Christian Thiel

Multiple Classifier Systems
Incorporating Uncertainty
The title page and cover show the painting *doctors* by Michael S. Parker (www.michaelsparker.com), 2003. Used with friendly permission of the artist.

Cover layout by Kreativfrosch Grafik + Design (www.kreativfrosch.eu), Wegscheid bei Passau, Germany.
Dissertation zur Erlangung des Doktorgrades
Doktor der Naturwissenschaften (Dr. rer. nat.)
der Fakultät für Ingenieurwissenschaften und Informatik
der Universität Ulm

vorgelegt von
Christian André Thiel

Angefertigt am Institut für Neuroinformatik
Einreichung der Dissertation: Dezember 2009
Tag der Promotion: 10. Juni 2010

Gutachter: Prof. Dr. Günther Palm
Gutachter: Prof. Dr. Heiko Neumann
Gutachter: Prof. Dr. Barbara Hammer

Amtierender Dekan der Fakultät:
Prof. Dr.-Ing. Michael Weber
The inclusion of uncertain class information into multi classifier systems (MCS) is the central theme in this thesis. A multi classifier system relies on multiple base classifiers, each of which is trained on a separate view of the problem at hand. Combining their answers will lead to a more accurate final decision. An example would be emotion recognition, with decisions based on observations of the mouth, the eyes or the pitch of the voice.

Traditionally in classification one sample is associated with exactly one class, for example anger. But, in practical applications, such a hard distinction is not warranted; instead a sample should have soft class memberships, thus being associated fuzzily with multiple classes at the same time. The inclusion of this uncertain information into various, but isolated building blocks of a MCS has been tackled by a great many researchers. This thesis places these approaches in the greater MCS context and assesses their utility. Remaining problems are identified and in many cases a solution is proposed.

Bayesian probability is the most obvious tool for modelling class uncertainty, but perhaps the Dempster-Shafer theory of evidence, fuzzy logic or fuzzy sets, or even a distribution of opinions are much more viable in a classification context. These formal uncertainty theories, as well as some others, are assessed regarding their aptitude to support the core flavours of uncertainty in MCS, as identified in this work: vagueness, imprecision, and certainty. For the very fitting Dempster-Shafer theory, practical applications are reported.

Some base classifiers have been extended to be trained on and answer with uncertain labels: learning vector quantisation, self-organizing maps, and most notably support vector machines (SVMs). The latter are an already very powerful breed of classifiers, and based on the idea of duplication, the underlying optimisation problem could be altered to accept fuzzy labels. Obtaining soft outputs from the binary SVMs is not trivial, but complete solutions are provided for the One-vs-Rest and One-vs-One multiclass decomposition architectures. Experiments do confirm the effectiveness of the fuzzy trained machines over their hard trained
counterparts, and yield answers as to which variants and parameter settings are to be preferred. Ideas on including certainty weighted information into many classifiers, and a study on the (surprisingly low) impact of label noise complete the classifier chapter.

Once all classifiers have provided their answers, the following fusion stage requires an assessment of how certain these opinions are. This can be accomplished by looking at the structure of the answer, for example using the Gini function or an index of fuzziness. A rating of certainty can also be provided by the classifiers themselves. Both approaches are closely related to measuring the fuzzy accuracy of an algorithm, with Binaghi’s fuzzy error matrix being one option. Rejection experiments provide further insight in this regard.

Existing fuzzy classifier fusion schemes are presented and categorised, among them those derived from uncertainty theories. For four fusion functions (decision templates, linear associative memory, naive Bayes and pseudoinverse solution) it can be shown that they all share the confusion matrix as the core element. Experiments again confirm the superiority of MCS architectures over single classifiers, and give pointers as to which fusion scheme to use in which situations.

Numerous experiments were conducted to assess each mentioned aspect. Of the four real-world applications that served as test cases, two are especially noteworthy. The setup for the recognition of emotions in video sequences of facial expressions includes 14 base classifiers, operating on four different areas of the face and utilising PCA, edge orientation, and motion based features. The accuracy of the system is slightly higher than for the average human, and apparently competitive with approaches described in the literature. Applied to land cover mapping, the fuzzy semi-supervised SOMs provide much smoother soft answers than hitherto proposed techniques, and the award-winning\(^1\) fuzzy SVMs produce a rendering of the satellite images that exhibits very compact regions, perfect for easy expert interpretation, and the highest accuracy of all tested classifiers.

---

\(^1\)My initial publication on fuzzy-input fuzzy-output support vector machines [260] won the “Best Paper” award at the KES 2007 conference, out of 411 contributions. In collaboration with Ferdinando Giacco, applying the SVMs in a land cover mapping scenario placed us among the winners [156] of the 2008 DFTC Fusion Contest of the IEEE Geoscience and Remote Sensing Society.
Zusammenfassung

Die Berücksichtigung von Unsicherheit in Mehrklassifikatorsystemen (MKS) steht im Zentrum dieser Arbeit. Anwendungen finden sich beispielsweise in der Emotionserkennung, wenn die Stimmung einer Person automatisch in eine bestimmte Klasse wie glücklich, wütend oder überrascht eingeordnet werden soll. Mehrere Klassifikator-Programme spezialisieren sich dann jeweils auf ein Merkmal der Person, etwa die Augen, den Mund oder die Stimmhöhe. Die Entscheidungen der Programme sind mit Unsicherheit behaftet, zum Beispiel weil die Umgebung sehr laut war. Auch soll sich ein Klassifikator nicht auf eine Emotion festlegen; eine Person kann ja wütend und überrascht zugleich sein. Also wird seine Antwort in einer sogenannten vagen oder weichen Zuordnung bestehen. Um stabilere und genauere Entscheidungen zu bekommen, fasst das MKS die Einzelmeinungen der Programme per Fusion zusammen.


