Master's thesis

Review of k-Nearest Neighbor advances for brain tissue segmentation

Author:
Youri Tjang

Supervisors:
Petronella Anbeek
Koen Vincken

May 18, 2009

Universiteit Utrecht

Contents

1 Introduction **4**

2 The k-NN method **5**
2.1 k-NN example 5
2.2 Brain Tissue Segmentation 8
2.3 Other appliances 10
2.3.1 Advances per category 12

3 Techniques **14**
3.1 Principal Component Analysis 14
3.2 Genetic algorithms 15
3.3 Evaluating results 16
3.3.1 Gold standard 16
3.3.2 Comparisons 16

4 Advances **19**
4.1 Learning 19
4.1.1 Features 19
4.1.2 Feature normalization 19
4.1.3 Feature weighting 19
4.1.4 Clustering feature space 20
4.1.5 Composite Distance Key 22
4.1.6 Automated learning 23
4.1.7 K 24
4.2 Searching 24
4.2.1 Distance weighting 24
4.2.2 Difference weighting 25
4.3 Computational workarounds 27
4.3.1 Feature space partitioning 27
4.3.2 Feature space clustering 28

5 Comparison **33**
5.1 Learning 33
5.1.1 Feature weighting and selection 33
5.1.2 Automated training 33
5.1.3 Optimizing K 34

5.2 Searching 34
5.2.1 Distance function 34
5.3 Computational workarounds 35

6 Discussion **36**

Glossary

k-NN	k-Nearest Neighbor.
AFFIRMATIVE	Attenuation of Fluid by Fast Inversion Recovery with Magnetization Transfer Imaging with Variable Echoes.
BTS	Brain Tissue Segmentation.
CBIR	Content based image retrieval.
CBR	Case based reasoning.
CDK	Centroid Distance Key.
CDT	Composite Distance Transform.
CSVD	Clustered Singular Value Decomposition.
DF-WKNN	Difference Weighted k-NN .
distE	Effective distance.
DS-WKNN	Distance Weighted k-NN .
FSE	fast spin-echo.
GA	Genetic Algorithm.
GM	Gray matter.
KLT	Karhunen-Loève Transform.
MRI	Magnetic Resonance Imaging.
MST	Minimum Spanning Tree.
MTC	Magnetization Transfer Contrast.
NMSE	Normalized Mean Square Error.
PCA	Principal Component Analysis.
PCE	Percentage Of correctly estimated tissue volumes.
POE	Percentage Of Overestimated tissue volumes.
PUE	Percentage Of Underestimated tissue volumes.
SI	Similarity Index.
WM	White matter.

1 Introduction

Brain Tissue Segmentation (BTS) is used in clinical research to quantify the different types of brain tissue depicted in MR images. The segmentation can be done manually by an expert (radiologist), who points out voxels and labels them. This is very time-consuming and laborous work and is hard to reproduce.

Over the years automated segmentation methods have been proposed and succesfully applied. Some of these segmentation methods use algorithms from pattern recognition. One of those methods is the k-Nearest Neighbor (k-NN) method. The k-NN method has been succesfully applied, not only in BTS, but also other appliances. This thesis will review and compare adaptations to the k-NN method which are applicable to brain tissue segmentation. We sought methods that can improve classification quality and/or improve the speed of the classification.

In Section 2 we will explain the k-NN method in general and, afterwards, how k-NN is applied in BTS more specifically. Section 3 will explain some important techniques used in the articles. In Sections 4 and 5 the advances of k-NN are reviewed and compared. In Section 6 we will propose a number of methods we think are the most promising for BTS specifically.

2 The k-NN method

The aim of the k-NN method is to classify samples based on a number of their n quantified features. Each sample is represented by a vector with n elements and is placed in an n-dimensional space, called the feature space.

2.1 k-NN example

To take an everyday example for the average Dutchman: classification of the weather conditions of a day e.g. bad weather or good weather. First we monitor the temperature and rainfall of a couple of days. This is shown in Table 1 and Figure 1 for 10 days.

	Measurement		
Day	Temperature	Precipitation	Classification
1	24	40	good
2	18	50	good
3	-12	100	bad
4	6	200	bad
5	30	20	good
6	0	150	bad
7	18	100	bad
8	4	60	bad
9	12	0	good
10	34	200	good
11	20	120	?

Table 1: Overview of articles ordered by application

To each day we manually assign a label: good or bad, based on what we think is good or bad weather. In Figure 1 the x-axis denotes the temperature and the y-axis denotes the rainfall. Each plot mark represents a day.

Now we monitored the 11^{th} day and want to know if it's good or bad. The most intuitive way is to plot our new day in the graph and look at the neighborhood of the point, like we did in figure 1. If the new day in the neighborhood of many good days, the new day is likely to be good as well. Analogous if the new day in the neighborhood of many bad days, the new day is likely to be bad as well. The amount of days we look at, around our new day, is known as k. Hence we search for the k days closest to our new day.

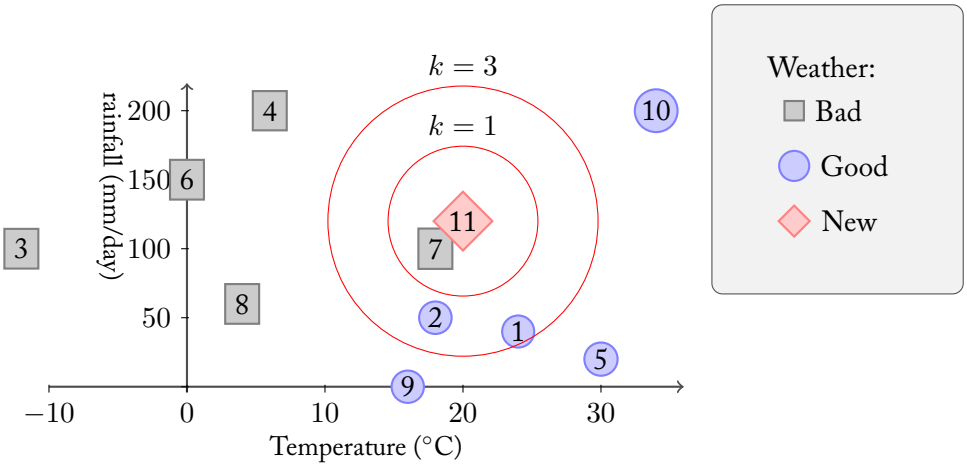


Figure 1: Example $2D$ feature space

This is how k-NN works, but of course it doesn't work only on weather conditions. The wide variety of applications is shown later on. To talk about the similar elements of the k-NN method we introduce some conventions:

- The days with corresponding measurements are known as samples or points,
- The measured entities are known as features or dimensions,
- The 10 days which are preclassified are known as the training set,
- The collection of all features is known as the feature space,
- Assigning good or bad weather to a day is known as classification, where good is a class and bad is a class.

Features This example can easily be extended. We now chose only 2 features, the temperature and the rainfall. We could also have added wind speed, clouds, humidity, etc. These features would have been added to a $3^{rd}, 4^{th}, \ldots$ axis in the graph (making it impossible to show it on paper). The feature space will then become an n-dimensional space, depending on the amount of entities measured

Feature range From the weather condition example we can see that the temperature has a range of $-10°C$ to $35°C$ and rainfall goes from $0mm$ to $200mm$. This is the range of the features. It's easy to see that when different features have ranges that differ a lot, the neighborhood of a sample will differ compared to uniform feature ranges. In an additional step the feature range can be adapted to the preference

of the application. This is known as feature weighting. Enlarging a feature range will make that feature less important, reducing it will make it more important.

Distance function The points in the neighborhood (in the feature space) of a new point are calculated by a distance function. The distance function is usually the Euclidean Distance. In $2D$ this is also known as the Pythagorean metric:

$$a^2 = b^2 + c^2$$

or

$$a = \sqrt{b^2 + c^2}$$

Given points x_1 and x_2, b is the difference in the first dimension and c is the difference in the second dimension, then a is the distance between points x_1 and x_2.

In general we don't look only at $2D$. So in ND we have points $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n)$. The Euclidean distance is then defined by:

$$d_{Euc}(X, Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Another possible distance function is the Manhattan distance, defined by:

$$d_{Man}(X, Y) = \sum_{i=1}^{n} \|(x_i - y_i)\|$$

For a new sample m we calculate the distance to each point in the learning set S and return the k points closest to m. The amount of calculations for the distance function is thus dependent on S and the amount of dimensions. The growth of the running time with increasing features/dimensions is also known as *'The Curse of Dimensionality'*. This is one of the reasons why we need to look for advances in the k-NN method .

