Multiple Classifier Fusion Incorporating Certainty Factors

Diplomarbeit an der Universität Ulm
Fakultät für Informatik
Abteilung: Neuroinformatik

vorgelegt von
Christian Thiel

1. Gutachter: *Prof. Dr. Günther Palm*
2. Gutachter: *Dr. Friedhelm Schwenker*

September 2004

www.ChristianThiel.com

Abstract

Multiple classifier systems do fuse the answers of several classifiers to boost the combined accuracy. The new approach presented does only look at the answers of the single classifiers and derives from them a certainty measure indicating how likely the output is correct. Two such measures, based on the Gini and entropy function, are utilised and compared. The fusion of the classifiers is achieved using the Dempster-Shafer theory, a mathematical framework for representing and combining measures of evidence. Here, the certainty factors are being used as basis to model the doubt in the final decision by a technique called discounting, which in fact determines the influence of each classifier for every individual sample. The experiments on two data sets show that the technique can not beat the accuracy of the single best classifier when rejecting no samples due to high doubt. This can only be achieved by including knowledge from the training phase, adjusting the certainty factors globally for all samples according to the optimal weight of the classifiers in the fusion process. Some guidelines for this adjustment could be found. The system is not able to outperform the decision templates fusion technique on one of the data sets (cricket songs).

Zusammenfassung

Die Antworten mehrerer Klassifikatoren können vereinigt werden, um eine höhere Gesamtgenauigkeit zu erreichen. Der dafür hier vorgestellte neue Ansatz betrachtet nur die Ausgaben der einzelnen Klassifikatoren und leitet von ihnen jeweils einen Sicherheitsfaktor ab, der angibt, wie wahrscheinlich es ist, dass die Antwort korrekt ist. Zwei solcher Maße, aufbauend auf der Gini-Funktion und der Entropie, werden vorgestellt und verglichen. Die Fusion der Klassifikatorausgaben geschieht dann mit Hilfe der Dempster-Shafer Theorie, einem mathematischen Ansatz zur Repräsentation und Kombination von Vertrauen in Ereignisse. Die Sicherheitsfaktoren werden dabei im Rahmen der so genannten Abschlag-Technik als Basis genutzt, um den Zweifel an der kombinierten Antwort und das Gewicht der einzelnen Klassifikatoren für die Kombination zu ermitteln. Die Experimente zeigen dass, solange keine Datenpunkte wegen zu hohen Zweifels zurück gewiesen werden, die Grundform des neuen Ansatzes die Genauigkeit des besten einzelnen Klassifikators auf dem jeweiligen Datensatz nicht überbieten kann. Dies wird erst erreicht durch die Verwendung von Wissen aus der Lernphase, indem die Sicherheitsfaktoren global so angepasst werden, dass die Klassifikatoren das optimale Gewicht im Fusionsprozess haben. Für diese Anpassung konnten einige Richtlinien gefunden werden. Auf einem der beiden Datensätze (Grillengesang) ist der Ansatz nicht in der Lage, die Klassifikationsleistung der Decision-Templates Fusionsmethode zu schlagen.

Contents

Chapter 1

Introduction

A classifier is a system that takes the data representation of an object and answers with a label telling to which class it thinks the sample belongs. Applications are numerous: the classification of handwritten numerals, images of human tissue, or sensor signals, to name a few. In fact, classifiers are of use in almost every discipline of science, but also in many applications. Because these systems do make mistakes, there is an interest to combine the answers of a number of those experts to get a more accurate overall answer.

This work presents an approach for classifier fusion. The new idea is to look at the answer of each classifier and algorithmically estimate how likely it is to be correct, resulting in a certainty factor. This factor is then taken into account by the fusion process, which operates on the basis of Dempster-Shafer theory. That mathematical framework is a generalization of Bayesian reasoning and a tool for representing and combining measures of evidence, able to express the notions of conflict and doubt, a feature which will be explored in this work. The new classifier fusion approach will be evaluated experimentally on two real-world data sets.

In chapter 2 the concepts of some important classifiers are presented, in more detail those that will be used in the experiments. Chapter 3 gives an overview over the idea of classifier fusion and describes some existing methods. Desired properties of cer-

tainty factors and two that possess those, based on the Gini and entropy functions, are presented in chapter 4. They will be employed in chapter 5 where the new Dempster-Shafer fusion approaches and their theoretical background are introduced. Details on the data and the applications used are given in chapter 6. The results of the experimental evaluations are reported in chapter 7. A summary of the work, together with a general classification of the approach and possible further developments, finalises the work in chapter 8.

Chapter 2

Basic Classifiers

2.1 On Classifiers

A classifier C takes a feature representation $x \in \mathbb{R}^d$ of an object or concept, and maps it to a classification label y. For example, it takes a picture of a handwritten numeral and tells which number 0-9 it represents. The point x from the feature space $\mathbb{R}^d$ is often in the form of a vector $\in \mathbb{R}^d$. Before a classifier can be used, it has to be trained in a learning phase, employing training data points that are each labeled with their correct class $\omega \in \Omega$.

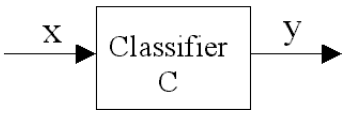

Figure 2.1: Basic classifier concept.

The output classification label y expresses to which of the l possible classes the classifier thinks the input data point x belongs. This answer can take the following different forms [64, 30]:

- At the *abstract* or *hard* or *crisp* level, the classifier outputs a unique label. Example: $y = \{\mathrm{p}(\omega_1), \ldots, \mathrm{p}(\omega_l)\} = \{0, 1, 0, \ldots, 0\}$

- In the special case of the *rank* level, the answer is a queue with $\omega_i \in \Omega$, with the topmost element of this queue being considered as the most likely by the algorithm. This level is seldom used today. Example: $y = \{\omega_3, \omega_1, \omega_8, \ldots, \omega_4\}$

- At the *measurement* or *soft* level, the classifier assigns each ω a value, representing its belief that the input x belongs to this class[1]. Example: $y = \{p(\omega_1), \ldots, p(\omega_l)\} = \{0.15,\ 0,\ 0.4, \ldots,\ 0.2\}$

In the new methods that will be proposed and in the experiments, always classifiers with an output on the measurement level will be used. If a crisp decision is required, the soft answer can be hardened using the *maximum membership rule* [31], that is selecting the class with the highest assigned value.

Today, many different approaches to the problem of building classifiers exist. Some typical ones will be briefly explained, before going into more detail about those actually used in the experiments.

2.1.1 Decision Trees

This natural method is based on a divide-and-conquer approach. The knowledge about the data is represented in a *decision tree*. In each of its nodes, a test is performed on a variable/dimension, usually the comparison with a constant. Each node has children nodes that are reached depending on the outcome of the test. An unknown sample x begins at the root and is routed through the tree, according to the outcome of the tests in the nodes. Once a leaf is reached, the sample can be classified according to the class label assigned to the leaf.

One method to construct the decision trees from the training data is the so-called C4.5 by Ross Quinlan [45].

[1] Answering with a whole subset of Ω can be modelled on the measurement level. In [64], it is an extension to the abstract level.

2.1.2 Association Rules

Unlike the trees, the association rules look at more than one variable/dimension at once. A typical rule would be:
IF $x_1 > a$ AND $x_2 > b$ AND $x_5 \leq c$ THEN $y_1 = u$ AND $y_3 = v$
Restraining the right side of the equation to one result yields the classification rules, giving a class label to the input. Unfortunately, there may be multiple rules that match a given input, leading to a conflict that has to be solved.

One method to construct association rules is the Apriori algorithm introduced by Agrawal et al. [1]. Instrumental in it are the notions of $support/coverage$, that is the relative frequency with which the left side of the rule is true in the training data, and of $confidence/accuracy$, the fraction of samples classified correctly by the rule on the training data.

2.1.3 Multilayer Perceptrons MLP

Multilayer perceptron networks belong to the class of supervised neural classifiers. They consist of perceptrons that are organised in layers: an input layer, one or more hidden layers, and the output layer. Every perceptron in one particular layer is usually connected to every perceptron in the layer above and below. These connections carry weights w_i. Each perceptron calculates the sum of the weighted inputs, and feeds it into its activation function, regularly a sigmoid one. The result is then passed on to the next layer. The output layer has for example the same number of perceptrons as there are classes, and the perceptron with the highest activation will be considered the classification of the input sample.

Training is achieved by successively feeding all training samples into the network, and comparing the output with the true class label. This information is then used to adjust the weights in the network in the so-called *backpropagation* phase, employing the *generalised delta rule* [49].

2.1.4 Support Vector Machines

The original Support Vector Machine suggested by Vapnik [59] was a linear classifier for two-class problems. By solving a quadratic programming optimisation problem, it produces a maximum-margin hyperplane that separates the classes and also has the maximum distance (margin) to the closest training samples. The samples closest to the hyperplane are called the *support vectors*. To address the problem of not linearly separable classes, slack variables ξ_i can be introduced that include the distance of falsely classified samples to the nearest place of correct classification into the optimisation problem.

2.2 Classifiers Used in the Experiments

2.2.1 K-Means

The K-Means algorithm comes from the field of cluster analysis and tries to assign new data points to clusters according to their distance to prototypes that have been found by the algorithm. The 'K' in the name refers to the number of prototypes used. The concept was most likely proposed by MacQueen in 1967 in [34], and has been rediscovered many times since. It minimises the total within cluster variance. In our context, the prototypes are of interest as they will be employed as starting points for the K-NN algorithm. For that end, the algorithm will work on training data that contains only examples from one class, then proceed to the next training set to find prototypes for that class.

K-Means works by executing the two steps described below, until convergence occurs[2] or a pre-set maximum iteration time t_{max} is reached. To initialise, K prototypes (or

[2]Let *C(t), t = 0,1,...* be a series of partitions produced by the K-means algorithm. Then the within cluster Variance $D_{var}(C(t))$ is monotonically decreasing:

$$D_{var}(\,C(t)\,) = \sum_{j=1}^{k} \sum_{x_\mu \in C_j(t)} \|x_\mu - c_j(t)\|^2$$

Figure 2.2: Centers calculated by K-Means for handwritten numerals (see chapter 6.1).

centers) c_j must be chosen, for example randomly from the training data $x_\mu, \mu = 1,...,n$.

1. For each center we identify the subset of training points (its cluster C_j) that are closer to it than any other center, yielding the so-called minimum distance partition.

$$C_j(t+1) = \{x_\mu : \| x_\mu - c_j(t) \| = \min_{i=1,..,K} \| x_\mu - c_i(t) \| \} \quad j = 1,...,K;$$

2. The means for the data points in each cluster C_j are computed, and this mean vector becomes the new center $c_j(t+1)$ for that cluster. Then increase t by 1. If $t > t_{max}$ end else go to step 1.

$$c_j(t+1) := \frac{1}{|C_j(t+1)|} \sum_{x_\mu \in C_j(t+1)} x_\mu \quad j = 1,...,K$$

(Distance to nearest cluster center is not longer.)

$$\geq \sum_{j=1}^{k} \sum_{x_\mu \in C_j(t)} \min_i \|x_\mu - c_i(t)\|^2$$

(Regroup inner sum, sample x_μ now put into the cluster of the nearest center in pass *(t+1)*.)

$$= \sum_{j=1}^{k} \sum_{x_\mu \in C_j(t+1)} \|x_\mu - c_j(t)\|^2$$

(New center $c_j(t+1)$ minimises the variance within the cluster.)

$$\geq \sum_{j=1}^{k} \sum_{x_\mu \in C_j(t+1)} \|x_\mu - c_j(t+1)\|^2 = D_{var}(\, C(t+1)\,)$$

2.2.2 Radial Basis Function Networks (Gaussian)

The theoretical basis of radial basis function networks (RBF) lies in the field of the interpolation of multivariate functions [42]. It is often used for function approximation, performing nonparametric regression, but can also be employed for the task of classification [33, 40].

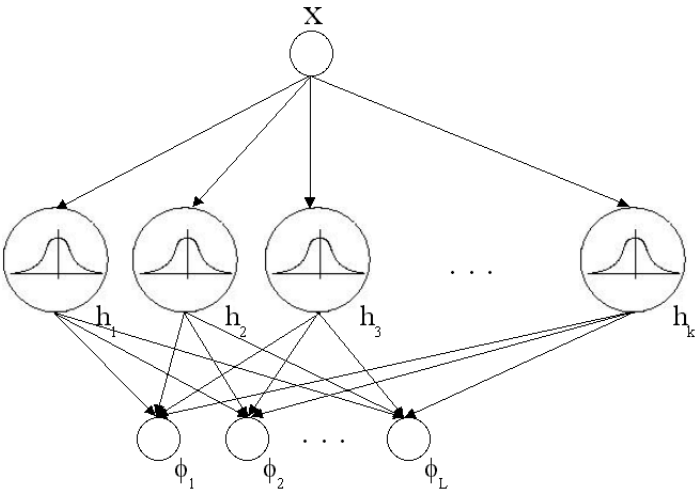

Figure 2.3: Concept of a RBF network.

Reducing the problem of interpolation to that of approximation lowers the number of required basis functions and allows the network to be modelled as a neural network [8]. This consists of one hidden layer with k basis functions, or neurons. At the input of each neuron, the distance (euclidian in most applications, also here) between the neuron center c and the input vector x is calculated. The basis function h is then applied to this value to produce the output of the neuron. An output unit ϕ of the RBF network is formed by a weighted (w) sum of the neuron answers.

$$\phi(x) = \sum_{j=1}^{k} w_j \, h_j(x) \tag{2.1}$$

For the task of classification, the network outputs can be interpreted as being proportional to the a posteriori probability that the input belongs to the corresponding class [4]. The function to be approximated is simply the one that has a value of 1 for each sample that is in the specific class for which the network output unit is trained, and 0

for all other samples. Hence, we have one network output unit $\phi(x)$ for each of the l classes to be learned. The outputs of all networks output units are normalised to sum up to 1.

The radial basis function used in this implementation is the commonly employed standard Gaussian[3].

$$h_j(x) = \exp(-\frac{\| x - c_j \|^2}{2\,\sigma_j^2}) \frac{1}{\sqrt{2\pi\sigma_j^2}}$$

A training of the network is accomplished with so-called two-phase learning [51]. In the first phase, the centers c_j and the *scaling* or *width* parameters σ_j are learned and fixed. The second phase is to find the optimal combination weights w_j.

To obtain centers for the gaussian function, often unsupervised clustering methods are used [38]. We employ the K-Means algorithm to produce $\frac{k}{l}$ prototypes per class by applying it consecutively to training data with samples only from that class, totalling in the aforementioned k prototypes (=centers) for the network. As an estimate for the σ_j^2 of each center, the mean euclidian distance of the center to each sample in the cluster is used[4].

In the second phase, the combination weights of each network output unit are determined [21] in one pass using the pseudoinverse solution [8], which is based on the overdetermined least-square criterion: Let (x^μ, y^μ), $\mu = 1, \ldots, m$, be the set of m training samples with feature vector x^μ, and the class labels $y^\mu \in \{0,1\}^l$ encoded through the relation $y_i^\mu = 1$ iff $\text{class}(x^\mu) = i$. And let $H_{\mu j}$ be the outcome of the j-th basis function with the μ-th feature vector x^μ as input, and $Y_{\mu j}$ the j-th component of the μ-th label vector y^μ. Then the desired approximation can be formulated as follows (effectively equaling l functions $\phi(x)$ as defined in equation 2.1):

$$Y = HW$$

The matrix W consists of the output weights w, one column with k weights for each

[3] Another choice of basis functions suggested by Bishop in [4] is the thin-plate spline function $h_j(x) = x^2 \ln x$ and the following: $h_j(x) = (x^2 + \sigma_j^2)^{-\alpha}, \quad \alpha > 0$

[4] That is, those samples nearer to this prototype than any other.

output unit[5]. The optimal W is now the minimum of the error function

$$E(W) = \|HW - Y\|^2$$

to wich the solution is explicitly given [51] in the form $W = H^+Y$, where H^+ denotes the pseudo inverse matrix of H which is defined as

$$H^+ = \lim_{\alpha \to 0+} (H^T H + \alpha \mathrm{Id})^{-1} H^T.$$

Given that $(H^T H)^{-1}$ is defined, the pseudo inverse matrix simplifies to

$$H^+ = (H^T H)^{-1} H^T.$$

To now classify a new sample z, it has to be fed to the k basis functions, assembling the answers into a vector $A = (h_1(z),\ h_2(z),\ \ldots,\ h_k(z))$. The final (soft) classification F of the sample is calculated as simple as $F = AW$, the l values in F representing the estimated a posteriori probabilities that z is in the corresponding class.

