2 shown in figur 2.3 for the cytoplasm. They are perpendicular to the line L, and is the longest line to each side fitting inside the object. The names of the features is 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 bound by 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 centrum (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.

This project has extracted features from the pictures in to ways. CHAMP is used as in the previous project of Byriel (1999). The second way, is by implementing the feature extraction in Matlab. A reason to use Matlab, was because the code can be made freely available, which is not the case with CHAMP since it is a commercial program. Other people can add new features or change the way features are extracted with the Matlab implementation, to get better features. Furthermore many of the operations that shall be performed, can be made relatively simple with the Matlab image toolbox. The feature extraction with Matlab is implemented by several M-files. These M-files can be seen on the accompanied cd-rom in appendix B.

In Landwehr (2001) and in Walker, Jackway, Lovell & Longstaff (1994), some new features was proposed. These features use information of the texture in the picture, by using an invariant GLCM approach. GLCM is an abbreviation of 'Gray Level Co-occurrence Matrix'. After feature selection, some of these new features seemed to give information that could be used. In the paper of Lin & Fu (1983), many other features was used with good results. These features and others could be added later on top of the feature extraction code made for Matlab in this project.

Feature extraction results

Features for the old data was extracted with CHAMP in Byriel (1999). Features from the new database was extracted with Matlab and CHAMP in this thesis. The features was compared, to verify the extracted Matlab features. Comparison of extracted features, were done with correlation, by using equation 2.1:

$$corr = \frac{\sum_{i=1}^{K}(x_i - m_1)(y_i - m_2)}{\sqrt{\sum_{i=1}^{K}(x_i - m_1)^2 \sum_{i=1}^{K}(y_i - m_2)^2}} \tag{2.1}$$

Here K is the number of data, x and y are the features to compare. m_1 and m_2 are the mean of all x and y, respectively. If $|corr|$ is close to 1, we have a high correlation between the features.

The correlated features are seen in table 2.2. The correlation is very high for most features. Cytoplasm elongation, cytoplasm roundness and nucleus relative position correlates worst.

When a cells nucleus is big enough, it can divide the cytoplasm into 2 parts, as shown in fig 2.4. CHAMP would analyse it, as if there were 2 cytoplasma, since they are not connected. All CHAMP cytoplasm measurements, except the cytoplasm with the biggest area, was thrown away. Unfortunately, the cytoplasm areas sorted away will be of some distance from the rest, and able to move the cytoplasm center of mass considerably. The Matlab feature extraction was made differently. All cytoplasm with borders to the same nucleus, are considered as one cytoplasm. This could account for the low correlation of some features.

Feature	**Cross-correlation value**
Nucleus area	0.9999
Cytoplasma area	0.9999
N/C ratio	0.9962
Nucleus brightness	1.0000
Cytoplasma brightness	0.9993
Nucleus shortest diameter	0.9921
Nucleus longest diameter	0.9939
Nucleus elongation	0.9462
Nucleus roundness	0.9283
Cytoplasm shortest diameter	0.9765
Cytoplasm longest diameter	0.9732
Cytoplasm elongation	0.7641
Cytoplasm roundness	0.8981
Nucleus perimeter	0.9972
Cytoplasm perimeter	0.9552
Nucleus relative position	0.6801
Maxima in Nucleus	0.9444
Minima in Nucleus	0.9445
Maxima in cytoplasm	0.9877
Minima in cytoplasm	0.9899

Table 2.2: Feature correlations between CHAMP and Matlab features, is shown.

In fig 2.5 the *nucleus area* from CHAMP and Matlab feature extraction is

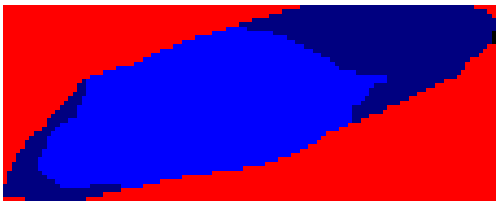

Figure 2.4: A Carcinoma in situ intermediate segmented cell. The colors are red,dark blue and light blue for background,cytoplasma and nucleus respectively.

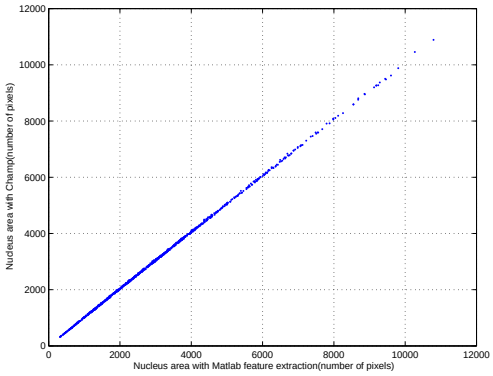

Figure 2.5: Nucleus area for CHAMP and Matlab is plottet against each other.

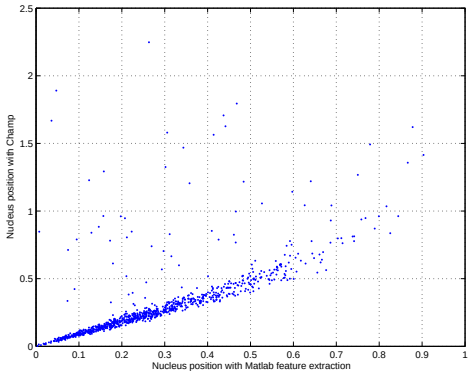

Figure 2.6: Nucleus relative position for CHAMP and Matlab is plottet against each other.

plottet against each other. We get a nice line, as expected because of the high correlation. In fig 2.6, the worst correlated feature, *Nucleus relative position*, is plottet from the CHAMP and Matlab feature extraction. Even though a poor correlation is obtained, we get a nice line. A few data points lie quite some distance to the line, and is the reason of the poor correlation. These few data points, are possibly caused by the difference in how champ and Matlab find the belonging cytoplasm.

Conclusion on feature extraction

The features extracted with CHAMP and Matlab correlate quite well, and there should therefore not be any great difference between them. The features extracted with Matlab are therefore used as the data for the new database.

2.2 Feature Selection

Feature selection is used to select subsets of features that give the best classification results. Some features can have some noise, others maybe don't help classification at all, and others maybe can't be utilized by the chosen classifier. Maybe even the available number of data samples are too little to justify the amount of features with a particular classifier.

The question could arise, of why it is necessary to use feature selection. A perfect classification algorithm would be able to diminish the effect of poor features. But the perfect classification algorithm does not exist. Some classification algorithms, have a number of parameters that are dependent on the number of features. In some cases the available dataset is quite limited and therefore it can be necessary to limit the amount of parameters that has to be found for the classification algorithm and thereby limiting the amount of features.

One way to select the right features, is by trying all combinations. When we have more than just a few features, the possible combinations explode in number. One approach to find the features is simulated annealing in combination with a wrapper approach, which is the method chosen.

Filters

Filters are often used to remove useless features with cross correlation as their fitness or maybe even with the average common information(Entropy measure). This approach is suitable if there exists too many features, and the worst has to be removed. Filters do not take into consideration which features, a chosen classifier algorithm performs best with, and which the classifier is not able to utilize. If using cross correlation as a fitness measure, only linear dependencies are found. Therefore useful non-linear features could risk being removed.

Wrappers

Wrappers use the actual classification algorithm that is going to be used in the classification, to estimate the error with the different feature combinations. The error is the fitness given by the wrapper. Since no classification algorithms are perfect, they each have their own up- and down-sides. By using the wrapper approach, a fitness is obtained that better reflects the performance of the features on the actual classifier algorithm.

Forward and Backward selection

When searching for feature combinations that work well together, forward and backward selection can be used. Forward selection starts by using a few features, and adding features that give better error estimates. Backward selection starts with all the features as a starting point and removes poor features. Backward selection has the advantage, that it will be better at finding combinations of features that work well together, but each by themselves does not work very well. Forward selection adds features, and when adding one feature at a time, features that only work well together will have a smaller chance of getting selected.

Simulated annealing

One way of searching for feature combinations, is using simulated annealing. Simulated annealing is a stochastic search method, that is guided by a fitness value. In our case the estimated error. When a new subset of features is tested, the better the estimated error is than before the step, the bigger the chance is the new state is kept. This chance is controlled by a parameter called the 'temperature'. The bigger it is, the greater the chance is accepting a new state with a lower fitness value.

Simulated annnealing is especially suited not to get trapped by a local minimum. It expects the estimated error rates to hold to some degree, else the algorithm can be caught in something that is not a minimum, but just a wrong low estimate of the error. Therefore the variance in the error calculation has to be minimized. Simulated annealing in this project will use all features in the start fase and from here remove un-useful features. This corresponds to backward selection. Every time a new subset of features is accepted, it is saved for later use. When simulated annealing is finished, all the different subsets of features accepted are saved, their fitness are then compared to find the one that performs the best. Simulated annealing, needs a fitness value to guide its search, which is obtained with a wrapper where K-fold cross-validation is used to calculate the error on the classification algorithm. The simulated annealing algorithm is shown in table 2.3.

```
1   Begin simulated annealing
2     T=chosen annealing temperature(typically around 0.2).
3     N=number of iterations
4     F=All features(Backward selection)
5     Actual_error=Error(F)
6     For N=1 to Max_iterations
7       N=N-1
8       Randomly chose to remove or add a feature to F.
9       New_error = Error(F)
10      If (e^((Actual_error−New_error)/T)) > rand[0..1]
11        Accept and keep the new features in F.
12        Actual_error = New_error
13      Else
14        Keep the old features.
15      End if
16    End for
17  End
```

Table 2.3: The simulated annealing algorithm used in this project for feature selection.

2.3 Defuzzification methods

Clustering algorithms like C-means and Gustafson Kessel clustering, finds a number of clusters, from the structure of the data. They do not tell what information the clusters contains, or how the information shall be used. This are practical matters, that has to be considered when classifying with the chosen clustering algorithms.

The clustering algorithms define how data points are given memberships to the different clusters. This possible fuzzy membership should be used to predict the class of a data point, but how it shall be used together with the information contained in the clusters, is not given.

4 different defuzzification methods will here be proposed and later investigated for both C-means and Gustafson Kessel. They will describe how to calculate what information a cluster contains, and how data points membership to several clusters are dealt with, when deciding on their class. The methods will be referred to as defuzzification methods, since they describe how to use a data points fuzzy membership to the clusters, and how to use clusters possible fuzzy membership to the different classes, when deciding on the data points crisp class.

Defuzzification method 1

This method is the one most often used, and is the most simple method. All training data are assigned the cluster nearest it. The class most often occurring in the training data assigned to a cluster, becomes the clusters class value. The cluster gets a total membership to one class, and zero to the rest for the training data. This information is kept in a vector $clusterU_i$ for cluster i, containing memberships for all classes. Index k in this vector will point to the membership of class k. The membership vectors from all clusters are joined into a 2D matrix called $clusterU$:

$$clusterU = \begin{pmatrix} clusterU_1 \\ clusterU_2 \\ \vdots \\ clusterU_c \end{pmatrix} \tag{2.2}$$

Test data is given the class value, the cluster nearest it has.

Defuzzification method 2

This defuzzification method uses a data points fuzzy membership to the different clusters for weighting the different clusters crisp class memberships.

The clusters are assigned class values with the training data, the same way as for defuzzification method 1, and stored in $clusterU$. The difference here, is how the class of the test data are found. Equation 2.3 calculates the membership U_{class} to the different classes from test data point k:

$$U_{class} = U_k \times clusterU \tag{2.3}$$

The product is ordinary matrix multiplication. U_k is the membership vector of data point k to the different clusters. The index in vector U_{class} for which it is biggest, determines the class of test data point k.

Defuzzification method 3

Until now, the clusters have only belonged to one class. Now we let each cluster have a fuzzy membership to all of the classes. Therefore the memberships in $clusterU_i$ for cluster i can be different from 0 or 1. But, still the total membership to all classes should sum up to 1.

Training data are assigned the cluster nearest it. The percentage of training data of each class belonging to cluster i, gives the clusters fuzzy membership to the different classes. F.ex if we have the classes 1 and 2, and the cluster i contains 20% training data of class 1 and 80% of class 2, then the memberships

should be given by $clusterU_i = [0.2, 0.8]$.
Test data are classified like in defuzzification method 2 with equation 2.3, but now with a fuzzy membership matrix $clusterU$.

Defuzzification method 4

The last defuzzification method presented here, will add a density measure. Clusters for which many training data points are assigned, will get a bigger influence than clusters with a small number of assigned training points. Clusters made up of relatively few training data, will have a bigger chance of overfitting the training data, than clusters with more data. So, by adding this density measure, overfitting will less likely occur. The density measure will minimize the influence of clusters that possibly overfit the data, which is the clusters with few data points. Unfortunately, clusters made up of only a few non-noisy data points will also have its influence minimized, even though these clusters can be just as good as the bigger ones.

Data belonging equally between a big and little cluster, will have a greater probability of belonging to the larger cluster than the little cluster, and is another reason the density measure could be a way to improve on the classification error. The density measure could improve the error rate, by assigning more of the data from the boundary to the bigger cluster.

Defuzzification method 4, is almost like defuzzification method 3. The only difference, is how the vector $clusterU_i$ for cluster i is made. In defuzzification method 3, $clusterU_i$ contains the percentage of training data with the different classes. Now, instead a frequency count of the training data is used. F.ex if we have the classes 1 and 2, and the cluster i contains 20 pieces training data of class 1 and 80 of class 2, then the memberships should be given by $clusterU_i = [20, 80]$. It should be noted, that the sum of $clusterU_i$ shall not sum up to 1, but to the number of training data in the actual cluster. Therefore the more training data associated with a cluster, the bigger an influence $clusterU_i$ will have.

The optimality of using this density measure, depends on the data. If the data naturally has many small clusters, this defuzzification method will not necessarily give a good result, since their effect are minimized. If overfitting is a problem, the results maybe can be improved by this method. If many data points is on the border between two clusters, the density measure can help minimize the error, by favoring the bigger cluster from a statistical point of view.

2.4 Supervised clustering

A method is here proposed that in this thesis will be called supervised clustering. It helps finding better cluster centers, if the natural clusters are not nicely separated because some overlap of the classes occur in most of the features.

Supervised clustering is sometimes referred to when an expert somehow guides the clustering proces, f.ex like saying how many clusters to use. In Duda, Hart & Stork (2000), supervised clustering is defined to be when the diagnose of the training data are used in the clustering proces. In this thesis, the definition of supervised and unsupervised clustering, will be as in Duda et al. (2000). Using this definition, C-means and Gustafson kessel clustering normally is unsupervised, and is here named like that. The new method here proposed, uses the diagnoses of the training data to find the clusters and will here be called supervised clustering.

C-means and Gustafson-kessel clustering normally works, by finding natural clusters in the data. The algorithms does not utilize the diagnoses of the training data. The clusters are found, assuming the classes are naturally separated in such clusters. If the different classes are overlapping in its features, then the natural clusters are not necessarily found with good cluster centers., this resulting in a poorer classification. The proposed procedure should find better centers by utilizing the diagnoses of the training data.

The proposed procedure, does not change how C-means and Gustafson-kessel clustering works. It is merely a simple way to use the algorithms and there results, to find better cluster centers. The procedure is the following:

1. First the training data are divided up into its diagnoses.
2. Then clusters are found for each diagnose separately, with the chosen clustering algorithm.
3. Afterwards all the found cluster centers from every diagnose are merged into one model.

Step 1 is the supervised step, since the training data are split into groups by their diagnose. Step 2 builds a clustering model for each of these groups of training data, and therefore finds cluster centers specific for every diagnose. Merging the clusters in step 3 is simply done by collecting all calculated cluster centers, and using them for the new merged model. GK clustering furthermore has a mahalanobian distance matrix associated to every cluster. This distance matrix has to follow the clusters after the merging. When the found clusters are merged into one model, they are treated as an ordinary model build by FCM or GK.

If normal and abnormal cells are quite overlapping in their features, the clusters should be found much better with this proposed method. Since the

clusters are fitted on a single diagnose one at a time, the clustering algorithms are able to fit the clusters more precise on the actual diagnose without being disturbed of data with other diagnoses. If the data are well separated in natural clusters, this supervised procedure probably is not helpful. But the error for well separated clusters should be quite low. So, when supervised clustering does not help is when the error already is quite optimal, but the errors obtained with supervised clustering should not be worse.
Supervised clustering is only applicable when the diagnoses of the training data are known before hand, which is the case in this project. Results will in later chapters be obtained for both unsupervised(the normal procedure) and supervised clustering.

Chapter 3

C-means clustering

The C-means algorithm is a clustering algorithm, that finds natural clusters in the data. C-means clustering has relatively few parameters to tune, and therefore it can be an attractive method to use. Furthermore the used data, has no limit on how many dimensions it can have, *i.e.* the number of features. But the downfall is, that the method will often give a worse result, because it is not good at utilizing and modelling its data. There exists two c-means algorithms: hard c-means and fuzzy c-means.

3.1 Hard C-means Theory

Hard c-means(HCM) starts out, by randomly choosing center positions for all clusters. All data are then assigned the cluster center nearest with the membership value 1, and a membership of 0 to all others. The Euclidean distance measure is used. New cluster centers are found by calculating the average position of the data points belonging to the cluster, as in equation 3.1, where $\mathbf{c}_i$ is the center of cluster i, C_i is the set of all clusters, and $\mathbf{u}_k$ is the membership of the data point k.

$$\mathbf{c}_i = \frac{1}{|C_i|} \sum_{k, \mathbf{u}_k \epsilon C_i} \mathbf{u}_k \tag{3.1}$$

Afterwards the data is again assigned the cluster center nearest it, and new cluster centers are again calculated. This process is iterative, until an objective function is below some threshold or the number of iterations exceeds some limit. The objective function is normally given by the sum of distances from all the cluster centers to their respective data points, as in equation 3.2. J is

the objective value and c is the number of cluster centers.

$$J = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} (\sum_{k,\mathbf{u_k} \epsilon C_i} \|\mathbf{u}_k - \mathbf{c}_i\|) \tag{3.2}$$

This objective function is minimized by iteratively finding the cluster centers. The possible minimum reachable by training, can be highly dependent on the random placement of the centers in the start. Every time new cluster centers are calculated, the membership for each data point can shift abruptly to another cluster. This is one of the reasons that HCM is not as god at finding optimal cluster centers. This problem is tackled with fuzzy c-means.

3.2 Fuzzy C-means Theory

Fuzzy c-means(FCM) modifies HCM, by allowing the data points belong to all of the clusters, with memberships in the interval [0..1]. The total membership to all clusters from a single data point, should be equal to 1. The memberships are calculated by equation 3.3.

$$m_{ik} = \frac{1}{\sum_{j=1}^{c} (\frac{d_{ik}}{d_{jk}})^{2/(q-1)}} \tag{3.3}$$

m_{ik} is the membership for data point k to cluster center i. K is the number of data points. d_{jk} is the distance from cluster center j to data point k. $q \epsilon]1..\infty[$, and is the exponent that decides how crisp the memberships should be. If $q \approx 1$, the membership will be crisp, almost like HCM, and $q \to \infty$ results in equal membership to all clusters, no matter the clusters position.
New cluster centers are calculated with these fuzzy memberships in equation 3.4.

$$\mathbf{c}_i = \frac{\sum_{k=1}^{K} m_{ik}^q \mathbf{u}_k}{\sum_{k=1}^{K} m_{ik}^q} \tag{3.4}$$

FCM is much less dependent on a good placement of the cluster centers when it starts, if an appropriate value for q is chosen. This is, because the clusters position is determined by all data points, but with an influence determined by their fuzzy membership. Therefore data points will not abruptly shift cluster. But still the random placement of cluster centers to start with, can have an influence on how optimal the final cluster centers are placed.

The objective value of how well the cluster centers are placed, is obtained from equation 3.5. J_i is the objective value for cluster i.

$$J = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \sum_{k=1}^{K} m_{ik}^q d_{ik}^2 \tag{3.5}$$

Just like HCM found the cluster centers iteratively, until the objective function was below som threshold or until the number of iterations exceeded some limit, likewise does FCM.

C-means Parameters

FCM has the parameter q, that decides the crispness of the memberships. When $q \approx 1$, FCM is quite like HCM, with a couple of exceptions. HCM can maksimal have as many clusters as data points. This limit is not in FCM. The other difference, is in calculating the memberships. In HCM, a data point can only belong to one cluster, even if the distance is equal to the 2 closest clusters. FCM will in this case let the data belong equally to both clusters. These subtle differences will in practice not separate these methods from each other, but give equal results for $q \approx 1$.

Both hard and FCM has to decide how many cluster centers to use. This is an important parameter, because too many clusters can result in overfitting the model to the noise in the data. The training of the cluster centers stop, when the objective function is below some threshold or when the limit of iterations are passed. These shall be chosen so the cluster centers have converged to a solution.

3.3 Fuzzy C-means implementation

The implementation of FCM, is available in the fuzzy toolbox in Matlab. Matlab is a programming language especially suited for implementation of technical computing.