Mit Support-Vektor-Maschinen (SVM), lernender Vektor-Quantisierung und selbstorganisierenden Karten werden einige Basis-Klassifikatoren so erweitert, dass sie


Allgemein wurden zahlreiche Experimente durchgeführt, um die genannten Problemstellungen besser verstehen und beurteilen zu können. Von den vier dazu verwendeten Anwendungsszenarien verdienen zwei besondere Erwähnung:

Contents

Abstract i
Zusammenfassung iii

1 Introduction 1

2 Applications 7
  2.1 Emotion Recognition from Facial Expressions 8
    2.1.1 Related Work 9
    2.1.2 Multi-Modal Emotion Recognition 10
    2.1.3 Databases and Labelling 12
    2.1.4 Features 16
    2.1.5 Classification Architecture, Setup 19
    2.1.6 Using Motion Energy Estimates 21
    2.1.7 Experimental Results 22
  2.2 Emotion Recognition from Speech 25
  2.3 Fruit Recognition 30
  2.4 Land Cover Mapping 32

3 Uncertainty Theories 41
  3.1 Probability theory 43
  3.2 Logic 45
  3.3 Distribution of Opinions 46
  3.4 Possibility Theory and Fuzzy Sets 49
  3.5 Rough Sets 53
  3.6 Fuzzy Logic and Fuzzy Control 54
  3.7 Dempster-Shafer Theory of Evidence 60
    3.7.1 Example 61
    3.7.2 Practical Applications 62
    3.7.3 Transferable Belief Model 63
  3.8 Uncertainty in Databases 63
  3.9 Uncertainty in Artificial Intelligence 65
  3.10 Comparison of Uncertainty Theories 65
    3.10.1 Context Model 69
3.10.2 Umkehrer Framework

4 Classifiers

4.1 Radial Basis Function Networks

4.2 $F^2$-K-Nearest Neighbour

4.3 $F^2$ Learning Vector Quantisation
  4.3.1 Basic Principles
  4.3.2 Adapting the Prototypes
  4.3.3 Adapting the Labels
  4.3.4 Experimental Setup
  4.3.5 Results and Observations

4.4 Self-Organizing Maps

4.5 $F^2$ Support Vector Machines
  4.5.1 Basic Principles
  4.5.2 Fuzzy-Input SVMs
  4.5.3 Fuzzy-Output SVMs
  4.5.4 Experimental Results
  4.5.5 One-Class SVMs

4.6 Certainty Weighted Labels

4.7 Label Noise
  4.7.1 Modelling label noise
  4.7.2 Experimental Results

4.8 Measuring Classifier Performance

5 Certainty Measures for Classification

5.1 Calculated from Fuzzy Answers

5.2 Fuzzy Accuracy

5.3 As Additional Classifier Answer

6 Multiple Classifier Fusion

6.1 Fusion Schemes

6.2 Experiments

6.3 Future Approaches

7 Summary

7.1 Theoretical Contributions

7.2 Experimental Results

Bibliography

Acknowledgments

Index
List of Figures

2.1 Image sequence from the JACFEE-Morph emotional faces database. 12
2.2 Image sequence from the Cohn-Kanade emotional faces database. 13
2.3 Face area labelling tool. 14
2.4 Emotion labelling tool. 15
2.5 Sobel edge detection filter applied. 17
2.6 Orientation histograms for emotional faces. 18
2.7 Optical flow in V1 and MT. 19
2.8 Motion energy in samples from the Cohn-Kanade database. 22
2.9 Distribution of fuzzy labels (max.) in the speech labelling experiment. 27
2.10 Samples from the fruit dataset. 30
2.11 ASTER satellite image of the area near Salerno. 33
2.12 DFTC contest: False colour composite and mapping. 38

3.1 Fuzzy sets for the fuzzy control application (input variables). 57
3.2 Fuzzy sets for the fuzzy control application (output variable). 57
3.3 Evaluation of the resulting fuzzy control application. 58
3.4 Generalised bell-shape membership function. 59

4.1 RBF network. 74
4.2 LVQ: Behaviour with added label noise. 83
4.3 LVQ: Prototype location changes. 84
4.4 SOM: Land cover mapping. 86
4.5 SOM: Rejection rate versus accuracy. 87
4.6 SOM: Class-wise rejection curves. 88
4.7 Classifier comparison: Rejection rate. 103
4.8 SVM architecture comparison: Land cover mapping. 104
4.9 One-Class SVM. 109
4.10 Label noise: Accuracy development. 114
4.11 Label noise: Soft-trained versus hard-trained. 115
4.12 Label noise: Agreement with original label. 116