K K is the one true parameter of k-NN . The choice of k depends on the data and the amount of training samples. A large k will smooth out the classification; it will become less susceptible to noise, but cluster boundaries become less distinct.

Election The distance function returns the k points closest to the new sample m. We will call the result set R. The points in R all have a class assigned to them. In

the case of binary segmentation the class occurring the most in R will be assigned to m. This is what is used most.

An other option is to label m with a set of values proportional to the amount of occurring classes in R, this is used in Anbeek et al. (2004) and is called probabilistic segmentation.

Samples When more samples are included in the training set, the classification becomes better; the classification error rate will drop. Theoretically, when an infinite amount of samples is selected, the error rate will approach the Bayes error rate, which is the minimum achievable error rate.

2.2 Brain Tissue Segmentation

This thesis' primary aim is to review k-NN advances for BTS. In this section we will explain how BTS works and how k-NN is used in it.

One of the reasons Magnetic Resonance Imaging (MRI) is used in research and clinical studies is because of its ability to produce multicontrast images. Multicontrast images of the brain can show different contrasts between tissues, this can help differentiating the tissues. The multicontrast images are acquired by different MRI scan sequences. Scan sequences which where used in Anbeek et al. (2004), Vrooman et al. (2007), Cocosco et al. (2003) are: T1-weighted, T2-weighted and proton density-weighted (PD). Anbeek et al. (2004) also included inversion recovery (IR) and fluid attenuation inversion recovery (FLAIR) He et al. (2005) Used an MR pulse sequence protocol called Attenuation of Fluid by Fast Inversion Recovery with Magnetization Transfer Imaging with Variable Echoes (AFFIRMATIVE). This includes Magnetization Transfer Contrast (MTC), FLAIR and fast spin-echo (FSE). The FSE images can be used for registration, because FSE is very fast it minimizes patient movement. AFFIRMATIVE is used to increase lesion-to-tissue contrast and seeks to identify and minimize the sources of false lesion classifications in segmented images(Bedell, 1996)

The contrasts of T1-weighted, T2-weighted, proton density-weighted (PD), inversion recovery (IR) and fluid attenuation inversion recovery (FLAIR) images are shown in Figure 2.

BTS aims at labeling the main brain tissues: white matter, gray matter and cerebrospinal fluid. Anbeek et al. (2004) also included: basal ganglia, ventricles and white matter lesions. This is done by assigning a set of probability values to voxels in an image. A segmented image can be used for detection of pathologic

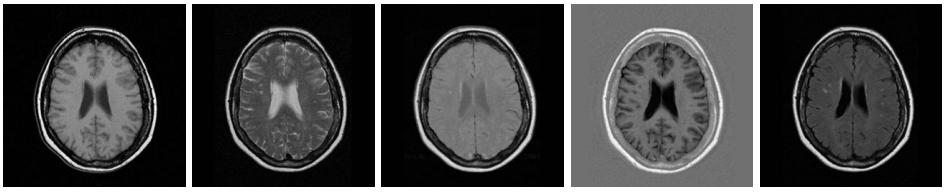


Figure 2: By altering the MR scan sequences images from the same subject are acquired with different contrasts. MR brain images from left to right: T1-weighted, T2-weighted, PD, IR, FLAIR

tissue in the human body. For example, Anbeek et al. (2004) aimed to detect white matter lesions in brain tissue.

Image preprocessing

Masking A large part of the voxels of an MR image do not contain usefull information, for example the voxels outside the body. These voxels are left out of all calculations so they do not contribute to the computational complexity. This is called masking.

Registration To improve the probability that a piece of tissue in the body will be represented by the same voxel position in different imaging modalities, the images can be registered. Registering aligns different images of the same subject. A typical application is to correct for movement of the subject in-between scans. Other applications of registration are: (1) to align scans of the same patient with are made over a large period of time (e.g. a month or year) or (2) to align scans of different patients. This can be useful to track the development of a piece of tissue.

Anbeek et al. (2004) used rigid registering, Cocosco et al. (2003), Lee and Nelson (2008) and Vrooman et al. (2007) used non-rigid registering.

The feature space is built from the voxel intensities of the images, every voxel is used as a sample. In addition, spatial information is used, because the likelihood of a certain tissue type in the brain is bound by a region. This feature space we can use with the k-NN algorithm as shown in Section 2.

Feature range Similar to the temperature and rainfall in the weather condition example the feature range of MR images acquired from different scan sequences can differ in voxel intensity range. The voxel intensity range is used as the corresponding

range of the axes of the feature space. The voxel intensity ranges can be transformed to be equal, which gives the features an equal weighting. Different feature range modifications are mentioned in paragraph 4.1.2 and 4.1.3.

Data dependency k-NN is highly data dependent (Cocosco et al., 2003). In this paper articles are discussed from different fields and therefore the methods are tested on different types of data. In this thesis we will only discuss published literature based on their tests and results.

2.3 Other appliances

BTS is a typical application of k-NN . But due k-NN to its easiness it is widely applied, as shown in this section. The advances proposed in the articles mentioned in this thesis often aim at a specific type of data or application. In table 2 we give an overview of the articles per field. These articles are discussed in more detail in Section 4.

CBIR Thomasian and Zhang (2007), Chang and Yeung (2006) used k-NN for content-based image retrieval. Content based image retrieval (CBIR) aims at searching an image from a large database of images. Metadata[1] of an image is one way to search images, but it is cumbersome to add to images, think of your own photo collection or all images on the internet. A CBIR system automatically analyses the images based on their colors, shapes, histograms, texture. This makes it possible to automatically search for images in large databases. A characteristic of CBIR is the high variability of image content. This causes the dataset to be highly nonlinear. k-NN is known for it's ability to perform well in nonlinear feature spaces. CBIR often uses a Mahalanobis distance based on the image histogram.

CBR In Case based reasoning (CBR) new problems are solved by the knowledge of old/solved problems. The system contains a knowledgebase with problems and their solutions. These are stored in a predefined format.

1. A new problem is presented and it will be rewritten into the format.
2. the system will retrieve the problems and solutions that are most similar to it.

[1]Metadata of an image can be text containing information about the content of the image

Field	Authors
Medical Imaging	Anbeek et al. (2004)
	Cocosco et al. (2003)
	He et al. (2005)
	Lee and Nelson (2008)
	Vrooman et al. (2007)
CBIR[a]	Chang and Yeung (2006)
	Castelli and Thomasian (2003)
	Thomasian and Zhang (2005)
	Thomasian and Zhang (2007)
CBR[b]	Ahn and Kim (2008)
	Zuo et al. (2008)
Gene expression	Xiong and wen Chen (2006)
General classification	Zhuang et al. (2007)
	Yu et al. (2007)
	Jagadish et al. (2005)

Table 2: Overview of articles ordered by application

[a]Content-Based Image Retrieval
[b]Case Based Reasoning

3. It can construct a solution to the new problem.

4. The solution can then be validated by the user

5. If needed the solution is added to the system

In step 2 a k-NN algorithm can be used to search for the similar problems, this is what is used in Ahn and Kim (2008). CBR can be applied to areas in which multi-variate decision making is common, including manufacturing, finance and marketing (Ahn and Kim, 2008).

general classification Others, like Jagadish et al. (2005), Zhuang et al. (2007), Zuo et al. (2008), Zhang and Zhou (2007)and Hu et al. (2008) tried to look at k-NN classification in a more general way and sought optimizations that don't aim

at one application or type of data specifically. They assume a dataset with high dimensionality, because this is a large performance issue of k-NN as explained in Section 2. Because we think these methods may prove to be advantageous with BTS, these methods have been included in this thesis.

2.3.1 Advances per category

The articles propose optimizations or extensions to the k-NN method . Some articles combine multiple methods at once. An overview per optimization-category is given in table 3.