It became clear, the algorithm delivered with Matlab was not very robust, meaning it did not always give the results which was expected. The algorithm would not work when stressing the algorithm, either by decreasing q close to 1 or having to many cluster centers in comparison with the number of data points. When the algorithm failed, it would return NaN's for the cluster centers and membership values. NaN is an abbreviation of **N**ot **A** **N**umber. For example the division $\frac{0}{0} = NaN$, since the result is undefined. A simple example has been made, that stresses the algorithm, so that the returned cluster centers are full of NaN's. It can be seen in the Matlab file '*fcm_problem.m*' on the accompanied cd-rom.

Since the algorithm was to be used to see the difference between FCM and HCM, and for stability reasons, Matlab's algorithm is modified. A step by step modification will now be explained, to document the functionality for future use.

Matlab fuzzy C-means algorithm

First of, the problem has to be identified.
The problem is found to happen when q is close to 1, or when the number of clusters are raised. To understand the problem, equation 3.3 is rewritten to 3.6.

$$m_{ik} = \frac{(d_{ik})^{-2/(q-1)}}{\sum_{j=1}^{c}(d_{jk})^{-2/(q-1)}} \tag{3.6}$$

This equation is more how the membership values are calculated in the Matlab algorithm. This equation can run much faster in Matlab, since the denominator only has to be calculated once for each cluster, and only one division has to be made for every membership value.

The problem with this equation is, that the exponentiation now is done before the division. Since the computer only has a limited accuracy, the expression $(d_{ik})^{-2/(q-1)}$, will give really big values, if $d_{ik} \ll 1$ and $-2/(q-1) \ll -1$. This is the case when q is close to 1 and the number of cluster centers increase, and therefore some data points will get yet smallere distances to some cluster centers. At some time, the values will be too big for Matlab to perform calculations on, and will get the value ∞(remember it is a negative exponentiation).

The value in the nominator of equation 3.6, will also appear in the summation in the denominator. Since all the values are positive, the summation will at least be as big as the nominator. Therefore, if the value ∞ is calculated in the nominator, the denominator will definitely not be smaller than ∞. This resulting in a membership given by $\frac{\infty}{\infty} = NaN$ for Matlab. Since all results are dependent on all the other results, a single iteration of the algorithm will propagate NaN's to all cluster centers and membership values.

This problem is solved, by scaling all of the distances from the data points to the actual cluster, so that the smallest distance becomes 1. In the case of the smallest distance being 0 (when a cluster center and data point have exactly the same coordinates), then any distance of 0 gets the value of 1 and all other distances gets the value ∞.

By scaling the values like this, the shortest distance, will always be 1 after the exponentiation, except in one case. When $q = 1$, we would get an exponent of $-\infty$. Matlab says $1^{-\infty}$ should give NaN. But to get the FCM algorithm to perform as intended, the result in the algorithm is made to give 1. Even though $q = 1$ is not defined for the C-means algorithm, it is defined for exponents very close to 1, that also can give $-\infty$ for the exponentian, since Matlab is limited in its accuracy. The reason to scale the values like this, is to make the implementation more robust to Matlab's limited precision. So when the precision becomes an issue, the results will still give expected results.

Another problem is when the membership values for the data points passed to the FCM algorithm does not have any membership at all to one or more of the cluster centers, even though FCM dictates it should have a membership,

maybe infinitesimal, but still not 0 in total from all data points. If the total membership to a particular cluster is 0, then it is impossible to calculate any new cluster center for it, since the clusters are determined by equation 3.4, and therefore by the membership values. A solution has been chosen, where 3 random data points are assigned a little membership to the corresponding cluster, but only if no data points has any membership to it. Now it will be possible to calculate a cluster center.

By solving the problems as described, we get an algorithm that is more robust to the precision errors in Matlab's calculations. The modified algorithm has four m-files against two before the changes. The files are '*fcm_mod.m*', '*stepfcm_mod.m*', '*calc_cent.m*' and '*calc_U.m*'. The last two files are the ones added, and are used to simplify the reading of '*stepfcm_mod.m*'. Hardly any changes are made in '*fcm_mod.m*', but most in the other files. The files can be found on the accompanied cd.

C-means results

The Hard & Fuzzy C-means algorithm will here be used to classify the 'old' and 'new' data presented earlier in chapter 1.7. An appropriate defuzzification method will be chosen as the first step. Classification results with and without feature selection will afterwards be obtained for unsupervised and supervised clustering.

Error estimates will be made with k-fold cross-validation by partitioning the data with care and rerunning k-fold cross-validation several times, so better error estimates are obtained(se section 1.6).

3.1 Defuzzification results

The 4 mentioned defuzzification methods in section 2.3, will here be tested, and compared on the old and new data. One defuzzification method will be chosen to be used with the FCM algorithm. This comparison will use unsupervised clustering.

To start with, the error will be calculated and plottet against the fuzziness exponent for each of the defuzzication methods. The error is measured with a 2-fold cross-validation with 20 reruns. This is done for 5,10 and 20 clusters, and all features are used. Figure 3.1 - 3.6 shows the results.

Defuzzification method 1 is the most common used defuzzification method. It is the method that in these figures seem to give the smallest error when the worst exponent is chosen, but not the smallest error for an optimal exponent. The other methods seem to perform better, but at the same time being more dependent on an optimal fuzzy exponent. The results for both the old and the new data supports this.

Defuzzification method 2 seems to have the most unstable error prediction of all methods. If a good error calculation shall be found it could be a poor method to chose. Later on feature selection is performed with simulated annealing, where a good error prediction helps in the search of a subset of good features. Therefore a stable error prediction is preferred. Defuzzification method 4, seems to be the one with the most stable error prediction.

In the figures it can be hard distinguishing between the graphs for the methods, so if interested a colored version of this report can be seen on the accompanied cd-rom in pdf format.

To being able to choose the best method, a near optimal fuzzy exponent is chosen for each method for a series of cluster count. These values can be seen in table 3.1. They are chosen from the shown figures, where a global mini-

	Method 1	**Method** 2	**Method** 3	**Method** 4
5 **clusters, old data**	$q = 1.60$	$q = 1.54$	$q = 1.40$	$q = 1.40$
10 **clusters, old data**	$q = 1.40$	$q = 1.45$	$q = 1.33$	$q = 1.30$
20 **clusters, old data**	$q = 1.53$	$q = 1.27$	$q = 1.32$	$q = 1.27$
5 **clusters, new data**	$q = 2.20$	$q = 1.00$	$q = 1.16$	$q = 1.10$
10 **clusters, new data**	$q = 1.00$	$q = 1.10$	$q = 1.10$	$q = 1.10$
20 **clusters, new data**	$q = 1.25$	$q = 1.15$	$q = 1.12$	$q = 1.10$

Table 3.1: Found fuzzy exponents q for the 4 different defuzzification methods for the old and new data.

mum seemed to occur. With these fuzzy exponents, the classification results were obtained, and are shown in table 3.2. The errors are calculated with a

	Method 1	**Method** 2	**Method** 3	**Method** 4
5 **clusters, old data**	10.57%	9.95%	9.76%	9.71%
10 **clusters, old data**	8.48%	7.44%	7.08%	6.97%
20 **clusters, old data**	6.20%	6.41%	5.69%	5.52%
5 **clusters, new data**	11.17%	11.22%	11.11%	11.16%
10 **clusters, new data**	11.05%	10.83%	10.83%	10.70%
20 **clusters, new data**	10.12%	9.84%	9.75%	9.79%

Table 3.2: The table shows the calculated errors for the different methods together with chosen cluster counts. The optimal exponents shown in table 3.1 is used.

2-fold cross-validation with 40 reruns. Results for defuzzification method 1 are quite a bit worse than method 2,3 and 4, both for the old and the new data. Method 2 gives worse results for the old data than method 3 and 4, and as mentioned earlier, its error calculation is the most unstable. Method 3 and 4 is quite equal, but a little better error is achieved with method 4. As mentioned earlier, method 4 seemed to give the most stable error calculation of all the methods.

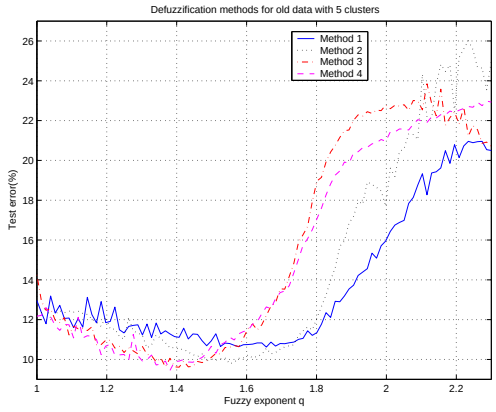

Figure 3.1: Exponent dependency on old data is shown. Defuzzification methods tested with 5 clusters.

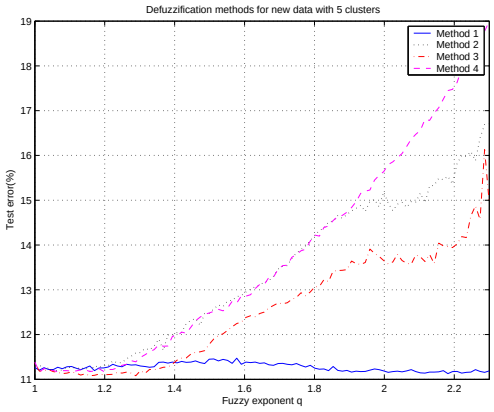

Figure 3.2: Exponent dependency on new data is shown. Defuzzification methods tested with 5 clusters.

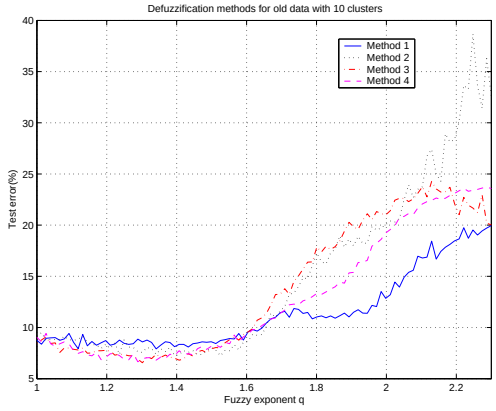

Figure 3.3: Exponent dependency on old data is shown. Defuzzification methods tested with 10 clusters.

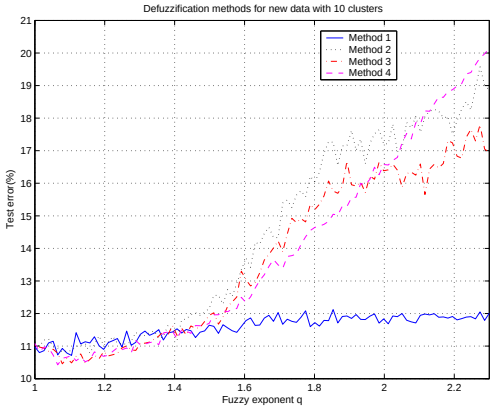

Figure 3.4: Exponent dependency on new data is shown. Defuzzification methods tested with 10 clusters.

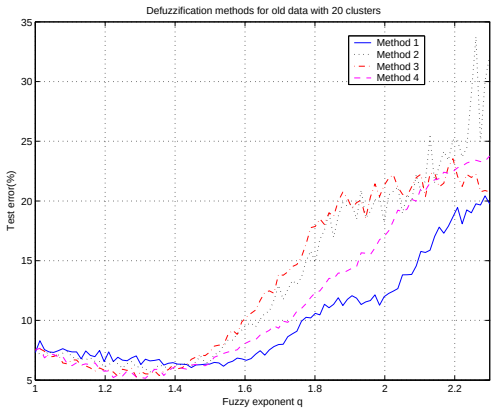

Figure 3.5: Exponent dependency on old data is shown. Defuzzification methods tested with 20 clusters.

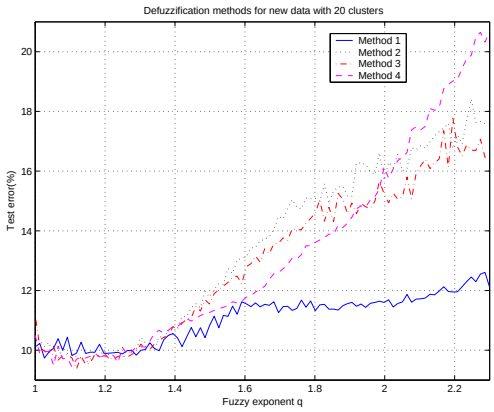

Figure 3.6: Exponent dependency on new data is shown. Defuzzification methods tested with 20 clusters.

Defuzzification Conclusion

For poor fuzzy exponents, method 1 is the best performing, but the worst for optimal exponents. It is a good choice, if no effort is going to be used in finding an optimal fuzzy exponent.

A lower error was calculated, using the fuzzy membership of data points to the clusters for method 2 compared to method 1, but an unstable classification error seemed to be a problem for it. By letting the clusters have a fuzzy membership to each class, a further decrease in the error was observed by method 3. By adding the density measure in method 4, a little smallere classification error is obtained. The defuzzification method to use, therefore is a choice between method 3 and 4. Since method 4 has a more stable error calculation in general and a little smallere error, defuzzification method 4 is chosen as the method to use with FCM from now on.

3.2 Hard C-means results

Results for supervised and unsupervised HCM will here be presented first for the old data, and afterwards for the new data. Results are obtained, by using the FCM algorithm, with the fuzzy exponent $q \approx 1$ since this in practice should give results equal to HCM.

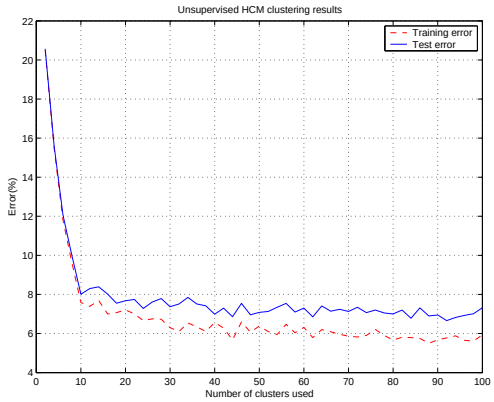

Figure 3.7: Unsupervised clustering error plottet against number of clusters.

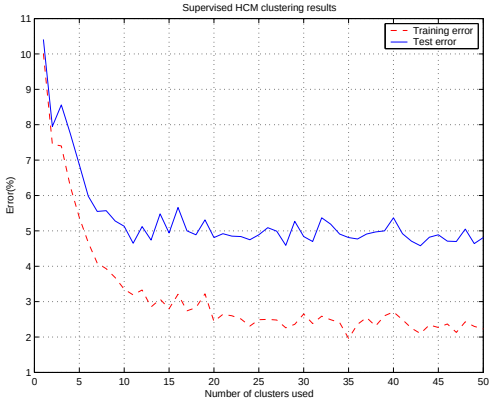

Figure 3.8: Supervised clustering error plottet against number of clusters.

Hard C-means on Old data

The number of clusters to use, was to be determined. This was done by measuring the error with 2-fold cross-validation with 20 reruns, for a series of different number of clusters. All features were used. The results for unsupervised clustering is seen in figure 3.7 and for supervised in figure 3.8. Both figures indicate, that using more clusters give a lower test error.

Since we know how many clusters to use, we can calculate the error for HCM. A 10-fold cross validation and 20 reruns is used to estimate the error. No feature selection is yet performed. 50 and 100 clusters is used for supervised and unsupervised clustering, respectively. The results is shown in table 3.3. The supervised clustering seems to give far the best results compared to

	Overall error	**FP%**	**FN%**
Unsupervised classification	5.99%	6.90%	5.38%
Supervised classification	4.03%	4.95%	3.42%

Table 3.3: Results for unsupervised and supervised HCM clustering with all features.

unsupervised clustering, and just barely acceptable with the 5% error limit. The unsupervised clustering errors are all above 5% and are not acceptable. To improve on the error, feature selection is performed with simulated annealing. The features found, is those shown in table 3.4. The chosen features, gives the error shown in table 3.5. Again 50 and 100 clusters for supervised and unsupervised clustering is used, respectively. A 10-fold cross validation and 20 reruns is used to estimate the error.

	Selected features
Unsupervised classification	1,3,4,5,6,7,10,11,12,16,18,20
Supervised classification	1,3,4,5,6,7,11,12,13,16,18

Table 3.4: Selected features for unsupervised and supervised clustering. Every number corresponds to a feature, se chapter 2.1.

	Overall error	FP%	FN%
Unsupervised classification	3.88%	4.17%	3.68%
Supervised classification	2.62%	2.80%	2.50%

Table 3.5: Results for unsupervised and supervised HCM after feature selection.

Now the error is below the wanted 5% for both the supervised and unsupervised method. The supervised clustering error is still much better than the unsupervised clustering error.

Hard C-means on New data

Again the number of clusters is to be determined, and is done in the same way as for the old data. The results can be seen in figure 3.9 and 3.10. Both figures indicate, that the more clusters used, the lower the test error seems to become. The error is calculated with 10-fold cross-validation and 20 reruns. 50 and 100 clusters is used for supervised and unsupervised clustering, respectively. The results is shown in table 3.6. The supervised clustering again gives a much

	Overall error	FP%	FN%
Unsupervised classification	9.71%	25.67%	4.00%
Supervised classification	8.77%	18.81%	5.24%

Table 3.6: Results for unsupervised and supervised HCM clustering with all features.

better result than unsupervised clustering. But both methods is a long way from 5% error, which is the goal. To improve on the error, feature selection is performed with simulated annealing. The subset of features found is shown in table 3.4.

The chosen features, gives the error shown in table 3.8. The error is again calculated with 10-fold cross-validation and 20 reruns. A lowering of the error

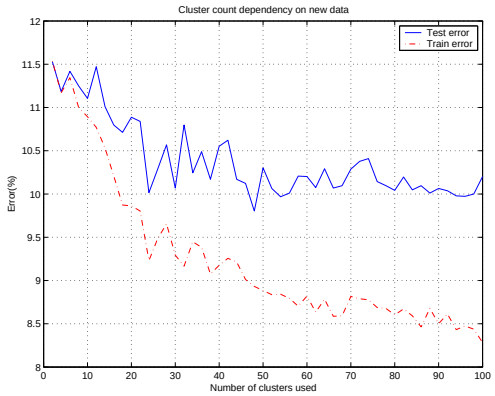

Figure 3.9: Unsupervised clustering error plottet against number of clusters.

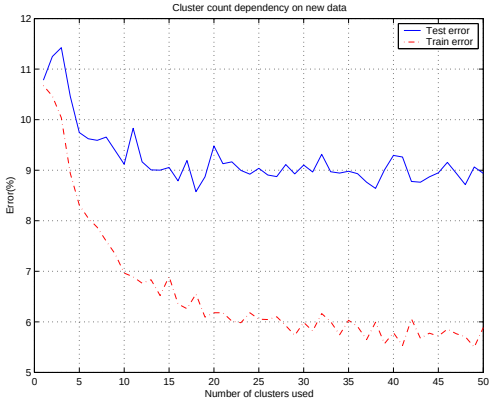

Figure 3.10: Supervised clustering error plottet against number of clusters.

	Selected features
Unsupervised classification	1,3,6,7,11,14,15,18,20
Supervised classification	1,2,3,4,5,6,7,8,11,12,14,15,16,17,18,19,20

Table 3.7: Selected features for unsupervised and supervised clustering. Every number corresponds to a feature, se chapter 2.1.

with around 1% is obtained after feature selection, but the error is still not below 5%, and therefore not acceptable.

	Overall error	FP%	FN%
Unsupervised classification	8.28%	21.81%	3.41%
Supervised classification	7.86%	17.12%	4.55%

Table 3.8: Results for unsupervised and supervised HCM after feature selection.

Hard C-means conclusion

It is shown that acceptable results can be obtained with HCM clustering for the old data. The error with the new data is way too high, and therefore shows HCM is not suitable for an error below 5% with the new data. The supervised clustering procedure, performs much better than unsupervised clustering for nearly all results. This could indicate the clusters are not nicely separable in natural clusters. Supervised clustering allows clusters to adapt to data of the same class, without interference from data of other classes. This is not the

case for unsupervised clustering.
It is interesting that more clusters used both for supervised and unsupervised clustering, gives a smaller test error result. The reason could be several clusters complement each other to form more complex clusters in the data.

3.3 Fuzzy C-means results

First results for the old data and afterwards the new data will be presented. Classification results for both unsupervised and supervised clustering is found. Near optimal parameters like the number of clusters and fuzzy exponent is determined and used. Afterwards classification results is obtained with the subset of better performing features found with feature selection.

Fuzzy C-means with Old data

First the influence of the fuzzy exponent and the number of clusters are determined. Afterwards, feature selection is performed to get a better classification error.

Fuzzy C-means exponent

A close to optimal exponent q is to be chosen. Therefore the cross-validation error is estimated for a series of different values of the exponent q. The errors are found by 2-fold cross-validation with 20 reruns. The findings are shown in figure 3.11 and 3.12. From the figures, an optimal exponent for unsupervised clustering is determined to be $q = 1.25$ and $q = 1.20$ for supervised clustering, since minimum error was obtained for these values. It is noticed, that if FCM is used with the exponent $q = 2$, which usually is seen as an acceptable value to use, results much worse than HCM will be achieved.

Number of clusters

The number of clusters to use is now determined. This is necessary, so the built model can describe the data best possible. If too many clusters is chosen, we can risk overfitting the noise in the data. To few clusters, and a poor classifier will be achieved. Therefore an analysis of the number of clusters against the cross-validation test error has been made. Figures 3.13 and 3.14 shows the results of this. The errors were found by 2-fold cross-validation with 20 reruns.