5.1 Distribution of the entropy certainty factor. 121
5.2 SVM multiclass variants: Class-wise rejection curves. 128
6.1 Classifier fusion: A hierarchy of schemes  134
6.2 Classifier fusion: Average  135
6.3 Classifier fusion: Decision templates  137
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Facial emotions: Confusion matrix for human labelling</td>
<td>16</td>
</tr>
<tr>
<td>2.2</td>
<td>Facial emotions: Confusion matrix for classifier architecture (%)</td>
<td>16</td>
</tr>
<tr>
<td>2.3</td>
<td>Facial emotions: Optimal classifiers</td>
<td>20</td>
</tr>
<tr>
<td>2.4</td>
<td>Facial emotions: Confusion matrix for classifier architecture (count)</td>
<td>23</td>
</tr>
<tr>
<td>2.5</td>
<td>Speech emotion labelling experiment: Confusion matrix of intended emotions against human majority vote</td>
<td>27</td>
</tr>
<tr>
<td>2.6</td>
<td>Speech emotion labelling experiment: Accuracy of the average human</td>
<td>28</td>
</tr>
<tr>
<td>2.7</td>
<td>Speech emotion recognition: Confusion matrix of the experiment</td>
<td>28</td>
</tr>
<tr>
<td>2.8</td>
<td>Land cover mapping: Distribution of classes (Salerno dataset)</td>
<td>35</td>
</tr>
<tr>
<td>2.9</td>
<td>DFTC contest: Confusion matrix</td>
<td>39</td>
</tr>
<tr>
<td>3.1</td>
<td>Dempster-Shafer: Example basic probability assignments</td>
<td>62</td>
</tr>
<tr>
<td>3.2</td>
<td>Comparison: Imprecision in uncertainty theories</td>
<td>66</td>
</tr>
<tr>
<td>3.3</td>
<td>Comparison: Vagueness in uncertainty theories</td>
<td>67</td>
</tr>
<tr>
<td>3.4</td>
<td>Comparison: Certainty in uncertainty theories</td>
<td>68</td>
</tr>
<tr>
<td>4.1</td>
<td>Facial emotion recognition: Classifier comparison</td>
<td>76</td>
</tr>
<tr>
<td>4.2</td>
<td>Land cover mapping: Classifier comparison</td>
<td>101</td>
</tr>
<tr>
<td>4.3</td>
<td>Speech emotion recognition: Classifier comparison (hard/soft SVM)</td>
<td>102</td>
</tr>
<tr>
<td>4.4</td>
<td>Bradley-Terry coupling: Influence of the Fermi parameter</td>
<td>105</td>
</tr>
<tr>
<td>4.5</td>
<td>Reduced training set size: Optimal C-parameter for SVMs</td>
<td>106</td>
</tr>
<tr>
<td>4.6</td>
<td>Reduced training set size: Soft-trained versus hard-trained SVM</td>
<td>107</td>
</tr>
<tr>
<td>4.7</td>
<td>Reduced training set size: SVM accuracy</td>
<td>108</td>
</tr>
<tr>
<td>5.1</td>
<td>Rejection rules comparison: Accuracy results (land cover mapping)</td>
<td>129</td>
</tr>
<tr>
<td>5.2</td>
<td>Rejection rules comparison: Confusion matrices on rejected</td>
<td>130</td>
</tr>
<tr>
<td>5.3</td>
<td>Rejection rules comparison: Confusion matrices on not rejected</td>
<td>130</td>
</tr>
<tr>
<td>6.1</td>
<td>Form and usage of the memory matrix in fusion functions</td>
<td>138</td>
</tr>
<tr>
<td>6.2</td>
<td>Fusion performance with added label noise: Soft-trained</td>
<td>143</td>
</tr>
<tr>
<td>6.3</td>
<td>Fusion performance with added label noise: Hard-trained</td>
<td>143</td>
</tr>
<tr>
<td>6.4</td>
<td>Fusion performance: Cohn-Kanade dataset</td>
<td>145</td>
</tr>
</tbody>
</table>
You are sitting in a bar, waiting for a friend to turn up. A couple comes in and is seated at the table beside you. They start a livid conversation in a foreign language. You are bored, and begin listening in. The conversation gets faster, apparently leading into a heated argument, and their voices grow louder. They don’t seem to agree, but on the other hand neither of them sounds angry. You can see his face, at first indifferent or somewhat serious, then beginning to show a big grin, while his eyes are shining. Obviously, the two of them are enjoying a mock discussion.

For a human, this assessment is quite simple. From a technical, machine point of view, the scenario could be mastered by a multi classifier system incorporating uncertainty. Each piece of software functioning as a classifier would look at a different feature of the situation, say the pitch of the voice, and try to decide which emotion it is that the couple has. The conclusions of the classifier would surely not pinpoint exactly one emotion, but give the most probable alternatives (disagreement and not angry in the example above). Each of the multiple classifiers looking at different features will come up with such uncertain answers. To get a final assessment of the situation, the opinions of all the multiple classifiers will have to be combined using a fusion function. The output of the system is the most probable emotion, or a set of these, perhaps accompanied by an indication from the system how certain it is about its verdict.

In the following, the reasons why the inclusion and treatment of uncertain information is necessary at every stage of such a multiple classifier system will be discussed. The final part of this introduction will outline what issues are treated in which part of this thesis.