Category	Subcategory	Autors	Method
Learning	Feature Weighting and selection	Anbeek et al. (2004)	zero-mean
		Cocosco et al. (2003)	range-matching
		Vrooman et al. (2007)	comparing
		Lee and Nelson (2008)	Genetic algorithm
	K	Ahn and Kim (2008)	Genetic algorithm
	Feature space Clustering	Jagadish et al. (2005)	iDistance
		Zhuang et al. (2007)	CDK
	Automated learning	Cocosco et al. (2003)	pruning
Searching	Distance function	Chang and Yeung (2006)	kernel-based
		Dudani (1976)	Distance weighting
		Jagadish et al. (2005)	iDistance
		Yu et al. (2007)	iDistance
		Zhuang et al. (2007)	iDistance
		Zuo et al. (2008)	Difference weighting
	Feature reduction	Hu et al. (2008)	PCA
		Thomasian and Zhang (2007)	KLT[a]
	Graphs and trees	Cocosco et al. (2003)	MST[b]
		Jagadish et al. (2005)	B^+-tree
		Zhuang et al. (2007)	B^+-tree
		Thomasian and Zhang (2007)	OP-tree
		Yu et al. (2007)	B^+-tree
Workarounds	Partitioning	Thomasian and Zhang (2007)	disk resident tree
		He et al. (2005)	FSP[c] and PCA

Table 3: Overview of articles ordered by advance type. Note that an article can have multiple optmizations and can occur multiple times in the table.

[a] Karhunen-Loève transform
[b] Minimum Spanning Tree
[c] feature space partitioning

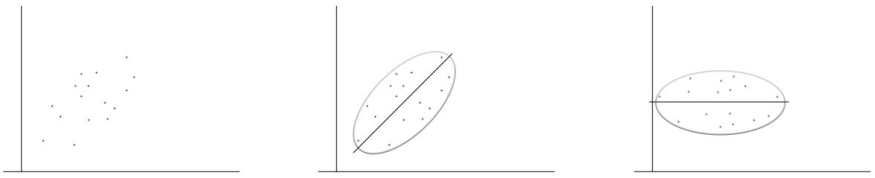
3 Techniques

The articles discussed in this thesis have used a wide variety of techniques. In this section we explain some of the important techniques.

3.1 Principal Component Analysis

The variability of dimensions in a high dimensional data set can differ a lot. A populair method to analyse and reduce the dimensionality while preserving the dimensions in which the variance is high is called Principal Component Analysis (PCA). This method is also refered to as the Karhunen-Loève Transform (KLT), for example in Thomasian and Zhang (2007). In this section we will shortly describe the PCA method, a more elaborate description is given in Pearson (1901)

First the feature space is centered by translating the coordinate system origin to the mean of the sample points. A kernel matrix K is made by augmenting the M data vectors $(x_1, x_2, \ldots, x_n)$ in a row-wise manner. K is now a matrix of dimensions $M \times n$. Then make a covariance matrix $\hat{K}$. Then, by solving the eigenvalue equation for $\hat{K}$, we get eigenvalues: $\epsilon_1 \geq \ldots \geq \epsilon_p$ and eigenvectors: $\alpha_1, \ldots, \alpha_p$. Where p is the dimensionality. When p is chosen smaller than the row rank of K, insignificant dimensions are ignored. The k principal components can now be choosen.



(a) Original $2D$ dataset (b) 1^{st} Principal Component (c) Axis Aligned

(d) Data projected (e) Reduced to $1D$

Figure 3: PCA: Step-wise data dimension-reduction.

Intuitively, a dataset can be seen as clustered points shaped like an ellipsoid (Figure 3a and 3b). In some directions the variability of the data is high and in other directions it is low. PCA finds the directions of the high variabilities. By projecting the data on those directions, the directions with low variability are removed (Figure 3e). In k-NN this results in a lower computational complexity, since the feature space is of lower dimension after PCA.

3.2 Genetic algorithms

The one true parameter of k-NN is k, but with the introduction of different optimizations of the method, it is possible other parameters are introduced. The values of the parameters can be heuristically set like in Cocosco et al. (2003). But it is also possible to automatically search for the optimum. Genetic Algorithms (GAs) are build on principles of genetics and evolution and implement 'survival of the fittest' and 'natural selection', with the aim of finding a heuristically optimum in the parameters.

Suppose we have a set of d values $\mathcal{S} = \{x_i : x \in F, 1 \leq i \leq d\}$ where F denotes all possible feature values and d the dimensions. The powerset of $\mathcal{S}$, $\mathscr{P}(\mathcal{S})$ contains all possible combinations of values. A GA searches for $\mathcal{S}_o \in \mathscr{P}$, which is heuristically optimal. This is done in iterations, which are called generations. We will describe the a GA in steps:

(Step one) In the first generation we chose a class of subsets of $\mathcal{S}$, $\mathscr{A}(\mathcal{S})$. In which the values of $\mathcal{S}$ are randomly chosen.

(Step two) With a fitness function f, we can calculate the performance of $\mathcal{S}$. In the case of k-NN f can be the distance function applied on a learning set and a test set and then be compared to the gold standard. The element of $\mathscr{A}_g$ that performs best $\mathcal{S}_i$ will be the prototype in the next generation.

$$\mathcal{S}_i = \{x : x \in \mathscr{A}, \text{argmax} f(x)\}$$

(Step three) In the next generation the values of $\mathcal{S}_i$ are the base for new sets $\mathscr{A}_{g+1}$. The values per set are differentiated by:

- altering only some values by random, called mutation or
- interchanging different subsets of S by random, called cross-over.

(Step four) Repeat steps 2 and 3 until a terminating condition is met, which can be for example:

- The fitness function reached a minimum score,
- a pre-set number of generations is reached or
- the result of f doesn't change enough

The generations cause the fitness function to converge to a heuristically optimal result, it is the best the algorithm found with the restrictions. The algorithm is terminated and will return the set which it found to be the best.

A problem with the algorithm arises when the converging gets stuck in a local optimum, but global sub-optimum. This can be overcome by cross-overs and mutations that change the value set a lot.

3.3 Evaluating results

3.3.1 Gold standard

For evaluation of a classification of a test set one can let human experts classify the test set and compare the results of the k-NN algorithm and the experts. The classification done by human experts is called the gold standard.

The "Leave-one-out" strategy is often used to validate the training set. The gold standard set minus one sample is used for the training, afterwards the sample which was left out is used to validate the training set by classifying it.

3.3.2 Comparisons

To compare classification a test set is usually already classified (the Gold standard), so the true classification is know. The k-NN algorithm is run on a training set, which doesn't contain samples from the Gold standard set, and the results are compared to the gold standard.

The most common way to compare the gold standard and the k-NN classification is the Similarity Index (SI). To calculate the SI you first have to count the:

- True-positives(TP): the amount of correctly classified positive voxels
- True-negatives(TN): the amount of correctly classified negative voxels
- False-positives(FP): the amount of incorrectly classified positive voxels
- False-negatives(FN): the amount of incorrectly classified negative voxels

Now we can calculate the Sensitivity (True positive Fraction), Specificity and False Positive Fraction).

$$\text{Sens} = \text{TPF} = \frac{\text{TP}}{\text{TP} + \text{TN}}$$

$$\text{Spec} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

$$\text{FPF} = 1 - \text{Spec} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

The similarity measures are the similarity index (SI), overlap fraction (OF), extra fraction (EF) also known as the percentage overestimated tissue volumes (POE), percentage underestimated tissue volumes (PUE), percentage correctly estimated tissue volumes (PCE):

$$\text{SI} = \frac{2(\text{Ref} \cap \text{Seg})}{\text{Ref} + \text{Seg}}$$

$$\text{OF} = \frac{\text{Ref} \cup \text{Seg}}{\text{Ref}}$$

$$\text{POE} = \text{EF} = \frac{\overline{\text{Ref}} \cap \text{Seg}}{\text{Ref}}$$

$$\text{PUE} = \frac{\text{Ref} \cap \overline{\text{Seg}}}{\text{Ref}}$$

$$\text{PCE} = \frac{\text{Ref} \cap \text{Seg}}{\text{Ref}}$$

Anbeek et al. (2004) used probabilistic segmentation and they adapted their evaluation method correspondingly with the Probabilistic Similarity Index, Probabilistic Overlap Fraction and Probabilistic Extra Fraction:

$$\text{PSI} = \frac{2P_{x,gs=1}}{\sum 1_{x,gs=1} + \sum P_x}$$

$$\text{POF} = \frac{\sum P_{x,gs=1}}{\sum 1_{x,gs=1}}$$

$$\text{PEF} = \frac{\sum P_{x,gs=0}}{\sum 1_{x,gs=1}}$$

where:

- $\sum P_{x,gs=1}$: Sum of all voxel probabilities, where in the gold standard the voxel value = 1,
- $\sum P_{x,gs=0}$: Sum of all voxel probabilities, where in the gold standard the intensity value = 0,
- $\sum 1_{x,gs=1}$: Sum of all voxels in the gold standard,
- $\sum P_x$: Sum of all probabilities in the probability map.

Probabilistic segmentation is a more generic way to express the result of a k-NN query. Instead of the result being the one class that occurs most in the k-nearest neighbors, with the probabilistic method the result is the set of fractions the k-nearest neighbors consist of. From this result a binary segmentation can be constructed by thresholding. Or a probability map can be constructed by making images per class of the result set.