Both unsupervised and supervised clustering seems to give better results, the

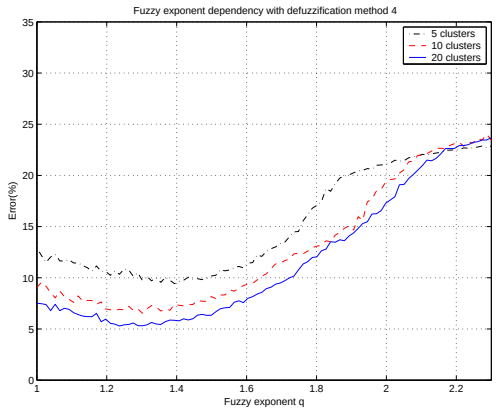

Figure 3.11: Fuzzy exponent versus test error for unsupervised clustering, using the old data.

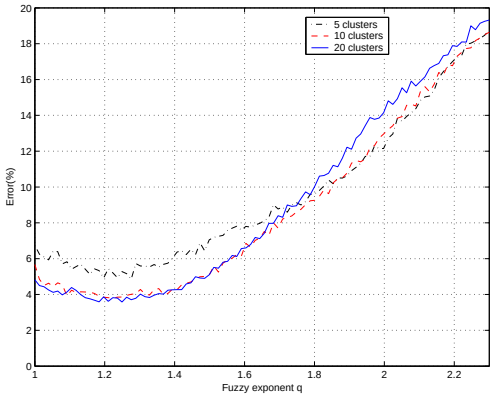

Figure 3.12: Fuzzy exponent versus test error for supervised clustering, using the old data.

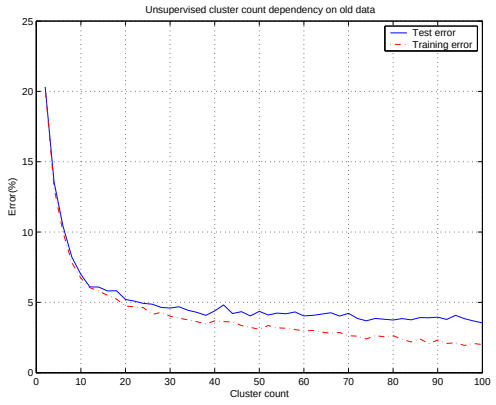

Figure 3.13: Cluster count versus error for unsupervised clustering, using the old data.

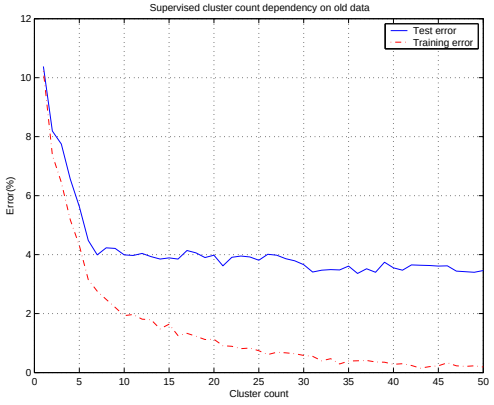

Figure 3.14: Cluster count versus error for supervised clustering, using the old data.

more clusters are used. Normally overtraining should occur when too many clusters are used. But no overtraining is noticed and could partly be because of the chosen defuzzification method 4, mentioned in chapter 2.3. Its density measure works against overfitting by giving smaller clusters less influence than larger clusters that possibly fits the data better. But the complexity of the data probably has a greater influence than the defuzzification method used. If the natural clusters in the data have complex shapes, it can be FCM describes these shapes better by combining a number of FCM clusters for every natural cluster in the data.

It is seen, that the training error for supervised clustering is almost 0% for 50 clusters, but much bigger for the unsupervised clustering with 100 clusters. The supervised clustering therefore seems to fit the structure of the data much better than unsupervised clustering.

Test errors with the chosen parameters is now found. The results are shown in table 3.9. The errors are obtained with 10-fold cross-validation and 20 reruns. The FN% is well below 5%, which was a requirement, but the FP%

	Overall error	**FP%**	**FN%**
Unsupervised classification	3.31%	5.60%	1.78%
Supervised classification	3.06%	4.17%	2.32%

Table 3.9: Overall error, False positive and False negative is shown for unsupervised and supervised FCM.

is still over 5% for unsupervised clustering and is not satisfying. The overall error is below 5%. The test errors are much better than those obtained in the project of Byriel (1999) with FCM, even though the same data is used and no feature selection is yet applied on this projects data. This lower error is probably caused by a more optimal fuzzy exponent, the large amount of clusters used and the chosen defuzzification method. Furthermore a test error obtained with k-fold cross-validation with a bigger confidence level probably also has some influence.

Feature selection

Feature selection is now performed. Simulated annealing is used to find the subset of features with a wrapper approach. Here we wrap the FCM algorithm into a black box, from which we get an estimated error. Simulated annealing uses this error as its fitness value. This error estimate is obtained by k-fold cross validation with a partitioning of the data so a maximal spread of the classes over all folds is achieved. The k-fold cross-validation is rerun several times, to get a good estimate of the error. See section 2.2.

A good error estimate is needed if simulated annealing is to have succes. Even though simulated annealing is able to tackle a local minimum or a false low estimated error value, too many will make it harder to find a good solution.

The features found with simulated annealing is shown in table 3.10.

	Selected features
Unsupervised classification	1,2,3,4,5,6,7,10,11,12,15,16,17,18,20
Supervised classification	2,3,4,5,6,7,10,11,12,15,16,18,20

Table 3.10: Selected features for unsupervised and supervised clustering. Every number corresponds to a feature, se chapter 2.1.

The errors calculated with the selected features, the same exponents and 50 and 100 clusters for supervised and unsupervised clustering respectively, is seen in table 3.11. 10-fold cross-validation and 20 reruns are used. The results are much better than before feature selection, an error reduction to half the error from before is achieved.

	Overall error	**FP%**	**FN%**
Unsupervised classification	1.81%	2.78%	1.17%
Supervised classification	1.64%	2.02%	1.38%

Table 3.11: Classification results for unsupervised and supervised FCM with selected features.

Fuzzy C-means with new data

The new data is investigated in the same way, as the old data. A near optimal fuzzy exponent and cluster count is found. A subset of features is afterwards found with features selection.

Fuzzy C-means exponent

The fuzzy exponent dependency is found just like in the case for the old data, and is shown in figure 3.15 with the unsupervised clustering, and in figure 3.16 for supervised clustering. The errors in the figure is calculated with 2-fold cross-validation and 20 reruns. From the figures the near optimal exponents is chosen to be $q = 1.13$ for unsupervised clustering and $q = 1.2$ for supervised clustering.

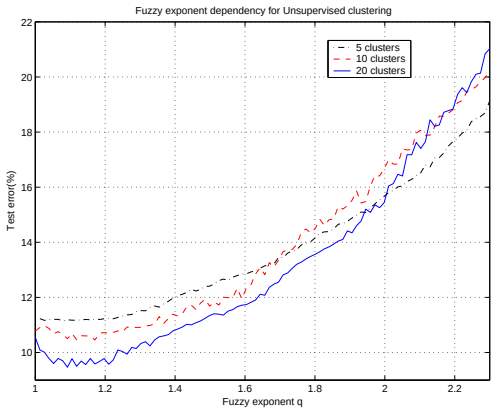

Figure 3.15: Fuzzy exponent versus test error for unsupervised clustering, using the new data.

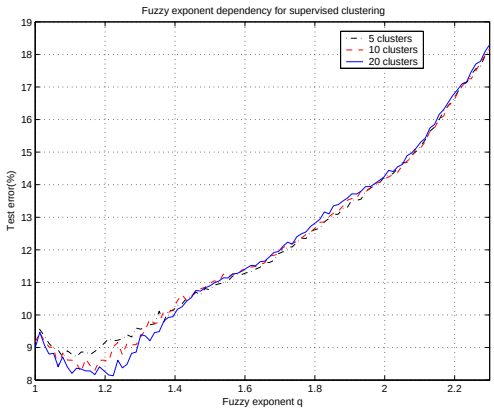

Figure 3.16: Fuzzy exponent versus test error for supervised clustering, using the new data.

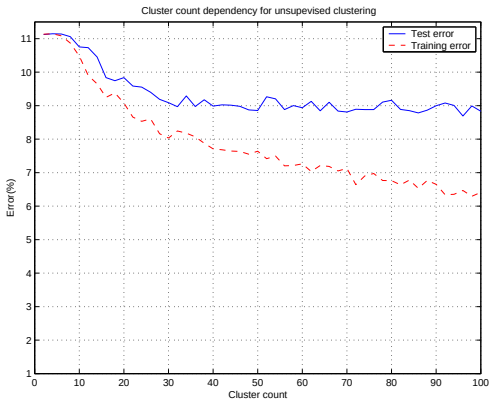

Figure 3.17: Cluster count versus error for unsupervised clustering, using the new data.

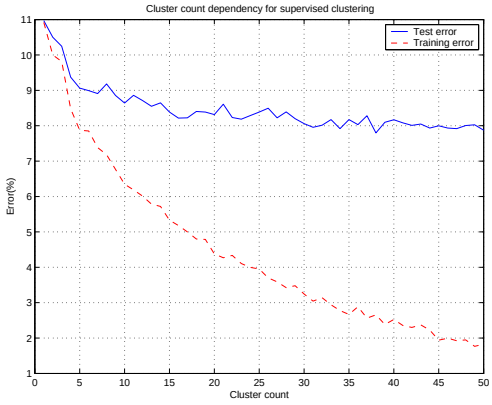

Figure 3.18: Cluster count versus error for supervised clustering, using the new data.

Number of clusters

Now the number of clusters to use is determined. An analysis of the number of clusters against the cross-validation test error is found, and is shown in figure 3.17 for unsupervised clustering and for supervised clustering in figure 3.18. The errors are calculated with 2-fold cross-validation and 20 reruns. The test error seems to be decreasing when more clusters are used for both unsupervised and supervised clustering. Therefore 50 clusters for supervised and 100 clusters for unsupervised clustering are chosen. The training error of the supervised clustering is seen to decrease towards zero much faster than for the unsupervised approach. This tendency is also noticed for the test error.

When using the chosen fuzzy exponents and number of clusters, the error is

calculated with 10-fold cross-validation and 20 reruns. The results are shown in table 3.12.

	Overall error	FP%	FN%
Unsupervised classification	8.72%	25.36%	2.75%
Supervised classification	7.56%	20.38%	2.96%

Table 3.12: Classification results for unsupervised and supervised FCM.

The goal is having and error below 5%, this is certainly not fulfilled with the calculated errors. The FP% errors are really poor whereas the FN% is low, which could be caused by a bias in the structure of the data towards the abnormal cells. Since their is many more abnormal cells than normal in the new database, this is probably partly the reason abnormal cells are favoured.

Feature selection

A subset of better performing features are now found with simulated annealing. The chosen features are shown in table 3.13.

	Selected features
Unsupervised classification	1,2,3,4,5,6,7,10,12,13,14
Supervised classification	1,3,4,5,6,7,10,14,15,17,18,19,20

Table 3.13: Selected features for unsupervised and supervised clustering. Every number corresponds to a feature, se chapter 2.1.

With these features, the errors are calculated with a 10-fold cross-validation and 20 reruns. The results is shown in table 3.14.

	Overall error	FP%	FN%
Unsupervised classification	8.08%	20.97%	3.45%
Supervised classification	6.10%	13.89%	3.29%

Table 3.14: Classification results for unsupervised and supervised FCM after feature selection.

After feature selection, a lower overall error rate is obtained. The FP% is lowered quite a bit, and a little higher FN% is obtained compared to before feature selection. But still the errors are much too high. Again supervised clustering outperforms unsupervised.

Fuzzy C-means conclusion

Acceptable results for the old data are achieved with well below 5% in error. The errors with FCM gives around halve the error obtained by HCM. The results on the new data is less uplifting. FCM is not able to come below 5% in error, which was the goal. Feature selection improved on the error, but not nearly enough. But still the errors of FCM is better than those of HCM.

The errors obtained with HCM after feature selection, has approximate the same value as those obtained with FCM before feature selection. This shows the importance of having a fuzzy exponent in the FCM algorithm. But if a really bad fuzzy exponent is chosen, FCM can perform worse than HCM. This can be seen in the figures 3.11,3.12, 3.15 and 3.16 by comparing the error for the fuzzy exponents $q \approx 1$(HCM) and $q = 2$(FCM). If defuzzification method 1 had been used instead of method 4, the error would had been less dependent on an optimal exponent, but a less optimal error would be obtained for an optimal exponent.

Feature selection improved the error considerably for both unsupervised and supervised clustering just as the case was for HCM.

In general it seems supervised clustering gives better results than unsupervised. Again an indicator that the clusters are not well separated in natural clusters.

It is interesting that the more clusters used both for supervised and unsupervised clustering, the lower an error is obtained. The reason probably being these clusters together form more complex cluster shapes, namely the natural cluster shapes of the data.

Chapter 4

Gustafson-Kessel clustering

The Gustafson-kessel(GK) clustering algorithm will here be introduced, and used to test the old and new papsmear data in this project. FCM produces spherical clusters and does not let a cluster change its shape dependent on the data. Gustafson and Kessel proposed a method so a cluster could adapt to hyperellipsoidal shapes. In the previous chapter, feature scaling for FCM was examined. This scaling was a linearly scaling of the features. GK clustering makes its own non-linear scaling, that is different from cluster to cluster.

4.1 Theory

The GK clustering algorithm, is quite similar to the FCM algorithm. The only difference is how the distances is calculated. FCM uses the euclidian distance measure. GK uses the mahalanobian distance measure shown in equation 4.1:

$$D_{ik}^2 = (x_k - c_i)A_i(x_k - c_i)^T \tag{4.1}$$

Where D_{ik} is the distance, c_i is the center for cluster i, x_k is data point k. A_i is the mahalanobian distance matrix. If A_i is the identity matrix, the euclidian distance measure is obtained. For GK, A_i is defined by equation 4.2:

$$A_i = p_i \cdot det(F_i)^{(1/N)} F_i^{-1} \tag{4.2}$$

Here N is the number of features, p_i is the volume cluster i shall have, F_i^{-1} is the inverse of the matrix F_i. F_i is the covariance matrix for cluster i, and is calculated with equation 4.3:

$$F_i = \frac{\sum_{k=1}^{K}(m_{ik})^q(x_k - c_i)^T(x_k - c_i)}{\sum_{k=1}^{K}(m_{ik})^q} \tag{4.3}$$

K is the number of data samples, q is the fuzzy exponent chosen and m_{ik} the membership defined by equation 3.3.

The objective function of GK is to minimize the distances from the cluster centers to the data points. The distances are given by equation 4.1, and controlled by the mahalanobian distance matrix A_i. The distances could be made smaller by using a less positive definite A_i, which corresponds to scaling the features so the clusters volume become smaller. Therefore some constraint is necessary on the clusters volume and therefore on A_i.

The eigenvectors and eigenvalues of A_i determines the shape, location and volume of cluster i. The hyperellipsoidal clusters are formed by vectors pointing in the direction of the eigenvectors and having lengths given by the N'th root of the corresponding eigenvalues. Since all the eigenvectors are perpendicular to each other, an approximate expression for the volume of a cluster can be made as the product of the N'th root of its eigenvalues. The volume will actually be calculated for a box shaped cluster, even though the cluster is cigar formed. The ratio between the calculated box volume and the actual cigar formed volume is a constant proportional factor, and can therefore be disregarded. The determinant of A_i can be calculated as the product of the eigenvalues in A_i. Therefore $\sqrt[N]{det(A_i)}$ is used to estimate the volume of cluster i.

In equation 4.2, A_i is calculated. A constant volume for A_i is wanted. The volume given for F_i^{-1} will quite likely change between every iteration of the algorithm. Therefore to get a constant volume, the factor $det(F_i)^{(1/N)}$ is multiplied with it, giving a constant volume of $det(det(F_i)^{(1/N)}F_i^{-1}) = 1$.

If a constant volume different from 1 for a cluster is wanted, this can be done with p_i in equation 4.2. Normally all clusters are given a constant volume of 1, and therefore p_i is often disregarded.

4.2 Gustafson-Kessel implementation

GK is not an algorithm in any of Matlab's toolboxes, therefore it is implemented in this project. It is located on the accompanied cd-rom and called '*gk.m*'. It has the following sub functions: '*step_ gk.m*', '*cal_ A.m*', '*dist_ gk.m*' and '*calc_ U.m*'. Since the determinant and inverse of a matrix is used in the algorithm, GK is prone to problems with covariance matrixes being close to singular. When having close to singular covariance matrixes, the calculated results in Matlab will have some error associated.

The algorithm is implemented with $p_i = 1$ from equation 4.2, restricting the volume of the clusters to 1.

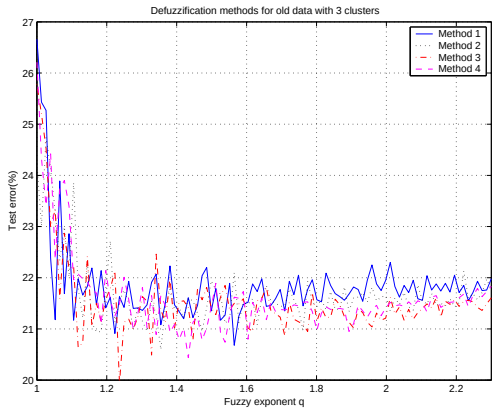

Figure 4.1: Exponent dependency with 3 clusters on old data for the different defuzzification methods.

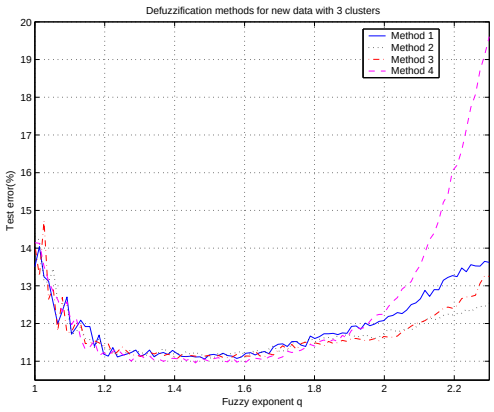

Figure 4.2: Exponent dependency with 3 clusters on new data for the different defuzzification methods.

4.3 Defuzzification results

The 4 mentioned defuzzification methods in section 2.3, will here be graphically compared on the old and new data with GK clustering. One of these defuzzification methods will be chosen and used later with all further GK clustering.

To start with, the error will be calculated and plottet against the fuzziness exponent for each of the methods. The error is measured by a 2-fold cross-validation with 20 reruns. This is done for 3 and 6 clusters for the old and the new data. All features are used. The defuzzification methods are tested on unsupervised clustering, as in section 3.1. The results are shown in figure 4.1,4.2, 4.3 and 4.4.

It can be hard reading which curves belong to which methods, since their graphs are so overlapping. This is more prominent when using 3 clusters than for 6 clusters. Using only 3 clusters, shows almost no difference between the defuzzification methods, when looking at the lowest obtainable error. This is because the fewer clusters there are, the more defuzzification methods 2,3,4 becomes like method 1.

When using 6 clusters some differences are observable. For the old data, method 2 and 4 has the lowest errors. For the new data, method 3 and 4 has the lowest errors. Since the defuzzification method chosen, is going to be used with a near optimal fuzzy exponent, the error for poor fuzzy exponents are not important. Therefore defuzzification method 4 is chosen as the method to use with GK clustering from now on.

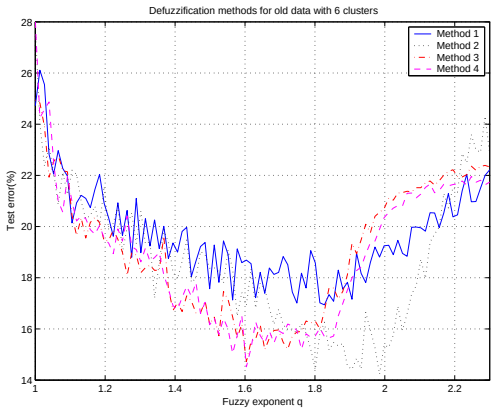

Figure 4.3: Exponent dependency with 6 clusters on old data for the different defuzzification methods.

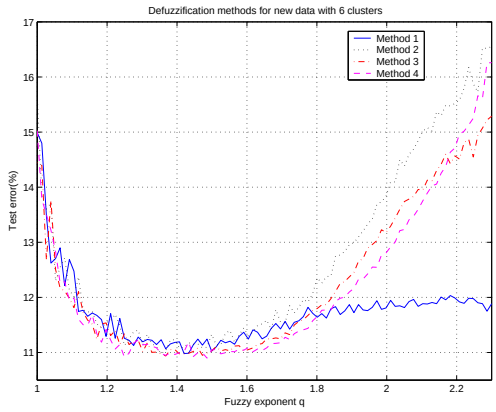

Figure 4.4: Exponent dependency with 6 clusters on new data for the different defuzzification methods.

4.4 Gustafson-kessel results

GK clustering is applied on the old and the new data with unsupervised and supervised clustering. First results for the old data will be obtained, thereafter results for the new data is obtained.