2.2.3 Fuzzy K-Nearest-Neighbour

The K-Nearest-Neighbour algorithm (KNN) [17] is used to classify feature vectors. Being simple, elegant and straightforward, it is often used today, yielding good classification results. The instance based classifier works by searching for the k nearest neighbours of the input vector $x \in \mathbb{R}^d$ among a set of prototypes or training samples. These neighbours are found using the L_p-norm ($p \in [1, \infty)$):

$$d_j^p(x, x^j) = \|x - x^j\|_p = (\sum_{n=1}^{d} |x_n - x_n^j|^p)^{\frac{1}{p}}$$

In our case the Euclidean distance $(p = 2)$ is used. Now let the k prototypes nearest to x be $N_k(x)$ and $c(z)$ be the class label of z. Then the subset of nearest neighbours within class $j \in \{1, \ldots, \text{number of classes } l\}$ is

$$N_k^j(x) = \{y \in N_k(x) : c(y) = j\}.$$

[5]The dimensions of the matrices hence are: H: $m \times k$, W: $k \times l$, Y: $m \times l$

For the K-Nearest-Neighbour algorithm, the classification result $j^* \in \{1,...,l\}$ is defined as a majority vote:

$$j^* = \text{argmax}_{j=1,...,l} |N_k^j(x)|$$

Only we do not need such a crisp answer, but the class membership of the input vector x for all l classes. This is achieved by using a Fuzzy K-Nearest-Neighbour classifier [56] who performs a mapping $\mathbb{R}^d \rightarrow [0,1]^l$. Let

$$\delta_j(x) = \frac{1}{\sum_{x^i \in N_k^j(x)} \|x - x^i\|_p}$$

be the support for the hypothesis that j is the true class label of x. We set $\delta_j(x) = 0$ if $N_k^j(x) = \emptyset$. After normalisation by

$$\Delta_j(x) := \frac{\delta_j(x)}{\sum_{i=1}^l \delta_i(x)}$$

we have soft outputs Δ_j with $\Delta_j(x) \in [0,1]$ and $\sum_{j=1}^l \Delta_j(x) = 1$.

Chapter 3

Multiple Classifier Fusion (MCF)

Multiple classifier fusion is the combination of the answers y_i of different basic classifiers pertaining to a sample x to an overall answer $y_{combined}$, which is hopefully more accurate than the answers of the single classifiers. Many fusion methods exist that aim to achieve this.

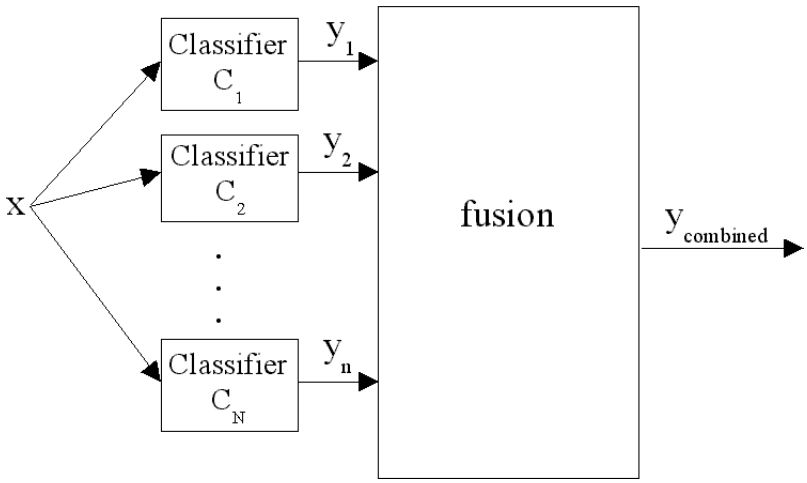

Figure 3.1: Basic classifier fusion concept.

Opposed to the idea of fusion is that of classifier selection [9]. It attempts to predict which of the single classifiers is most likely to produce the correct answer for a given sample [63]. Hybrid approaches that switch between the two paradigms are also available [28].

In this chapter, reasons for multiple classifier fusion are given, then approaches to build diverse classifiers are presented. Finally some common methods of classifier fusion are highlighted.

3.1 Motivations for Classifier Fusion

One reason for the combination of classifiers is the need for integration. The features of a sample may be presented in very diverse forms, making it impossible to use them as input for one single classifier. This point is much emphasised in [64].

Another rationale is the desire to boost efficiency by using simple and cheap classifiers that operate only on a small set of features. This approach will often be combined with the option to reject samples [43].

Perhaps the most obvious motivation for the combination is the possibility to boost the classification accuracy: combining classifiers with different errors [2], or combining local experts, will have that effect.

The fact that ”the best individual classifier for the classification task at hand is very difficult to identify, unless deep prior knowledge is available”[48] is not often mentioned in the literature, but also a motivation for multiple classifier fusion.

3.2 Building Diverse Classifiers

Concerning the construction of multiple classifier systems, two strategies can be distinguished [22]: *decision optimisation,* which strives to find the optimal decision function, and *coverage optimisation* aiming to construct mutually complementary classifiers whose combination ”covers” the data well. An integrated approach can be found in [48]. As only decision optimisation is addressed in this work, some remarks concerning coverage optimisation follow.

The classifiers constructed should have a high degree of error diversity [55]. An overview of methods how to arrive at diverse and complementary classifier ensembles is presented in [12] and [48]. The later also looks at various diversity measures.

One approach is to base the classifiers on the data from different sensors [24]. Another, employed in this paper, is to extract different features from the initial data. *Bagging* [7] and *Boosting* [16] are popular methods that re-sample the training data respectively assign different weights to the bootstrap data points.

3.3 Methods for MCF

Methods for the fusion of multiple classifiers can be categorised into two classes: *static* ones only look at the output of the classifiers, while *dynamic* methods take into account information from the training phase about the behaviour of the classifiers. An experimental comparison of the performance of many fusion methods mentioned here has been published by Kuncheva [27].

Static combination functions are quite simple. The *majority vote* takes the hard answers of classifiers and decides for the class the majority voted for [29]. With soft answers, one can take the *minimum*, *maximum*, *median*, *average* (see figure 3.2) or *probabilistic product* per class and compare them over all classes. The framework of *ordered weighted averaging operators* [65] can express most of the last mentioned techniques.

Dynamic fusion approaches are more elaborate. The *naive Bayes classifier*[1] works on hard classifiers, for l classes constructing a $l \times l$ confusion matrix M^j for each of the classifiers C_j out of the answers to the training samples $x_\mu \in S$:

$$M^j_{k,s} = \frac{\mid \{x_\mu \in S : C_j(x) = s\} \mid}{\mid \{x_\mu \in S : \text{label}(x) = k\} \mid}$$

Given the classifier outputs for a test sample, this confusion matrix is then used to construct conditional probabilities indicating to which of the classes the sample is likely

[1] The name *naive* stems only from the underlying assumption of independent classifiers.

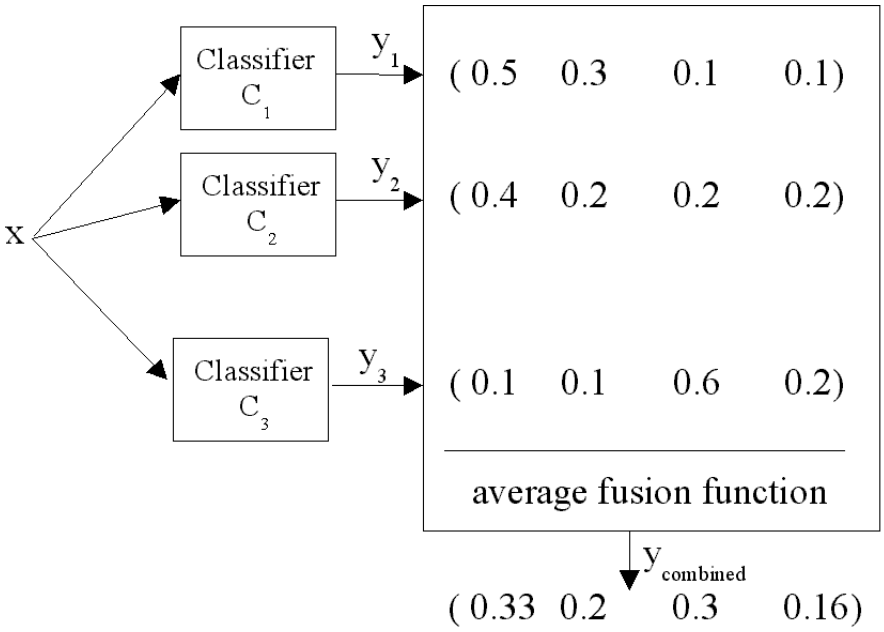

Figure 3.2: *Average* fusion.

to belong.

The *brute force* approach (also called *stacked generalization* in [62]) is a meta-method: It takes the classifier outputs as samples in a new intermediate feature space, that now have to be classified by a basic classifier, often a linear one or a neural network like the MLP (see chapter 2.1.3).

Another method for combining classifiers with crisp (or hardened soft) outputs is the *Behavior-Knowledge Space (BKS)* [23]. An answer from the k classifiers is considered as an index into a cell of the k-dimensional lookup table (BKS table). In the training phase, the true label of each sample is added to the cell which the classifier outputs for this sample point to. In the classification phase, one simply looks at the labels present in the cell indexed by a new sample. The drawback is that the lookup table will only be scarcely filled if there are not very many training samples. That means that there will often arise situations when there is no label in an indexed cell or a tie between labels (e.g. two labels of class one, but also two labels of class three). *Werneckes method* [60], actually proposed before BKS, addresses the last issue.

Decision Templates (DT) [27] are a relatively new approach. The basis of the technique

is a so called *decision profile* $DP(x)$ for each sample x, in form of a matrix consisting of one row for each classifier, with the classifier output in the rows (hence, with k classifiers and c classes, a $k \times c$ matrix). Now, for each class i a so called *Decision Template* DT_i is calculated as the average of $\{DP(x) : class(x) = i\}$. For the classification of a new data point z, the induced decision profile $DP(z)$ is compared to the DT_i of each class, the results of these comparisons then form the soft answer (see figure 3.3). As measures of similarity [13] between the fuzzy sets, quite some alternatives exists. However, in their experiments [27] Kuncheva et al. found that four of them were performing particularly well. In a later work [31] it was proven that each of them will lead to the same final classification, provided that the classifier outputs sum up to the same value. For the purpose of this paper, namely that of performance comparison between the newly proposed method and DTs, therefore one of these four measures, the similarity measure S_1, will be used:

$$S_1(A,B) = \frac{\| A \cap B \|}{\| A \cup B \|} = \frac{\sum_u \min\{\mu_A(u), \mu_B(u)\}}{\sum_u \max\{\mu_A(u), \mu_B(u)\}}$$

Here $\mu_A(u)$ and $\mu_B(u)$ are fuzzy membership functions on the fuzzy set U, $u \in U$, that is corresponding entries in the decision templates. So the measure S_1 is the sum of the pairwise minimums of the corresponding elements in the DTs divided by the sum of the pairwise maximums.

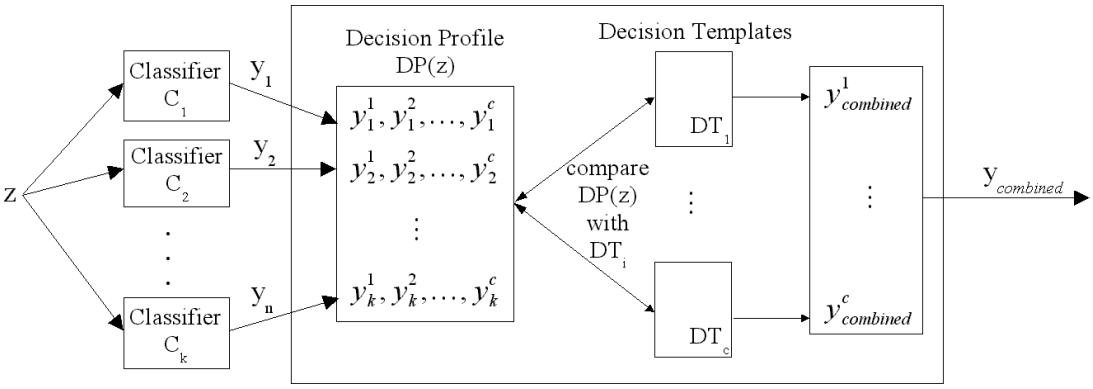

Figure 3.3: *Decision Template* fusion.

Techniques that use mathematical frameworks for the combination of fuzzy sets, like fuzzy theory or Dempster-Shafer, also exist, see chapter 5.

Chapter 4

Certainty Measurements for Classifier Outputs

The accuracy of classifiers varies depending on many different factors. One classifier may be an expert for a specific class, whereas others might be excellent for inputs from a certain part of the input space. Up to now, to estimate the accuracy of a classifier for the actual input, one has to rely on expert knowledge or a large sample database. The goal was to find functions that only look at the output of a classifier and estimate the accuracy of the output for the current input. These estimates, termed certainty factors here, will then be employed in the combination of multiple classifiers, and hopefully boost the accuracy of the combined answer.

In the following, the properties that are desired from a certainty factor are presented, then two measures that possess those.

4.1 Desired Properties of Certainty Factors

The certainty factors do give an estimation for the accuracy of the current output of a classifier, based only on this output. They should be high, or confident, if the classifier does classify the current input correctly, or if it is a least giving the correct label as

one of the most likely alternatives. That way such a classification result will be given a high importance in the combination of the classifiers. Hence an output that is wrong should be associated with a low certainty factor, telling the combination algorithm to take it into account only to a low degree. Unfortunately, there are the cases of good classification results associated with low certainty factors, and vice versa. These will actually have an averse effect on the performance of the combined classifiers, as they mislead the combination process. See table 4.1 for an overview. The experimental results show that the certainty factors for most of the tested classifiers do not fall clearly into the desired two categories.

	output correct	**output wrong**
certainty factor high	boost	decrease
certainty factor low	decrease	boost

Table 4.1: Effect on the performance of combined classifiers.

To be useful in our context, certainty factors should have the following properties:

1. The calculation of the certainty factors based on the classifier outputs must not be computationally expensive.

2. The resulting certainty factors must be in a known, predefined finite range. A value of ∞ would require special treatment and result in the such qualified classifier being the only one taken into account, which is not desirable. Also, a finite range allows to normalise the certainty factors, making a comparison between various classifiers possible.

3. The factor shall attain its maximum when applied to non-ambiguous and therefore certain classifier outputs, for example a (0 0 1 0) vector. Accordingly, the minimum should be awarded to evenly distributed, hence absolutely uncertain, classifier outputs the likes of $(\frac{1}{4}\ \frac{1}{4}\ \frac{1}{4}\ \frac{1}{4})$. Also, the formula used shall treat all classes equally and hence be symmetrical in its arguments[1].

[1]Note that the here stated properties were also postulated by Breiman et al. in [6] for measures regarding the impurity of regions in decision trees.

4. If a small fraction of the possible classes, but more than one, are presented as very likely, the certainty factor shall still be high. Thus the classifier can still contribute to the combination process, by in fact ruling out some very unlikely classes.

Two measures that fulfil these conditions seem to be suitable here: Shannon entropy and Gini function. Both are well studied (entropy: [45], Gini: [6]) in the field of decision trees [44, 46, 61].

4.2 Shannon Entropy

The entropy comes from the field of information theory. It was introduced by Claude Shannon in [54] as a means to determine the capacity of a channel required to transmit a message as encoded binary digits over telephone lines. Quickly it became evident that the entropy measure was of importance for other fields, too, as it can be considered a measure for the amount of information contained in a message, which is needed in this case. Entropy is defined as

$$H = -\sum_{i=1}^{n} p_i \log_2 p_i$$

where the p_i are the n elements of the classifier output vector. The p_i are probability values $0 \leq p_i \leq 1$, $\sum_{i=1}^{n} p_i = 1$, and the value of H is between 0 and $\log_2 n$. Its maximum value is achieved iff the output values are all $\frac{1}{n}$. If the classifier is really sure, so one of the p_i is 1, entropy is 0 ([50]). To make the entropies of different classifiers comparable, they are each normalised to have a maximum value of 1 by dividing by $\log_2 n$. To fulfil condition 2 formulated in 4.1, the entropy values are inverted (1 - normalised value).

In fact, the entropy function was found to be ideal to measure the ambiguity in a set of Dempster-Shafer hypotheses by Hutchinson and Kak [24]. They used it to automatically decide which real-world sensing operation should be done next, by predicting the hypothesis sets that might occur if a particular sensing operation was applied, and

looking at the resulting changes of ambiguity. But the entropy has also other uses in data mining, for example with decision trees [45], especially in multi-interval discretisation [15].