Emotions are a concept where it is intuitively clear that hard categories such as “he is angry” do not apply. He might be angry, but scared at the same time, and also a bit excited. Hence the need for a vague or fuzzy labelling of the situation, which could provide probabilities for each hard emotion category. With satellite images of the earth, one single atomic pixel will often refer to an area on the
Chapter 1. Introduction

ground that contains both a street and fields. A single category class would surely be inappropriate for this pixel. Or to give another example, if a patient is examined by a doctor, the medical expert will not come up with the one illness the patient could have, but (at least in his mind) a ranked list of probable causes that might even all be present to certain degrees.

A classifier learns from a training set, where the algorithm is presented with sample data points, and for each sample the correct class is provided in the associated label. But, as seen above, there are applications where these training labels are not hard categories, but consist of uncertain, vague class attributions. The labels might even be polluted by noise, for example because the expert labeller was distracted. Traditionally, however, classifier algorithms were only built to deal with certain, unambiguous training data. While this limitation can be healed quite easily for simple classifiers like the KNN, which looks at the nearest training samples, adapting involved architectures like SVMs requires deeper rooted changes.

Many established classifiers are able to answer with, or can be amended to give, fuzzy answers, yielding an estimate to which classes a new sample might belong to which degree. However, there is no indication how much this fuzzy answer can be trusted. One way to overcome this is to look at the structure of the answer, for example assessing its hilliness using the entropy measure. Another possibility is to have the classifier itself give a confidence value for its decision. This would be the preferred method, since only the classifier can know if it perhaps has too little training data or a poor model for this special kind of sample, or if the current sample has poor feature quality, for example due to bad camera lighting conditions.

Now that the classifier answers are fuzzy, accompanied by a self-certainty assessment, or perhaps coded according to a special uncertainty theory like Dempster-Shafer, another problem arises: how can the accuracy of a classifier be rated? It is not sufficient anymore to look at the percentage of samples for which it chose the correct hard class. The answer to this often lies in theoretical frameworks that deal with uncertainty. These uncertainty theories, like probability theory or fuzzy logic, then also offer various ways to formally represent uncertain class-assignments, and to combine them.

This is crucial for the step of classifier fusion. The answers of multiple classifiers concerning a new sample are combined to yield a final answer. Such an architecture is, for example, necessary if the application involves multiple modalities, such as video and voice recordings. On the same modality, there might also be differing methods to extract features, for example looking for edges and also for colour distributions in images. To achieve the highest fuzzy accuracy possible, a system would of course use as much information as can be made available to it. This mandates, inter alia, the inclusion of the full uncertain classifier answers
into the fusion process. Also, each classifier makes errors, but different ones than all the others, hence in combination the errors will tend to cancel each other out, reducing variance and increasing accuracy. An example is that happiness is best detected from video, while anger is better detected in speech.

As for the fusion functions themselves, they generally work by collecting statistics about the output-behaviour of the classifiers in the training phase. Due to the uncertain answers, this can no longer be done in a class-specific manner. But, most standard fusion algorithms can be adapted accordingly; the pseudoinverse fusion is especially versatile in this respect. Fusion operators derived from formal uncertainty theories, for instance the Dempster-Shafer orthogonal sum, play an important role, combining evidence from various classifier sources according to the principles laid down in each theory.

The final result, once all the classifier answers are combined, is a single, uncertain answer. Due to its uncertainty, it is much more valuable than a simple hard one-class decision. For example, samples on which the verdict is very fuzzy can be flagged and rejected by the system, possibly prompting a review by a specialist. The result would be a very high accuracy on the remaining samples. Another exciting use of certainty-rated answers is in the field of semi-supervised learning with co-training. In an application with plenty of samples, but where only a few of them are labelled, the base classifiers would first be constructed based on the limited training samples. Then, a portion of the unlabelled samples will be fed to the system. If the combined answer on a sample is then certain, it will be added to the training data to improve the accuracy of the base classifiers. Since the uncertainty associated with each final answer can be measured well in the proposed uncertain multi classifier system, it is now easy to select the most certain responses.

All of the uses of uncertainty in classifier fusion architectures introduced above are treated in detail in this work. Each chapter will give a motivation for its topic, and clarify the position of its contributions in the general field of research, citing plenty of references. The general thrust of each chapter will be showcased in the following overview.

Outline

The applications which serve as testbeds for experimentally evaluating the proposed algorithms and architectures are presented in Chapter 2. The most involved setup here, which also receives the most extensive description, concerns the recognition of emotions in videos of facial expressions. After reviewing the related work, also in the field of multi-modal emotion recognition, the databases are presented, with a special focus on the way in which the fuzzy labels were obtained.
from test persons. The feature extraction is explained, including a little detour on how to use the motion energy detected to boost classification accuracy on time series data. A description of the setup ensures that the experimental results can be compared with state-of-the-art systems documented in the literature.

Emotion recognition in speech as an application is then introduced along with the databases used, and the performance of our system is compared to that of the average human. The task of fruit recognition, for example telling red apples from orange oranges, is not a prime example for uncertainty, but serves well in basic experiments later. On the other hand, land cover mapping, that is the procedure of identifying structures in satellite images of landscapes, inherently calls for fuzzy labels, for pixels on the ground can correspond to a mix of classes. Here, not only are the features and data sets employed described, but we also reveal the exact setup that placed us among the winners of the 2008 IEEE GRSS data fusion contest.