Gustafson-Kessel on Old data

A fuzzy exponent and a number of clusters are to be determined for both unsupervised and supervised clustering. Figur 4.5 and 4.6, shows the results for unsupervised clustering. From here, a fuzzy exponent of $q = 1.91$ is chosen from figur 4.5, where the error seems smallest. It should be noted, that the curves for the testing error and training error, has an almost equal optimal fuzzy exponent. From figur 4.6 the number of clusters is seen to perform best with 7 clusters. More clusters give worse test errors. More than 7 clusters is overfitting the data. A visible bend of the curve for the training error can be seen when using just around 7 clusters, which indicates a good number of clusters to use. Occam's razor says simplere models should be preferred if they perform just as well on the training data, as the more complex models. In our case, a number of clusters greater than 7 gives a more complex model that is no better on the training data than the simpler one build with 7 clusters.

In figure 4.7 the fuzzy exponent dependency is shown, and in figure 4.8 the cluster count dependency is shown for supervised clustering. The supervised GK clustering gives surprising results. The test error seems to be smallest when the fuzzy exponent $q \approx 1$. So a crisp border between the clusters apparently performs the best.

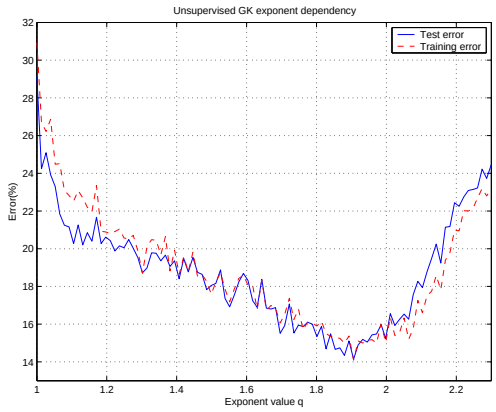

Figure 4.5: Unsupervised clustering error plottet against fuzzy exponent. 7 clusters are used.

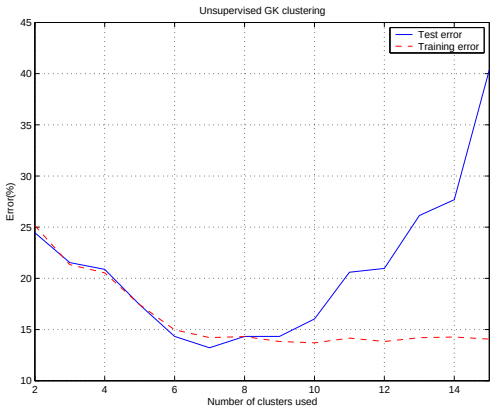

Figure 4.6: Unsupervised clustering error plottet against number of clusters. Fuzzy exponent $q = 1.91$ is used.

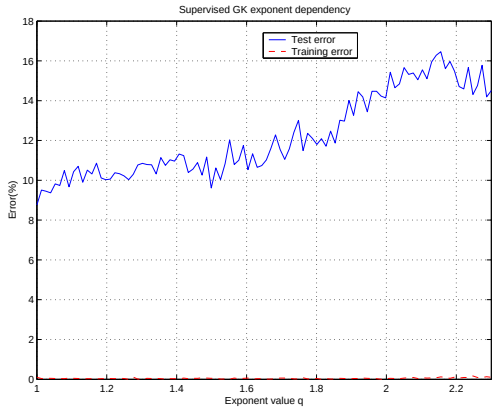

Figure 4.7: Supervised clustering error plottet against fuzzy exponent. 4 clusters are used.

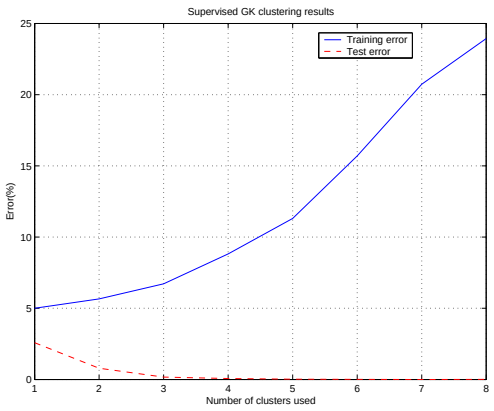

Figure 4.8: Supervised clustering error plottet against number of clusters. Fuzzy exponent $q \approx 1$ is used.

The optimal number of clusters for each class(normal and abnormal), seems to be when using only 1 cluster. With more clusters, the test error increases and overfitting is taking place.

The test error for GK, is calculated with the chosen parameters, and shown in table 4.1. 10-fold cross-validation and 20 reruns are used to calculate the error. Unsupervised clustering performs terrible, compared to supervised clus-

	Overall error	**FP%**	**FN%**
Unsupervised classification	14.34%	33.71%	1.31%
Supervised classification	4.44%	2.13%	5.97%

Table 4.1: Results for unsupervised and supervised GK clustering with all features.

tering. It could seem there is a problem with too high a bias towards abnormal cells, giving rise too high $FP\%$ and a low $FN\%$. The test error is well over 5%, and is therefore unacceptable.

Supervised clustering performs better, and does not show the same biasing problem. The error for supervised clustering is not acceptable, since its $FN\%$ is bigger than 5%.

To get a better error, feature selection with simulated annealing is performed. The chosen subset of features is shown in table 4.2.

	Selected features
Unsupervised classification	1,3,4,6,7,8,10,14,16,17
Supervised classification	1,2,3,4,5,7,8,9,10,11,12,14,15,16,17,18

Table 4.2: Selected features for unsupervised and supervised clustering. Every number corresponds to a feature, se section 2.1.

With these features, the errors are calculated with 10-fold cross-validation and 20 reruns. The results are shown in table 4.3.

	Overall error	**FP%**	**FN%**
Unsupervised classification	9.92%	23.83%	0.86%
Supervised classification	2.89%	4.46%	1.77%

Table 4.3: Results for unsupervised and supervised GK clustering with selected features.

Unsupervised clustering performs much better after feature selection than before. But still there is a problem with the bias towards the abnormal cells.

The test error is still way over 5%, and therefore not accepted.

Supervised clustering has improved on its error. Now its associated errors is all below 5%, and therefore supervised GK clustering performs well enough to be accepted.

Gustafson-Kessel on New data

The fuzzy exponent dependency for unsupervised clustering can be seen in figure 4.9, and the cluster count dependency can be seen in figure 4.10. The test error seems to be smallest when the fuzzy exponent is just around $q = 1.5$ and therefore is chosen as the fuzzy exponent. From the cluster count dependency, the test error for 4 clusters does not seem to be better using more or less clusters. The lowest training error is also for 4 clusters. A strange curve for the training error is seen, where it most of the time is larger than the curve for the test error. Normally the training error would be smaller than the test error. It could be thought, the training and test errors were somehow switched, but the rise in test error and not in the training error when enough clusters was used, indicates that no switch has been made. If the unsupervised GK clusters are fitted to well on the training data, the training data will only belong to a single cluster, no matter how mixed the cluster is of the different classes representing it. Therefore the training data will get assigned the class mostly represented in the cluster because there is no membership to any other cluster. The test data on the other hand probably fits the clusters less optimal and therefore will have a membership spread more out over several clusters. Clusters highly mixed of several classes has a smaller influence on the classification than the clusters almost only containing one class. This is determined by how the clusters is weighted in the defuzzification method (se section 2.3).

For supervised clustering, the fuzzy exponent dependency is shown in figure 4.11, and the cluster count dependency in figure 4.12. Just like in the case when using the old data, the fuzzy exponent is chosen to $q \approx 1$, since it clearly is seen the error decreases for a decreasing q. A cluster count of 1 per class shows the best performance, but also shows how effective supervised GK clustering is in adapting to the structure of the data, because more clusters will just overfit the data. It even can be supervised GK clustering is already overfitting the data with only one cluster per class.

The test errors are now calculated with the chosen parameters and shown in table 4.4. Unsupervised clustering shows a big bias towards the abnormal cells, by a $FP = 41\%$ against a $FN = 0.04\%$. Supervised clustering has a much lower error than unsupervised, but the $FP = 18\%$ is a bit high compared to a $FN = 3.1\%$. Both unsupervised and supervised clustering has errors much over 5% and is therefore unacceptable.

Feature selection is now performed. The obtained results are shown in table

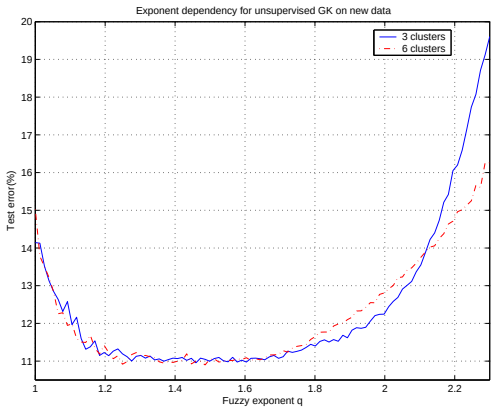

Figure 4.9: Fuzzy exponent dependency with unsupervised GK on new data.

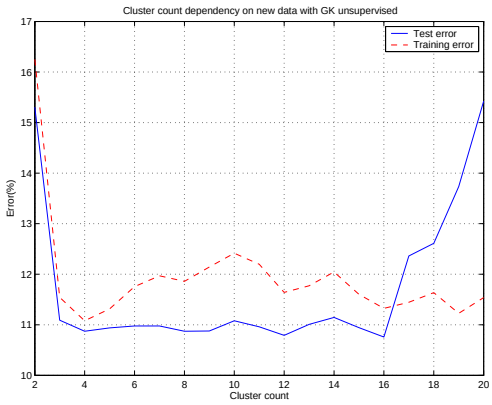

Figure 4.10: Cluster count dependency with $q = 1.4$ for unsupervised GK on new data.

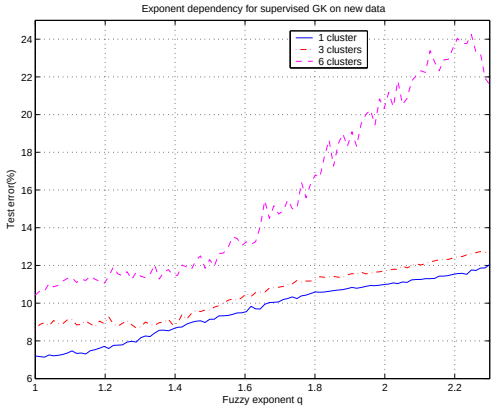

Figure 4.11: Fuzzy exponent dependency with supervised GK on new data.

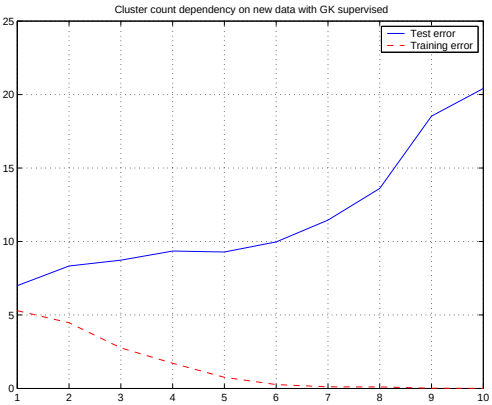

Figure 4.12: Cluster count dependency with $q = 1.0$ for supervised GK on new data.

	Overall error	FP%	FN%
Unsupervised classification	10.90%	41.32%	0.04%
Supervised classification	7.04%	17.99%	3.11%

Table 4.4: Results for unsupervised and supervised GK clustering with all features.

4.5. It is interesting, that the subset of features found for unsupervised clustering, all is a subset of the features found for supervised clustering. The features selected for supervised and not for unsupervised, must be features inducing a poor separation of the clusters. With the chosen features and parameters, the

	Selected features
Unsupervised classification	1,6,7,8,9,10,13,14,17,18,20
Supervised classification	1,2,4,5,6,7,8,9,10,12,13,14,15,16,17,18,19,20

Table 4.5: Selected features for unsupervised and supervised clustering. Every number corresponds to a feature, se section 2.1.

test errors is calculated and shown in table 4.6. The errors for unsupervised

	Overall error	FP%	FN%
Unsupervised classification	7.68%	22.66%	2.31%
Supervised classification	6.06%	13.88%	3.26%

Table 4.6: Results for unsupervised and supervised GK clustering with selected features.

clustering are lowered considerably, but still far from being as good as the supervised clustering. There is still too high a bias towards the abnormal cells, giving a high $FP\%$. It is interesting, that the supervised clustering before feature selection performs better than unsupervised clustering after feature selection.

Gustafson-Kessel Conclusion

Defuzzification method 4 was chosen be to used with GK. It performs better than the other methods, when using many clusters. Since the results obtained here is for a relatively small number of clusters, the effect of choosing this defuzzification method does not have a big influence.

Unsupervised clustering gives much worse results than supervised clustering,

showing that the data is not nicely separated in natural clusters and therefore it does not find a good natural boundary. The supervised clustering is told where the boundary is, and therefore better fits the clusters on the data. The results obtained with supervised clustering before feature selection is better than the results with unsupervised clustering after feature selection. This shows that using supervised clustering instead of unsupervised, is more important than performing feature selection with the actual data.

It is noted that supervised clustering is better at utilizing its features, since almost all the features chosen with feature selection for unsupervised clustering are a subset of the features selected for supervised clustering.

Both supervised and unsupervised GK clustering has shown to be overfitting the data with a relatively few clusters. Supervised clustering performed the best with only one cluster per class, possibly already overfitting the data.

The results obtained with GK, was just able to get test errors below 5% for the old data when using supervised clustering with feature selection. The results with unsupervised clustering was too high. For the new data, neither supervised and unsupervised clustering gave errors below 5% after feature selection.

Chapter 5

Direct and Hierarchical classification

Until now, the classification problem has been to decide between the normal and abnormal cells. In the following, classification will be done on all the diagnoses of the cells. Since GK has shown to be much poorer at classifying papsmear data in this project, only FCM will be used in the direct and hierarchical classification.

5.1 Theory

Direct classification, classifies all cells in a single step. Hierarchical classification is defined to be a multistage classification algorithm.

Sometimes the classification problem is quite complex, therefore the built classifier model also has to be quite complex to obtain god results. The problem is, the more complex the classification model has to be, the less the chance is finding a model that fits very well. Therefore if we can divide the problem up into subproblems and tackle these one by one, better results can possibly be obtained. Parameters like fuzzy exponent, cluster count and feature selection can be found so they fit a subproblem specifically.

The project of Landwehr (2001), showed this was the case. The project used data from single cell papsmear cells, equal to the kind used in this project, on a hierarchical classification approach. In the project of Lin & Fu (1983), papsmear data is classified with a binary tree approach, with a k-nearest neighbor classifier in every tree node. The results from this project seemed good, but other kinds of features are used. It seems, splitting the problem up in subproblems, would be a good idea. This idea is here called hierarchical classification, just like in Landwehr (2001). In figure 5.1 and 5.2, the difference between the direct and hierarchical approach is shown. A node is represented

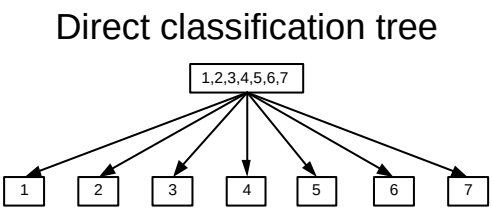

Figure 5.1: Direct classification. Every number corresponds to a class/diagnose.

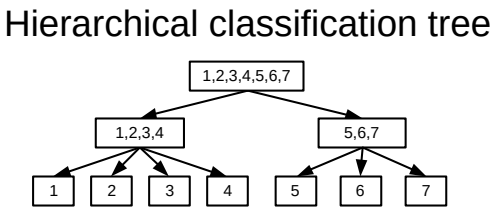

Figure 5.2: Hierarchical classification. Every number corresponds to a class/diagnose.

by a square box. Every parent node represents a classifier model. Leaf nodes do not contain a classifier. The numbers in the nodes, tell which cell diagnoses should be expected as input to the node. In the direct approach, all data is classified in one step. The hierarchical approach divides the data until all the diagnoses are represented by each their own node. In some way, the hierarchical approach can be thought of as a decision tree, where each decision node is an ordinary classifier algorithm using the features at hand.

5.2 Direct results

Results for the direct method will here be presented. This is done for the old and the new data. The selected features, chosen cluster count and exponent are chosen in the same manner as earlier when doing a normal/abnormal classification.

Old data results

The fuzzy exponent dependency on the test error is shown in figur 5.3 for unsupervised clustering and in figur 5.4 for supervised clustering. An optimal fuzzy exponent of $q = 1.5$ for unsupervised and $q = 1.4$ for supervised clustering is chosen. The curves made with 20 clusters are used since they seem to give the best results. The fuzzy exponents are chosen where the curves are roughly estimated to give a minimum error.
In figur 5.5 and 5.6 the cluster count dependency is shown for unsupervised and supervised clustering. The more clusters used for both unsupervised and supervised clustering, the lower an error is seen. Therefore 50 clusters are chosen for supervised and 100 for unsupervised clustering.
Feature selection is performed, which give the features shown in table 5.1.

The results with the selected features, cluster count and fuzzy exponent is then calculated with a 10-fold cross-validation and 20 reruns. Table 5.2 and 5.3

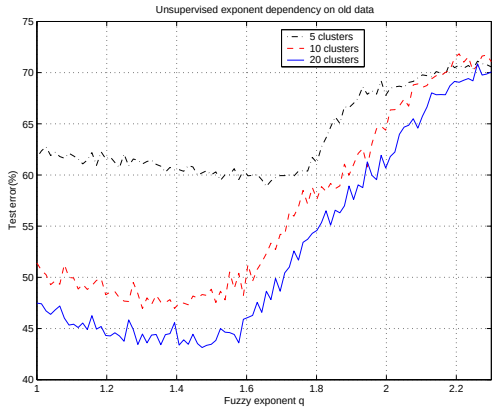

Figure 5.3: Fuzzy exponent versus test error for unsupervised clustering for the old data.

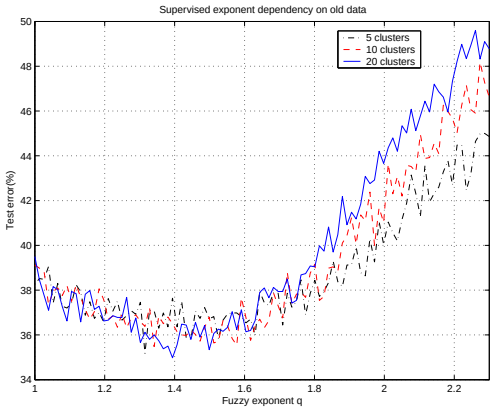

Figure 5.4: Fuzzy exponent versus test error for supervised clustering for the old data.

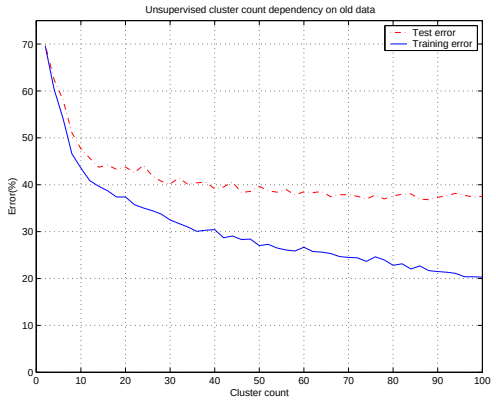

Figure 5.5: Cluster count versus error for unsupervised direct clustering for the old data.

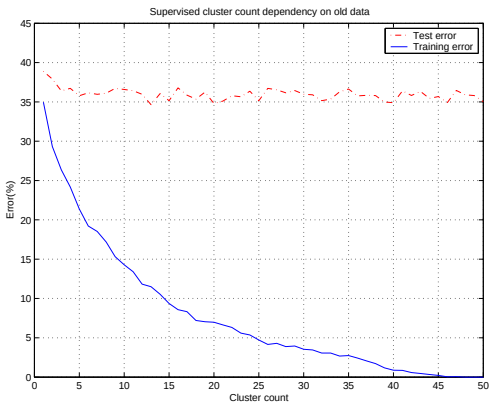

Figure 5.6: Cluster count versus error for supervised direct clustering for the old data.

	Selected features
Unsupervised classification	1,2,3,4,5,6,7,8,11,12,14,16
Supervised classification	1,2,3,4,5,6,7,10,11,14,16

Table 5.1: Selected features for unsupervised and supervised direct classification for the old data. Every number corresponds to a feature, se chapter 2.1.

shows the results for unsupervised and supervised clustering. The results are shown in a confusion matrix, where percentages are shown instead of sample count.

	Predicted diagnose						
	COL	**PAR**	**INT**	**SUP**	**MIL**	**MOD**	**SEV**
COL	86.4%	0.0%	0.0%	0.0%	0.2%	3.9%	9.5%
PAR	4.0%	92.0%	0.0%	0.0%	0.0%	3.3%	0.7%
INT	0.0%	0.0%	84.6%	13.6%	1.8%	0.0%	0.0%
SUP	0.0%	0.0%	13.3%	86.7%	0.0%	0.0%	0.0%
MIL	1.0%	2.45%	0.0%	0.0%	55.05%	24.25%	17.25%
MOD	1.0%	0.05%	0.0%	0.0%	26.6%	39.7%	32.65%
SEV	0.55%	1.25%	0.0%	0.0%	8.3%	24.85%	65.05%

Table 5.2: A confusion matrix containing the direct **unsupervised** classification results for the old data. The vertical column is the correct class and the horizontal row is the predicted class.