4.3 Gini Function

The Gini function is a measure of inequality or impurity, commonly used in the two fields of economics and data mining. In economics, the so-called Gini index shows the inequality of the distribution of wealth or income [18]. In data mining, the Gini function is used within decision tree algorithms to assess the impurity of regions [6]. Applied to classifiers, a 'pure' answer means that the classifier casts a crisp vote. The Gini function is defined as

$$G = 1 - \sum_{i=1}^{n} p_i^2 = \sum_{\substack{i=1 \\ p_i \neq p_j}}^{n} \sum_{j=1}^{n} p_i \, p_j$$

where again the p_i are the n elements of the classifier output vector, representing probability values. And as with the entropy, the minimum of 0 is reached if the classifier is sure, the maximum value is attained with an equal distribution among the p_i. The normalisation is done by dividing the resulting Gini value by $\frac{n-1}{n}$. Again, to comply with condition 2 in section 4.1, the values are inverted (1 - normalised value).

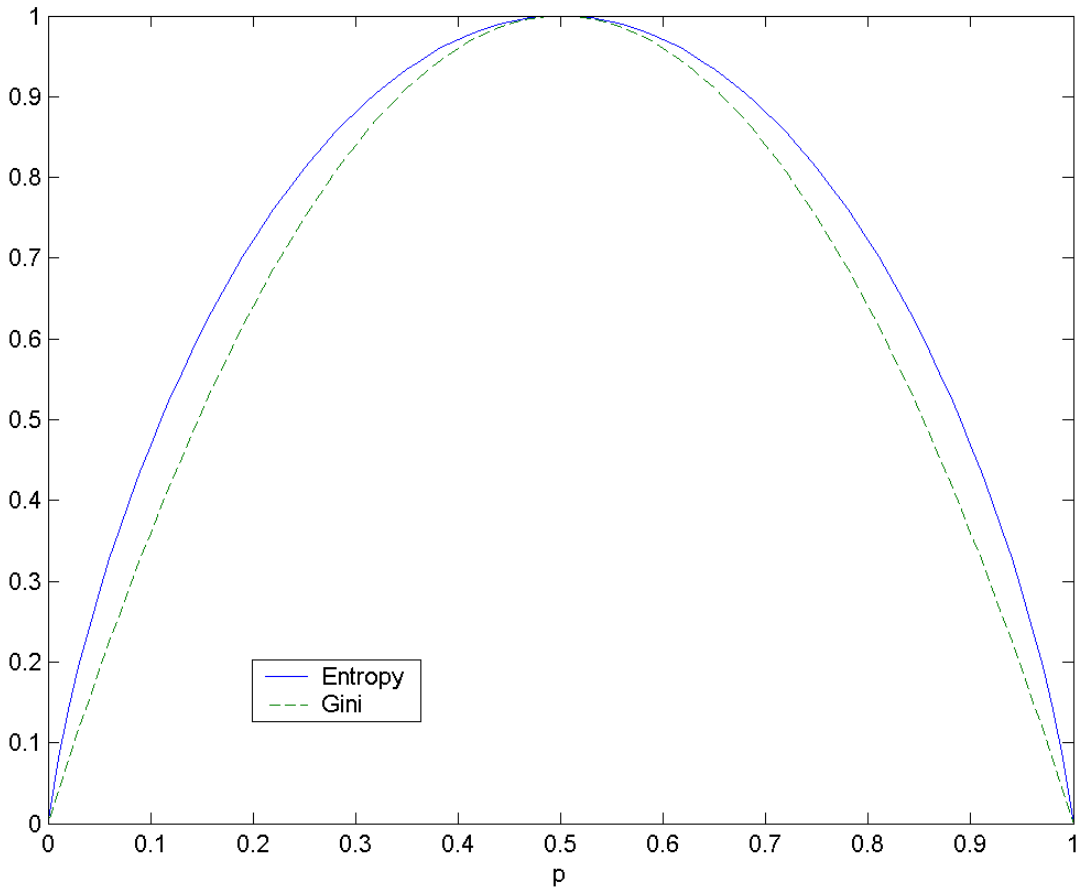

Figure 4.1: Not inverted entropy and Gini values, for a classifier output vector with two elements, p and $(1-p)$. If there are more than two elements, entropy is not generally higher!

Chapter 5

Multiple Classifier Fusion Using the Dempster-Shafer Theory of Evidence

The Dempster-Shafer theory of evidence, also known as the theory of belief functions, is a tool for representing and combining measures of evidence. Being a generalization of Bayesian reasoning, it does not require probabilities for each question of interest, but the belief in a hypothesis can be based on the probabilities of related questions. Contributing to its success is the fact that the belief and the ignorance or uncertainty concerning a question can be modelled independently.

Dempster-Shafer theory has been chosen over probability theory [39], possibility theory [14] and fuzzy theory [66, 67] because of its straightforward application to the problem, and experimental results [27] showing that it performs well in the area of classifier fusion. A classification of the mentioned theories with respect to the constance of their behaviour and the information they depend on can be found in [5].

The Dempster-Shafer theory was brought forward by Dempster [10] and Shafer [52], then came to the attention of artificial intelligence researchers in the early 1980s when they were trying to adopt probability theory to expert systems [53] and is still used in this field [19, 41]. Since then, many approaches have been taken to incorporate the Dempster-Shafer theory into methods for combining classifiers. They will be briefly

presented after an introduction into the theory, followed by the description of two new approaches to incorporate uncertainty and quality measurements into the classifier fusion process.

5.1 Introduction into the Basic Concepts of the Dempster-Shafer Theory

The Dempster-Shafer (DS) theory starts by assuming a universe of discourse, or *frame of discernment*, consisting of a finite set of mutually exclusive atomic hypotheses $\Theta = \{\theta_1, ..., \theta_q\}$. Let 2^Θ denote the set of all subsets of Θ. Then a function $m : 2^\Theta \rightarrow [0,1]$ is called a *basic probability assignment (bpa)* if it satisfies

$$m(\emptyset) = 0 \quad \text{and} \quad \sum_{A \subseteq \Theta} m(A) = 1$$

So, according to the conditions above, belief can not only assigned to an atomic hypothesis, but some set $A = \{a_1, ...a_n\}$, $a_i \in \Theta$. Hence, our belief in $m(A)$ represents our ignorance, which can not be subdivided among the subsets of A. Each element B with $m(B) \geq 0$ is called a *focal element*. Now with m being a basic probability assignment, the *belief function bel* : $2^\Theta \rightarrow [0,1]$ is defined as

$$bel(B) = \sum_{A \subseteq B} m(A)$$

It represents the minimum trust we can have in B because of the supporting *As*. Looking at the definition, it can be noticed that there is a one-to-one correspondence between the belief function and the basic probability assignments. If A is an atomic hypothesis, $bel(A) = m(A)$. Furthermore, if every focal element is an atomic hypothesis, we find ourselves in the situation of the probability theory. To get an intuitive understanding, one can consider a basic probability assignment as a generalization of a probability density function and a belief function as a generalization of a probability function [47].

The most interesting part of the theory is the possibility to combine two basic probability assignments m_1 and m_2 on Θ with the *orthogonal sum* $m_{12} = m_1 \oplus m_2$ which is

defined as

$$m_1 \oplus m_2 \;=\; m_{12}(C) = K \sum_{A,B:A\cap B=C} m_1(A)\cdot m_2(B)$$

where

$$K^{-1} = 1 \;- \sum_{A,B:A\cap B=\emptyset} m_1(A)\cdot m_2(B) = \sum_{A,B:A\cap B\neq\emptyset} m_1(A)\cdot m_2(B)$$

The factor K is an indicator how much m_1 and m_2 are contradictory, with $\log(K)$ being called the *weight of conflict*. The orthogonal sum $\oplus$ only exists if $K^{-1} \neq 0$, if $K^{-1} = 0$ the sources are said to be *total contradictory*. Combining several basic probability assignments is simple, as the orthogonal sum $\oplus$ is commutative and associative and can even be generalised for $m = m_1 \oplus \ldots \oplus m_n$, with $m(\emptyset) = 0$ as follows:

$$m(A) = K \sum_{\cap A_i=A,\ 1\leq i\leq n} \prod m_i(A_i)$$

$$K^{-1} = 1 - \sum_{\cap A_i=\emptyset,\ 1\leq i\leq n} \prod m_i(A_i) = \sum_{\cap A_i\neq\emptyset,\ 1\leq i\leq n} \prod m_i(A_i)$$

To get a more intuitive understanding of the formulas above, see [20]. The Dempster-Shafer theory does not stop here, but for example includes guidelines on how to update when new evidence is found. Alongside *bel* there exist other measures that encode the same information, but have another interpretation, for example the $plausability$ or $upper\ probability\ function$

$$plausability(A) = 1 - bel(\neg A)$$

It expresses how much we should belief in A if all currently unknown facts were to support A, thus forming a trust or knowledge interval $[bel; plausability]$.

Example

Let there be a frame of discernment with two atomic hypotheses θ_1 and θ_2, and based on it three basic probability assignments m_1, m_2 and m_3 as shown in table 5.1.

Then calculating the orthogonal sum $m_{12} = m_1 \oplus m_2$ is straightforward:

$m_{12}(\theta_1) = K \cdot m_1(\theta_1) \cdot m_2(\theta_1) = K \cdot 0.8 \cdot 0.6 = K \cdot 0.48$

$m_{12}(\theta_2) = K \cdot m_1(\theta_2) \cdot m_2(\theta_2) = K \cdot 0.2 \cdot 0.4 = K \cdot 0.08$

	m(θ_1)	**m(θ_2)**
m_1	0.8	0.2
m_2	0.6	0.4
m_3	0.25	0.75

Table 5.1: Initial situation in the example.

The normalising factor K is calculated as:

$K^{-1} = m_1(\theta_1) \cdot m_2(\theta_1) + m_1(\theta_2) \cdot m_2(\theta_2) = \sum_{\forall \text{ focal elements } f \in m_{12}} m_{12(\text{un-normalised})}(f) =$

$0.48 + 0.08 = 0.56.$

Resulting in:

$m_{12}(\theta_1) = 0.8571, \quad m_{12}(\theta_2) = 0.1429$

Fusing m_{12} and m_3 in the same way yields:

$m_{123}(\theta_1) = 0.6666, \quad m_{123}(\theta_2) = 0.3333$

Thus, after the combination process, the hypothesis θ_1 is the most likely, but θ_2 is not ruled out altogether, because there was some belief in it, especially in m_3.

5.2 Some Approaches to Include the Dempster-Shafer Theory in Multiple Classifier Fusion

As early as 1988, Mandler and Schürmann [35] developed a method to make use of the Dempster-Shafer theory in the process of multiple classifier fusion. Applied to on-line script recognition, they used a prototype based classification system. Each classifier, termed *expert*, produced a basic probability assignment that was calculated as the likelihood ratio of the intra-class-distance model to the inter-class-distance model, a measure that is also motivated by DS theory. The *bpa*s of three experts where then combined using the orthogonal sum.

In their influential 1992 paper on methods of combining multiple classifiers [64], Xu, Krzyzak and Suen also tackled the problem to include DS theory in the combination process. They used crisp classifiers, whose answer is a vote for exactly one class

or rejection of the sample. On a test set, they estimated the recognition rate ε_r^k and substitution rate ε_s^k for each of the K classifiers. Out of those, a basic probability assignment function was constructed by basically defining

$$m_k(\theta_c) = \varepsilon_r^k, \;\; m_k(\neg\theta_c) = \varepsilon_s^k, \;\; m_k(\Theta) = 1 - \varepsilon_r^k - \varepsilon_s^k$$

with $\theta_c \in frame\,of\,discernment$ referring to one of the C possible classes. The introduction of $m_k(\Theta)$ is necessary to account for the rejection rate. The K classifiers were then combined using the orthogonal sum $\oplus$. Because each of the *bpa*s has only two focal elements, the combination method can be optimised to have computation costs of only $O(M)$, with M being the number of classifiers.

Rogova [47] applied the DS concept to the combination of multiple classifiers with fuzzy, not crisp, outputs. She built a basic probability assignment for each class c for each classifier k[1], then combined the *bpa*s per class with the orthogonal sum. The final decision was made for the class c with the highest pro-c evidence. The interesting twist is that the information concerning the other $\neg c$ classes was used in the construction of the first per-class-per-classifier *bpa*s. Let d_c^k be the output of classifier k concerning class c. Then the first *bpa* calculated was

$$m_c^k(\theta_c) = d_c^k, \;\; m_c^k(\Theta) = 1 - m_c^k(\theta_c)$$

The second *bpa* took into account all votes of the classifier that did not vote pro-c

$$m_{\neg c}^k(\neg\theta_c) = 1 - \prod_{i \neq c}(1 - d_i^k), \;\; m_{\neg c}^k(\Theta) = 1 - m_{\neg c}^k(\neg\theta_c)$$

The basic probability assignments m_c^k and $m_{\neg c}^k$ were again fused using the orthogonal sum $\oplus$. Kuncheva, Bezdek and Duin included this approach in their experimental comparison [27] of the decision templates (*DT*) fusion technique with other, established methods. They came to the conclusion that the method "rated comparatively high on both data sets. It had a little lower final rank than the best DT techniques, and can be put basically in the same group."

[1] Rogova used another notation, k for the classes and n for the classifiers, this was only changed to be more consistent with the previous examples.

Al-Ani and Deriche came up with a new variant in 2002 [3]. Their idea was to improve on the way Rogova calculated the basic probability assignments for each classifier and class. In her work, it was based on the distance between the classification of an input vector and a reference vector, which was calculated as the mean classification output on training data for the class. Now, the reference vectors are adopted so that the mean square error between the final output per class and the training vectors is minimised. The adoption is achieved with an iterative gradient descent technique. Alongside the reference vectors, the so called value of ignorance g_n of each classifier is optimised, being the basis for the calculation of the DS measure $m_n(\Theta)$. And this is somewhat similar to the new technique proposed below to integrate uncertainty measures into the classifier fusion process.

5.3 Two New Approaches to Include Uncertainty

For the purpose of this paper, a straightforward method is used to construct the basic probability assignments: the output of each classifier is normalised to sum up to one, and then forms the basic *bpa* that the approaches below work with.

5.3.1 Orthogonal Sum Without Normalisation and the Transferable Belief Model (Conflict)

The combination via the orthogonal sum is not without problems. When basic probability assignments are combined, later the result does not allow for any conclusions if there had been a conflict between the sources. In fact, Zadeh [68] constructed an example where the resulting *bpa*s are the same, regardless if the sources were totally agreeing or quite contradictory. To get around this, it was suggested [20] to show a measure of conflict with each combination of two *bpa*s m_1 and m_2:

$$conflict(m_1, m_2) = -\log(1-K)$$

Here K is the same as defined in section 5.1, the log making sure that normal minor deviations are not overrated.

But there is an even better way to preserve information about the conflicts of the merged sources: Using the orthogonal sum $\oplus$ without its normalising denominator. The idea was brought up by Smets in the framework of the *transferable belief model*, which is based on the work of Dempster-Shafer. As explained in [57], the distribution of the probability mass missing to the sum of 1 among the remaining *bpa*s A_i with $m(A_i) > 0$ is not good, inter alia, because $plausability(A_i)$ is defined as the maximal degree of support that can be assigned to A_i, and must not be increased by the combination. Instead, the missing mass should be assigned to some contradictory state, which is shown to be the empty set $\emptyset$. Thus, the belief $m(\emptyset)$ is a good measure for the conflict between all the sources that have been combined using the orthogonal sum $\oplus$ without normalisation.

Example

Using the same framework as in the example in chapter 5.1, the combination of the basic probability assignments m_1, m_2 and m_3 is calculated as follows:

$m_{12}(\theta_1) = m_1(\theta_1) \cdot m_2(\theta_1) = 0.8 \cdot 0.6 = 0.48$

$m_{12}(\theta_2) = m_1(\theta_2) \cdot m_2(\theta_2) = 0.2 \cdot 0.4 = 0.08$

These values are not normalised by a factor K, but the mass missing to 1 is assigned as belief in $\emptyset$:

$m_{12}(\emptyset) = 1 - m_{12}(\theta_1) - m_{12}(\theta_2) = 1 - 0.48 - 0.08 = 0.44$

Now the combination $m_{12} \oplus m_3$ can be calculated as if there were no $m(\emptyset)$, except that the beliefs in the hypotheses are lower. In fact, calculating $m(\emptyset)$ can be postponed until after the combination of all basic probability assignments. The final results are:

$m_{123}(\theta_1) = 0.12, \quad m_{123}(\theta_2) = 0.06, \quad m_{123}(\emptyset) = 0.82$

The hypothesis θ_1 is still the most likely, but the high value of $m_{123}(\emptyset)$ reflects that there has been quite a conflict between the combined *bpa*s, which is true especially between m_1 and m_3.

5.3.2 Discount Factors (Doubt)

The discount factors are an idea that dates back to Shafers initial work [52] and have later been justified [58] in the context of the *transferable belief model*. But up to now, nearly no attempts [2] seem to have been made to use this handy technique in the combination of multiple classifiers.