4 Advances

4.1 Learning

This section is about the initial filling of the feature space also known as the learning phase. The quality of the classification is dependent on the learning set. A carefully chosen set can result in better classifications.

4.1.1 Features

k-NN Is known for its sensitivity to parameter scaling and the presence of irrelevance or noisy features (Lee and Nelson, 2008). There are different methods proposed which aim at optimizations based on features, some of those methods are mentioned in this section. This section is about the optimization based on the feature space.

4.1.2 Feature normalization

The multi-spectral data from MRI has different feature ranges, Anbeek et al. (2004) used variance scaling in order to align the features. The variance scaling was achieved by:

$$x' = \frac{x - \bar{x}}{\sigma}$$

in which x is the feature value, $\bar{x}$ is the mean feature value and σ is the standard deviation.

Cocosco et al. (2003) uses a method called histogram range-matching to normalize the features. It cuts off a pre-set percentile from the absolute maximum and minimum[2].

Vrooman et al. (2007) compares that method to a method that rescales to have zero mean and unit variance. After testing the two methods it is concluded that the range-matching method (Cocosco et al., 2003) gives a better result, as shown in section 5.1.1.

4.1.3 Feature weighting

Lee and Nelson (2008) implemented feature weighting and feature selection with the use of a GA (Ga's are explained in section 3.2). Their data consists of conventional MRI data like T1 and T2 weighted imaging and diffusion, perfusion and

[2]Cocosco et al. (2003) heuristically determined that $4/0.5/4\%$ percentiles are adequate for T1/T2/PD mr images,

Weight	n_{dim}	Mean $\pm$ standard deviation		
		Sensitivity	Specificity	Az
none	38	0.73 ± 0.16	0.75 ± 0.07	0.77 ± 0.10
1-bit	7	0.78 ± 0.18	0.79 ± 0.06	0.80 ± 0.08
2-bit	11	0.79 ± 0.14	0.78 ± 0.06	0.81 ± 0.07
3-bit	17	0.78 ± 0.15	0.78 ± 0.06	0.79 ± 0.06
4-bit	23	0.72 ± 0.22	0.80 ± 0.07	0.78 ± 0.12

Table 4: Comparison of k-NN classifiers: without feature optimalization, 1-bit, 2-bit, 3-bit and 4-bit feature weights. With Az as the area under the ROC curve. (Lee and Nelson, 2008)

spectroscopic images. Altogether they built a feature space of 38 dimensions. The aim of the GA is to find optimal weights for the features.

For a set $\mathcal{W}$ of feature weights: $\mathcal{W} = \{w_1, w_2, \ldots, w_d\}$, w_i is randomly chosen and weights of 0 are allowed (in which case a feature is filtered out). In their experiments they used an n-bit representation of the weights, where $n \in \{1, 2, 3, 4\}$, this is the precision of the weight value.

The weights are used in the distance function, for 2 points $X = (x_1, x_2, \ldots, x_d)$ and $Y = (y_1, y_2, \ldots, y_d)$:

$$d_w(X, Y) = \sqrt{\sum_{i=1}^{d} (w_i X_i - w_i Y_i)^2}$$

Take a class of random $\mathcal{W}$-sets. From the sets the performance is calculated as explained in section 3.2. The aim is to find a feature weight set in which the weights are heurisitically optimal. A known drawback of GA's is that the result can converge to a local optimum, but global sub-optimal solution. To overcome this problem they included highly disruptive crossover, mutation events and many generations.

Their results are shown in Table 4.

4.1.4 Clustering feature space

Jagadish et al. (2005) introduced iDistance. The high-dimensional feature space is partitioned in clusters and sub-clusters using k-means clustering. Then a reference point, which usually is the cluster centroid, is determined per (sub-)cluster. For points in each cluster the distances from those points to the cluster reference point

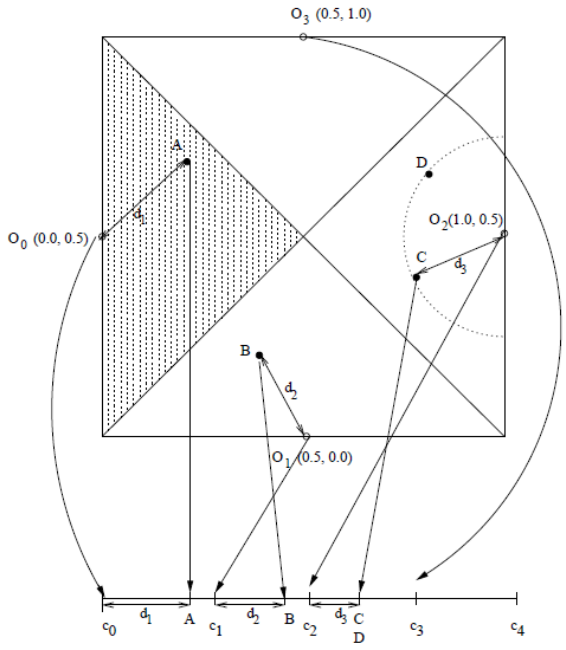


Figure 4: Mapping of points in feature space to their distance to reference point.

are calculated. Per cluster a 1-Dimensional space is stored by the distances as shown in figure 4

Afterwards the points are indexed using a B^+-tree. This is shown in figure 5

A B^+-tree is a tree datatype with multiple (a pre-set minimum and maximum) children per node.

Each internal node stores keys (other reference points). Only external nodes stores data (real sample points). k-NN Search can now be done on the B^+-tree. This is efficient because only part of the feature space is searched.

A drawback of this method is that it is very dependent on the quality of the clustering of the data (Zhuang et al., 2007). Cocosco et al. (2003) has shown that brain tissue data is generally very clustered, thus this method can be applied to brain tissue data.

Yu et al. (2007) used the *iDistance* and made some extensions. They proposed PCA (explained in section 3.1) to find the most important dimensions and possibly reduce the feature space to those dimensions.

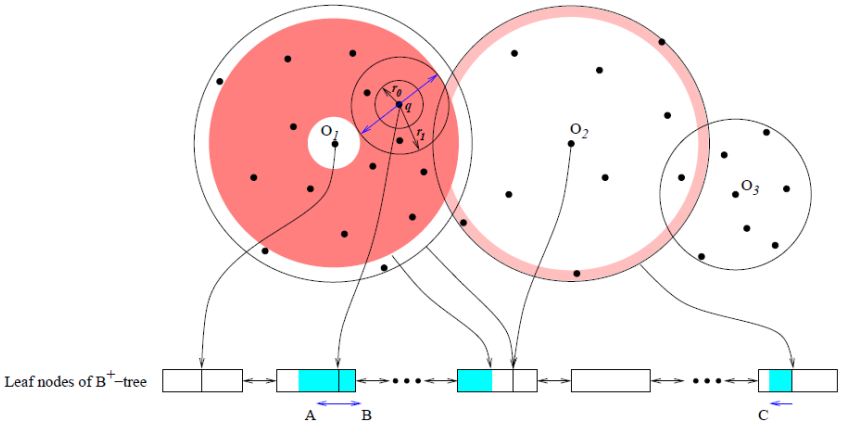


Figure 5: search regions for NN query q

4.1.5 Composite Distance Key

Zhuang et al. (2007) proposed an indexing method which makes use of a Composite Distance Transform (CDT). The first steps of the method are similar to those of the iDistance method used in Jagadish et al. (2005). The difference is that Zhuang et al. (2007) doesn't prune the points by reducing it to a single-dimensional space. The clusters are hyperspheres which are layered (sliced) by radius, this is shown in Figue 6.

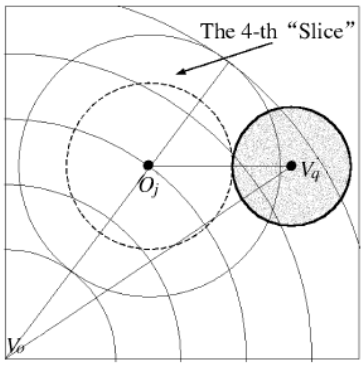


Figure 6: Corresponding slices of a cluster hypersphere

With the use of a Centroid Distance Key (CDK) which contains information about the centroid radius, centroid distance and start distance (an absolute space

origin), they make an index. Pre-query they have a hash-table of B^+-trees each containing clusters. The query will be mapped to the correct hash-table index, resulting in the query only having to search in a subset of the clusters. The search will make a searchsphere, starting out with a small radius and iteratively enlarging to eventually contain at least k data points S. From those data points, the $(|S| - k)$ farthest points are removed. The result is the k nearest neighbors data points.