The general concept of uncertainty is discussed at the beginning of Chapter 3, and the nomenclature for those kinds of uncertainty that might occur in classification is fixed. The rest of the chapter looks at various formal uncertainty theories, evaluating and eventually tabularly summarising to what extent each of them is suited to deal with classification-related uncertainty. The theories examined are probability theory, logic, distribution of opinions, possibility theory, rough sets, fuzzy logic and control, as well as the Dempster-Shafer theory of evidence. Especially for the latter, practical applications are presented. A short foray into the treatment of uncertainty in the fields of artificial intelligence and database systems is followed by a brief outline of two meta-theories for uncertainty treatment, the context model and the Umkehrer framework.

A major part of this thesis (Chapter 4) concerns itself with classifiers that are uncertainty-aware. Some algorithms, like k-nearest neighbour and radial basis function networks, already are or only require a few changes. A method to optimise variable kernels in RBF networks was re-discovered in the process of research. Including soft labels into learning vector quantisation not only imposes a fuzzy label-distance measure, but also a soft-labelling of the prototypes. The classic semi-supervised learning algorithm of self-organizing maps is extended to award soft labels, then evaluated on the satellite data set, while paying special attention to the form of the resulting fuzzy labels.

Support vector machines (Chapter 4.5) are powerful large-margin classifiers; their basic concept is anchored in hard binary labels. Extending them to accept and produce meaningful fuzzy labels is a major contribution of this work. After an introduction to the basic SVM concepts, the suggested algorithms are detailed. The fuzzy-input approach is based on the duplication of each fuzzy labelled sample, which requires some changes to the underlying optimisation problem. A fuzzy-output behaviour is achieved, for example, by using a coupling of the pair-
wise classifiers, based on sigmoid-transformed distances. Among other things, the extensive experiments illustrate a superiority of SVMs when compared to the accuracy of other classifier algorithms, and a definitive advantage of soft-trained SVMs over hard-trained ones. As an idea for future research, a fuzzy extension to one-class SVMs is outlined.

The rest of the classifier chapter deals with more general issues: If the labels carry certainty weights, is there a standard way to include this information into the classification process? What statistical methods should be used to judge the performance of uncertain classifiers? What happens once noise is added to the training labels? The last issue is explored more closely by laying out an algorithmic model for imparting noise and experimentally comparing the performance of soft-trained versus hard-trained classification architectures under the influence of label noise.

How certainty estimates for fuzzy classifier answers can be obtained is discussed in Chapter 5. To directly assess the structure of the answers, measures like entropy and the index of fuzziness are presented. Then, more involved approaches that allow each classifier to judge the certainty of its proper answers are reported. Also covered are ways to rate the accuracy of fuzzy classifiers, from involved metrics based on formal uncertainty theories to our proposal of the $S_1$-vector for simple soft answers.

Chapter 6 revolves around the classifier fusion step. Established fusion schemes are categorised and compared regarding their core elements, then methods to include uncertainty are surveyed; the focus is again on the Dempster-Shafer theory. Experiments demonstrate the usefulness of multi classifier approaches, and strive to establish a ranking between the fusion schemes. A section indicating promising venues for future research concludes.

A summary of the most important contributions and findings of this thesis can be consulted in Chapter 7, which is completed by references to my corresponding publications.
2 Applications

The intention of this chapter is to present the applications based on which we conducted our experiments. Only general experimental results are shown here, the ones pertaining to the fields of research we put a focus on can be found in the respective chapters.

Emotion recognition from facial expressions (Chapter 2.1) and speech (Chapter 2.2) is a perfect example for the presence of uncertainty in the labels, with multiple emotions being present simultaneously to different degrees. The fruit object recognition setup (Chapter 2.3) has crisp labels, but will allow to study noise on the training data. Land cover mapping based on satellite data (Chapter 2.4) again helps to evaluate soft classification performance, since due to a finite resolution one so-called mixed pixel may indeed cover multiple classes on the ground.
2.1 Emotion Recognition from Facial Expressions

Computers are generally rather indifferent towards the person in front of them. The software running on them does not care if the user is bored or angry, or stressed. This is because up to now, the software does not know about the emotional state of the human, which, to complicate matters, is also not clearly advertised. There are no easy signs to learn about the current emotion, the state must always be inferred from features of the human body and behaviour. Observing the movements of the body and extremities is one option, directly reading the brain another. But, most researchers focus on the more easily available features of speech (see Chapter 2.2) and facial expressions, the latter of which we will deal with in this section.

The recognition of emotions from facial expressions is useful in diverse fields of application [53], for example in teaching software which controls the engagement of the user, in ticket vending machines offering help when sensing unhappy customers, or in the gaming industry by automatically adapting the gameplay to suit the player’s mood. In a hospital scenario, a nurse could be automatically called to a patient showing signs of pain or discomfort.

The description of emotions has been the subject of discussion in many disciplines, like biology (Charles Darwin), philosophy (René Descartes) and psychology (William James). In technical applications, two representation schemes prevail: the activation-evaluation space and the six categories used by Ekman.