Assuming we have a certainty measure *cm* concerning the output of a classifier (in our case, $0 \leq cm \leq 1$, $cm = 1$ indicating a very certain classifier). Then the basic probability assignments induced by the output are scaled, or discounted, with *cm*:

$$m(A_i) = cm \cdot m(A_i), \quad m(\Theta) = 1 - cm \cdot (1 - m(\Theta))$$

Thus, $m(\Theta)$ represents the belief that the discounted belief function is based on evidence that might not be legitimate. The advantage is that this doubt is taken into account automatically when now combining sensors with the orthogonal sum. But basically, $m(\Theta)$ is based on the product of the inverted certainty measures, meaning that its value will decrease with each combination. This has to be taken into consideration when comparing doubt values, preferably restricting this to *bpa*s that have been calculated over the same number of $\oplus$ combination steps.

Discounting can be done at different stages of the fusion process, for example for each classifier or even each class of each classifier, giving ample opportunity to incorporate expert knowledge.

Example

Again, the same values as in the example in chapter 5.1 are used, but now with a certainty measure associated with each *bpa* as given in table 5.2.

Taking those certainty measures into account alters the *bpa*s as follows:

$m_1(\theta_1) = m_1(\theta_1) \cdot cm_{m_1} = 0.8 \cdot 0.7 = 0.56$

[2] Bloch [37] used discount factors in an approach to the problem of anti-personnel mine detection, but her sensors provided a measure for the certainty of the output themselves.

	m(θ_1)	**m(θ_2)**	*cm*
m_1	0.8	0.2	0.7
m_2	0.6	0.4	0.3
m_3	0.25	0.75	0.9

Table 5.2: The initial situation with certainty measure *cm* given.

$m_1(\theta_2) = m_1(\theta_2) \cdot cm_{m_1} = 0.2 \cdot 0.7 = 0.14$

$m_1(\Theta) = 1 - cm_{m_1} \cdot (1 - m_1(\Theta)) = 1 - 0.7 \cdot (1 - 0) = 0.3$

Proceeding likewise for m_2 and m_3 yields the situation shown in table 5.3.

	m(θ_1)	**m(θ_2)**	$m(\Theta)$
m_1	0.56	0.14	0.3
m_2	0.18	0.12	0.7
m_3	0.225	0.675	0.1

Table 5.3: Situation after the certainty measures have been applied.

Because of the value assigned to $m(\Theta)$, which represents the doubt towards this *bpa*, the combination process $m_1 \oplus m_2$ is more complicated:

$m_{12}(\theta_1) = K \cdot (m_1(\theta_1) \cdot m_2(\theta_1) + m_1(\theta_1) \cdot m_2(\Theta) + m_1(\Theta) \cdot m_2(\theta_1)) = K \cdot (0.56 \cdot 0.18 + 0.56 \cdot 0.7 + 0.18 \cdot 0.3) = K \cdot 0.5468$

$m_{12}(\theta_2) = K \cdot (m_1(\theta_2) \cdot m_2(\theta_2) + m_1(\theta_2) \cdot m_2(\Theta) + m_1(\Theta) \cdot m_2(\theta_2)) = K \cdot (0.14 \cdot 0.12 + 0.14 \cdot 0.7 + 0.12 \cdot 0.3) = K \cdot 0.1508$

$m_{12}(\Theta) = K \cdot m_1(\Theta) \cdot m_2(\Theta) = K \cdot 0.3 \cdot 0.7 = K \cdot 0.21$

The normalising factor K is calculated as follows:

$K^{-1} = m_1(\theta_1) \cdot m_2(\theta_1) + m_1(\theta_2) \cdot m_2(\theta_2) + m_1(\theta_1) \cdot m_2(\Theta) + m_1(\Theta) \cdot m_2(\theta_1) + m_1(\theta_2) \cdot m_2(\Theta) + m_1(\Theta) \cdot m_2(\theta_2) + m_1(\Theta) \cdot m_2(\Theta) = \sum_{\forall \text{ focal elements } f \in m_{12}} m_{12(\text{un-normalised})}(f) = m_{12}(\theta_1) + m_{12}(\theta_2) + m_{12}(\Theta) = 0.5468 + 0.1508 + 0.21 = 0.9076$

Accordingly, the normalised results are:

$m_{12}(\theta_1) = 0.6025, \quad m_{12}(\theta_2) = 0.1662, \quad m_{12}(\Theta) = 0.2314$

Calculating the combination $m_{12} \oplus m_3$ in the same way yields:

$m_{123}(\theta_1) = 0.4458, \quad m_{123}(\theta_2) = 0.5125, \quad m_{123}(\Theta) = 0.0416$

In this case, the most likely hypothesis is now θ_2, because m_1 who was supporting it strongest was discounted harder than m_3. The value of $m_{123}(\Theta)$ seems quite low, because there was quite some discounting applied, but one has to bear in mind that the value is basically the product of the inverted certainty factors.

Chapter 6

Data and Applications

Two real world data sets have been used in the experiments. They and the features extracted from them are presented in the following, together with details as to which classifiers were applied later to them.

6.1 Handwritten Numerals

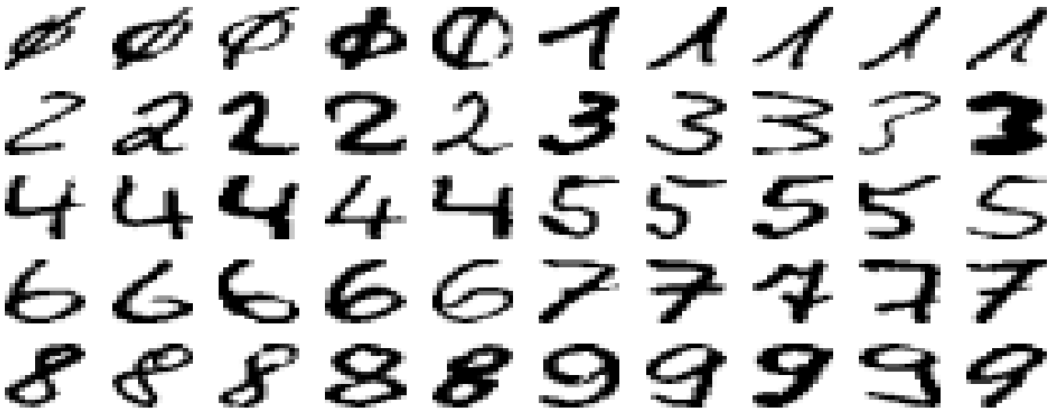

Figure 6.1: A subset of 50 samples from the handwritten numerals data set.

This is a very large dataset containing handwritten numerals for which the correct class labels are available. It consists of 10000 entries, 1000 for each of the classes 0-9. The

entries are originally represented by a 16x16 matrix containing grey-values from 0 to 255. The dataset is part of a bigger one that has been used for the evaluation of machine learning techniques in the STATLOG project [36]. Using various architectures, very good classification results can be achieved on this data. For our purpose, we do not operate on the whole picture matrix alone, but do extract the different features listed below.

- A histogram with 5 bins with equal width for the grey values of each entry (*histogram*).
- The data projected onto the first 8 principal components of the training data, extracted by means of a Karhunen-Loève transformation (*PCA*) [25, 32].
- The sums over the rows and columns. The picture is rotated ten times in the process, the sums are simply concatenated into one big feature vector with a dimension of 320 (*RowColumnSums*).
- The matrix flattened to a 256-dimensional vector (*pictures*).

Assuming the features are independent, they are each used in conjunction with different classifiers: The *RowColumnSums* and the *PCA* features will be classified using the *Gaussian Radial Basis Function Network* from chapter 2.2.2 (abbreviations: Gaussian, g_pca, g_row), for which the 5 prototypes per class have been produced using the *K-Means* algorithm from chapter 2.2.1 (20 iterations). The classification using *histogram* ($k = 5$) and *pictures* ($k = 10$) features will be accomplished with the *Fuzzy K-Nearest-Neighbour* algorithm from chapter 2.2.3 (abbreviations: fuzzy, f_histo, f_pictures).

6.2 Cricket Songs

These bioacoustic time series and the framework to access them are part of the PhD project of Christian Dietrich, who provided them kindly. Details on them can be found in [11].

Males of crickets (*Grylloidea*) 'sing' by using specialised, chitinous structures on wings and legs as stridulatory apparatus. These songs already play an integral part within descriptions of new species. Crickets are a very species-rich group with about 8000 described species, and reach their highest diversity in the tropics. As tropical habitats are destroyed on an increasing rate, there is a need to quickly identify areas rich in biodiversity. Sound recordings are a very valuable and non-invasive tool here. In addition, many crickets are sensitive indicator species for habitat quality.

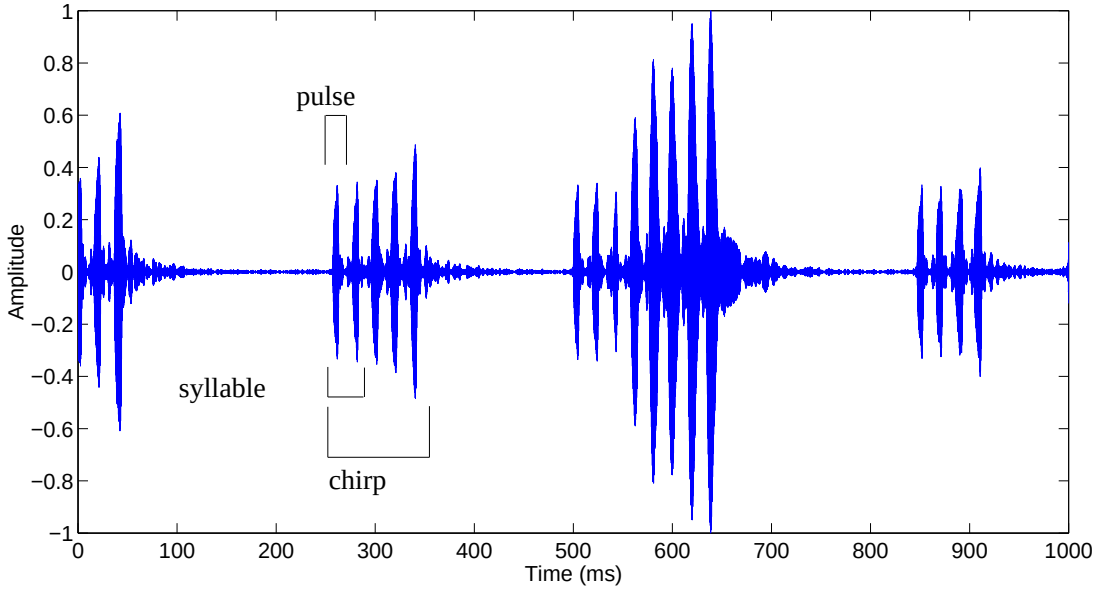

Figure 6.2: Global view of the song of a Homoeoxipha Lycoides.

The features used to classify the songs pertain to pulses. The location of these is estimated on the basis of the signal's energy, using a previously filtered signal. In the context of this paper, only local features which are characteristic for a single pulse are used:

- The basis to extract the *temporal structure of the pulses (t)* are the distances between the pulses. They are then used employing a d-tuple encoding scheme, taking into account *d* adjacent pulses.
- *Pulse length (l)* is the mean pulse length of d neighbouring pulses.
- The *pulse frequency (f)* is the average frequency over *d* pulses, where the average

frequency of one pulse is defined as $\overline{f}^k = \frac{1}{m}\sum_{j=1}^{m} f_j^k$ with $f^k \in \mathbb{R}^m$ being the frequency contour of the k-th pulse.

The cricket dataset contains 137 recorded songs from 30 different *Grylloidea* species from Thailand and Ecuador. Each feature is used as input for the *Fuzzy-K-Nearest-Neighbour* classification algorithm from chapter 2.2.3 (abbreviations: l, f, t). Accuracy is tested using 6-fold cross validation, complicated by the fact that frequently there are under 6 samples per class. So in some runs of the cross validation, there is no sample for a specific class. The algorithm takes into account that specialty.

Chapter 7

Experimental Results

The basis of the experiments are the *handwritten numerals* and *cricket songs* data sets as presented in chapter 6, were you can also find the description of how they are used in conjunction with the basic classifiers. All experiments were conducted using the Matlab software, and all results are based on stratified k-fold cross validation [1] (k = 6 for the *cricket* data and k = 10 for the *numerals* data). It is worth mentioning again that the basic probability assignments in the Dempster-Shafer combination method are simply the normalised outputs of the basic classifiers.

In this chapter, first the distribution of the certainty factors will be examined to get acquainted with the problem. Then the experimental results for the core questions are presented and discussed: Performance of the Dempster-Shafer fusion with certainty factors, changes if those certainty factors are adjusted, performance of the not normalised Dempster-Shafer fusion, and advantages of Gini or entropy certainty factors.

[1] *Stratified k-fold cross validation* is a technique to reduce the variance of the results of an (classification) experiment [26]. The data set is randomly divided into k parts. Stratification means that all classes are present in each of the samples in the same proportion as in the whole data set. Then there will be k runs of the experiment, and the average of these results will be taken. In each run, $(k-1)$ parts are used to train the classifier, and the one part left out (of course a different one for each run) is used for the evaluation of the classifier.

7.1 Distribution of the Certainty Factors

To understand and improve the combination process, it is very important to know the distribution of the certainty factor values for the different classifiers. This knowledge allows to identify several problems and to suggest solutions for them.

The diagrams 7.1 and 7.2 present some histograms with the distribution of the certainty factor values over all test samples classified by a classifier. Diagrams for all classifiers can be found in annex A. The long vertical line indicates the mean value of the certainty factor. There are two diagrams for each classifier, to make the distinction between the samples which were classified correctly and those who were not. The hope is, of course, that the histograms for the two cases are well different, falling into the two desired categories described earlier in table 4.1.

data set	**classifier on feature**	**accuracy**	**mean certainty factor on false**	**mean certainty factor on correct**
numerals	**pictures**	87.98%	0.735	0.995
numerals	**RowColumnSums**	92.00%	0.171	0.257
numerals	**PCA**	87.56%	0.122	0.314
numerals	**histogram**	16.13%	0.625	0.647
cricket	**f**	22.76%	0.750	0.753
cricket	**t**	66.04%	0.569	0.663
cricket	**l**	30.61%	0.392	0.443

Table 7.1: Accuracy of the classifiers and mean entropy certainty factors on the correctly and falsely classified samples.

Unfortunately, the classifiers do not behave so distinctly. In spite of being wrong about the current sample, they will produce a confident answer that yields a high certainty factor. The noble exception is the classifier operating on the pictures feature, when being right it will nearly always give a very crisp answer, and express doubt else. Still there is some difference in the distributions even with the other classifiers: the mean certainty factor for the samples classified correctly is higher than the mean certainty

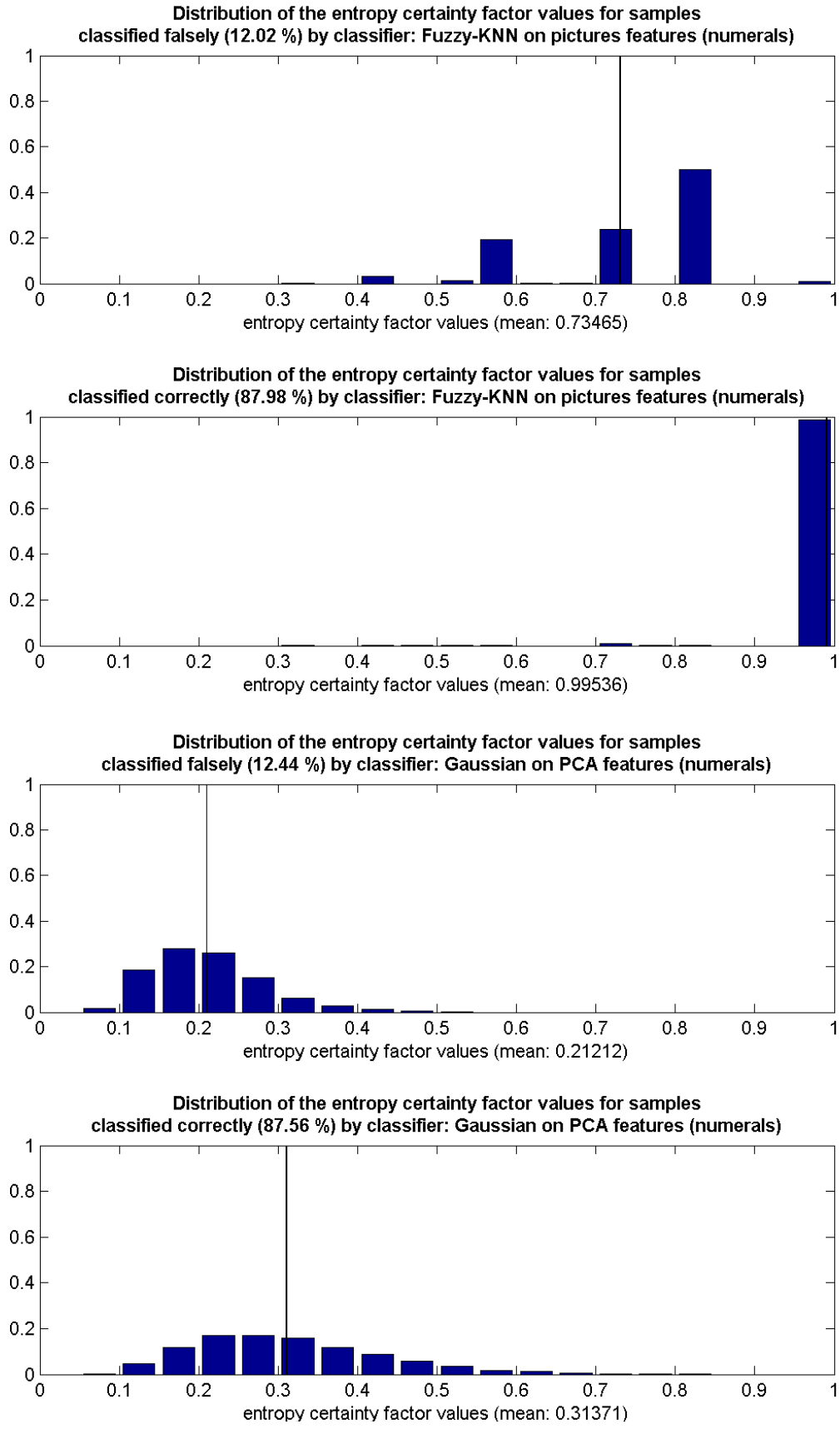

Figure 7.1: Distribution of the entropy certainty factor values on the numerals data (pictures and PCA features).