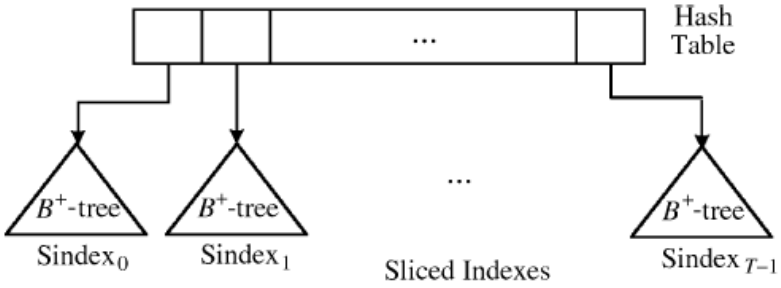
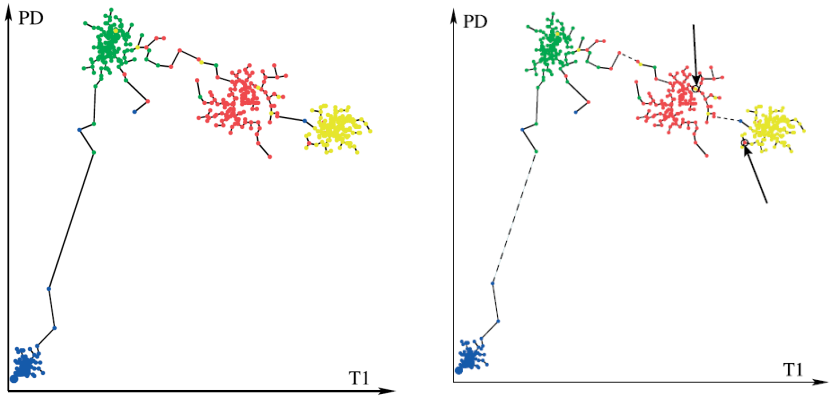


Figure 7: CDT indexing architecture.

4.1.6 Automated learning

The points the algorithm is trained with will define how new samples are classified. These points can be selected manually, by an expert, but this is prone to inter-, intra-observer variability and labor-intessive and time-consuming (Vrooman et al., 2007). To overcome these problems Anbeek et al. (2004), Cocosco et al. (2003) and Vrooman et al. (2007) described methods that use automated training and selection of training samples.

Cocosco et al. (2003) applies a method that has similarities to the previous clustering methods. They have a priori information that the amount of clusters is set to N. Each cluster has a 1-on-1 relation to a class. They make a Minimum Spanning Tree (MST) (Kruskal, 1956) of the points in feature space. The MST is broken up into N clusters by iteratively removing the longest edges (inspired by: Duda et al. (2001)). Each point in a cluster should now be of the same class. If a point is not of the same class it is pruned. Because of the extra a priori knowledge put into the system (the amount of classes and their clusterability) this is called semi-supervised training.



(a) Feature space with MST

(b) 2 Arrows pointing to samples which are pruned

Figure 8: MST, (Cocosco et al., 2003)

4.1.7 *K*

The only real parameter of the k-NN method , k, also influences the classification (Anbeek, 2005). Ahn and Kim (2008) used a GA(explained in section 3.2) to optimize k. They tested their algorithm on purchase prediction and stock market prediction, so we won't try to compare their results for reasons mentioned in section 2.2. They, however, did show an improvement. A drawback of this method is the high computational complexity. The method is inefficient compared to k-NN with a heuristically chosen k. This method can be useful when a high prediction accuracy is required.

4.2 Searching

This section is about extensions of k-NN that aim at improving the Searching, also called querying. When we have a new sample we want to classify, the classic k-NN method will search the nearest neighbors in feature space. When these are found the new sample is classified as the class occurring the most in the neighboring samples. The most important aspect with querying is the distance function. This function determines the quality of the classification.

4.2.1 Distance weighting

Dudani (1976) introduced an extra weighting for deciding the class of a sample x. This metric is based on the distance between x and it's k-nearest neighbors. For deciding the class of x that distance is taken into account. This method is called Distance Weighted k-NN (DS-WKNN).

The k nearest neighbors $X = (x_1, x_2, \ldots, x_k)$ of a new sample point x, where X is sorted by increasing order according to the distance between a x and $x_i \in X$. A weight w_i is assigned. Where w_i is:

$$w_i = \frac{d(x, x_k) - d(x, x_i)}{d(x, x_k) - d(x, x_1)}$$

The DS-WKNN will give neighbors closer to x a higher weighting, so that they contribute more to the classification.

4.2.2 Difference weighting

Zuo et al. (2008) made an extension on DS-WKNN, Difference Weighted k-NN (DF-WKNN). Not only are they looking at the distance of a sample and it's neighbors, but also at the correlation of the neighbors. This is shown in figure 9.

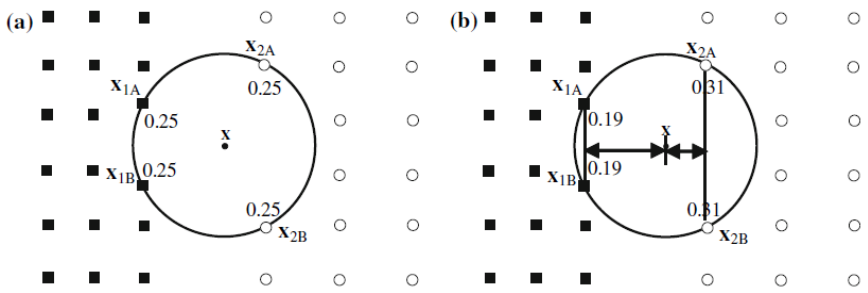


Figure 9: Assignment of weights to the nearest neighbors using: a) the distance-weighted knn rule and b) the difference weighted knn rule.

For a new sample x, get it's k nearest neighbors $(x_1, x_2, \ldots, x_k)$. Then the difference between the nearest neighbors $D = (x_1 - x, x_2 - x, \ldots, x_k - x)$ is calculated. The weight is calculated by solving the system of linear equations:

$$(DD^T + \eta \text{tr}(DD^T)/k)w = 1$$

Where $\eta = 10^{-3} \sim 1$, which is a regularization parameter. And tr(DD^T) is the trace of the matrix DD^T.

They extended DF-WKNN to a kernel version KDF-WKNN. The kernel function maps the space to a higher dimensional space and performs the non-kernel, linear function there. This is the equivalent of a nonlinear function. The result is a function which takes into account the nonlinear structure information. Figure 10 shows a kernel-version of PCA.

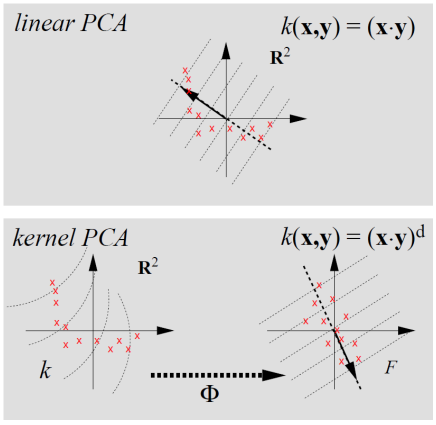


Figure 10: (Müller et al., 2001) An example showing the kernel version of PCA. The function works in a nonlinear way.

Two populair kernel functions are:

- Radial basis function (RBF) kernel:

$$k(x, x') = exp(-|x - x'|^2/2)$$

- polynomial kernel:

$$k(x, x') = (1 + x \cdot x')^d$$

Where x and x' are 2 samples.

The kernel distance function is then defined as:

$$k(x, x') = k(x, x) - 2k(x, x') + k(x', x')$$

They experimented with 30 datasets from the UCI Machine Learning Repository (Blake and Merz, 1998). They compared the results of WDF-WKNN, KNN, DS-WKNN and some other non-knn classifiers. The overall average classification

rate of KDF-WKNN is 87.67% , KNN is 85.19% and DS-WKNN is $85,34\%$. KDF-WKNN also outperforms the non-k-NN classifiers. The details are published in Zuo et al. (2008).

Although KDF-WKNN has shown to improve the classifiaction accuracy, the method suffers from a higher computational coplexity. The complexity of KDF-WKK is $O(md + k^2d + k^3)$ compared to k-NN which is $O(md)$, where m is the size of the learning set, d is the amount of dimensions and k is the amount of nearest neighbors.