	Predicted diagnose						
	COL	**PAR**	**INT**	**SUP**	**MIL**	**MOD**	**SEV**
COL	85.8%	0.0%	0.0%	0.0%	0.0%	0.7%	13.5%
PAR	4.0%	92.0%	0.0%	0.0%	0.0%	2.3%	1.7%
INT	0.0%	0.4%	88.6%	10.0%	1.0%	0.0%	0.0%
SUP	0.0%	0.0%	6.0%	94.0%	0.0%	0.0%	0.0%
MIL	0.95%	1.1%	0.0%	0.0%	61.45%	25.6%	10.9%
MOD	1.0%	0.0%	0.0%	0.0%	22.6%	56.95%	19.45%
SEV	0.65%	0.9%	0.0%	0.0%	6.5%	31.75%	60.2%

Table 5.3: A confusion matrix containing the direct **supervised** classification results for the old data. The vertical column is the correct class and the horizontal row is the predicted class.

The normal cells shows really good classification results with above 80%

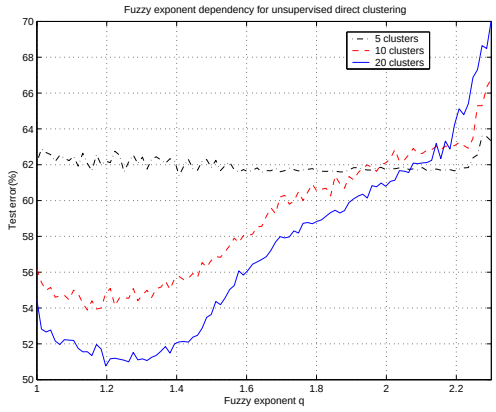

Figure 5.7: Fuzzy exponent versus test error for unsupervised clustering for the new data.

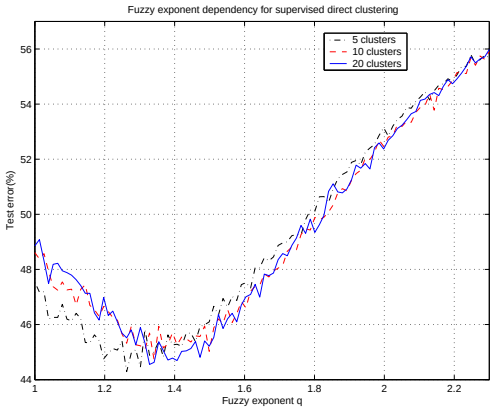

Figure 5.8: Fuzzy exponent versus test error for supervised clustering for the new data.

classification accuracy. The abnormal cells show a poor separation from each other, but this was expected. The results seem to show, that some of the cells form groups of diagnoses. These groups are $[COL, PAR]$, $[INT, SUP]$ and $[MIL, MOD, SEV]$.

New data results

The fuzzy exponent dependency is shown in figur 5.7 for unsupervised and in figur 5.8 for supervised clustering. A optimal fuzzy exponent of $q = 1.25$ for unsupervised and $q = 1.26$ for supervised clustering is chosen. The fuzzy exponents are chosen, where they seem to have a global minimum error.

In figure 5.9 and 5.10, the cluster count dependency is shown for unsupervised and supervised clustering. Figure 5.9 shows that the more clusters used for unsupervised clustering, the lower an error. Therefore 100 clusters is chosen. In figur 5.10 no decrease in test error is seen when more than 5 clusters is used. Therefore 5 clusters is chosen for supervised clustering.

	Selected features
Unsupervised classification	3 7 14 15 17 18
Supervised classification	1 2 3 7 11

Table 5.4: Selected features for unsupervised and supervised direct classification. Every number corresponds to a feature, se chapter 2.1.

Feature selection is performed, giving the features shown in table 5.4. It is surprising how few features was selected with feature selection. The results with the selected features, cluster count and fuzzy exponent is calculated with

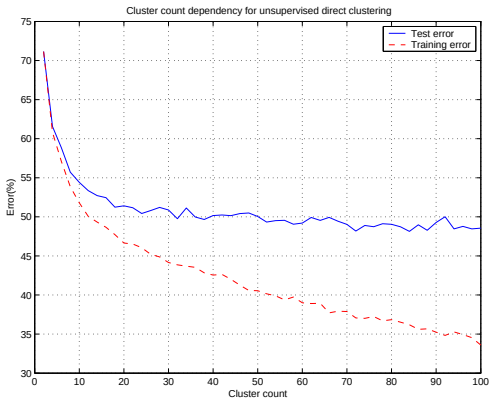

Figure 5.9: Cluster count versus error for unsupervised direct clustering for the new data.

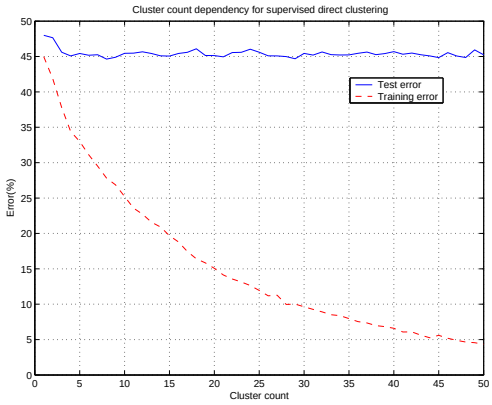

Figure 5.10: Cluster count versus error for supervised direct clustering for the new data.

a 10-fold cross-validation and 20 reruns. Table 5.5 and 5.6 shows the resulting confusion matrixes for the direct unsupervised and supervised clustering.

From the results, some groupings are noticed. The $[INT, SUP]$ is seen to be a group, just as in the old data. The diagnoses $[MIL, MOD, SEV, CIS]$ is definitely hard to separate from each other and is a group. The normal cells $[COL]$ could possibly also belong to this group, when looking at the obtained errors, but it is known to be a normal cell and therefore should not belong to this group.

	Predicted diagnose						
	SUP	**INT**	**COL**	**MIL**	**MOD**	**SEV**	**CIS**
SUP	85.64%	14.36%	0%	0%	0%	0%	0%
INT	9.43%	87.86%	1.28%	1.43%	0%	0%	0%
COL	0%	0.87%	52.33%	6.79%	13.55%	20.96%	5.50%
MIL	0%	0.50%	2.06%	62.42%	20.91%	12.98%	1.13%
MOD	0%	0%	4.27%	32.50%	32.11%	18.94%	12.18%
SEV	0%	0%	11.16%	12.53%	15.04%	34.81%	26.46%
CIS	0%	0%	4.63%	2.27%	9.13%	35.43%	48.54%

Table 5.5: Direct **unsupervised** classification results with selected features for new data, in a confusion matrix. The vertical column is the correct class and the horizontal row is the predicted class.

	Predicted diagnose						
	SUP	**INT**	**COL**	**MIL**	**MOD**	**SEV**	**CIS**
SUP	89.57%	10.43%	0%	0%	0%	0%	0%
INT	9.21%	89.79%	0%	1.0%	0%	0%	0%
COL	0%	0%	55.48%	7.73%	11.28%	23.16%	2.35%
MIL	0%	0.93%	1.40%	79.09%	9.40%	9.15%	0.0%
MOD	0%	0%	5.02%	41.24%	26.29%	21.66%	5.79%
SEV	0%	0%	8.14%	11.15%	10.65%	41.47%	28.59%
CIS	0%	0%	3.3%	2.97%	7.36%	21.87%	64.5%

Table 5.6: Direct **supervised** classification results with selected features for new data, in a confusion matrix. The vertical column is the correct class and the horizontal row is the predicted class.

5.3 Hierarchical results

The hierarchical tree structures here decided on, are chosen so the easiest separable cells are classified first, the cells hardest to separate is left for the final classification nodes. But the most important classification, is between the normal and abnormal cells, therefore the trees parent node is a classification on normal and abnormal. Furthermore, the diagnoses to split up in a node, are chosen so the number of cells are balanced almost equal to the branches of the node. The decided tree structures are based on the groups formed by the classification results from the direct classification. Hierarchical results with FCM, for both the old and the new data with unsupervised and supervised clustering, will be calculated in the following.

Old data results

The tree structure decided on, can be seen in figure 5.11. The numbers in the nodes is abbreviations for the diagnoses, shown in table 5.7. Notice, that in the direct classification on the old data, we found some groupings in the results. These corresponds to the groupings in the nodes $[1, 2]$,$[3, 4]$ and $[5, 6, 7]$ in figure 5.11. The classification tree is formed by letting the diagnoses split in a way, so the easy classifications take place before harder classifications, but by still trying to keep a balanced tree.

Every non-leaf node, will need to have a classifier associated with it. Therefore every node, needs the selected features and parameters that performs best at the actual stage to be determined. We have 2 parameters, the fuzzy exponent and the number of clusters to find. These parameters are obtained in the same manner, as when finding the optimal parameters in earlier chapters, by using plots of the error versus the parameters value. But only the values

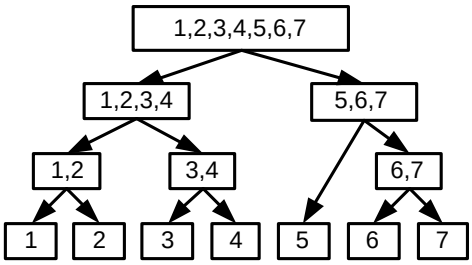

Figure 5.11: The figure shows the hierarchical tree structure chosen for the old data. The numbers in the nodes is abbreviations for the diagnoses, shown in table 5.7.

Abbreviation	Cell type
1	Columnar epithelial(COL)
2	Parabasal epithelial(PAR)
3	Intermediate epithelial(INT)
4	Superficial epithelial(SUP)
5	Mild dysplastic(MIL)
6	Moderate dysplastic(MOD)
7	Severe dysplastic(SEV)

Table 5.7: Abbreviations used for the old data used in the tree structure of figure 5.11

decided on for the parameters will here be shown, not the plots used for every node. The parameters and features chosen, is shown in table A.1 for unsupervised clustering, and in table A.2 for supervised clustering in appendix A. No feature selection is performed for node 1234, since the error was too close to an error of 0%. All features is therefore used in that node.

With the chosen parameters, the errors are calculated and shown in table 5.8 for unsupervised clustering and in table 5.9 for supervised clustering. The errors are calculated with a 10-fold cross-validation and 20 reruns.

New data results

The tree structure decided on for the new data is shown in figure 5.12. The numbers in the nodes, is abbreviations of the diagnoses, and shown in table 5.10. The classification decided on in every node, is based on the results obtained with the direct classification with the new data. The diagnoses easiest to separate from each other, is the classifications first carried out. At the same

	Predicted diagnose						
	COL	**PAR**	**INT**	**SUP**	**MIL**	**MOD**	**SEV**
COL	91.8%	0%	0%	0%	0%	0.7%	7.5%
PAR	0%	97.2%	0%	0%	0%	0.3%	2.5%
INT	0%	0%	91.7%	8.3%	0%	0%	0%
SUP	0%	0%	0.2%	99.8%	0%	0%	0%
MIL	1.0%	1.75%	0.1%	0%	47.75%	35.9%	13.5%
MOD	1.0%	0%	0%	0%	14.35%	65.5%	19.15%
SEV	0.05%	0.1%	0%	0%	3.85%	31.6%	64.4%

Table 5.8: Hierarchical **unsupervised** classification results with selected features for old data, in a confusion matrix. The vertical column is the correct class and the horizontal row is the predicted class.

	Predicted diagnose						
	COL	**PAR**	**INT**	**SUP**	**MIL**	**MOD**	**SEV**
COL	94.8%	0%	0%	0%	0%	1.50%	3.7%
PAR	0%	97.6%	0%	0%	0%	0%	2.4%
INT	0%	0%	90.3%	9.5%	0.2%	0%	0%
SUP	0%	0%	0%	100%	0%	0%	0%
MIL	1.0%	1.85%	0.15%	0%	48.35%	37.9%	10.75%
MOD	1.05%	0%	0%	0%	10.0%	70.85%	18.1%
SEV	0.25%	0.05%	0%	0%	3.5%	34.45%	61.75%

Table 5.9: Hierarchical **supervised** classification results with selected features for old data, in a confusion matrix. The vertical column is the correct class and the horizontal row is the predicted class.

time, a balanced tree structure with the number of cells split into the branches being as equal as possible, was the goal.

The selected features, exponents and clusters counts in the different nodes, are shown in table A.3 for unsupervised clustering, and in table A.4 for supervised clustering in appendix A. No feature selection is performed for node 123, since the error was too close to 0%. All features is therefore used for this node.

With the chosen parameters, the errors are calculated and shown in table 5.12 for unsupervised clustering and in table 5.12. The errors are calculated with a 10-fold cross-validation and 20 reruns.

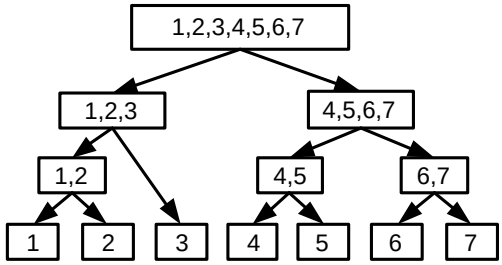

Figure 5.12: The figure shows the hierarchical tree structure chosen for the new data. The numbers in the nodes is abbreviations for the diagnoses, shown in table 5.10.

Abbreviation	Cell type
1	Superficial epithelial(SUP)
2	Intermediate epithelial(INT)
3	Columnar epithelial(COL)
4	Mild dysplastic(MIL)
5	Moderate dysplastic(MOD)
6	Severe dysplastic(SEV)
7	Carcinoma in situ(CIS)

Table 5.10: Abbreviations used for the new data cell diagnoses in the tree structure of figure 5.12

	Predicted diagnose						
	SUP	**INT**	**COL**	**MIL**	**MOD**	**SEV**	**CIS**
SUP	92.76%	7.24%	0%	0%	0%	0%	0%
INT	6.07%	92.36%	0.79%	0.78%	0%	0%	0%
COL	0%	0%	51.48%	0.99%	10.11%	33.1%	4.32%
MIL	0%	0%	0.72%	67.36%	24.38%	6.82%	0.72%
MOD	0%	0%	2.21%	31.28%	44.86%	17.76%	3.89%
SEV	0%	0.03%	6.15%	6.61%	11.71%	47.56%	27.94%
CIS	0%	0%	2.27%	1.13%	5.87%	45.07%	45.66%

Table 5.11: Hierarchical **unsupervised** classification results with selected features for new data, in a confusion matrix. The vertical column is the correct class and the horizontal row is the predicted class.

Comparison of results & Conclusion

It can be hard to compare the results from the direct and hierarchical classifiers. Therefore an average classification accuracy for each classifier is cal-

	Predicted diagnose						
	SUP	**INT**	**COL**	**MIL**	**MOD**	**SEV**	**CIS**
SUP	93.53%	6.47%	0%	0%	0%	0%	0%
INT	10.21%	87.79%	1.21%	0.79%	0%	0%	0%
COL	0%	0%	66.81%	1.86%	5.79%	18.71%	6.83%
MIL	0%	0%	0.71%	68.09%	24.86%	5.13%	1.21%
MOD	0%	0%	1.98%	28.99%	46.02%	17.37%	5.64%
SEV	0%	0.48%	6.39%	6.65%	8.38%	48.19%	29.91%
CIS	0%	0%	2.53%	2.67%	4.6%	34.1%	56.1%

Table 5.12: Hierarchical **supervised** classification results with selected features for new data, in a confusion matrix. The vertical column is the correct class and the horizontal row is the predicted class.

culated by taking the average of the correct classification of each cell. The results obtained for the direct classification is shown in table 5.13 and in table 5.14 for the hierarchical classifiers. The classification accuracy for the direct classifiers is much lower than for the hierarchical classifiers. But it should be remembered, that the hierarchical classifiers is much more complex, having to find good parameters and features to use in every node. Furthermore, a direct classification is a necessary step, if the tree structure is decided on in the same way as here.

The supervised clustering shows much better results than unsupervised clustering. Actually, using the supervised clustering on the direct classifiers, seems to perform almost as well as unsupervised clustering on the hierarchical classifiers. Since the implementation of supervised clustering is quite simple compared to finding the hierarchical model, it shows the importance of using supervised clustering when using this kind of data.

The number of features selected in every node for hierarchical classification, is found with feature selection by using simulated annealing. It seems more features were selected using supervised clustering, than unsupervised clustering. To show this, the average number of features over the nodes of the hierarchical classifier, is calculated and shown in table 5.15. The table shows, that feature selection on supervised clustering indeed chooses more features than unsupervised clustering. The reason possibly being, that supervised clustering is less dependant on features giving a good separation of the natural clusters, than unsupervised clustering. Therefore more features can be utilized by the supervised clustering. It is interesting, that the number of features found for the new data is much smallere than for the old data. Not only the classification accuracy of the new data is much worse than the old data, but also fewer features was usable.

Classification results obtained for the old data, shows that the accuracy for

	Direct classification	
	Unsupervised clustering	**Supervised clustering**
Old data	72.5%	77.0%
New data	57.67%	63.74%

Table 5.13: Classification accuracy for the classifiers using direct classification

	Hierarchical classification	
	Unsupervised clustering	**Supervised clustering**
Old data	79.74%	80.52%
New data	63.15%	66.65%

Table 5.14: Classification accuracy for the classifiers using hierarchical classification

	Average no. features for	
	Unsupervised clustering	**Supervised clustering**
Old data	11.8	13.6
New data	7.6	10.2

Table 5.15: Average number of features selected, when building the hierarchical classifiers.

the normal cell diagnoses all is above 90% for the hierarchical classifiers. The direct classifiers almost have a 5% worse accuracy for each of the normal cell diagnoses. The poor separation between the abnormal cells was expected, since this also is one of the hardest classifications for cyto-technicians.

Classification results with the new data, shows a good separation of the superficial and intermediate normal cells from the abnormal. But the Columnar epithelial cells had a very poor separation from the abnormal cells. Compared to the separation between the columnar epithelial cells and the abnormal cells in the old data, this is surprising. The pictures in the new database has been verified by a cyto-technician and no problems with diagnoses of the cells was found.

Chapter 6

Investigations

Several investigations relating to the classifiers and the data used in this project are performed in the following sections. The investigations are the following:

- Feature scaling with the FCM classifier.
- Bias control of the classes in the data.
- Robustness analysis of FCM and GK clustering.
- Influence of the amount of training data on the overall classification error.
- Comparison of the new data versus the old data.

6.1 Feature scaling

Many classification algorithms need some preprocessing of the features to get good results. One such thing is standardizing or normalizing the features. Standardization scales the features so they have a fixed standard deviation and mean value. Normalization scales the features, so all values are inside a certain interval. Some of the more sophisticated classification algorithms perform their own scaling of the data. This is the case with GK.

C-means classification uses the distance measure to find the cluster centers, and afterwards to find the classification of test data. If a certain feature has a standard deviation much greater than all other features, it will likely have a greater influence on the algorithm because of the greater distances it causes than other features. It often is practice scaling the features so they all have an equal mean and standard deviation. This investigation shows the importance of using some kind of standardization or normalization.

Procedure

The FCM algorithm will here be used to show the importance of scaling the features. All features start out having a standard deviation $\sigma = 1$ and mean $\mu = 0$. The performance of the actual feature scalings is measured with k-fold cross validation, and rerunning of the k-fold cross validation is applied, to get better error estimates, as described in section 1.6. Simulated annealing will be used to find new and better scalings of the features. This is done by changing the scaling of a random feature by keeping the mean $\mu = 0$, and changing the standard deviation σ to some random value. Simulated annealing will then decide if the new scaling is better performing or not.

Feature scaling results

Results for both the old and the new data are found. Only supervised clustering is used, since the previous results has shown it performs better than unsupervised clustering. The found feature scalings for the old data is shown in table 6.1, and for the new data in table 6.2. The feature numbers can be looked up in section 2.1. It is noted that the feature scalings for the old and the new data, are quite different even though the data should not be too different. It was found, that the feature scalings simulated annealing found, was not the same every time it was run. The reason probably being, that the space to search in is much more complex than for ordinary feature selection. Ordinary feature selection is only a binary choice of keeping or removing a feature, whereas it for feature scaling is a decision of giving a value from the whole interval [0..1]. Since similar feature scalings are not always obtained, it is quite likely non-optimal feature scalings that are found.

Feature nr.	**Feature scaling**	**Feature nr.**	**Feature scaling**
1.	0.1770	11.	0.2917
2.	0.6853	12.	0.1203
3.	0.2647	13.	0.0772
4.	0.0798	14.	0.0305
5.	0.7075	15.	0.7695
6	0.8565	16.	0.4637
7.	0.9113	17.	0.6900
8	0.0059	18.	0.5562
9.	0.4914	19.	0.4467
10.	0.8652	20.	0.0511

Table 6.1: Chosen feature scalings with simulated annealing for the old data.