The activation-evaluation space describes emotions by their position in a two-dimensional space spanned by two axes: The vertical axis represents evaluation or valence of the emotion, telling if it is a positive or negative one. The horizontal axis indicates the activation level, that is how likely a subject with this emotion is to act on it. For example, according to studies by Whissel [280] and Plutchik [189], disgust would have activation 5 and evaluation 3.2 (angle: 161.3°), while happy would be at activation 5.3 and evaluation 5.3 (angle: 323.7°).

1 Emotion recognition from gestures [89] is feasible in conjunction with other features. In the wake of the declaration of the “war on terror”, security companies saw a market in systems that can detect suspicious behaviour for example at airports, but (until now [207]) such detecting activity is still fortunately effectuated by specially trained humans and not computer systems [208].

2 For example the Australia-based company Emotiv Systems (www.emotiv.com) will begin to sell (scheduled in 2009) a headset-helmet to consumers at around 300 USD that produces an EEG of the brain activity using 16 electrodes. Once taught, the EPOC system is purported to be able to discern 4 “thinking states” and 4 emotional states like frustration and excitement. In addition, facial expressions shall be detected by reading the brain’s signals to the face muscles. At the time of writing of this footnote (12/2008), the system shows still a reluctance to fully cooperate even with company staff.
Ekman and Friesen introduced their facial action coding system (FACS) [70] to describe all visually discernable facial movements via the introduction of so-called action units (AUs). For example, AU number 5 signifies a raising of the lid. Combinations of action units then correspond to one of six archetypal emotions: anger, fear, happiness, disgust, sadness and surprise. It is these emotions (or a subset thereof, possibly with the addition of a neutral class) that are employed in the majority of face-emotion related research. We will use these six categories in the following application, aiming to classify emotions in facial expressions.

The two emotion representation schemes presented are not uncontested, see for example [220] advocating a componential model. Especially the insistence on distinct basic emotions is criticised (see [55] for a summary of arguments), a point which is reinforced by our own findings that there is quite some disagreement between human emotion-judges (see Figure 2.9 and, for the faces, the average agreement rate of a human with the majority of 81.7%). Therefore, we will also adopt fuzzy emotion labels in our experiments.

A general overview of face identification and face emotion recognition can be found in [227], summing up and categorising research publications and citing key findings. As the title suggests, computational, physiological (cognitive science based) and psychophysical aspects are covered.

### 2.1.1 Related Work

There is a rich body available on the detection of emotions in human facial expressions. It can be divided into several categories according to the features used and the classification approaches and architectures.

While ordinary 2D camera data is still the most common input, there are alternate approaches like using a thermal image camera [293]. Two-dimensional data can also be used together with 3D models [174, 252]. In [174], a face model is pregenerated, and then fitted to the current sample. From the resulting model parameters, the nearest prototypical expression can be determined. In [252] a sample face presented is modeled using an active appearance model, which is a generative parametric model based on prototype vectors obtained by principal component analysis (PCA). The classification is accomplished using layered generalised linear discriminant analysis (LDA).

But, meaningful features can also be extracted directly from plain 2D images.

---

3 The translation of action units into emotions was possible using a dictionary the original authors produced, called EMFACS. Nowadays, it has been replaced by a database called FACSAID that is or will be available commercially from the website www.face-and-emotion.com.

4 For example, even concentrating on one emotion, Russell and Fehr ([211], p. 186) argue “that membership in the category of anger is a matter of degree and that borders separating anger from ‘not-anger’ are fuzzy.”
The high level family of them provides information on a more abstract level, for example the “degree of mouth openness” \cite{118} or the form of parts of the face, like the eyes \cite{299}.

Common low level features \cite{124, 158} are statistical ones like those based on PCA and LDA \cite{155, 291}. Gabor features are also frequently used \cite{158}. Very basic (3 \times 3 pixel) rectangular features masks are used in \cite{124}, together with boosting. This illustrates that with low level features, a cascade architecture is often suitable for classification. Flatter classifier techniques like k-nearest neighbour \cite{291} or support vector machines \cite{43} also yield good results.

Not all facial regions are equally important to recognise emotions. One option is to lay a net of perpendicular lines onto each sample image, using prominent facial landmarks like eyebrows, resulting in eight different regions \cite{131}. Similarly in \cite{155}, where weights were assigned to each of the disjoint regions according to their estimated contribution to the emotion. Six prominent regions were marked in \cite{299}, and based on movements inside, emotions were classified based on four simple rules.

Most of the approaches introduced use static images as source \cite{299, 124, 131, 155, 252, 174, 118, 291, 158}. In order to model the temporal succession of the input frames, which gives valuable information, hidden Markov models (HMM) \cite{278} or optical flow \cite{228} can be used.

Of course, the classification doesn’t have to be effectuated on a single feature, but the decisions based on multiple features can be combined, as for example done in \cite{158} using a simple neural network with one hidden layer. We will pursuit the classifier fusion approach in the following.