4.3 Computational workarounds

The running time of the k-NN method depends greatly on the amount of features and samples that are selected. The features make up the dimensions in a N-dimensional hypercube of feature space. The amount of memory it takes to store this hypercube will be large which high-dimensional hypercubes. Also the time it takes to build the cube can be long, which is disadvantageous when rebuilding the cube. In this section some methods are described which deal with these problems.

4.3.1 Feature space partitioning

Building a high-dimensional feature space will be demanding on the memory requirements of a PC. According to He et al. (2005) a four-dimensional feature space with dynamic intensity range of $[0, 255]$ has a total size of $256^4 = 4GB$. Higher dimensional feature spaces are not manageable with a PC (the average PC anno 2009 has 4GB of RAM). To be able to run the k-NN method with high-dimensional feature spaces He et al. (2005) proposed a method that partitions the feature space in equally divided hypercubes, called subspaces. The division has the advantage that the feature space doesn't have to be stored as a whole in the computer RAM at once. Because a subspace is smaller, it can be loaded on a PC. For example if the feature space mentioned before is subdevided into $4^4 = 256$ subspaces, each subspace will have the size of 16MB.

By serializing the feature space it can be stored on the hard drive. The feature space can be read out in a single lookup, thus minimizing time. (A single lookup on a hard drive is faster than partial lookups). Within a subspace k-NN search can be done. k-NN search within a subspace will only contain the points in that subspace, thus saving a lot of time.

Borders A problem arises at the borders of each subspace; suppose we are looking for the nearest neighbors of a point near the border. The algorithm will only search within the subspace the point is in, but it may have neighbors in adjacent subspaces. Therefore there is an overlap between the subspaces. The overlap has the size of the Effective distance (distE). For each point the distE is assumed, points beyond the distE are assumed not to influence the point(Warfield (1996) and Cuisenaire and Macq (1999)). An overlap at the borders of distE thus minimizes the classification error.

Results For testing the method He et al. (2005) used MR images acquired with the AFFIRMATIVE pulse sequence. They compared a data set of 12 segmented MR brain images to reference volumes generated by experts (Gold standard) using 4 feature dimensions. And they used a data set of 19 segmented MR brain images to reference volumes generated by experts (Gold standard) using 3 feature dimensions. They compared the SI, Percentage Of Overestimated tissue volumes (POE), Percentage Of Underestimated tissue volumes (PUE) and Percentage Of correctly estimated tissue volumes (PCE) of the two setups. The results are shown in figure 11 and 12. Overall, the results show the setup using 4 features has higher scores.

The use of 4 features compared to 3, shows an improvement in segmentation of White matter (WM) and Gray matter (GM). Because of the feature space partitioning the subspaces became small enough to fit in the PC memory. It can be concluded that the method makes it possible to include more features and still run on a PC.

4.3.2 Feature space clustering

Like He et al. (2005), Thomasian and Zhang (2007) uses a partitioning algorithm on the learning data-set.

Clustering Thomasian and Zhang (2007) first clusters the data with the aim of getting class-homogeneous clusters. This is achieved with Clustered Singular Value Decomposition (CSVD) Castelli and Thomasian (2003) and Thomasian et al. (1998). Many datasets of various application domains show local correlations. The clustering separates the heterogeneous data into clusters which itself are more homogeneous. As shown in Cocosco et al. (2003) this also holds for BTS-data. SVD or PCA will introduce less error on homegeneous data. For clustering, the k-means

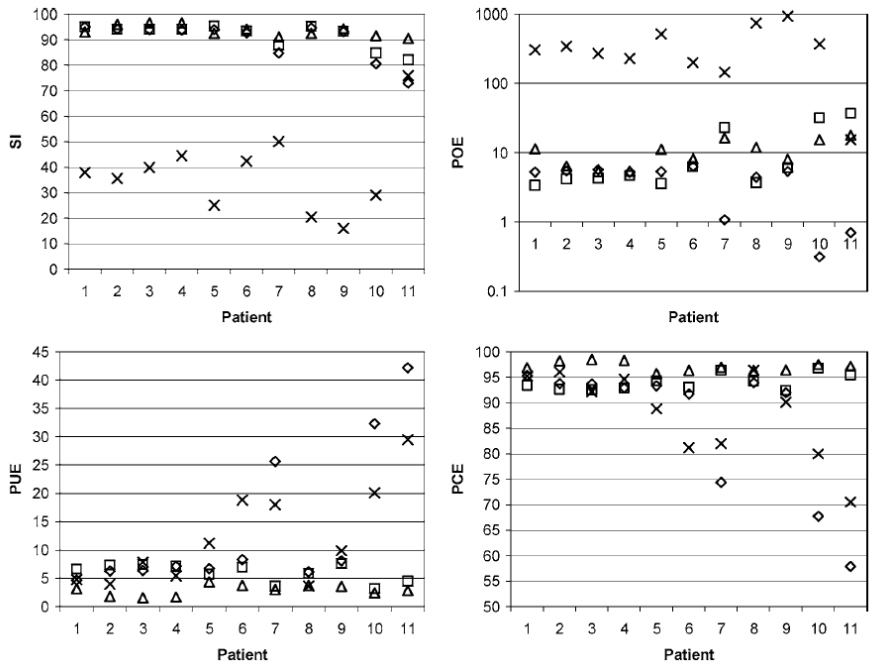


Figure 11: Results from He et al. (2005) using 4 features: Quantitative comparison of AFFIRMATIVE images. The symbols $\diamond$, $\square$, $\triangle$ and $\times$ represent gray matter, white matter, CSF and lesion respectively.

clustering algorithm (Steinhaus, 1956) is used. As a post-condition each cluster size has to fit into the main memory.

PCA On each cluster PCA is performed, to reduce the dimensionality. After the PCA the eigenvalues are obtained and sorted in nondecreasing order. The eigenvalues and corresponding feature dimensions are iteratively eliminated until the overall Normalized Mean Square Error (NMSE) reached a set target:

Thomasian and Zhang (2007):" Given an $M \times N$ matrix X of M images with N features, PCA computes the covariance matrix $C = X^T X/M$, which is decomposable as $C = V \Lambda V^T$. The eigenvectors of matrix V are the principal components and Λ is a diagonal matrix of eigenvalues: $\{\lambda_1, \lambda_2, \ldots, \lambda_N\}$, whose elements are assumed to be in nondecreasing order. The NMSE is:

$$\text{NMSE} = \frac{\sum_{h=1}^{H} m_k \sum_{i=n_h+1}^{N} \lambda_{h,i}}{\sum_{h=1}^{H} m_h \sum_{i=1}^{N} \lambda_{h,i}}$$

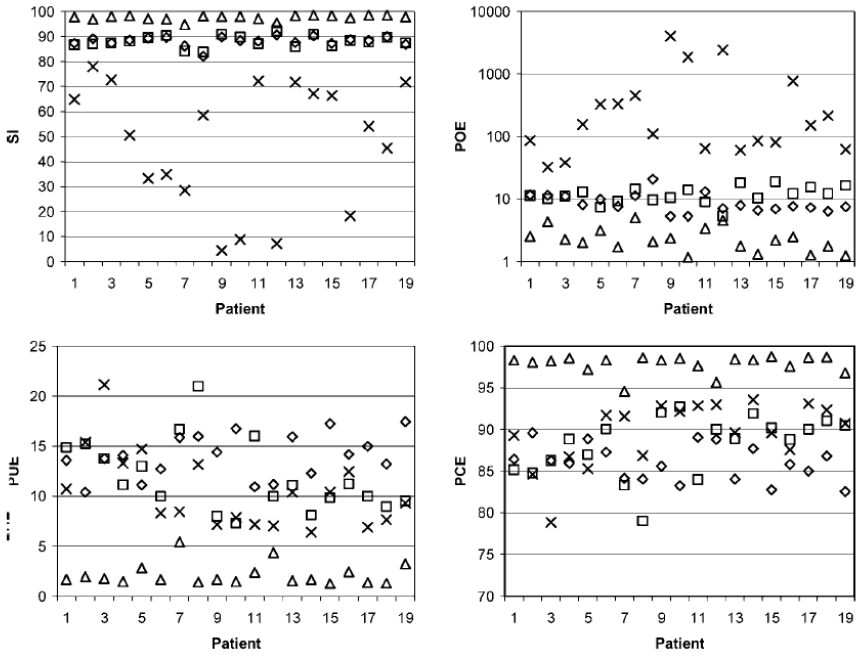


Figure 12: Results from He et al. (2005) using 3 features: Quantitative comparison of AFFIRMATIVE images. The symbols $\diamond$, $\square$, $\triangle$ and $\times$ represent gray matter, white matter, CSF and lesion respectively.

where m_h is the number of points in the hth cluster, n_h is the number of retained features in that cluster, $\lambda_{h,i}$ is the eigenvalue corresponding to the ith feature in the hth cluster."