Feature nr.	Feature scaling	Feature nr.	Feature scaling
1.	0.8967	11.	0.9174
2.	0.5966	12.	0.0573
3.	0.7933	13.	0.5015
4.	0.9819	14.	0.7770
5.	0.2438	15.	0.4239
6	0.4440	16.	0.2426
7.	0.8334	17.	0.7469
8	0.0213	18.	0.3484
9.	0.0066	19.	0.4637
10.	0.2098	20.	0.7975

Table 6.2: Chosen feature scalings with simulated annealing for the new data.

The error estimates achieved are shown in table 6.3. The errors are calculated with a 10-fold cross-validation. The found feature scalings give a considerable decrease in error compared to when just a standard scaling of $\sigma = 1$ is used. If the features had been used without any standardization, the results would probably be worse than for a standardization with $\sigma = 1$.

The results show, that ordinary feature selection performs better on the old data, than feature scaling does. On the new data, the difference between the errors for feature scaling and feature selection, comes to a tie since they almost are equal. The reason feature scaling is not performing at least as well as ordinary feature selection, must be caused by the much more complex data space to search through, resulting in solutions that are not optimal.

The results obtained with feature scaling, should theoretically be able too at least get results as good as ordinary feature selection with a $\sigma = 1$ scaling, since the choices of using or not using a feature corresponds to a feature scaling with $\sigma = 1$ or $\sigma = 0$, respectively. The feature scaling works as a kind of feature selection, where the features can be chosen to participate to a degree, instead of a binary participation in the case of feature selection.

	Old data, OE%	New data, OE%
Features scaled by $\sigma = 1$	3.06%	7.56%
Feature scaling	2.15%	6.07%
Feature selection	1.64%	6.10%

Table 6.3: Error estimates are shown for a standard scaling ($\sigma = 1$), for the chosen feature scalings and for ordinary feature selection($\sigma = 1$ scaling) on both the old and new data.

Conclusion

The importance of feature scaling with the FCM algorithm is showed. Results with feature scaling are near those obtained with feature selection, even though it theoretically should be possible to at least get the same error. With the used data the conclusion is, that using simulated annealing for feature scaling gives a less optimal result than with simulated annealing on feature selection. The reason being, the search space of feature scaling is too complex compared to the search space of the feature selection. If fewer features had been available, the complexity of the search space for the feature scaling would be reduced and possibly give at least as good results as feature selection.

6.2 Bias control

Cyto-technicians often want a biasing of the results, so when in doubt of a cells diagnose, the cell is more likely to be diagnosed as abnormal than normal. When favoring the abnormal diagnosis a lower FN% and an increased FP% will occur. Biasing the results give a shift in error between FN% and FP%. A rough way to control the bias will here be proposed and tested.

The results obtained for FCM and GK until now, has had a tendency of a bias towards the abnormal cells. The number of data used for each diagnose has some influence on the bias, but also the underlaying structure of the data has an influence. The choice of using defuzzification method 4 influences the results by weighing the bigger clusters more than the smaller clusters.

The procedure of how the bias can be controlled will now be explained. First we introduce $B_{desired}$ which is a vector containing the desired bias to each diagnose. The sum of the vector shall equal 1. F.ex. if there are two classes where equal biases are wanted, then $B_{desired} = [0.5, 0.5]$. If $B_{desired} = [0, 1]$, then the bias is total to class 2 and therefore all data points will be classified as this class. All data points of class 1 will with certainty be calculated wrong. How sensitive the model is to a bias, will depend on the built model and on the structure of the data used.

The bias control is added directly in the defuzzification methods, by changing the memberships calculated by these. In eq. 6.1, the modification of the memberships to the different classes, is given by the vector $U_{class-biased}$:

$$U_{class-biased}(i) = \frac{B_{desired}(i)}{Diag\%(i)} \cdot U_{class}(i) \tag{6.1}$$

Here $Diag\%(i)$, is the percentage of training data points having the class i. U_{class} is the vector defined by equation 2.3 for defuzzification method $2-$

4. Now $U_{class-biased}$ substitutes U_{class} in the defuzzification methods, so the classification decision is made on this new biased membership.

Biasing like this, gives us no guarantee that $B_{desired} = [0.5, 0.5]$, will give a $FN\% = FP\%$. This depends on the actual used data, the built model and the structure of the data. Using $Diag\%(i)$ should help minimize the effect of imbalances in the amount of data for each class. The fuzzy exponent has an influence on the sensitivity to a bias in equation 6.1. If the fuzzy exponent $q \approx 1$ is used, the bias control will only have a little influence on the actual bias. Since the memberships to the different clusters are either..or for $q \approx 1$, only clusters not being well represented by having a high percentage mixing of the different classes, will be sensitive to a bias. F.ex a cluster itself can have a membership of 0.2 to the normal cells and 0.8 for the abnormal cells. Only defuzzification method 3 and 4 uses a fuzzy membership from each cluster to every diagnose, therefore they are the only methods sensitive to a bias control, when using $q \approx 1$. To rely on having clusters highly mixed of the training data's different classes, is not a good idea. Instead the fuzzy exponent should be chosen greater than 1, enabling fuzzy memberships from data to the clusters. The value of the fuzzy exponent will furthermore have an influence on the sensitivity to the bias control.

Results with bias

Results are obtained for FCM and GK for both the old and the new data. Test errors are calculated with a 10-fold cross-validation and 20 reruns. The found features and parameters in the previous chapters for FCM and GK are used with supervised clustering, except the fuzzy exponent $q = 1.2$ is used instead of $q \approx 1$ for GK. The reason being $q \approx 1$ will give a crisp membership on which the bias will have no effect, which is mentioned earlier.

Results for the old data can be seen in figure 6.1 for FCM and in figure 6.2 for GK. Results with the new data are seen in figure 6.3 for FCM and in figure 6.4 for gk.

The bigger the bias is on the normal cells, the lower a FP% is noticed. But at the same time the FN% becomes greater because of the lower bias towards the normal cells. This shows that the simple bias control works with the proposed method. The overall error for both FCM and GK clustering on the old and the new data, shows the overall error is smallest when a bias of 0.4 on the normal cells are used. This reflects that there are fewer normal cells than abnormal, and therefore when in doubt of the diagnose of cells, the best overall results are obtained by favoring cells with the diagnose abnormal.

It is seen, that a simple bias like the one here introduced, can change the bias to the classes wanted. It can also be used to minimize the effects of an imbalance in the number of cells to the different classes. The actual bias

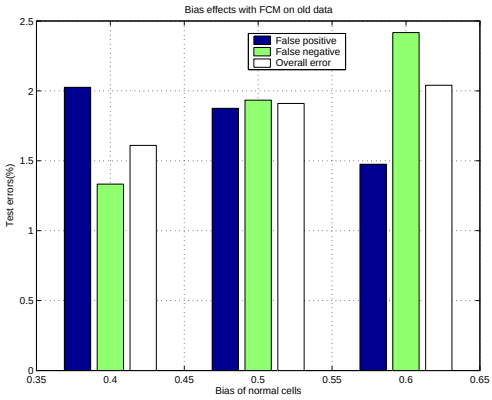

Figure 6.1: Test errors on old data for different biases of the normal cells when using FCM.

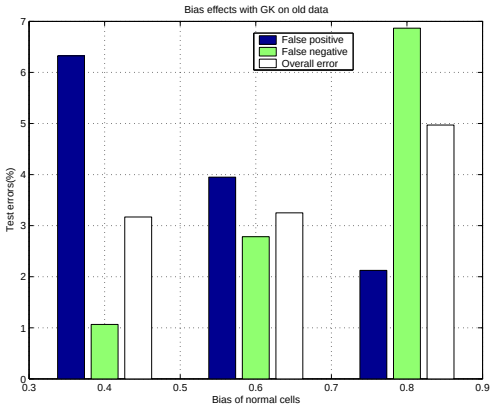

Figure 6.2: Test errors on old data for different biases of the normal cells when using GK.

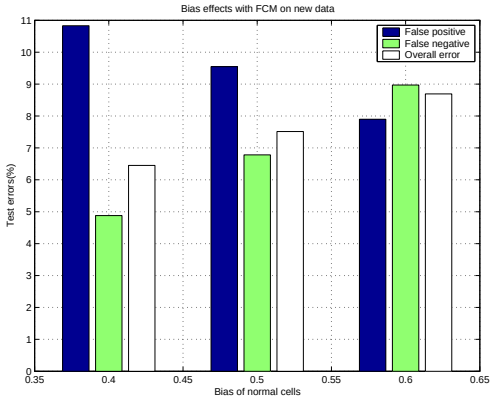

Figure 6.3: Test errors on new data for different biases of the normal cells when using FCM.

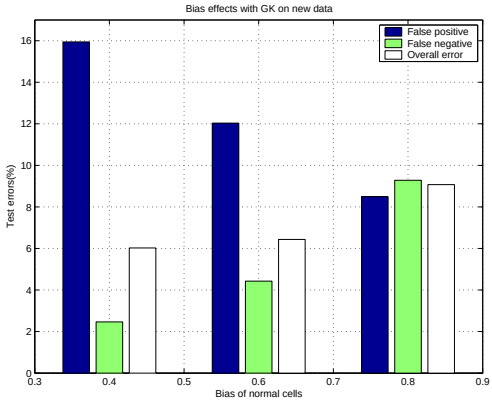

Figure 6.4: Test errors on new data for different biases of the normal cells when using GK.

necessary for a certain ratio between FP% and FN% is not given. This bias must be found experimentally.

6.3 Robustness Analysis

The features used to build a classifier, can be more or less noisy. The classification results therefore depends on how sensitive the classifier is to noise in the features. The noise can be accumulated while extracting the feature, or maybe even exists naturally.

When building the classifier, the noise has some influence on the final classifier. Noise in the features not yet seen by the classifier, can also give rise to

an error. Here the effect of how sensitive the classifier is to noise in unseen data samples, is investigated. The robustness of FCM and GK for both the old and new data is investigated.

Robustness measure

The membership of data to the different diagnoses, could have been used to calculate some robustness measure. But these memberships are quite dependent on the chosen fuzzy exponent. If the fuzzy exponent $q \approx 1$, a crisp membership would be obtained; for a very big exponent, almost equal membership to all diagnoses would be obtained. Instead a simple measure is here defined. The robustness is defined as the possible deviation for a selected feature in a data point unseen by the classifier, before it is classified wrong. If the data point already is being classified wrong, it gets the stable distance value 0. From this it follows that every data point tested, will have a robustness measure for every feature. The data point being tested, shall be a data point not used to build the classification model. The robustness is measured for every feature separate, even though there can be noise in several features of the same data point, simultaneously. Furthermore, noise in several features can be correlated. F.ex the area of a cells nucleus will have some correlation with the perimeter of the cells nucleus. These interdependencies between features in the data, are not investigated here.

Procedure

The distance a feature can deviate before the data point is mis-classified, is found by trying a series of changes in the actual feature, until it is classified wrong. The possible deviation before a wrong classification is made, is the robustness measure for the actual feature in the data point investigated. The approach is very simple, and applicable without knowing how the chosen algorithm works. The disadvantage is that the measured change can be wrong. The feature is changed in a series of steps. If the steps are too big, the class obtained from the classification can shift back and fourth inside a step. Therefore the smallere the steps, the better the chance a shift in the classification will be detected. Another problem exist, when features can change without the predicted class ever changing. To find these cells and corresponding features, the features should be scanned in an interval of $[-\infty, \infty]$. This is of course not possible. Therefore an upper limit and lower limit in which the feature change shall be examined is needed in excess of the step size.

A procedure that resembles LOOCV, is used to get measurements for all data points. When building the model, all data points except one are used to build the model. The robustness on the data point not used to build the

model, is measured. This proces is done iteratively so all data points have been left out once and tested.

Robustness results

The robustness is measured for the FCM and GK clustering algorithms. The optimal parameters like cluster count and fuzzy exponents found in earlier chapters are used. The comparison between FCM and GK should be on the same features and data, which they will not be if feature selection is applied. Therefore all features are used for the investigation. Defuzzification method 4 is used when finding a data points classification, since this one was chosen for FCM and GK previously. Supervised clustering gave the best classification results in the previous chapters, and is therefore chosen. All features are standardized with mean $\mu = 0$ and the standard deviation $\sigma = 1$ for both FCM and GK. Scaling the features for GK should not have any influence, but is done to make it easier comparing the results between FCM and GK.

Every feature has a measurement from every data point both for FCM and GK. These results are shown in a plot for every feature. Results both for FCM and GK, amounts to 40 plots for the old data and 40 plots for the new data. All the plots are not shown here, only plots from a few selected features are shown. The rest of the plots can be found on the accompanied cd-rom as pictures in the format as encapsulated postscript and portable networks graphics files. Whenever not enough detail can be seen in the following plots, it is recommended viewing them from the cd-rom.

The interval investigated is set to $[-3, 3]$, and the step size is set to 0.01. First the results for the old data and afterwards for the new data, will be presented.

Old data

Robustness results for FCM and GK will here be shown for a few selected features. In figure 6.5 and 6.6 the robustness results for the feature 'nucleus area' are shown. The plot is read by having all the data points actual standardized feature value on the x-axis, and their stable distance in standard deviations on the y-axis. Data points already classified wrong before any change in the feature value was applied, has a stable distance of 0 on the y-axis. Data points that were insensitive to a change inside the $[-3, 3]$ interval, is shown at 3 on the y-axis. A little displacement of the abnormal cells downwards on the y-axis is made for those having the stable distance 3, so it is easier differentiating the results between normal and abnormal cells.

The robustness for FCM on the feature containing the nucleus area, shown

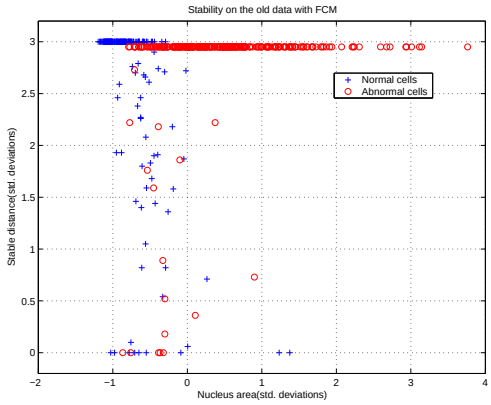

Figure 6.5: Robustness on nucleus area with the FCM algorithm for the old data.

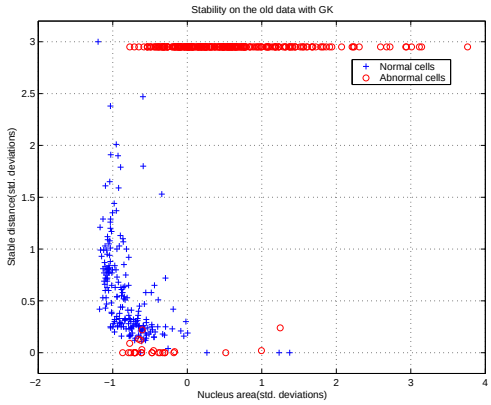

Figure 6.6: Robustness on nucleus area with the GK algorithm for the old data.

in figure 6.5, seems to be surprisingly good. Only relatively few data points are sensitive to a change in the feature at all. It should be remembered that we are talking about a deviation of ± 3 standard deviations from the actual value of the feature. For the same feature, the GK algorithm gives a much worse result shown in figure 6.6. Almost all the normal cells can be classified wrong, if a ± 1.5 standard deviations is applied on the value of the feature. But the abnormal cells are insensitive to these changes. This could reflect a big bias towards the abnormal cells for this feature.

In figure 6.7 and 6.8 the robustness results for the feature 'N/C ratio' are shown for FCM and GK. Again the FCM algorithm was relatively insensitive to changes. The GK algorithm again showed a very high sensitivity to changes in the feature. Now it is mostly the abnormal cells that are sensitive to changes, therefore having a big bias towards the normal cells for this feature. The GK clustering algorithm scales all its features locally for every cluster it finds. The high sensitivity, therefore could be caused by these local scalings of the features. If this is the reason, the sensitivity on some of the other features could be very little. The feature with which the best robustness for GK seemed to occur, was for the feature 'Nucleus brightness'. The results with this feature can be seen in figure 6.9 for FCM and in figure 6.10 for GK. Again the robustness for FCM seems good. The robustness for GK with this feature is much better than for the previously shown GK robustness results. The difference between FCM and GK in robustness is not big, and this even though it is the feature with the best robustness for GK.

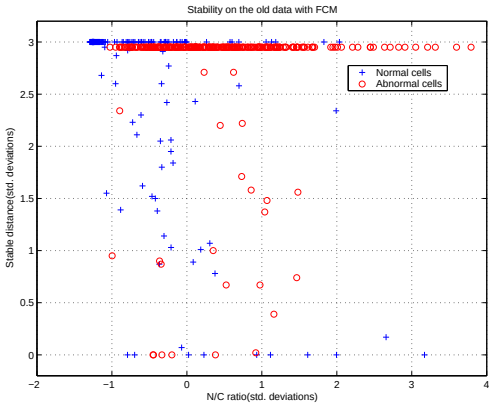

Figure 6.7: Robustness on N/C ratio area with the FCM algorithm for the old data.

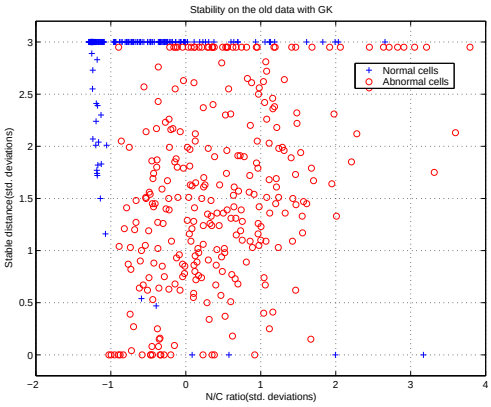

Figure 6.8: Robustness on N/C ratio area with the GK algorithm for the old data.

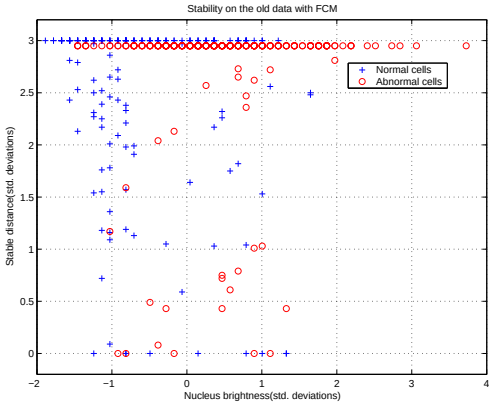

Figure 6.9: Robustness on Nucleus brightness with the FCM algorithm for the old data.

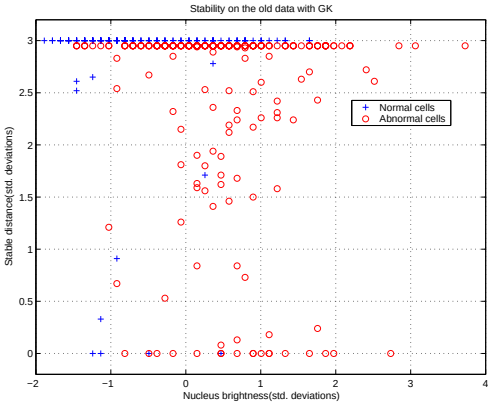

Figure 6.10: Robustness on Nucleus brightness with the GK algorithm for the old data.

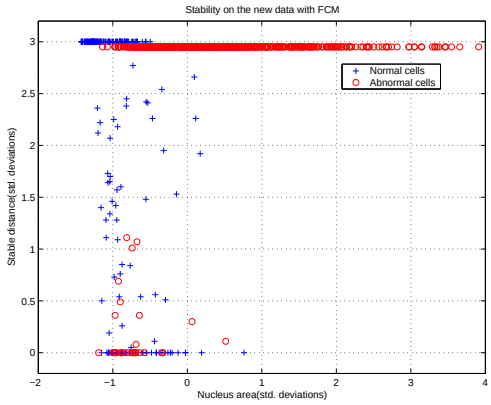

Figure 6.11: Robustness on nucleus area with the FCM algorithm for the new data.

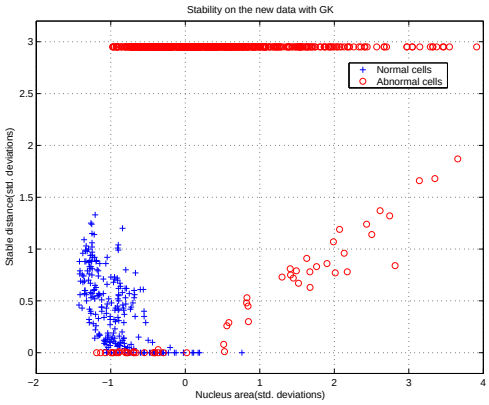

Figure 6.12: Robustness on nucleus area with the GK algorithm for the new data.

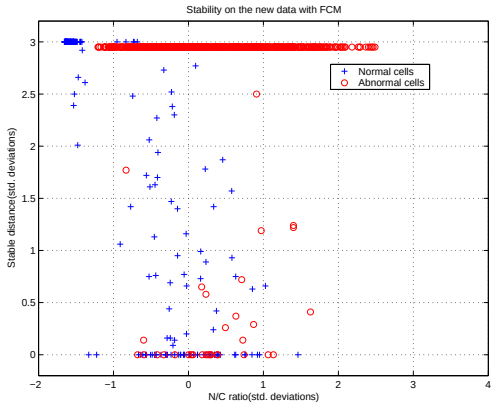

Figure 6.13: Robustness on N/C ratio area with the FCM algorithm for the new data.