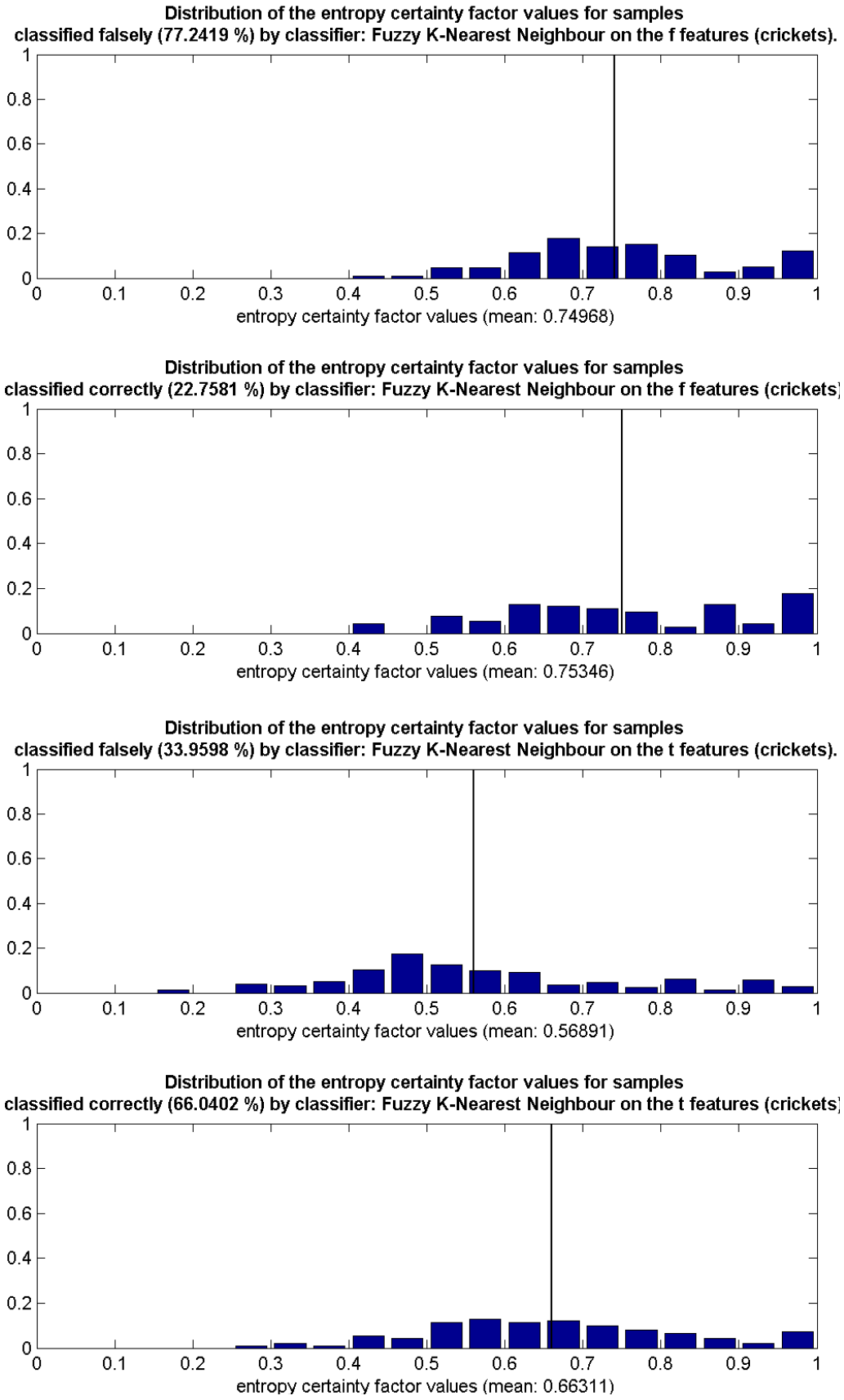

Figure 7.2: Distribution of the entropy certainty factor values on the cricket data (f and t features).

factor on the samples who elicited a wrong answer. This *gap* is the lever that will allow the Dempster-Shafer fusion process to distinguish if the answer of the classifier is useful.

The gap of the mean certainty factors has of course its highest value of 26% for the pictures feature with its crisp answers. It drops to 6%-10% for most other classifiers. The very low values of 2% for the classifiers on histogram and f features indicate that the combination algorithm will not be able to make any use out of their certainty factors. This is not to say, however, that those classifiers should be omitted, as they can (and in the case of the one operating on the f feature, will) make independent errors and thus contribute to the overall accuracy.

7.2 Classification Results

The goal is to improve the classification performance by combining several classifiers. But to see if we are successful, we have to know what we are comparing against. Three measures will be the base for comparison here: The performances of the single best classifier, the hardened probabilistic product, and the decision template fusion.

	numerals	**cricket**
single best classifier	92.00%	66.04%
probabilistic product	91.58%	55.24%
decision templates	69.53%	84.02%

Table 7.2: Accuracy on the numerals and cricket data set.

The single best classifier on the numerals data is the one operating on the RowColumn-Sums feature, having an accuracy of 92%, the best one on the cricket data operates on the t feature with 66% accuracy. The values for all classifiers are presented in table 7.1.

Combining the classifiers using a probabilistic product per class, then hardening the

decisions (see chapter 3.3), yields an accuracy of 91.58% [2] on the numerals data, and 55.24% on the cricket data. Note that this method is equivalent to the plain Dempster-Shafer fusion as described in chapter 5.1. Several experiments showed that the classifier operating on the histogram data can not contribute to the accuracy of the combined results, and will therefore not be used.

The application of decision templates for classifier fusion (see again chapter 3.3) yielded surprising results[3]: accuracy on the numerals data was as low as 69.53%, but rose to an astonishing 84.02% on the cricket data. On this set, the standard deviation of the accuracy in the cross-validation was 0.08, reflecting the big impact of the training samples on the classifier when, like here, only few samples are available. Contributing to the success of decision templates on the cricket data (a constellation also employed in [11]) is the fact that the classifier operating on the f feature has a poor accuracy, but does possess a characteristic answer pattern for each class, something which can be exploited by the combination method. A look at the normalised mean per-class entropy in the decision templates found during the training phase backs this finding: the value of 0.3451 on the numerals data means that the classifiers agree far more often than on the cricket data set with a value of 0.1055. The especially high performance of the DT fusion on the cricket data can not be beaten by the newly proposed method, and hence the performance of the single best classifier will mainly be used as a reference in the following.

[2] Accuracy is 91.85% when leaving out the classifier operating on the histogram data, dropping to 39.76% when including it.

[3] Because the classifier output of the training phase was not easily available in this phase of the work, the following approach has been taken: With classifier outputs being available in k sets from the cross validation process, $(k-1)$ of them were used as training data, and one as test set. To arrive again at a k-fold cross validation, k passes of this were done with a different test set each time. This means that the accuracy of true decision template fusion would be higher, because the templates constructed would be specific to the trained classifiers actually used for testing. But as this would be especially true for the cricket data set with few training samples and hence a big difference in the trained classifiers, and the method performing very well on this data set, one can assume that the effect is not grave.

7.2.1 Certainty Factors in Dempster-Shafer Fusion (Doubt)

The different classifiers will be fused using the certainty factors as discount factors as described in chapter 5.3.2. During the process, there is a value assigned to $m(\Theta)$, representing the doubt that the decision is based on correct evidence. Now all samples with a doubt value over a certain threshold will be rejected, thus trading an increased rejection rate for accuracy on the not rejected samples. This system works well for the numerals data set, as shown in figure 7.3.

When no samples are rejected, the accuracy is lower than the one achieved using the simple probabilistic product. This is because the certainty factors/discount factors alter the impact each classifier has in the combination process. In case of the numerals data set, the certainty factor values of the classifier on the pictures feature are very dominant (compare figure 7.1 and table 7.1), so dominant in fact that the combined accuracy equals exactly the accuracy of this classifier. On the crickets data, the certainty factor values for the worst classifier (operating on the f feature, see figure 7.2) have the biggest impact. This problem is dealt with in the next section.

7.2.2 Adjusting the Certainty Factors

As shown in chapter 7.1, the distribution of the certainty factor values is quite different for each classifier. This automatically gives a classifier with a higher mean certainty factor more weight in the combination process (see chapter 5.3.2), if this is justified or not. Hence, the certainty factors have to be adjusted to give each classifier the optimal weight.

In this process, one can also try to enhance the width of the gap between the mean certainty factor values for the correctly and falsely classified samples. To this end, for each classifier a threshold is defined above which the certainty factors are increased artificially by multiplying them with a adjustment value greater than one, while discounting the certainty factors below the threshold.

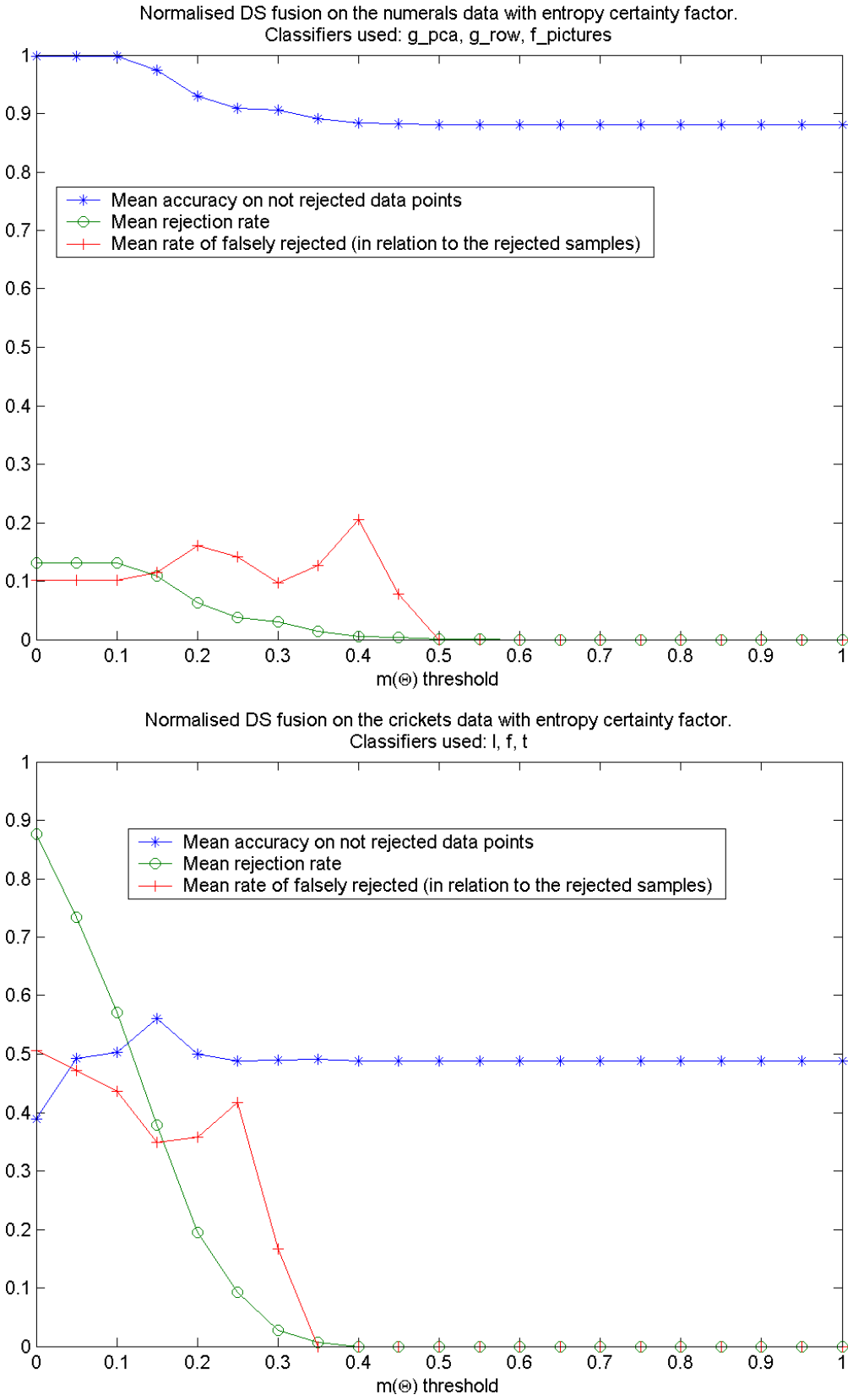

Figure 7.3: Results of the normalised Dempster-Shafer fusion on the numerals and cricket data.

But unfortunately, choosing the right adjustment values to give the classifier the optimal weight is a complex issue, depending on the following features of the classifier:

- Its accuracy
- The width of the certainty factor gap
- The independence of its error rate

Hence, there is no straightforward and simple way to predict the best threshold and adjustment factor values. But the observations made in the experiments allow the conclusion that the threshold value should be around the value of the mean certainty factor on the falsely classified samples. Just making the the gap wider here, for example by using the adjustment values (0.5,2) does yield only bad results. A good technique to begin with is to adjust the mean certainty vector of each classifier to match its accuracy. If simultaneously widening the gap helps the overall accuracy seems to depend on the actual data set.

To illustrate the above abstract guidelines on adjusting the certainty factor, the following presents the optimised settings which yielded optimal results.