Indexing The clusters are indexed by their centroid position and radius. The radius in this case being the distance from the centroid to the farthest point in the cluster.

OP-tree The indexed clusters can be stored on the computer hard disk by serialization. Storing the feature space on the hard disk has the advantage that the feature space can become bigger than the computer RAM and also isn't volatile like the RAM. The feature space is optimized so clusters can be retrieved at low cost using the indices. An example is shown in figure 13 (Thomasian and Zhang, 2007)

k-NN search Within the CSVD feature space searching goes as follows:

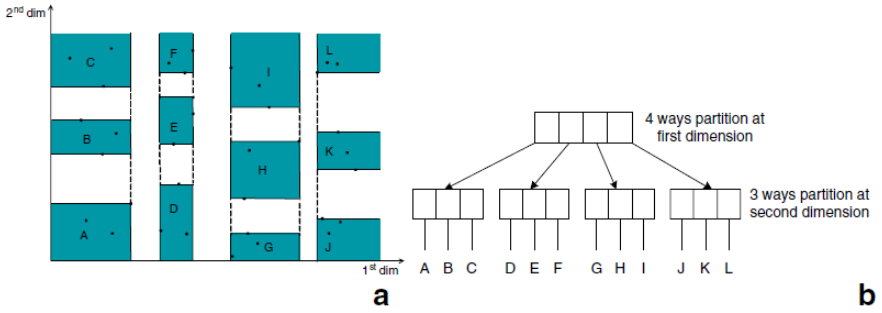


Figure 13: a) The partition of a two-dimensional OP-tree. b) The corresponding hierarchical structure.

- For a new point p, find the cluster it belongs to. That is, find the cluster $\mathcal{C}'$ which centroid has the smallest distance to p.
- Order other clusters on their distances.
- Compute the distances to the k-Nearest Neighbor points in the cluster.
- Compute the maximum distance d_{max}.
- Now check other clusters $\mathcal{C}^h \in \mathcal{C}$, for an overlap with $\mathcal{C}'$, such that:

$$d_{max} > \text{dist}(P, \mathcal{C}^h)$$

  - Perform a range query with radius d_{max} centered at P (A range query will retrieve all points within distance d_{max} of a query point).

  - Compute the original distances of all retrieved points, and if their distance is less than dmax insert the point into the heap for k nearest neighbors and update d_{max}.

Results Experiments were conducted on datasets: TXT55 (real-world texture dataset with 79,814 points and 55 dimensions) and SYNTHETIC64 (synthetic dataset with 99,972 points and 64 dimensions). Results are shown in figure 14. They compared the linear scan (normal k-NN) to the CSVD method, with dif-

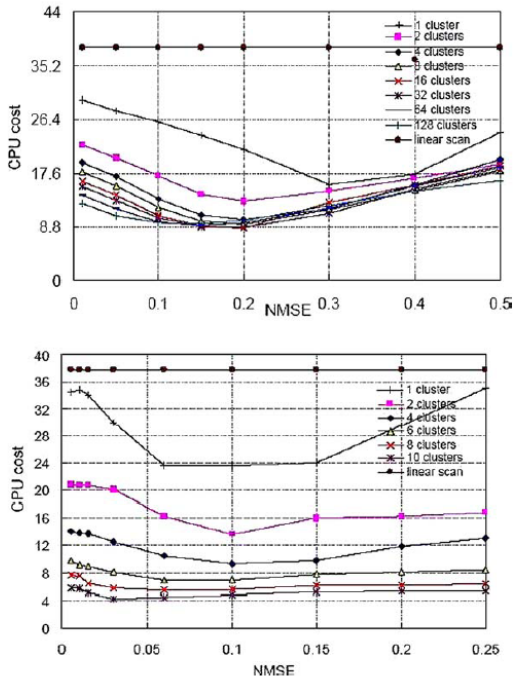


Figure 14: Results from Thomasian and Zhang (2005). Top: TXT55 dataset, bottom: SYNTHETIC64 dataset

ferent cluster sizes and different NMSE's). SVD (1 cluster) shows significant improvement over the linear scan. CSVD shows an ever bigger improvement. With a low NMSE, little dimensions are eliminated. As more dimensions are eliminated the error increases and CPU Cost decreases to some minimum. With higher NMSE the CPU increases, which can be explained by the increased amount of range queries executed because of the higher amount of false points are retrieved.

5 Comparison

Some methods aim at computational speed, others at the quality of the classification.

5.1 Learning

5.1.1 Feature weighting and selection

A number of groups (Anbeek et al. (2004), Cocosco et al. (2003), Vrooman et al. (2007) and , Lee and Nelson (2008)) proposed methods for feature weighting and/or selection. Vrooman et al. (2007) compared the methods proposed in Anbeek et al. (2004) (zero-mean-unit-variance) and Cocosco et al. (2003) (range mathing). They concluded the range matching method performs better, however a drawback of this method is that it is data-dependent, opposed to the zero-mean method. Their results are shown in Table 5 and Table 6.

	Zero-mean-unit-variance	Range matching
BG	0.02	0.002
CSF	0.31	0.03
GM	0.16	0.01
WM	0.06	0.009

Table 5: Comparison of feature normalization methods. Standard deviation of the normalized feature values for different methods.

	Zero-mean-unit-variance	Range matching
CSF	90.9	92.2
GM	91.8	92.9
WM	93.4	94.3

Table 6: Comparison of feature normalization methods. Similarity measures (%) for the different tissue types, using different normalization methods.

5.1.2 Automated training

Cocosco et al. (2003) made an improvement with the use of selection of the training samples. They used a MST to optimize the learning set. In this way samples which are calculated to not belong in a cluster are pruned from the training set.

The training set then contains clusters of data points, which are homogeneous in classification. For the use of BTS they showed this to be a valid assumption. This method was used in Vrooman et al. (2007) in combination with non-rigid registration to further improve classification. In Vrooman et al. (2007) the automated training method is compared to a manual version. It is concluded that there is no significant difference between the two.

5.1.3 Optimizing K

The only true parameter of k-NN is k. Choosing a to low or to high k can cause inaccuracy (Duda et al., 2001). Anbeek (2005) showed tests with different k's, but were all chosen by hand. To train different training sets this can be a cumbersome work. It is possible to search for k by iteratively increasing it and testing a set. But due to the computational complexity of k-NN search, this is not desired. Ahn and Kim (2008) used a GA to search for a heuristically optimal k. They compared their method to other methods that try to optimize k. Their results showed that the method is as good as or sometimes better than the others. This method is still very computationally complex, because for each k it has to calculate the performance of a training set. Additionally, as with most GA's, the algorithm does not assures to find an optimum. However, for applications in which an optimal k is required, for high accuracy, this method can be useful.

5.2 Searching

5.2.1 Distance function

Dudani (1976) and Zuo et al. (2008) aimed at improving the accuracy of the distance function. Dudani (1976) assignes a weight, based on the distance between points. Zuo et al. (2008) includes the correlation of points in the distance function (KDF-KNN). They showed this can improve classification accuracy for some datasets. They experimented on 30 data sets of different types of data and compared the classification rates of different classifiers, among others: k-NN , DS-KNN and KDF-WKNN. The average classification rates are 85.19%,85.34% and 87.67% for k-NN , DS-KNN and KDF-WKNN respectively. The KDF-WKNN shows an improvement in accuracy, but the computational complexity is also higher. If m denotes the amount of training samples, and d the amount of dimensions, the computational complexity of:

- k-NN is $O(md)$

- KDF-WKNN is $O(md + k^2d + k^3)$

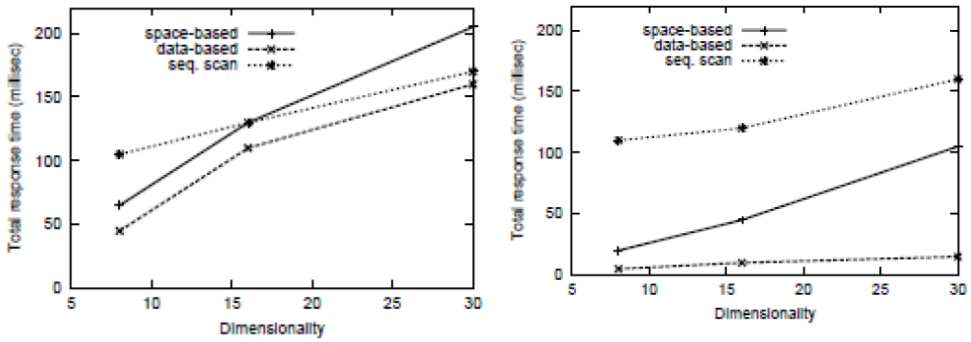
In terms of multiplications in the distance function.