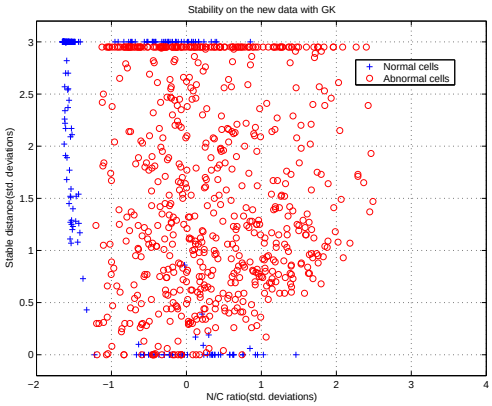

Figure 6.14: Robustness on N/C ratio area with the GK algorithm for the new data.

New data

The robustness will be shown for the same features that was shown with the old data. In figure 6.11 and 6.12, results for the feature 'Nucleus area' are shown. Results for the feature 'N/C ratio' are shown in fig 6.13 and 6.14. Lastly results for the feature 'Nucleus brightness' are shown in the figures 6.15 and 6.16. The observations made for the old data are also made for the new data. The robustness of GK clustering with the new data, is less optimal than with the old data. For the feature 'Nucleus area', now only a less than ± 1 standard deviation is necessary to classify the normal cells wrong. For the feature 'N/C ratio', the robustness also seems to be lowered.

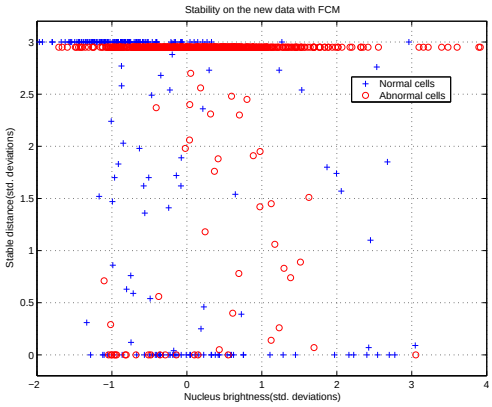

Figure 6.15: Robustness on Nucleus brightness with the FCM algorithm for the new data.

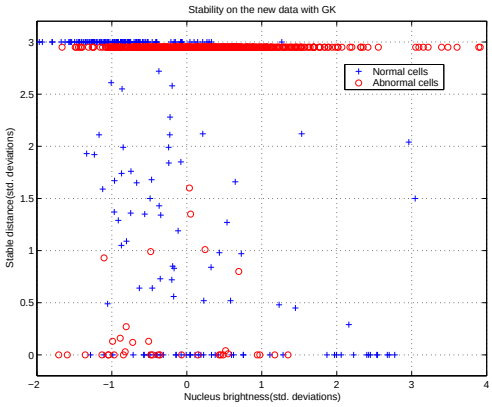

Figure 6.16: Robustness on Nucleus brightness with the GK algorithm for the new data.

Conclusion

There was a surprisingly big robustness margin on most of the correct classified cells with FCM. A few were less stable, indicating they were cells on the border to be classified wrong. The robustness of the noise in the features of GK, was much smaller than FCM. Even though almost equal classification results were obtained for FCM and GK on the new data, they show totally different stabilities to noise in the features. Robustness ought to be a parameter when choosing a classification algorithm.

The poor robustness of GK compared to FCM is not only present in the shown features. Plots of all the features can be seen on the accompanied cd-rom.

We know the classification results obtained with GK was worse than those obtained with FCM. But this difference can not explain the difference in robustness between these to algorithms. The poor robustness for GK is probably because of the feature scaling that is performed by the GK algorithm for every cluster found, and because the clusters of GK are limited to hyperellipsoidal clusters.

The exceptional good robustness results with FCM, could reflect that many of the features contain redundant information. Even though changing one of the features to a total different value, it is highly likely that it still has enough information in the rest of the features to make a correct classification, no matter which feature has been changed. This kind of robustness is not always present in classification systems. Decision tree's is an example. Here single features can be used to guide it down through the tree. But if the parent decision node of the tree uses the feature that the noise is added to, a wrong path down the tree could be taken resulting in a wrong classification.

The good results with FCM, could indicate that the many clusters used for FCM clustering, can form more complex clusters that fit the actual shape of the data better than GK, and therefore being less sensitive to noise in the features.

Feature selection in the previous chapters, is implemented by minimizing the classification error without considering the robustness. The robustness can possibly have become worse after feature selection, because the redundancy of information between the features now is smaller. If feature selection had been implemented not only by using the classification error, but also by incorporating the robustness in the fitness, the redundancy of information could be kept. Hereby only sorting features away that have a low redundancy of data and no usable information. The classification results would probably not be as good, but a more stable classification algorithm could be obtained.

6.4 Data size dependency

Some projects use huge amounts of data to build and test classifiers. When the data is at hand, it is a good idea using it. But when it is expensive and hard getting high quality data in big amounts, it is worth considering how much data are needed to get an acceptable error. The amount of data necessary, will likely be dependent on the method used to build the model. An investigation into the amount of necessary data is made, when using supervised FCM and GK clustering on a classification between normal and abnormal cells.

The optimal parameters and features found in earlier chapters for FCM and GK is used. The overall errors with different amounts of training data is obtained by building the model many times with different training and test data, where the mean of the errors is the calculated error. The training data is randomly sampled from the data, though in a way that spreads the different classes best possible over the training and test data. It should be noted that this investigation, does not show if the data available describes all the diagnoses fully, or if the diagnoses only are partially described. It is merely an investigation of how the amount of training data influences the classification error for FCM and GK with the data available.

Old data results

The results for the old data can be seen in figure 6.17. The figure shows the overall test error on the y-axis and the number of training data used on the x-axis. The errors for both FCM and GK clustering are showed. It is seen the results for GK clustering is much worse than those for FCM, no matter how much training data is used. Especially for less than 100 training data,

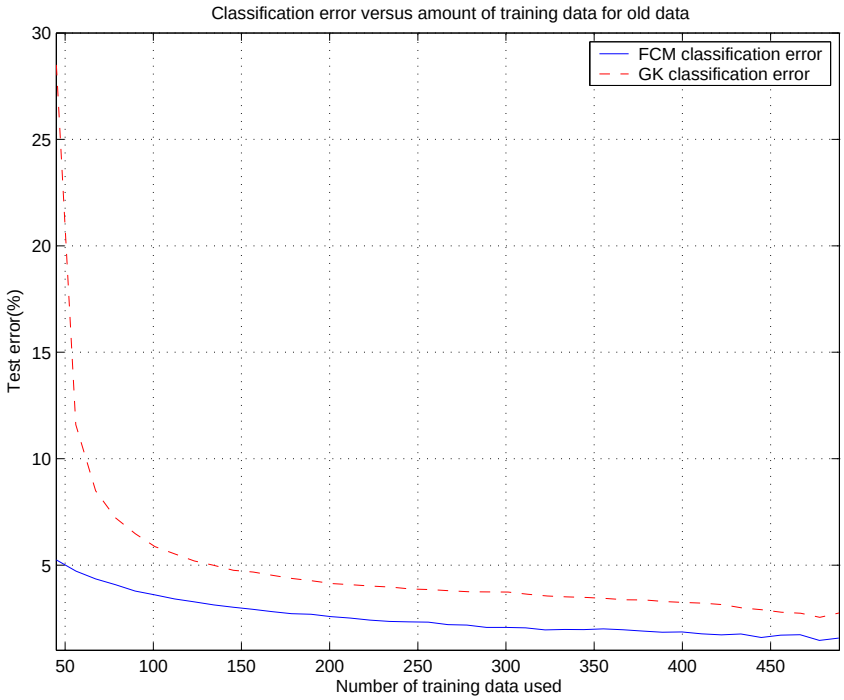

Figure 6.17: The figure shows the test error versus the number of training data used for FCM and GK on the old data.

the classification error rate rises fast for GK the less training data used, since it has to determine the shape of the hyperellipsoidal clusters in excess of the clusters position. FCM seems to perform better than GK, when only using a few training data, since it only has to determine the clusters position and not the shape of its clusters. Its error is smaller, and its curve is not showing the same increase in error when fewer training data used.

For more than 150 training data, the decrease in error by using more training data gives similar error reductions for FCM and GK. Their curves have become quite flat, and therefore would not benefit much from more training data.

New data results

The results for the new data can be seen in figure 6.18. The difference between GK and FCM clustering is not very big. When using less than 100 training data, the classification error for FCM is below GK. GK does not only need to find the centers for the clusters, it also has to find their shape. Therefore GK performs worse than FCM with less than 100 training data.

It is seen that when using between 100 and 700 training data, GK performs better than FCM. GK clustering assumes hyperellipsoidal clusters, whereas the many clusters of FCM together form the data's clusters. The assumption

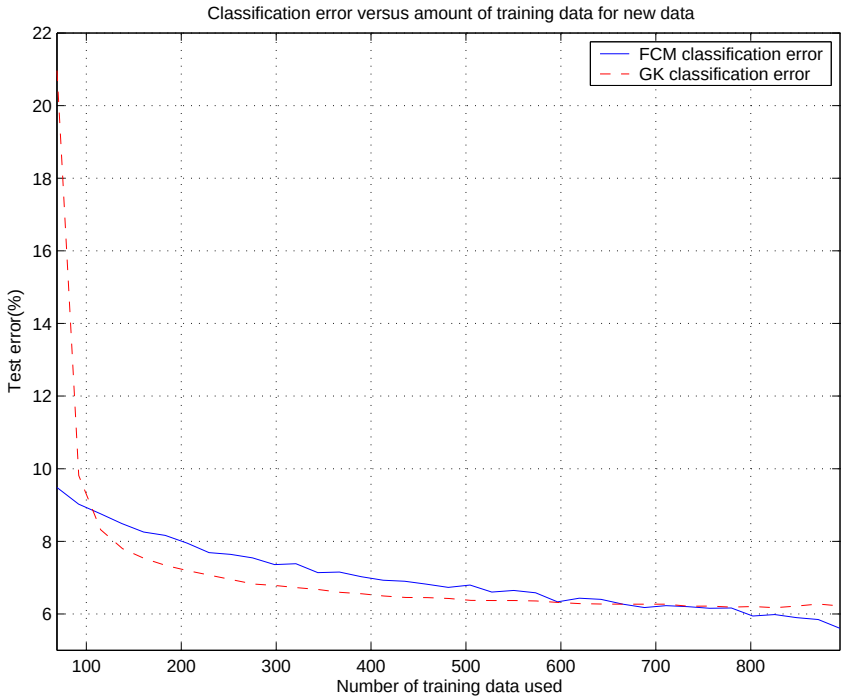

Figure 6.18: The figure shows the test error versus the number of training data used for FCM and GK on the new data.

of using hyperellipsoidal clusters, seems to be good when using between 100 and 700 training data. FCM has possibly not yet found cluster shapes that fit as well as the hyperellipsoidal clusters of GK.

When using more than 700 training data, FCM again performs better than GK. Since the cluster shapes of GK is limited to hyperellipsoidal clusters, more complex shapes can not be adapted. The curve with the classification error for GK clustering supports this, since its curve has become quite flat when using around 700 training data. The curve for FCM is still decreasing considerably, and therefore probably still adapting to more complex cluster shapes. The error limit of FCM is therefore probably not reached with the amount of data available here.

Conclusion

The hyperellipsoidal clusters of GK did not fit the old data as well as the clusters found with FCM. The GK clusters fitted the new data better than the FCM clusters, when using between 100 and 700 training data. But when enough training data was used, FCM performed best, whether the old or the new data was used. Since the overall classification error was below 5% for the old data for more than 50 training data with FCM and more than 100 training

data for GK, their is enough old data. Having more than 500 data in the old data would probably not give a much lower error, since the curve for FCM had flattened out.

The overall classification error for the new data is to high for both FCM and GK. Since the error curve for FCM did not seem to flatten out, it maybe is possible to obtain an overall error $< 5\%$ if enough training data were available. The assumption that more training data could lower the error, is based on any data added to the training data, should not deviate in any special way from the data already in the database, because else the error curves calculated would be useless.

6.5 New data versus Old data

The results obtained on a classification between normal and abnormal cells, gave an overall error just around 6% for the new data. This was surprising, since overall error results with the old data was obtained below 2%. In the direct and hierarchical classification, the reason was identified. The new data's columnar cells were classified as severe dysplastic for around 20% of its cells and with about 5% to moderate dysplastic and roughly 5% as carcinoma in situ. In the classification on the old data, the columnar cells were nicely separated from the dysplastic cells. The classification showed a good separation between the superficial and intermediate cells from the dysplastic cells, both for the old and the new data.

Differences between the new and the old data, are going to be shown graphically in the following. The columnar and severe dysplastic cells are picked out of the old and the new data. Interesting features are then plottet against each other. In figure 6.19 the nucleus area is plottet against the cytoplasma area for the columnar and severe dysplastic cells in the old data. In figure 6.20, the same is done for the new data. The plot with the new data shows, the columnar and severe dysplastic cells are mixed much more together and overlapping each others clusters considerably more than in the old data.

In figure 6.21 the nucleus area is plottet against the nucleus relative position for the columnar and severe dysplastic cells in the old data. Likewise is done with the new data, in figure 6.22. Again the cells are mixed more for the new data than for the old data. The separation between the columnar and severe dysplastic is harder in the new data, than in the old data.

The figures showed, that the new data's columnar and severe dysplastic cells are not as well separated as the old data's. This explains why results with the new data are worse than those obtained for the old data. It also shows, that the cells of the new data are selected in a somehow different way. It has not been the intention of the cyto-technicians to select the cells differently, but if

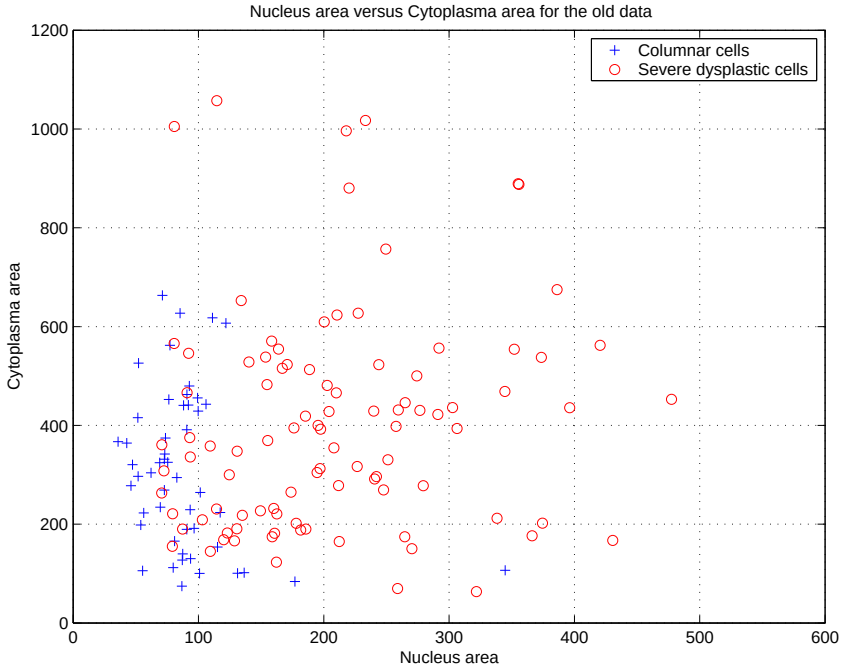

Figure 6.19: Nucleus area versus cytoplasma area for the **old data**'s columnar and severe dysplastic cells.

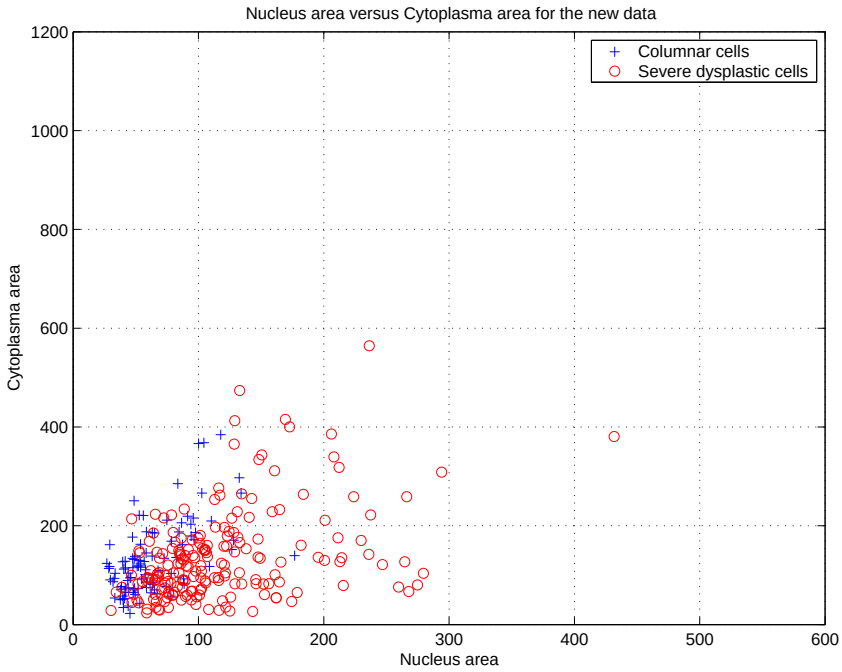

Figure 6.20: Nucleus area versus cytoplasma area for the **new data**'s columnar and severe dysplastic cells.

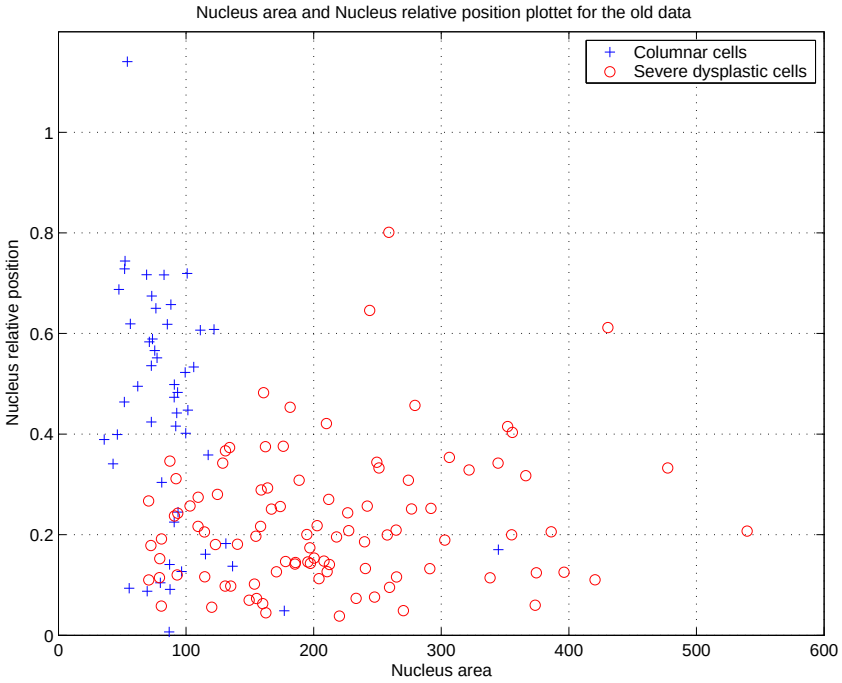

Figure 6.21: Nucleus area versus nucleus relative position for the **old data**'s columnar and severe dysplastic cells.

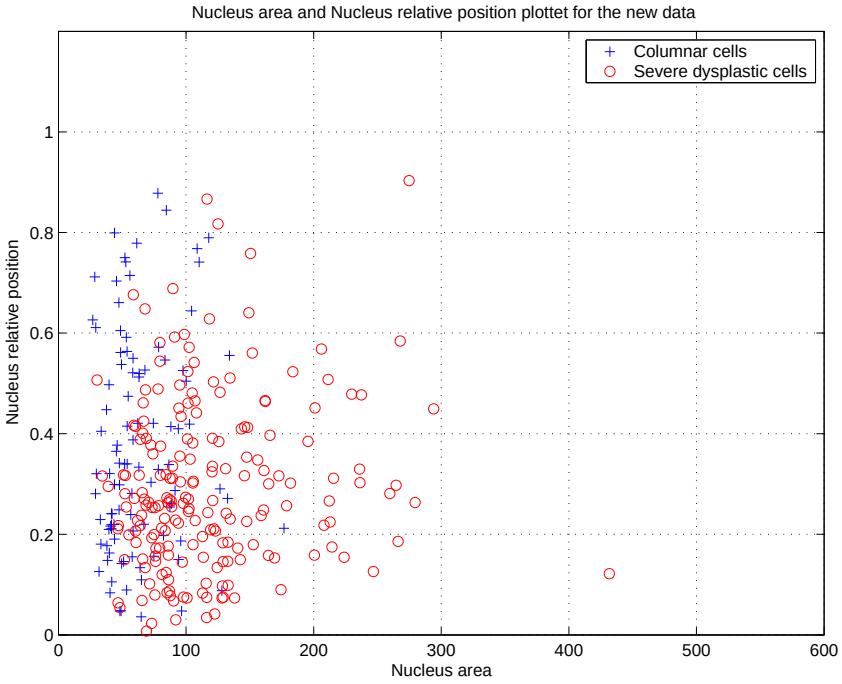

Figure 6.22: Nucleus area versus nucleus relative position for the **new data**'s columnar and severe dysplastic cells.

the cyto-technicians who made the new database were more skilled, than those who made the old database, it could explain the differences. A more skilled cyto-technician will be able to distinguish a cell better when being closer to the border of another cells diagnose than a less skilled cyto-technician, and therefore accepting cells in the new database which were not accepted in the old database. This is possible reason, not necessary the correct reason to the differences seen. It could be argued that some mistake happened in the feature extraction for the new data in this project. In section 2.1 it was shown that the features extracted with Matlab correlated very well with those extracted by the CHAMP software for the new data. The champ software was also used to extract the features from the old data in the project of Byriel (1999). A high correlation indicates that the features contain the same information, not that they have the same linear scaling. Therefore a scaling error would not be detected by the correlation. If the figures shown, had different scalings of the features between the new and old data, it could explain the conglomeration of the cells. But still the orientation of the clusters seem to be different and more overlapping in the new data compared to the old data. The scaling of the features can not influence on the overlapping seen between the columnar and dysplastic cells. Classification with FCM was made with standardized features and should therefore not be influenced by a different scaling. The results with FCM showed a poorer classification with the new data than with the old data, also indicating differences between the old and new data is more fundamental than a wrong scaling.