On the numerals data set, the distribution of the certainty factors of the classifier based on the pictures feature gives an excellent example for the merits of the approach: correct answers have an certainty factor above 0.95, while nearly all false answers have a certainty factor below 0.85 (see figure 7.1). The obvious choice for a threshold is 0.95, below which the certainty factor will be set to 0. Certainty factors on the classifiers using PCA and RowColumnSums features are adjusted as noted in figure 7.4, choosing the mean value of the certainty factor on the false samples as threshold, tuning the adjustment factors so that the mean certainty factor on correctly classified samples is 0.49 for each, giving them equal influence. The distribution of the adjusted certainty factors for the classifier working on the PCA feature is shown in figure 7.4. Comparing the performance of the combined classifier with adjusted versus natural certainty factors (figure 7.4 versus figure 7.3) yields encouraging observations: Now even without rejecting any samples, the accuracy is two percent points higher than the one of

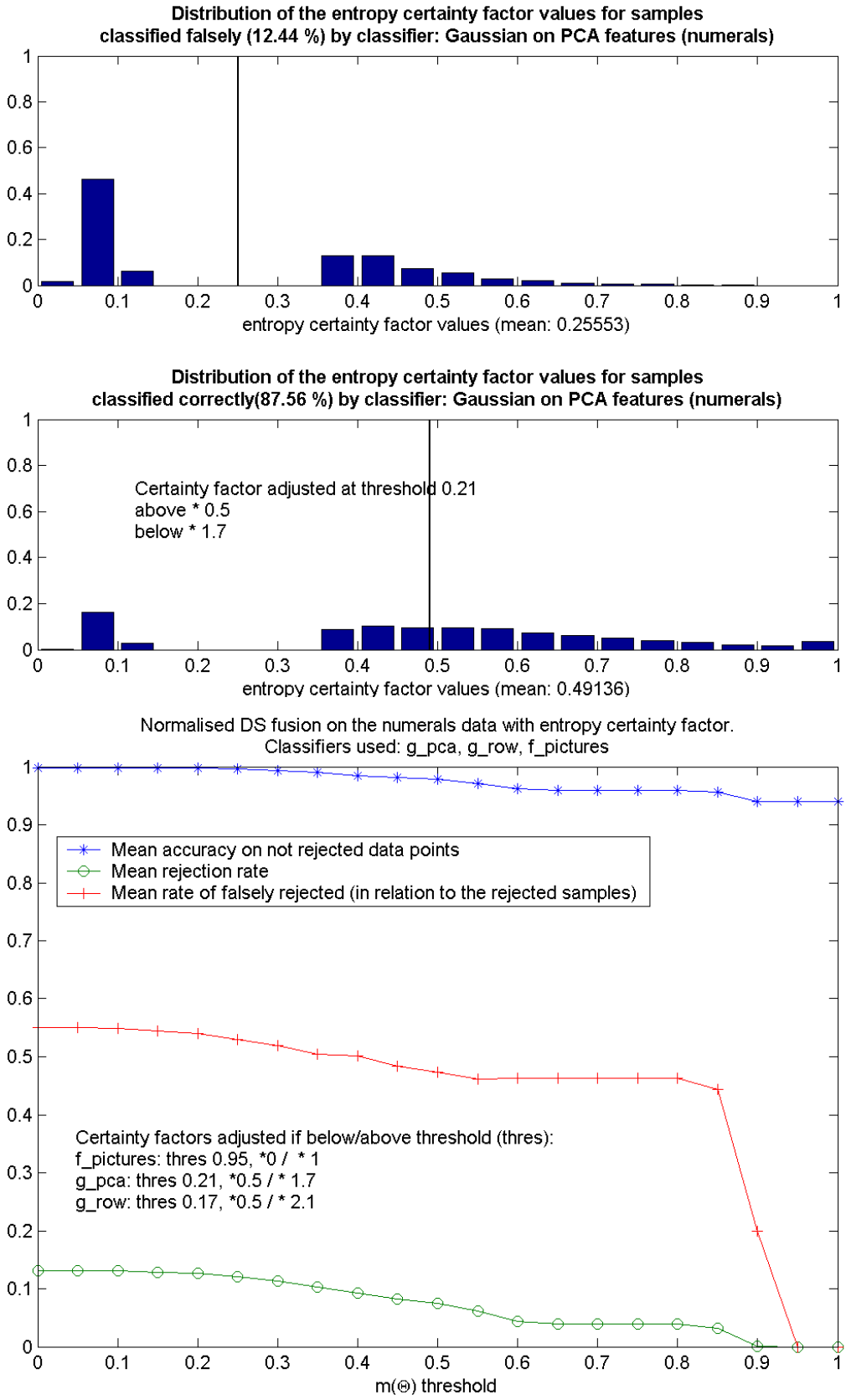

Figure 7.4: Distribution of the adjusted certainty factor values on the numerals data (PCA feature); Results of the normalised Dempster-Shafer combination with adjusted certainty factors.

the best single classifier, and significantly better than without the adjusted certainty factors! Also, the rejection rate rises more gently, permitting to increase the accuracy while still rejecting few samples.

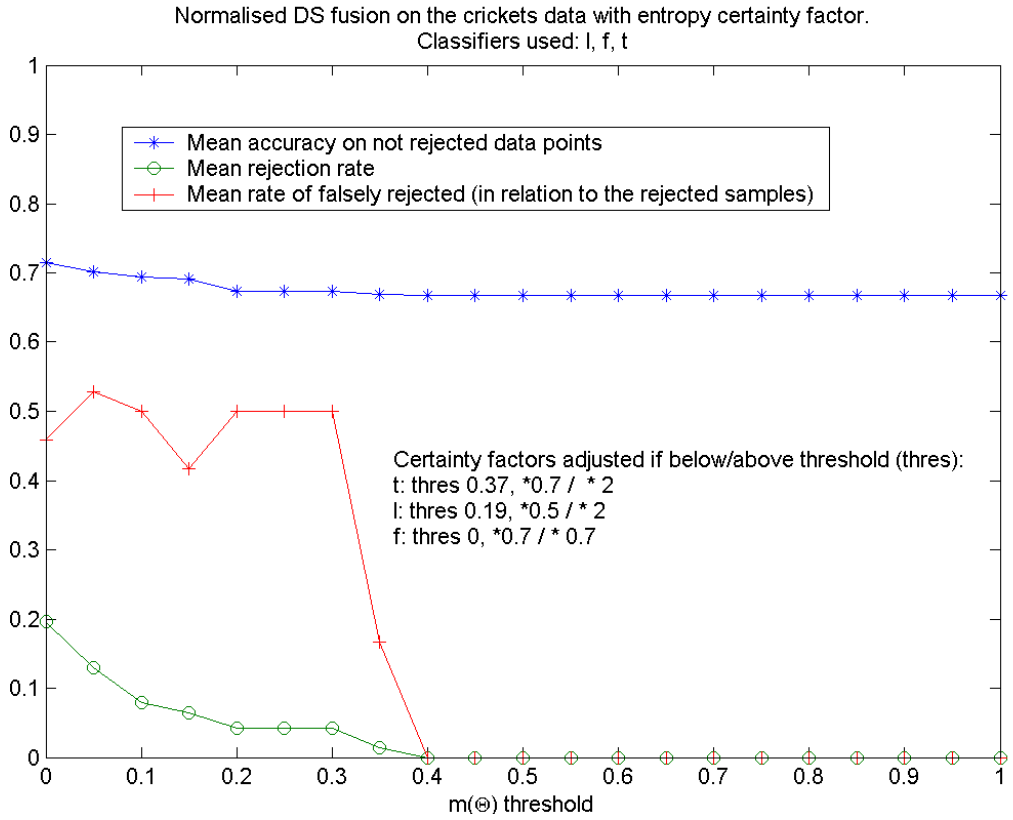

Figure 7.5: Results of the normalised Dempster-Shafer combination on crickets data with adjusted certainty factors.

The situation is not so clear on the crickets data. The optimal thresholds for adjusting the certainty factors of the classifiers on the t and l features are 0.2 below the above recommended value (see figure 7.5). The certainty factors here are not adjusted to have an equal mean value, but in a way that classifiers with higher accuracy have a higher mean certainty factor. It is very interesting to see that the worst classifier (on the f feature) with also a nearly nonexistent certainty factor gap has to take part in the classification process, certainty factors adjusted by 0.7, in order for the combined results to be better (by 0.75 percent points) than the single best classifier. This shows that its classification errors are independent of the other classifiers'. As on the numerals

data, the rejection rate rises much slower than with natural certainty factors.

These results do spoil the hope that the combined accuracy could be boosted by the certainty factors without including any knowledge from the training data, because the mean certainty factor of the falsely classified samples is an important hint on where to set the adjustment threshold. Where this puts the proposed combination technique in relation to others in the literature will be discussed in chapter 8.

7.2.3 Not Normalised Dempster-Shafer Fusion (Conflict)

The benefit of not normalising the beliefs after a combination via the orthogonal sum is that the conflict between the two combined sources is retained as belief in $\emptyset$ (see chapter 5.3.1). Other than the the certainty factors, this technique does not affect the rang order of the classes in the combined result, but gives an additional measure that can be used to reject possible falsely classified samples.

The approach works quite well, as shown in figure 7.6. Rejecting all samples whose combined classification is associated with a doubt $m(\emptyset)$ higher than a certain threshold boosts the accuracy. The cost of the boost is the increased rejection rate, which rises gently with the numerals data, but does a sudden jump to 34% at the beginning with the cricket data.

Combining not-normalised fusion and certainty factors does not prove to be successful. For the numerals data, no increase in accuracy can be achieved over fusion with certainty factors by rejecting samples with a conflict higher than a threshold [4]. An explanation for this failure is that the belief in Θ is partly redistributed among the θ_i, thus reducing the obvious conflict between the sources that is caused by the very hard answers of the classifier operating on the pictures feature. The accuracy on the cricket data can be increased, but only at the price of a very high rejection rate. If the certainty

[4] Except for one case: if the classifier operating on the histogram feature is not excluded (see chapter 7.2), an $\emptyset$-threshold of 0.9 gains 2.4 percent points accuracy, in effect together with not-adjusted certainty factors leading to an accuracy of 99% with a rejection rate of only 15%, Θ-threshold set to 0.

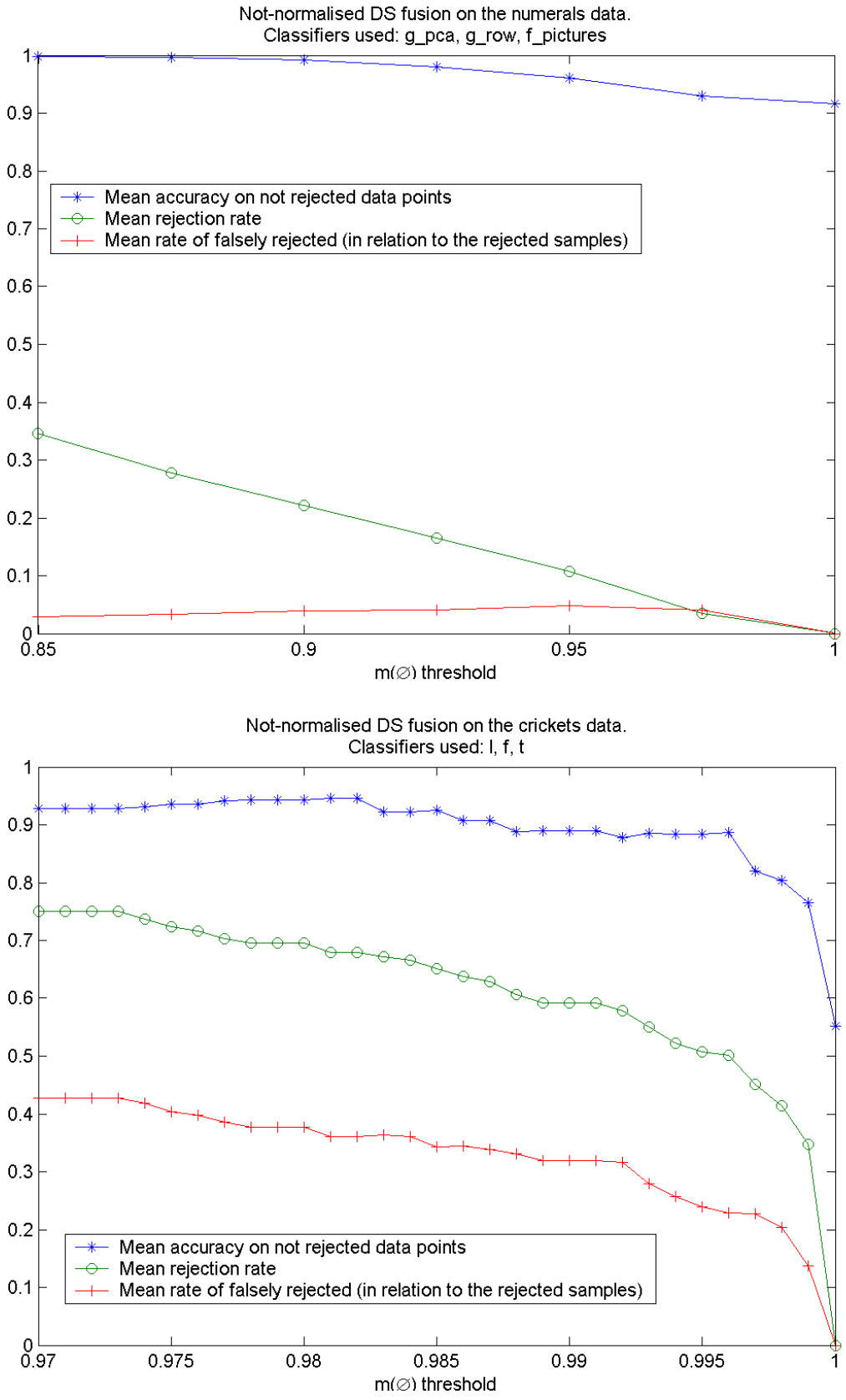

Figure 7.6: Not-normalised Dempster-Shafer fusion on the numerals and cricket data. (The x-axes are scaled differently.)

factors are adjusted, the possible gain remains little (70.9% accuracy with a rejection rate of 70.9%). Despite that, an encouraging result is the fact that very rarely a sample would be rejected because it violated the conflict- as well as the doubt-threshold. This means that these two parameters can be optimised independent of each other!

7.2.4 Entropy or Gini as Certainty Factor

In the fusion experiments so far, only the entropy certainty factor has been used. This restriction is justified by the following experimental results.

Generally, the distribution of the Gini certainty factors on the samples is a little bit less smooth than the distribution of the entropy certainty factors (see figure 7.7). The Gini values are also significantly lower, which may come as a surprise given the two-element characteristic of the functions as shown in figure 4.1.

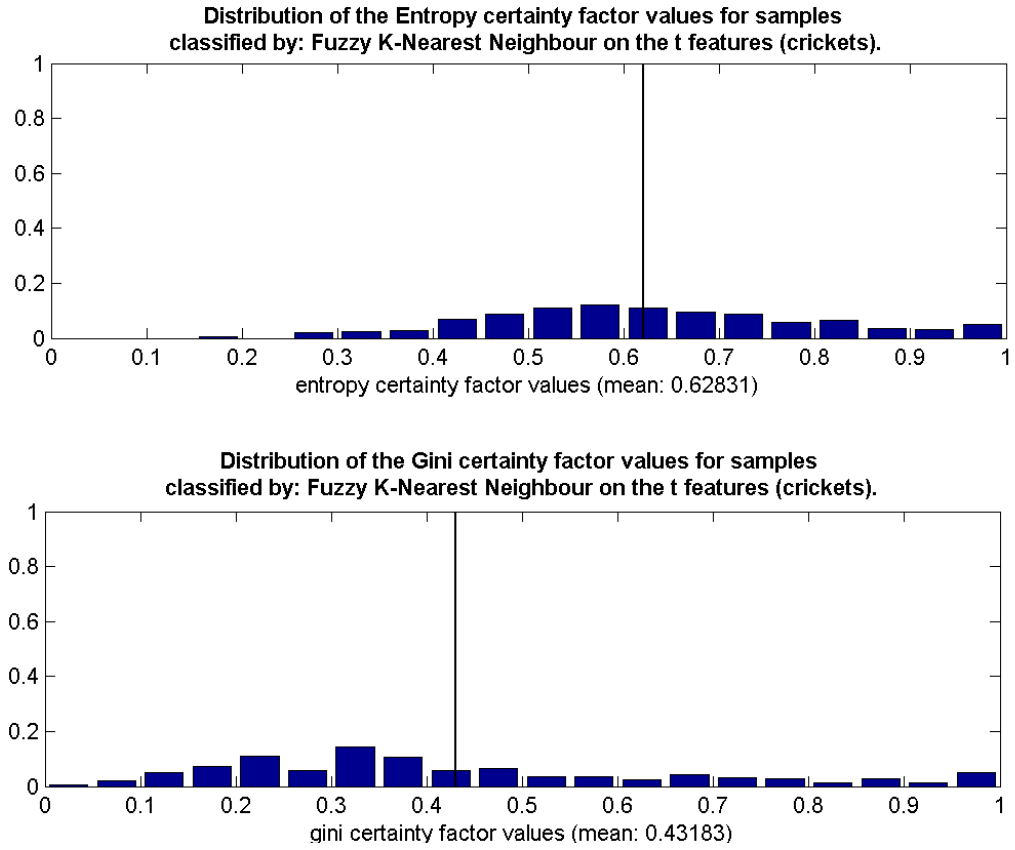

Figure 7.7: Gini and entropy certainty factors on the crickets data set (t feature).

Performing a Dempster-Shafer fusion on the cricket data with a Gini certainty factor yields a performance that is 1.6 percent points higher than with the entropy certainty factor, and other than there, it can be increased by 8 percent points if accepting a high rejection rate of up to 48 %. On the numerals data, the different certainty factors show about the same performance, only the rejection rate rises a little more gradually.

The picture is quite different with the Dempster-Shafer fusion using adjusted certainty factors. Here, the performances on the numerals data set are the same. The rejection rate rises faster when using the Gini certainty factor, but the performance/rejection ratio remains the same. On the cricket data set, the performance using the entropy certainty factor is always 2 percent points higher. As the fusion technique with adjusted factors has the best overall results, it is recommended to use the entropy certainty factor.

Chapter 8

Summary, Classification of the Approach, and Future Research

8.1 Summary of the Results

Certainty factors derived from the outcome of the classifiers and then used in the Dempster-Shafer fusion method do not lead to a magnificently improved accuracy of the combined classifiers. But at least on the numerals data set, the correctness can be increased to near 100% with this simple method, if a certain amount of rejected samples is permitted (figure 7.3).

The answers of the classifiers alone do not allow the certainty factors derived from them to predict how accurate the output is, the gap between the mean certainty factor of false respectively correct answers is not big enough (chapter 7.1). Hence it is necessary to include knowledge from the training phase, by adjusting the certainty factors so that they give their classifier the optimal influence in the decision fusion process. It was only possible to find very general rules for this adjustment (chapter 7.2.2), and should again be stressed that the widening of the gap is of secondary importance. This approach led to very good results (figure 7.4), although it could not beat the *decision templates fusion* on the cricket data set (figure 7.5).