5.3 Computational workarounds

Thomasian and Zhang (2007, 2005), Zhuang et al. (2007), Yu et al. (2007), He et al. (2005) did some partitioning in feature space. The two variants are:

- partitioning based on the data and
- partitioning based on the feature space.

Jagadish et al. (2005) compared the two variants, which are shown in figure 15. Because of the data dependency they tested on two types of data: uniform data and



(a) (b)

Figure 15: A comparison of partitioning methods: 15a on uniform data 15b on clustered data.

clustered data. They compared these types with the default k-NN algorithm (sequential scan), the iDistance (data-based clustering), and space based clustering. It is shown, for uniform data, with a dimensionality of 8 that the space-based partitioning uses 60% of the time sequential scan uses and data-based partitioning uses 45%. for clustered data, it is shown that space-based partitioning uses 20% the time of sequential scan and data-based partitioning only 10%.

6 Discussion

k-NN method can be easy to implement and can give good results when used with BTS, as shown in Vrooman et al. (2007). To increase the quality of the classification and improve performance of the method we sought adaptations to the k-NN method .

In this section we will discuss methods usable in BTS. We divided the advances in optimizations of the learning, searching and computational workarounds.

The building of the feature space allows us to do a first optimization. The weighting of the features has an effect on the result of the distance function. In BTS we have a priori information that different classes will exist in different clusters (Cocosco et al., 2003). Feature weighting can be implemented to optimize the discriminability of the classes.

Feature weighting and selection To find heuristically optimal feature weights, the use of a Genetic Algorithm (Section 3.2) has shown good results (Lee and Nelson, 2008). Lee and Nelson (2008) applied this method in BTS, but compared to Cocosco et al. (2003), Anbeek et al. (2004), Vrooman et al. (2007), used a lot more information: By including MR spectroscopy, diffusion and perfusion images they acquired 38 features. The GA finds heuristically optimal feature selection and weighting, but restricted to at most 4-bit accuracy. They reported overfitting with higher bit accuracy. To be able to apply the GA method on a data set with lower dimension ($d < 10$, like in Cocosco et al. (2003), Anbeek et al. (2004), Vrooman et al. (2007)), the problem of overfitting has to be overcome. This can be done by increasing the amount of training samples. To further make use of GA the bit accuracy can be increased to find more accurate feature weights.

Clustering Because the high amount of test samples in brain tissue segmentation studies it is useful to reduce the computational complexity of the search fase. As said before, the data in BTS is highly clustered itself, the use of a clustering method, like Thomasian and Zhang (2007), can decrease the search time to about $\frac{1}{5}$ of default k-NN , as shown in figure 14. This method clustering method also includes PCA and a persistent OP-Tree. The combination of these techniques is promissing to reduce search time while preserving classification accuracy.

As a last note we'd like to say that one of the main reasons of research groups for choosing the k-NN method is the easyness to implement and understand it. Many advances discussed in this thesis extend k-NN with methods which are more

difficult to understand. We think k-NN should not be stained with too much of those methods. In this way the method can also be applied in the wide variety of fields as it is now.

References

*Hyunchul Ahn and Kyoung-jae Kim. Using genetic algorithms to optimize nearest neighbors. *Ann oper Res*, 163:5--18, 2008.

*P. Anbeek, K. Vincken, M. v Osch, R. Bisschops, and J. vd Grond. Probabilistic segmentation of white matter lesions in mr imageing. *Neuroimage*, 21:1034--1044, 2004.

Petronella Anbeek. *Probabilistic Segmentation of MRI Brain Images*. PhD thesis, Image Sciences Institute, University Medical Center, Utrecht, 2005.

Barry Joseph Bedell. Automatic quantitation of multiple sclerosis lesions on mr images. *ETD collection for Houston Academy of Medicine-Texas Medical Center*, 1996. URL http://digitalcommons.library.tmc.edu/dissertations/AAI9700321.

C.L. Blake and C.J. Merz. Uci repository of machine learning databases. 1998. URL http://archive.ics.uci.edu/ml/.

Vittorio Castelli and Alexander Thomasian. Csvd, clustering and singular value decomposition for approximate similarity search in high-dimensional spaces. *IEEE Trans Knowl Data Eng*, 15:671--685, 2003.

*Hong Chang and Dit-Yan Yeung. Kernel-based distance metric learning for content-based image retrieval. *Image and Vision Computng*, 25:695--703, 2006.

*C. Cocosco, A. Zijdenbos, and A. Evans. A fully automatic and robust brain mri tissue classification method. *Medical Image Analysis*, 7:513--527, 2003.

O. Cuisenaire and B. Macq. Fast euclidean distance transformation by propagation using multiple neighborhoods. *Computer vision and Image Understanding*, 76: 163--172, 1999.

Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley & Sons, Inc, 2001.

*S.A. Dudani. The distance-weighted k-nearest-neighbor metric. *IEEE Trans Syst Man Cybern*, 6:325--327, 1976.

*Renjie He, Balasrinivasa Rao Sajja, and Ponnada A. Narayana. Implementation of high-dimensional feature map for segmentation of mr images. *Annals of Biomedical Engineering*, 33(10):1439--1448, 2005.

*Qinghua Hu, Daren Yu, and Zongxia Xie. Neighborhood classifiers. *Expert Systems with Applications*, 34:866--876, 2008.

*H. V. Jagadish, Beng Chin Ooi, Klan-Lee Tan, Cui Yu, and Rui Zhang. idistancce: an adaptive B^+-tree based indexing method fornearest neighbor search. *ACM Transactions on Database Systems*, 30(2):364--397, 2005.

Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society.*, 7(1):48--50, 1956.

*Micheal C. Lee and Sarah J. Nelson. Supervised pattern recognition for the prediction of contrast-enhancement appearance in brain tumors from multivariate magnetic resonance imaging and spectroscopy. *Artificial Intelligence in Medicine.*, 43:61--74, 2008.

Klaus-Robert Müller, Sebastian Mika, Gunnar Rätsch, Koji Tsuda, and Berhard Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transaction on neural networks*, 12(2):181--202, 2001.

Karl Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 6(2):559--572, 1901.

H. Steinhaus. Sur la division des corp materiels en parties. *Bull. Acad. Polon. Sci.*, 4(3):801--804, 1956.

*Alexander Thomasian and Lijuan Zhang. Exact k-nn queries on clustered svd datasets. *Information Processing Letters*, 94:247--252, 2005.

*Alexander Thomasian and Lijuan Zhang. Persistent clustered main memory index for accelerating k-nn queries on high dimensional datasets. *Multimed Tools Appl*, 38:253--270, 2007.

Alexander Thomasian, Vittorio Castelli, and Lijuan Zhang. Clustering and singularvalue decomposition for approximate indexing in high-dimensional spaces. *Proc. conf. on information and knowledge management*, pages 267--272, 1998.

*Henri A. Vrooman, Chris A. Cocosco, Felde van der Lijn, Rik Stokking, . Arfan Ikram, Meike W. Vernooij, Monique M.B. Breteler, and Wiro J. Niessen. Multi-spectral brain tissue segmentation using automatically trained k-nearest-neighbor classification. *NeuroImage*, 37:71--81, 2007.

Simon Warfield. Fast k-nn classification for multichannel image data. *Pattern Recognition Letters*, 17:713--721, 1996.

*Huilin Xiong and Xue wen Chen. Kernel-based distance metric learning for microarray data classification. *BMC Bioinformatics*, 7:299, 2006.

*C. Yu, B. Cui, S. Wang, and J. Su. Efficient index-based knn join processing for high-dimensional data. *Information and Software Technology*, 49:332--344, 2007.

Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40:2038--2048, 2007.

*Yi Zhuang, Yue-Ting Zhuang, and Fei Wu. Composite distance transformation for indexing and k-nearest-neighbor searching in high-dimensional spaces. *Journal of Computer Science and Technology*, 22(2):208--217, 2007.

*W. Zuo, D. Zhang, and K. Wang. On kernel difference-weighted k-nearest neighbor classification. *Pattern Anal Applic*, 11:247--257, 2008.