Chapter 7

Summary of results & discussion

Data preparation

A old and a new database was to be used in this project. The new single cell papsmear database was prepared. Feature extraction on the new data was implemented in Matlab, and compared with the same features extracted by the CHAMP software. Most features showed a high correlation. The features with the lowest correlations, was caused by different procedures in how to determine which cytoplasma belonged to a cell.

Defuzzification

In this project, the clustering algorithms Fuzzy C-means(FCM) and Gustafson kessel(GK) was used. These algorithms give a fuzzy memberships from the data to the different clusters. How the fuzzy membership is to be used and what information the clusters contain is to be decided on. 4 different defuzzification methods was proposed, one was to be selected. Defuzzification method 4 was chosen for both FCM and GK clustering, because it gave the smallest overall test errors when choosing an optimal fuzzy exponent. Defuzzification method 1 gave the best results, when a poor fuzzy exponent was chosen. FCM gained a considerable decrease in error when using the chosen method, where GK nearly did not gain anything, because of the relatively few clusters it used.

Supervised clustering

Clustering algorithms like FCM and GK normally find clusters in the data, from the natural structure in the data. The clusters is normally formed without regards to the classes of the data. Since the database in this project have a diagnose for each cell, this information could somehow be utilized. A procedure was proposed, called supervised clustering, that was easy applicable on clustering algorithms like FCM and GK. The procedure enables a better usage of the data's given diagnoses. The results with supervised clustering outperformed the results with unsupervised clustering.

Classification results

One of the goals in this project was classification on single cell papsmear pictures. Two clustering algorithms were chosen for the classification, FCM and GK. The FCM algorithm was available in the Matlab fuzzy logic toolbox, but unfortunately it was not very robust. Therefore it was modified to being more robust before used in this project. The GK algorithm was not found in the toolboxes of Matlab, and therefore implemented.

The classification were between normal and abnormal cells. Results were obtained for HCM, FCM and GK, with a 10-fold cross-validation and rerunning it 20 times. Feature selection was performed with simulated annealing, which gave a lower classification error. A summary of the results after feature selection, is for the old data shown in table 7.1 for unsupervised clustering and in table 7.2 for supervised clustering.

All results on the old data was acceptable and below 5% error, except for the unsupervised GK. The results obtained for HCM were worse than FCM as anticipated, but better than GK on the old data. The supervised clustering showed much better results than unsupervised clustering.

	Overall error	**FP%**	**FN%**
Unsupervised HCM	3.88%	4.17%	3.68%
Unsupervised FCM	1.81%	2.78%	1.17%
Unsupervised GK	9.92%	23.83%	0.86%

Table 7.1: Results for unsupervised clustering after feature selection. Old data.

	Overall error	**FP%**	**FN%**
Supervised HCM	2.62%	2.80%	2.50%
Supervised FCM	1.64%	2.02%	1.38%
Supervised GK	2.89%	4.46%	1.77%

Table 7.2: Results for supervised clustering after feature selection. Old data.

Results with the new data is shown in table 7.3 and 7.4 for unsupervised and supervised clustering, respectively. All results with the new data, had an overall error above 5%. HCM, FCM and GK could not give acceptable results. The worst results were obtained with HCM. FCM and GK gave about the samme error for supervised clustering, but GK gave the best results for unsupervised clustering. Supervised clustering again showed to give much better results than the unsupervised clustering.

HCM and FCM performed better the more clusters was used. It is possible that if more clusters was used, than in the interval investigated, an overfitting of the data could take place giving worse results. 50 and 100 clusters was chosen

	Overall error	FP%	FN%
Unsupervised HCM	8.28%	21.81%	3.41%
Unsupervised FCM	8.08%	20.97%	3.45%
Unsupervised GK	7.68%	22.66%	2.31%

Table 7.3: Results for unsupervised clustering after feature selection. New data.

	Overall error	FP%	FN%
Supervised HCM	7.86%	17.12%	4.55%
Supervised FCM	6.10%	13.89%	3.29%
Supervised GK	6.06%	13.88%	3.26%

Table 7.4: Results for supervised clustering after feature selection. New data.

for supervised and unsupervised, respectively. The many clusters chosen, does not necessarily reflect how many real clusters there is in the data. The many clusters chosen, together formed more complex shapes of the actual clusters in the data.

GK on the other hand, performed best with relatively few clusters. For supervised clustering, only one cluster for each class performed the best. But for unsupervised clustering, more clusters was needed. The poorer results obtained with GK, was assumed to be caused by the clusters being limited to hyperellipsoidal shapes. Furthermore, since GK performed best with few clusters, it showed clusters of GK did not supplement each other very well, so the hyperellipsoidal clusters together were poor at making more complex shapes than hyperellipsoidal clusters. Since FCM clusters supplemented each other well and GK could not, the difference must be caused by the shapes induced on the GK clusters.

Feature scaling

FCM does not have any scaling of the features built into the algorithm, like GK has. The clusters is found by using the distance to the different data points in the data. Therefore features having values over a large interval, will have more influence than features spanning a little interval. On this basis an investigation was made to find feature scalings for each feature separately that could decrease the error. A linearly scaling was chosen, where the standard deviation of each feature then had to be determined. Other nonlinear scalings, like GK could have been used, but would have given a more complex solution space to search through. Simulated annealing was used to find the scalings of the features.

The results for feature scaling on the old data, showed a lowering of the error compared to a standardization of $\sigma = 1$ for all features. The overall error

of feature selection gave better results than the feature scaling could obtain.

The new data, also showed feature scaling lowered the overall error compared to a normal standardization. The overall error after feature scaling and feature selection showed an equal error fro the new data.

In theory, feature selection is a subset of the solutions possible with feature scaling, and therefore feature scaling should at least be able to get just as good results as feature selection. But the problem was finding the scaling of each feature. Since the results for the old data gave a poorer result for feature scaling than feature selection, the problem of finding the right feature scalings was too complex for the simulated annealing algorithm. The feature scalings found with simulated annealing did not give equal feature scalings when running the algorithm an extra time. The scalings would for some features bee much different. This supports the argument that the solution space was too complex to search through. Therefore it probably is a good idea, using feature selection instead of feature scaling, to decrease the amount of solutions to search through.

Bias control

Cyto-technicians often favor the sick cells, when in doubt of a cells diagnose being normal or abnormal. With automated classification it therefore would be nice, if some control on the bias could be implemented. Since the memberships to each diagnose is fuzzy from the data, a simple bias was implemented. The results showed that it was possible controlling the bias. But the bias method used, did need a fuzzy membership of the data to the clusters, and therefore a non crisp fuzzy exponent was necessary.

Robustness analysis

Quite often, only the actual error obtained with a classification algorithm is of interest. The robustness of a classification algorithm is often not investigated at all. The features extracted, will have a normal varians in their features, as also is found in mother nature, which can be looked upon as noise. The process of segmentation and feature extraction can add further noise. The influence of noise was therefore investigated.

A simple robustness measure was introduced, where all features of every data point was investigated, to see how much their values each by them self could be changed in either directions, before the data point was classified incorrectly. The robustness measure therefore did not measure changes in several features simultaneous, even though the features could have some correlation.

A robustness analysis was made for both the old and new data, for FCM and GK clustering. The results from these were shown in several plots. FCM showed a good robustness for both the old and the new data, where most features could be changed with several standard deviations, before being classified

incorrectly. GK gave results that showed it tolerated much smallere changes in the features, before classifying the data points incorrectly. The poor robustness was possibly caused by the non-linear scaling happening locally for every cluster in GK and clusters fitting the data less optimal. Even though GK gave classification results similar to FCM for the new data when using supervised clustering, there was a big difference in robustness to noise. Since FCM also performed the best on the old data, the choice between FCM and GK algorithms was simple, and in favor of FCM.

Data size dependency

An investigation into how the amount of training data influenced the overall classification error, was carried out.

The results for the old data showed, that the database was big enough to achieve a below 5% in overall error. The error curve was falling slowly, when using more than 150 training data. Therefore a bigger database could give better results, even though the error would probably not be lowered much.

The results for the new data gave error curves well above 5%, which was not acceptable. The error curve of GK had become quite flat when using around 700 training data. FCM had a more decreasing error curve, so more training data could possibly had given an error below 5% for FCM. The hyperellipsoidal clusters of GK seemed to give better errors than FCM, for much of its curve. This showed that the hyperellipsoidal clusters represented the data well, when not enough training data is present to form more complex clusters in FCM. But eventually when enough training data was used, FCM outperformed GK, because their then was enough training data to form more complex shapes.

It was seen for both the old and the new data, that FCM performed better than GK when very few training data was used. This reason being, GK did need more data to be able to form the hyperellipsoidal clusters good enough from the non-linear scaling of every cluster.

Direct and Hierarchical classification

A classification on every cells diagnose was performed with direct and hierarchical classification. The direct classification being the simple procedure and hierarchical being the more advanced where the classification problems were partitioned up into smallere problems.

The hierarchical classifiers showed in average a better accuracy for each cells diagnose, than the direct classifiers could. Supervised clustering showed to perform better than unsupervised clustering for both the direct and hierarchical classifiers. The gain in error using supervised versus unsupervised clustering, was almost as big as using a direct versus hierarchical classifier. Since the supervised clustering is simple to implement and not making the classification problem harder, like having to find a lots of extra parameters, it is important

considering supervised clustering.

The average number of features used for supervised versus unsupervised clustering, showed that more features was selected for the supervised clustering. This indicates supervised clustering was better at utilizing its features. The problem of why not as good results was obtained for the new data compared to the old data, was found. The columnar cells in the new data, had a high percentage off cells classified as abnormal cells, especially as severe dysplastic cells. The other of the normal cell diagnoses was well separated from the other diagnoses. The abnormal cells showed to be hard to separate from each other, which they also is for a cyto-technician.

New data versus Old data

It was found when making the classification on each diagnose, that the problem of the poor results for the new data, likely was the columnar cells in the new data. The columnar cells were in particular, poorly separated from the severe dysplastic cells. A little investigation in what the problem could be, was then made. The columnar and severe dysplastic cells were separated from the rest of the cells, for the old and new data. Selected features was then plottet against each other. The plots for the old data, showed a better separation between the columnar and the severe dysplastic cells, than was observed for the new data. The cells of the new data was more overlapping and mixed. This was probably caused by choosing the columnar cells in a different way in the new data, than was done in the old data. This poorer separation in the new data, were likely the reason for the poor classification.

Chapter 8

Conclusion

On the new database created by Herlev hospital feature extraction was implemented in matlab, and compared with the same features extracted by the CHAMP software from the new data. A high correlation was achieved, showing the matlab feature extraction was just as good as the commercial CHAMP software.

Classification on the diagnoses normal/abnormal was carried out with Fuzzy C-means(FCM) and Gustafson-Kessel(GK) clustering for the old and new data. FCM and GK assigns data fuzzy memberships to the different clusters. Therefore 4 defuzzification methods was suggested, to convert this information into a diagnose. Defuzzification method 4 was chosen because it gave the lowest classification error.

A supervised clustering approach was suggested, to decrease the classification errors. The results with the supervised clustering compared to ordinary unsupervised clustering, gave much better results. Feature selection was successfully implemented with simulated annealing, removing features that increased the classification error.

The FCM algorithm does not change the scaling of the features. Normally all features are standardized, giving them equal importance. The possibility of using a linear feature scaling different for every feature was tried. Simulated annealing was used to find each features scaling. Since feature selection is part of the solutions possible with feature scaling (by giving the features standard deviations of 0 and 1), feature scaling should in theory at least perform as well as feature selection. The errors with the found feature scalings, was a bit higher than those obtained with feature selection. The reason probably being, the solutions space was too complex to search through, when having to find a linear scaling for each feature. Therefore, when using 20 features of this kind of data, it was better using ordinary feature selection. If the data were less complex or less features were available, it is possible feature scaling could perform better than feature selection.

False-positive% and false-negative% errors well below 5% were obtained for the old data, but the new data gave unacceptable errors well over 5%. FCM

performed best on the old data, but on the new data GK and FCM performed equal. FCM performed best with many clusters. The many clusters supplemented each other, by forming more complex cluster shapes than they each by them self could form. Supervised GK clustering performed the best when only using one cluster per class, and did not show to be able to let several GK clusters form more complex cluster shapes.

A robustness analysis was performed on FCM and GK, which showed GK had a poor robustness to noise in the features compared to FCM. The good robustness obtained for FCM, was caused by the high redundancy in information between the features. Often feature selection is performed, choosing the features giving the lowest classification error. But when removing features, the redundancy of information becomes less, resulting in a poorer robustness. It therefore would be a good idea, that feature selection in excess of lowering the classification error, also tried keeping a high robustness.

A simple bias control procedure was applied. It showed it was possible to control the bias of the classification towards the normal or abnormal cells.

It often is a problem not knowing how much data are needed when building a database. An investigation into how the amount of data influenced the classification results was made. This investigation was performed for FCM and GK clustering. The investigation showed enough data were available for the old data, and fewer could have been enough. Too high an error was obtained with the new data, but the decreasing error curve showed a lower error possibly could be obtained if more data was available. The classification results on the old data were so good that there almost was no place for improvement with a better classifier. The new data on the other hand gave much poorer results and therefore giving room for improvement with a better classifier. When considering that the new data is collected by skilled cyto-tecnicians, the confidence in the data quality is high. It therefore seems that the new data is a better choice for benchmarking classifiers, in comparison to the old data.

A classification on the exact diagnosis, was performed by a direct and hierarchical classification approach. The hierarchical approach of splitting the problem up into subproblems, gave better results than the direct classifier.

The features found in every node of the hierarchical classification, showed that supervised clustering in average could utilize more of the features than unsupervised clustering.

Erik Martin
(s960863, em@kampsax.dtu.dk)

Bibliography

Babuska, R. (1998), *Fuzzy modeling for control*, Kluwer Academic Publichers.

Bjerregaard, B. (2002), Computerstyret automatisk udstyr til screening for livmoderhalskræft, Technical report, Amtssygehuset Herlev.

Byriel, J. (1999), Neuro-fuzzy classification of cells in cervical smears, Master's thesis, Technical University of Denmark(DTU), Dept. of Automation, Bldg 326, 2800 Lyngby, Denmark.

Duda, R. O., Hart, P. E. & Stork, D. G. (2000), *Pattern Classification*, 2 edn, A Wiley-Interscience Publication.

Gonzalez, R. C. & Woods, R. E. (1993), *Digital image processing*, Addison-Wesley Publishing Company.

Jang, J.-S. R., Sun, C.-T. & Mizutani, E. (1997), *Neuro-Fuzzy and Soft Computing - A computational approach to Learning and Machine Intelligence*, Prentice Hall.

Jantzen, J. (1998), Neuro fuzzy modelling, Technical report, Technical University of Denmark, Dept. of Automation, Bldg 326, 2800 Lyngby, Denmark.

Landwehr, D. (2001), Web based pap-smear classification, Master's thesis, Technical University of Denmark(DTU), Dept. of Automation, Bldg 326, 2800 Lyngby, Denmark.

Lin, Y. K. & Fu, K. S. (1983), 'Automatic classification of cervical cells using a binary tree classifier', *Pattern recognition* **16**(1), 69–80.

Thomsen, U., Wahl, S. & Ringsholt, M. (1999*a*), *Gynækologisk Cytologi - Abnorme Fund*, 5, 1 edn, Hospitalslaborantskolen i København.

Thomsen, U., Wahl, S. & Ringsholt, M. (1999*b*), *Gynækologisk Cytologi - Normale Fund*, 4, 2 edn, Hospitalslaborantskolen i København.

Walker, R. F. & Jackway, P. (1996), 'Statistical geometric features - extensions for cytological texture anal.', *Proceedings of the 13th International Conference on Pattern Recognition* **vol. 2**, p. 790–4.

Walker, R. F., Jackway, P. & Longstaff, I. D. (1995), 'Improving co-occurrence matrix feature discrimination', *Conference Proceedings DICTA-95. Digital Image Computing: Techniques and Applications* pp. p. 643–8.

Walker, R. F., Jackway, P., Lovell, B. & Longstaff, I. D. (1994), 'Classification of cervical cell nuclei using morphological segmentation and textural feature extraction', *Intelligent Information Systems, Proceedings of ANZIIS '94* pp. p. 297–301.

Walker, R. F., Jackway, P. T. & Longstaff, I. D. (1997), 'Recent developments in the use of the co-occurrence matrix for texture recognition', *Digital Signal Processing Proceedings, 13th International Conference on Digital Signal Processing* **1**(9), p. 63–65.

Appendix A

Hierarchical clustering parameters

A.1 Old data

	Chosen features	Exponent	Clusters
Node 1234567	1,2,3,4,5,6,7,10,11,12,15,16,17,18,20	$q = 1.25$	100
Node 1234	All	$q = 1.17$	100
Node 12	3,4,5,9,10,12,15,16,17,18,19	$q = 1.6$	100
Node 34	1,4,7,12,14	$q = 1.76$	6
Node 567	1,2,3,4,5,7,9,10,11,12,13,14,15,16,17,18	$q = 1.27$	40
Node 67	1,2,3,4,5,7,11,14,15,16,17,19	$q = 1.5$	28

Table A.1: Chosen features for unsupervised hierarchical clustering with old data.

	Chosen features	Exponent	Clusters
Node 1234567	2,3,4,5,6,7,10,11,12,15,16,18,20	$q = 1.2$	50
Node 1234	All	$q = 1.17$	50
Node 12	1,2,3,4,5,6,7,8,10,12,13,14,15,16,17,18,19,20	$q = 1.38$	50
Node 34	3,4,5,6,7,8,10,12,14	$q = 1.27$	1
Node 567	1,2,3,4,5,7,9,10,11,12,13,14,15,16,17,18	$q = 1.31$	12
Node 67	1,2,3,4,5,7,11,14,15,16,17,19	$q = 1.57$	50

Table A.2: Chosen features for supervised hierarchical clustering with old data.

A.2 New data

	Chosen features	Exponent	Clusters
Node 1234567	1,2,3,4,6,7,10,12,13,14	$q = 1.13$	100
Node 123	All	$q = 1.74$	30
Node 12	5,7,11,14	$q = 1.9$	60
Node 4567	2,3,5,6,10,15	$q = 1.25$	96
Node 45	1,4,5,7,10,14,16,17,19	$q = 1.3$	96
Node 67	1,4,5,7,10,14,16,17,19	$q = 1.3$	80

Table A.3: Chosen features for unsupervised hierarchical clustering on the new data.

	Chosen features	Exponent	Clusters
Node 1234567	1,3,4,5,6,7,10,14,15,17,18,19,20	$q = 1.2$	50
Node 123	All	$q = 1.8$	8
Node 12	1,5,7,8,11,14,20	$q = 1.9$	3
Node 4567	1,2,3,4,5,10,13,17,19	$q = 1.4$	50
Node 45	1,2,4,5,6,10,14,16,18,19,20	$q = 1.27$	49
Node 67	1,2,4,5,6,10,14,16,18,19,20	$q = 1.25$	48

Table A.4: Chosen features for supervised hierarchical clustering on the new data.

Appendix B

Accompanied CD-rom

A short summarize of the cd-rom's contents:

- The file 'README.txt' summarizes the contents of the cd-rom.
- Microsoft Excel file 'new_database.xls' and 'old_database.xls' containing feature extraction results for the new and old data, respectively.
- Directory 'Stabililty results (plots)' contains plots of the robustness analysis.
- Directory 'Matlab feature extraction' containing Matlab *.m files where feature extraction is implemented on the new data.
- Directory 'Classification files' containing the Matlab *.m files implemented and used for the classification in this project. The Fuzzy C-means and Gustafson-kessel algorithms is found in this directory.
- Directory containing the pictures from the new database containing the cells and their segmented counterparts.