Using the not normalised Dempster-Shafer fusion allows to increase the overall accuracy be rejecting samples with a high conflict measure $m(\emptyset)$. The combination with certainty factors did not prove successful, but did no harm either (chapter 7.2.3).

8.2 Classification of the Approach

The Dempster-Shafer classifier fusion approach incorporating certainty factors operates on soft classifier answers. It is not a technique for *coverage optimisation*, that is one for constructing diverse classifiers, but for *decision optimisation*, the fusion of many classifiers' answers [48]. Regarding the aspect of $fusion$ versus $selection$ [9], the approach is a hybrid: it fuses the results of the classifiers via the orthogonal sum, but using the certainty factors also chooses the classifier that should have the biggest impact. It does not go so far like the *brute force* or *stacked generalization* [62] methods who treat the classifier outputs as a new intermediate feature space, but still attaches meaning to the classifier answers. The classification of the approach as $static$ or *dynamic* is not clear, because it generally does only look at the classifier output, but incorporates some information from the training phase in form of the adjusting of the certainty factors. Being basically static, it does not make much sense to combine it with dynamic classifier fusion methods who can draw much more information than provided by certainty factors from the training phase. This would be different of course if, like in [37], the classifiers could provide those measures themselves!

8.3 Future Research

There are some aspects of certainty factors which, in the opinion of the author, are worth further investigations.

The basic probability assignments are just the classifier outputs per class here. A more intricate approach is conceivable, for example following Rogova [47] as described in

chapter 5.2, but additionally weaving the certainty factors into the process of finding the per-class-per-classifier *bpa*s.

Despite decision templates being a dynamic classifier fusion technique, certainty factors could be included. For example when comparing templates, the distance to a classifier labeled uncertain would have less impact. On an earlier stage, employing certainty factors in the creation of the templates could make these more distinct.

Certainty factors could also be used as a measure to construct basic classifiers that can better shape their answers, depending if they are most likely right or wrong. Likewise, the measures could be employed as a lead to find diverse classifiers for the fusion ensemble, classifiers which would make different, independent errors.

Bibliography

[1] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216.

[2] Kamal M. Ali and Michael J. Pazzani. On the link between error correlation and error reduction in decision tree ensembles. Technical Report 95-38, Department of Information and Computer Science, University of Calfornia, Irvine, 1995.

[3] Ahmed Al-Ani and Mohammed Deriche. A new technique for combining multiple classifiers using the Dempster-Shafer theory of evidence. *Journal of Artificial Intelligence Research*, 17:333–361, 2002.

[4] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 2000.

[5] Isabelle Bloch. Information combination operators for data fusion: A comparative review with classification. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 26(1):52–67, 1996.

[6] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[7] Leo Breiman. Bagging predictors. *Machine Learning*, 24(12):123–140, 1996.

[8] D. S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–351, 1988.

[9] B. V. Dasarathy and B. V. Sheela. A composite classifier system design: Concepts and methodology. *Proceedings of the IEEE*, 67(5):708–713, 1979.

[10] Arthur P. Dempster. A generalization of Bayesian inference. *Journal of the Royal Statistical Society*, 30:205–247, 1968.

[11] Christian Dietrich, Friedhelm Schwenker, Klaus Riede, and Günther Palm. Classification of bioacoustic time series utilizing pulse detection, time and frequency features and data fusion. Ulmer Informatik-Berichte 2001-04, University of Ulm, Germany, 2001.

[12] Thomas G. Dietterich. Ensemble methods in machine learning. In Josef Kittler and Fabio Roli, editors, *Multiple Classifier Systems*, pages 1–15. Springer, 2002.

[13] Didier Dubois and Henri Prade. *Fuzzy Sets and Systems: Theory and Applications*. Academic Press, 1980.

[14] Didier Dubois and Henri Prade. Combination of information in the framework of possibility theory. In Mongi A. Abidi and Rafael C. Gonzalez, editors, *Data Fusion in Robotics and Machine Intelligence*, pages 481–505. Academic Press, 1992.

[15] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1022–1027. Morgan Kaufmann, 1993.

[16] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *13th International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann, 1996.

[17] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, New York, 2nd edition, 1990.

[18] Deltas George. The small sample bias of the Gini coefficient: Results and implications for empirical research. *Review of Economics and Statistics*, 85(1):226–234, 2003.

[19] Jerzy W. Grzymala-Busse. *Managing Uncertainty in Expert Systems*. Kluwer Academic Publishers, Boston, 1991.

[20] Jochen Heinsohn and Rolf Socher-Ambrosius. *Wissensverarbeitung. Eine Einführung.* Spektrum Akademischer Verlag, 1999.

[21] John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, 1991.

[22] Tin Kam Ho. Complexity of classification problems and comparative advantages of combined classifiers. In Josef Kittler and Fabio Roli, editors, *Multiple Classifier Systems*, pages 97–106, 2000.

[23] Y. S. Huang and C. Y. Suen. A method of combining multiple experts for the recognition of unconstrained handwritten numerals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(1):90–94, 1995.

[24] S. A. Hutchison and A. C. Kak. Multisensor strategies using Dempster-Shafer belief accumulation. In Mongi A. Abidi and Fafael C. Gonzalez, editors, *Data Fusion in Robotics and Machine Intelligence*, pages 165–209. Academic Press, 1992.

[25] K. Karhuhnen. Zur Spektraltheorie stochastischer Prozesse. *Annales Academiae Scientiarum Fennicae*, 34, 1946.

[26] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, Montreal*, pages 1137–1143. Morgan Kaufmann, 1995.

[27] Ludmila I. Kuncheva, James C. Bezdek, and Robert P. W. Duin. Decision templates for multiple classifier fusion: An experimental comparison. *Pattern Recognition*, 34:299–314, 2001.

[28] Ludmila I. Kuncheva. Switching between selection and fusion in combining classifiers: An experiment. *IEEE Transactions on SMC, Part B*, 32(2):146–156, 2002.

[29] Ludmila I. Kuncheva. Classifier fusion. Tutorial handed out at a conference.

[30] Ludmilla I. Kuncheva. Clustering-and-selection model for classifier combination. In *4th International Conference on Knowledge-Based Intelligent Engineering Systems*, 2000.

[31] Ludmilla I. Kuncheva. Using measures of similarity and inclusion for multiple classifier fusion by decision templates. *Fuzzy Sets and Systems*, 122(3):401–407, 2001.

[32] M. M. Loève. *Probability Theory*. Van Nostrand, Princeton, 1955.

[33] David Lowe. The handbook of brain theory and neural networks. chapter Radial Basis Function Networks, pages 779–782. MIT Press, Cambridge, 1995.

[34] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–298, 1967.

[35] Eberhard Mandler and Jürgen Schürmann. Combining the classification results of independent classifiers based on the Dempster/Shafer theory of evidence. *Pattern Recognition and Artificial Intelligence*, pages 381–393, 1988.

[36] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, editors. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, New York, 1994.

[37] Nada Milisavljevic and Isabelle Bloch. Sensor fusion in anti-personnel mine detection using a two-level belief function model. *IEEE Transactions on Systems, Man and Cybernetics, Part C*, 33(2):269–283, 2003.

[38] J. Moody and C. J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1:184–294, 1989.

[39] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[40] Tomaso Poggio and Federico Girosi. A theory of networks for approximation and learning. Technical Report AIM-1140, MIT, Cambridge, 1989.

[41] James J. Pomykalski, Walter F. Truszkowski, and Donald E. Brown. Expert systems. In J. Webster, editor, *Wiley Encyclopedia of Electronic and Electrical Engineering*. 1999.

[42] M. J. D. Powell. Radial basis functions for multivariate interpolation: A review. In J. C. Mason and M. G. Cox, editors, *Algorithms for Approximation*, pages 143–168. Clarendon Press, Oxford, 1987.

[43] P. Pudil, J. Novovicova, S. Blaha, and Josef V. Kittler. Multistage pattern recognition with reject option. In *Proceedings of the 11th International Conference on Pattern Recognition*, volume 2, Conf. B, pages 92–95. IEEE Computer Society Press, 1992.

[44] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[45] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[46] Brian D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[47] Galina Rogova. Combining the results of several neural network classifiers. *Neural Networks*, 7(5):777–781, 1994.

[48] Fabio Roli and Giorgio Giacinto. Design of multiple classifier systems. In H. Bunke and A. Kandel, editors, *Hybrid Methods in Pattern Recognition*, pages 199–226. World Scientific Publishing, 2002.

[49] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. Rumelhart and J. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*, volume 1, pages 318–362. 1986.

[50] Uwe Schöning. *Algorithmen - Kurz Gefasst*. Spektrum Akademischer Verlag, 1997.

[51] Friedhelm Schwenker, Hans A. Kestler, and Günther Palm. Three learning phases for radial-basis-function networks. *Neural Networks*, 14:439–458, 2001.

[52] Glenn Shafer. *A Mathematical Theory of Evidence*. University Press, Princeton, 1976.

[53] Glenn Shafer. Dempster-Shafer Theory. `http://www.glennshafer.com/assets/downloads/article48.pdf`, ∼2002.

[54] Claude Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.

[55] Amanda J. C. Sharkey. Multi-net systems. In *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems*, pages 3–30. Springer, 1999.

[56] Sameer Singh. 2D spiral pattern recognition with possibilistic measures. *Pattern Recognition Letters*, 19(2):141–147, 1998.

[57] Philippe Smets. The nature of the unnormalized beliefs encountered in the transferable belief model. In *Proceedings of the 8th Conference on Uncertainty in Artificial Intelligence*, pages 292–297, San Mateo, California, 1992. Morgan Kaufmann.

[58] Philippe Smets. Belief functions: The disjunctive rule of combination and the generalized Bayesian theorem. *International Journal of Approximate Reasoning*, 9:1–35, 1993.

[59] Vladimir Vapnik. *Theory of Pattern Recognition [in Russian]*. Nauka, Moscow, 1974.

[60] K.-D. Wernecke. A coupling procedure for discrimination of mixed data. *Biometrics*, 48:497–506, 1992.

[61] Ian H. Wittan and Frank Eibe. *Data Mining - Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco, 1999.

[62] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[63] Kevin Woods, W. Philip Kegelmeyer, and Kevin Bowyer. Combination of multiple classifiers using local accuracy estimates. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 19(4):405–410, 1997.

[64] Lei Xu, Adam Krzyzak, and Ching Y. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(3):418–435, May/June 1992.

[65] Ronald R. Yager and Janus Kacprzyk, editors. *The Ordered Weighted Averaging Operators*. Kluwer Academic Publishers, 1997.

[66] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[67] L. A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1:3–28, 1978.

[68] L. A. Zadeh. Book review: A mathematical theory of evidence. *AI Magazine*, 5(3):81–83, 1984.

Appendix A

Histograms of the certainty factors

The diagrams below present histograms with the distribution of the entropy certainty factor values over all test samples classified by each classifier. The long vertical line indicates the mean value. There are two diagrams for each classifier, to make the distinction between the correctly and falsely classified samples.

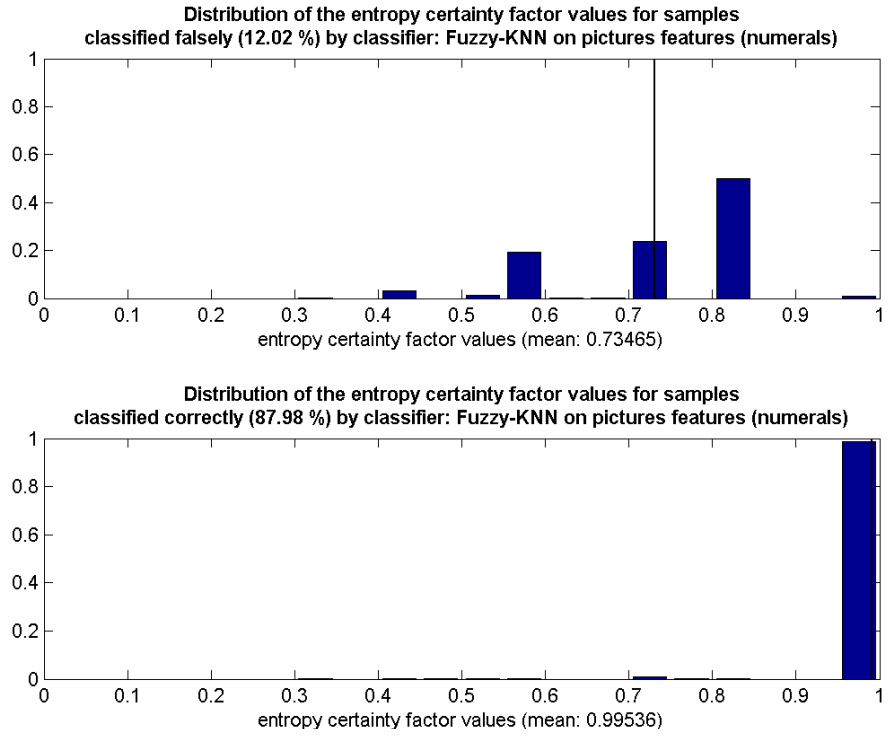

Figure A.1: Distribution of the entropy certainty factor values on the numerals data (pictures feature).

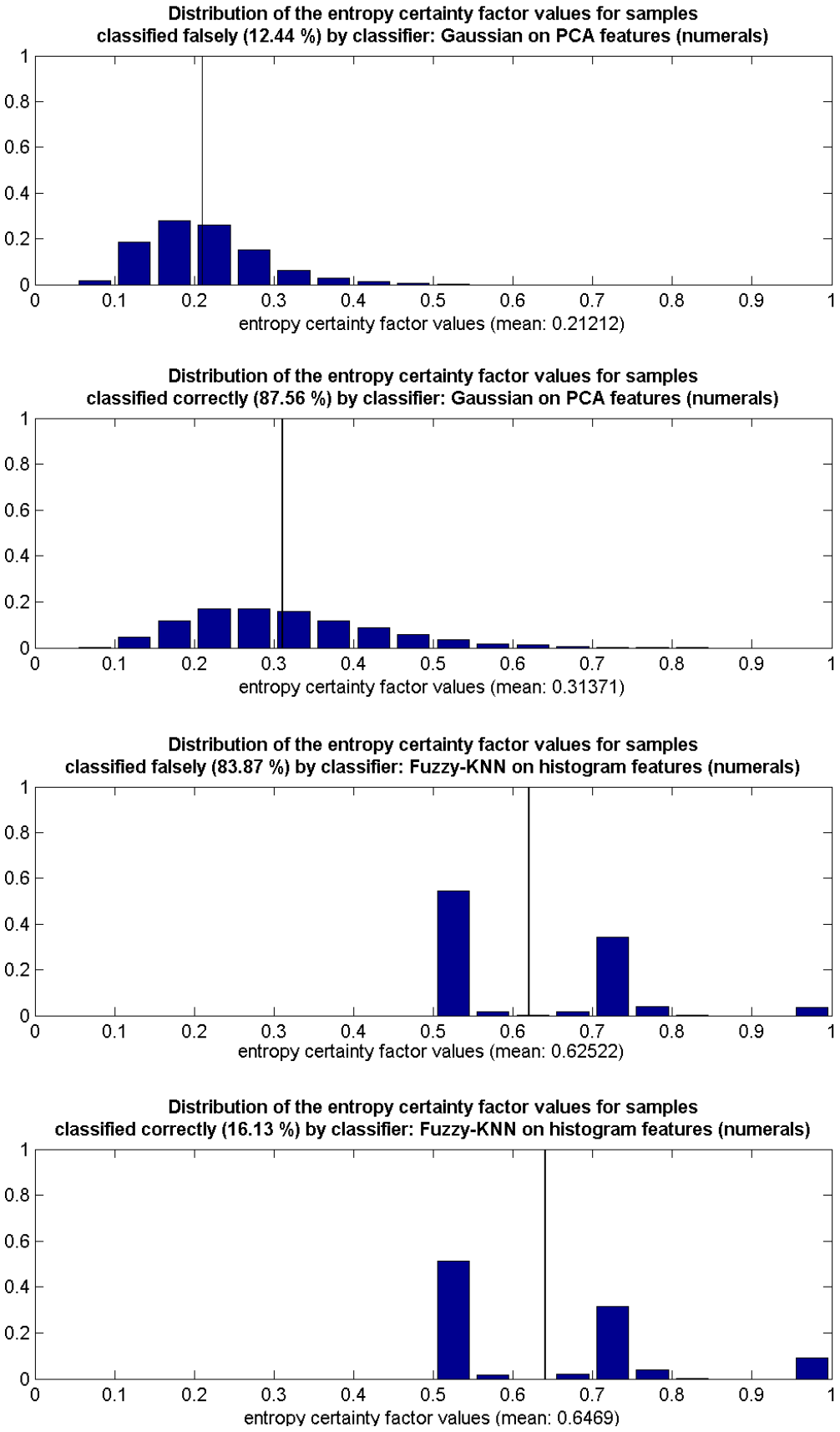

Figure A.2: Distribution of the entropy certainty factor values on the numerals data (PCA and histogram features).