### 2.1.2 Multi-Modal Emotion Recognition

A short remark on multi-modal emotion recognition is in order. Of course it should be beneficial for the recognition rate to use all possible traits of the human body as input features. But unfortunately, “material which combines speech and video is still rare” \cite{53}, p. 70). Existing databases, like from the HUMAINE project \cite{47}, are generally not available to outsider researchers. A very promising approach is to tap a huge source of readily available material with acted emotional content: TV series and movies. After recording, the data only has to be labelled. This approach has for example been taken by \cite{50} \footnote{The authors worked on emotion recognition from speech, using very basic prosodic features.}, and in future will be made easier spurred by tools developed in the field of automatic content recognition and segmentation for television \cite{214}. A similar application area is the tracking of interactions in meeting situations, where recently advanced web-based tools to
annotate, organise and visualise the data were developed [44].

There are two noteworthy studies on multi-modal emotion recognition [43, 47]. In [43], the setup contains data from one actress, whose emotions are classified into the four categories of anger, sadness, happiness and neutral. The speech-related features are statistical ones, the face motion is captured using 102 markers on the actress’ face, the face feature calculation then based on the markers’ 3D positions (reduced via PCA). Recognition rates on the face feature alone are significantly higher than those on the speech features. The authors also experimentally compare the performance of a combination of the two modalities, on the feature level and on the decision level. The decision level fusion is effectuated with simple static functions (see Chapter 6.1) like voting or product, with product being chosen as the best. Both bimodal systems improve over the face-based classifier. However, the feature-level integration is especially good for the anger and neutral categories, while the decision-level system excels with the sadness and happiness emotions, bringing the authors to the conclusion that “the best approach to fuse the modalities will depend on the application” ([43], p. 210). The setup of [47] is more involved, with 10 subjects expressing 8 different emotions. The speech features used are rather broad, ranging from pitch to MFCC. Body and gesture movement is tracked based on the silhouette and hand blobs. The face related features use MPEG-4 facial action parameters. The experiments again show an accuracy improvement over single modality performances, but decision fusion (with 74.6% accuracy) is not as successful as feature level fusion (with 78.3%). Unfortunately, unlike in [43], the confusion matrices of the final responses are not given, so a more detailed comparison on the emotion-level is not possible. As a conclusion of the two studies shortly presented here, it is clear that the fusion of modalities yields improved recognition rates, but the optimal level for this fusion (features or decisions) depends on the application.

With no good multi-modal databases being publicly available, we see the biggest merit in limiting our research to decision fusion inside a single modality, based on different independent features there. This especially as the topic of feature level fusion with related questions like automatic feature selection is entirely outside the scope of this work. Furthermore, the accuracy we can obtain working on a single modality (77%) is very close to the one obtained in [47] using three modalities (78%).

\[\text{But again, “Because of the large amount of personal information included in this highly natural conversational-speech data, it is not possible to make the entire corpus publicly available” ([44], p. 3). Which is deplorable, because of the several thousand hours of recording, some include emotional behaviour, which is for example exploited for laughter recognition [221].}\]

\[\text{The 77%-figure for our results is obtained for our best architecture, if we assume that all emotions have the same number of test samples, as is the case in [47]. Their experiments there are based on 8 emotions, ours only on six. On the other hand, we have close to a hundred different subjects, while they only have 10. For accuracy comparisons all based on the Cohn-Kanade database, see Chapter 2.1.7.}\]
2.1.3 Databases and Labelling

For some experiments, we used the JACFEE-Morph database, which is based on the well-know JACFEE (Japanese and Caucasian facial expressions of emotion [168]) collection. JACFEE includes frontal images of 8 different persons, each enacting a neutral expression and the six emotions of anger, disgust, fear, happiness, sadness and surprise. In order to obtain image sequences for emotion detection, Hoffmann, Traue, Bachmayr, and Kessler [105] used their facial expression morphing tool which implements morphing techniques to obtain a smooth transition from the neutral face to the one with the full-blown emotion.

The resulting JACFEE-Morph database contains 11 images for each emotion sequence, with the images having a resolution of 500x400 pixels (see Figure 2.1). When presented, the emotion in the video sequence has an unnatural feeling. This is because for each emotion here, it takes the same time to go from neutral to full-blown state, which is not the case for natural emotions [105].

The database most heavily used [228, 288, 51, 180, 283, 291] in facial emotion recognition is the so-called Cohn-Kanade set made public by Kanade, Cohn, and Tian in 2000 [127]. The authors instructed their subjects to perform 23 different facial displays, each of which began and ended with a neutral face (see Figure 2.2). The recordings were frontal, with the 30 degree side-angled version that was also shot apparently not having been published. The sequences recorded with a Panasonic WV3230 camera were digitised to have a resolution of 640x490 (sometimes 480) pixels. Unfortunately, the pictures captured were originally shown in

---

8To obtain a copy of the database, contact the Affect Analysis Group at the University of Pittsburgh, USA. The database is copyright by Jeffrey Cohn.
2.1. Emotion Recognition from Facial Expressions

Figure 2.2: One of the 488 sequences from the Cohn-Kanade database. The length of each sequence is different. In the database the authors distribute, the sequences start from a neutral expression and end with a display of the full-blown emotion.

interlaced mode, that is with twice the frame rate, but each time showing only the odd or even numbered lines, leading to artifacts in the digitised material. For example, the mouth had big location jumps from frame to frame. To counter this, we used a field extension deinterlacing technique, interpolating the missing from the presented lines. Since our goal is to detect emotions based on different areas of the face, these needed to be identified. Using our segmentation tool (see Figure 2.3), in each sequence the face area, left and right eye, and the mouth were identified. This segmentation could also have been accomplished completely automatically, using for example the face recognition tool in the OpenCV library.