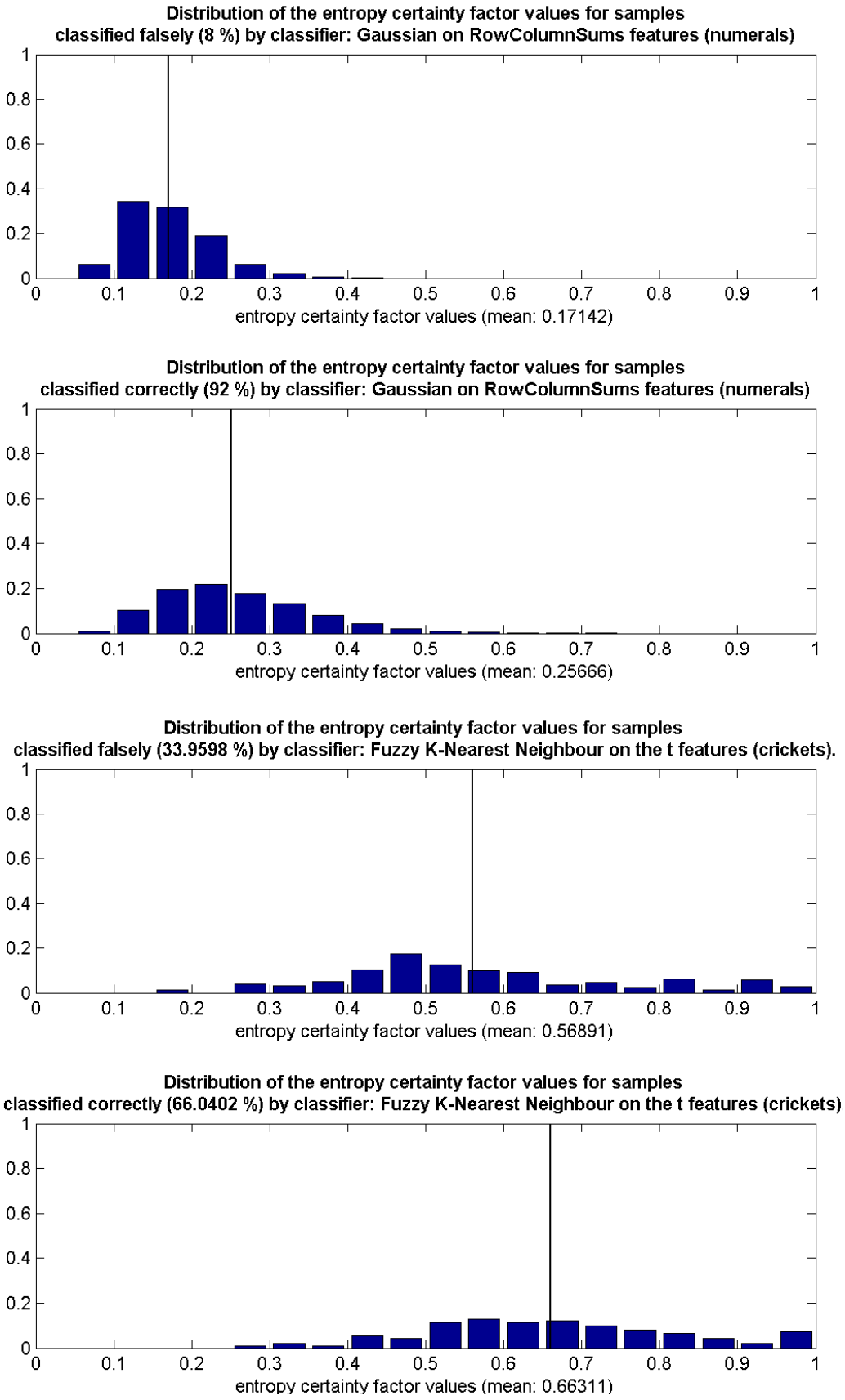

Figure A.3: Distribution of the entropy certainty factor values on the numerals and cricket data (RowColumnSums and t features).

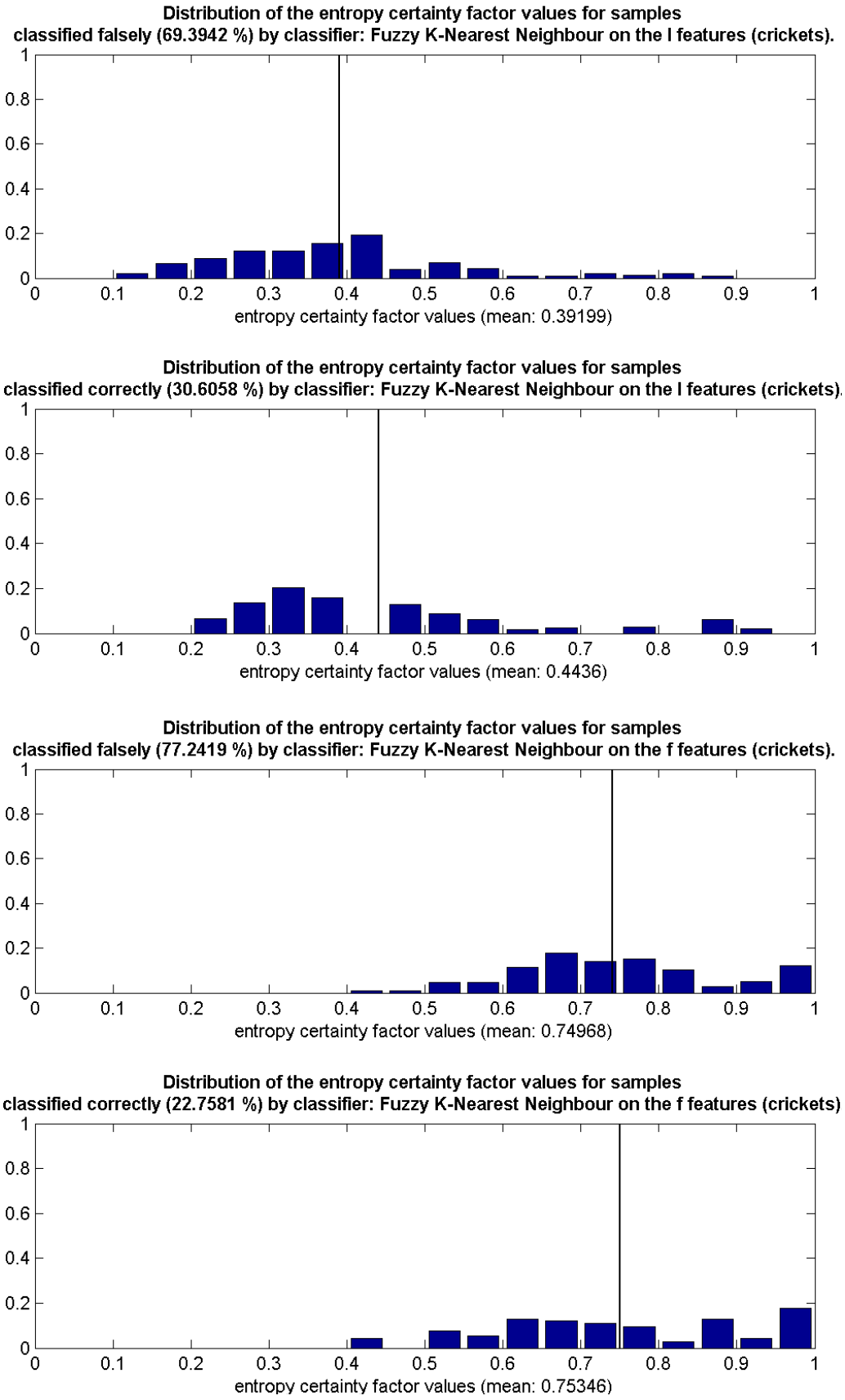

Figure A.4: Distribution of the entropy certainty factor values on the cricket data (l and f features).

Appendix B

Technical Details

For the experimental evaluations, the software Matlab 6.5 was utilised. This appendix helps to use the programs written for the work. First, the location of the different data sets will be explained, then a few interesting difficulties encountered are mentioned. The last section gives an overview over the functionality of each program file.

Data

The raw numerals data is available in the subdirectory `numerals_data`. Using the scripts in subdirectory `not_needed_any_more`, features were extracted, the classifiers trained and tested, and the k-fold cross-validated results saved in the files `Ziffern_result_fuzzyknn_histogram.mat`, `Ziffern_result_fuzzyknn_pictures.mat`, `Ziffern_result_gaussian_pca.mat`, `Ziffern_result_gaussian_rowcolumnsums.mat`, the labels in `Ziffern_teach.mat`. The classifier answers to the cricket data are in the files in subdirectory `cricket_data`. The loading phase of the script `performance_ds.m` is the best place to learn how the classifier results are stored in these files.

For all functions, it is assumed that class labels are integers, ascending from 1. Where this is not the case, a mapping function would be necessary.

Interesting difficulties

Some minor difficulties encountered during the realisation could be of interest to someone embarking on a similar task:

- When normalising the output of RBF networks, one has to keep in mind that it can be negative (compare `gaussian_classify_fuzzyresult.m`).
- When fusing two absolutely sure but contradicting sensors with DS and discounting, the straightforward formula for normalisation does not work and $m(\Theta)$ has to be set to 1 as a special case.
- During DS fusion with certainty factors, a factor of 0 leads to an sensor output of all 0s and $m(\Theta) = 1$. DS fusion works in a way that this sensor automatically has no effect on the fusion results.

Functionality of the Programs

The abbreviations in parenthesis tell the purpose of the program. Implemented algorithms are marked by *algo*, the *helper* functions just contain unimportant service code. The *test* scripts try out algorithms, and *dia* scripts produce diagrams, for example with performance results. Features are extracted from data sets by *feat* programs. More detailed information on the functionality and call-parameters of each program is available in its code or via the Matlab `help` command (`help <program name>`).

`adjust_certainty_factors.m` (helper): Adjusts the certainty factors with the given factors, widens the gap at the given threshold.

`calculate_prototypes_with_k_means.m` (algo): Calls `k_means.m` to produce prototypes for each class.

`certainty_factors_histograms_crickets.m` (dia): Draws diagrams with histograms of the distribution of the certainty factors on the cricket data. Allows the adjustment of

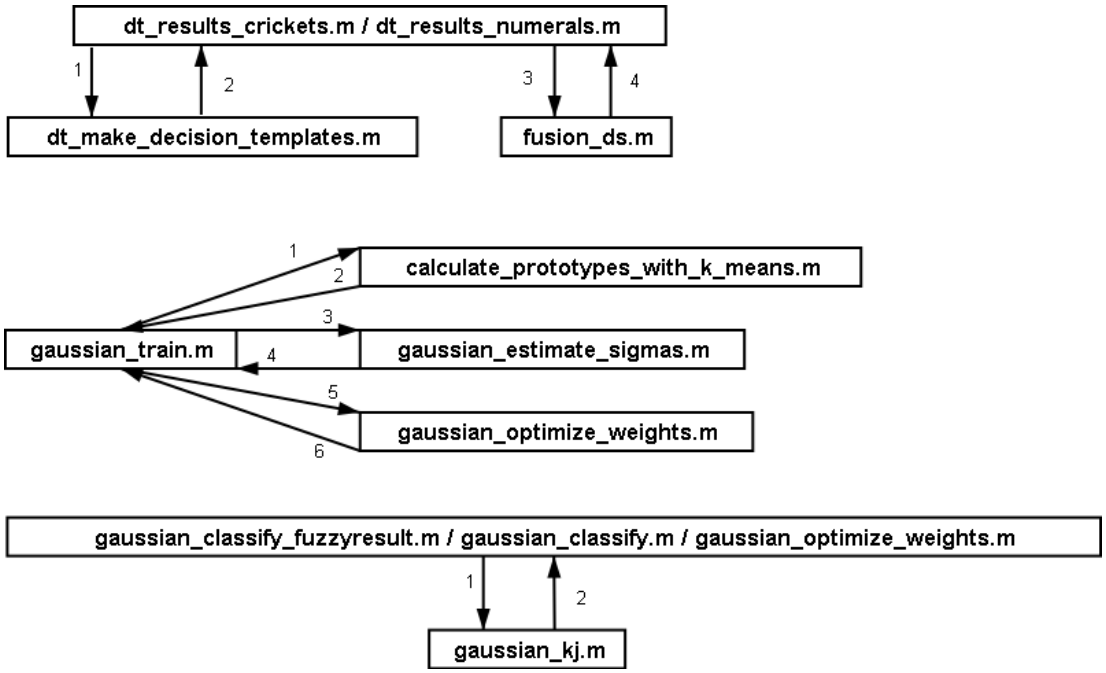

Figure B.1: Structure of calls and data passed of some major programs.

the certainty factors.

`certainty_factors_histograms_numerals.m` (dia): Draws diagrams with histograms of the distribution of the certainty factors on the numerals data. Allows the adjustment of the certainty factors.

`classification_performance.m` (helper): Calculates the classification performance out of the passed answers and labels.

`dt_fusion.m` (algo): Performs the *decision template* fusion (similarity measure: S_1), given the decision templates from the training and the answers of the classifiers to the new samples.

`dt_make_decision_templates.m` (algo): Calculates the decision template for each class, given classifier output for the training samples.

`dt_results_crickets.m` (dia): Calculates the results of the decision-template fusion scheme for the crickets data.

`dt_results_numerals.m` (dia): Calculates the results of the decision-template fusion

scheme for the numerals data.

`fusion_ds.m` (algo): Performs the *Dempster-Shafer* fusion in the following modes: standard orthogonal sum, not normalised orthogonal sum, including certainty factors, not normalised with certainty factors. Certainty factors can be adjusted.

`fusion_maximum.m` (algo): Performs the static *maximum* fusion.

`fusion_probabilistic_product.m` (algo): Performs the *probabilistic product* fusion.

`fuzzy_knn.m` (algo): Implements the *Fuzzy K-Nearest-Neighbour* algorithm.

`fuzzy_knn_test.m` (test): Performs a test of the fuzzy_knn classifier with the numerals data.

`gaussian_classify.m` (algo): Classify the sample point given with the parameters given using a Gaussian Radial Basis Function Network. Gives a hard answer.

`gaussian_classify_fuzzyresult.m` (algo): Classify the given sample point with the parameters given using a *Gaussian Radial Basis Function Network*. Gives a fuzzy answer.

`gaussian_estimate_sigmas.m` (algo): Estimate the σs for the RBF network.

`gaussian_kj.m` (algo): Implements the hidden sigmoid function $h_j(x)$ used in the RBF network.

`gaussian_optimize_weights.m` (algo): Optimises weights for the RBF network.

`gaussian_test.m` (test): Performs a test of `gaussian_classify.m` with the numerals data.

`gaussian_train.m` (algo): Trains the Gaussian RBF-network with the given samples. Produces prototypes, weights, and σs.

`get_entropy.m` (algo): Calculates the normalised, not inverted *entropy* values of the samples passed.

`get_gini.m` (algo): Calculates the normalised, not inverted *Gini* values of the samples passed.

`harden_maximum.m` (algo): Hardens the fuzzy decisions passed using the *maximum membership* rule.

`harden_maximum_second.m` (algo): Hardens the fuzzy decisions passed using the maximum membership rule, but taking the second-highest value.

`histogram.m` (feat): Extracts the histogram feature from the numerals data.

`k_means.m` (algo): Implements the *K-Means* algorithm, returns prototypes.

`load_cricket_data.m` (helper): Loads the cricket data into the variables used, for example, by `performance.m`.

`normaliz.m` (helper): Normalises the values of each row of the matrix passed.

`pca.m` (helper): Does a Karhunen-Loève transformation on the passed data.

`performance_ds.m` (dia): Produces diagrams or numeric results showing the performance of the Dempster-Shafer fusion (`fusion_ds.m`) in all modes. Can adjust certainty factors.

`performance_numerals_single_classifiers_simply_hardened.m` (test): Calls different classifiers with the numerals data, tells how they perform.

`Plotnumber.m / Makenumberpage.m` (dia): Visualise the handwritten numerals data.

`principal_components.m` (feat): Transforms the input training data and test data by projecting the samples onto the eight first principal components of the training set. Uses `pca.m`.

`rot_features.m` (feat): Extracts the RowColumnSums feature from numerals data passed.

`sort_matrix.m` (helper): Sorts the rows of the matrix passed according to the values in the last column (ascending).

Selbstständigkeitserklärung

Ich erkläre, dass ich die Diplomarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den 6.9.2004

(Christian Thiel)