The dataset we received contained a total of 488 sequences from 97 individuals. Some of the sequences simply show the performance of single action units. We decided to omit these and a few more complex displays which do not correspond to any discernable emotion: of the 53 sequences left out (our retained total: 432), the majority represents action unit 25, “Lips Part”. This brings us to an important point concerning the comparability of studies based on the Cohn-Kanade database: the number of sequences used varies largely. Quite some studies do start from 488 (for example [283, 292]) and eliminate some, while some publications base their finding on data sets with lower series numbers. This leads to a

9In the context of this application, we use left and right in mirror speak, that is when we talk about the left eye, what we mean is the person’s right eye.

10The Open Computer Vision library was originally developed by Intel, and is now available as open source project at http://sourceforge.net/projects/opencvlibrary. An introduction can be found in [34].

11For example, [288] reports a total size for the database of 96 individuals and 391 sequences, and reduces their experimental setup to 365 sequences by eliminating those that are shorter than 11 frames. In [51], the authors had 104 individuals, and used only the sequences of those 53 individuals for which all 6 emotions were displayed. The number of individuals in [180] was
situation where the results of experiments on the Cohn-Kanade database can not be easily compared, aggravated by the fact that some authors reduce the sample size considerably, without giving clear details on the reduction criteria. Another uncertainty in the interpretation of accuracy results based on the Cohn-Kanade database is the mapping of sequences to emotion classes. The authors provide for each sequence an annotation in terms of action units, for example “6+12+25”, and a translation table, which for each emotion has a number of action unit sequences that will be produced by such an emotion. The example mentioned above would clearly map to happy. However, there are many cases, for example “1+2+5+16+20+25”, in which a mapping based on this table is not possible. That in essence means that for every study recognising the 6 emotions based on the Cohn-Kanade database, a large subjective factor in this translation will be present.

Being interested in fuzzy emotion labels for each sequence, we did not consult a single emotion specialist, but 15 amateurs. That is, we recruited 15 test persons, and presented all 488 emotional sequences to each of them (in random order). The test person could see the current video as often as desired, and then decided which of the six emotions it presented. The user interface for this experiment is shown in Figure 2.4. Thus, for each sequence, we have 15 votes, which can be interpreted as soft labels, and which were used as groundtruth for our experiments described later. When taking the majority vote for each sequence, we can built a confusion matrix between majority and individual decisions, as shown in Table reported at 90, with a total of 276 sequences.

**Figure 2.3:** Our tool to mark the following face areas in each sequence: face, left/right eye, mouth. Areas could be adapted inside each sequence.
2.1. Emotion Recognition from Facial Expressions

Figure 2.4: The tool used by the test subjects to label each of the emotional sequences. The “play” button will show the current sequence again, selecting one of the emotions (captions on the buttons in German) will go forward to the next sequence. The user could log in to the system, that is he did not have to (in fact, it was discouraged) go through all of the 488 videos in one session.

2.1. The accuracy of the average human in this setup can thus be computed as 81.9\% (for Cohn-Kanade; 85.5\% for the JACFee-Morph database)\textsuperscript{12}, meaning in 4 out of 5 cases a human will detect the same emotion as the majority of his peers.

It is very interesting to compare the human confusions (Table 2.1) with the ones a classifier makes: Table 2.2 shows the confusion matrix for the best classifier architecture (as identified later). Both matrices detail similar recognition rates and confusion patterns. For example, humans confuse fear with surprise and disgust, the classifier too, but with additional confusions with happiness.

\textsuperscript{12}To set this into relation with the emotions that were encoded by the original translation table (making a “best guess” through visual inspection if the table failed): in 22\% of the cases, the table-emotion and majority-voted emotion did not agree. Do bear in mind that this figure is still for the full set of 488 sequences, of which we later removed 53 sequences that did not show an apparent emotion.
Table 2.1: Confusion matrix of the majority vote decision (rows) against the individual decision (columns), given as mean of fractions. The total number of samples for each emotion, as determined by majority vote among the human labellers, is given in the last column. For example, 25 sequences have been labelled as containing fear by the majority vote, but the consensus was only 56%. Database: Cohn-Kanade

<table>
<thead>
<tr>
<th>maj. \ indiv.</th>
<th>hap.</th>
<th>ang.</th>
<th>sur.</th>
<th>disg.</th>
<th>sad.</th>
<th>fear</th>
<th>samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>happiness</td>
<td>0.99</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.01</td>
<td>105</td>
</tr>
<tr>
<td>anger</td>
<td>0</td>
<td>0.80</td>
<td>0</td>
<td>0.12</td>
<td>0.07</td>
<td>0.01</td>
<td>49</td>
</tr>
<tr>
<td>surprise</td>
<td>0.01</td>
<td>0</td>
<td>0.78</td>
<td>0</td>
<td>0.01</td>
<td>0.19</td>
<td>91</td>
</tr>
<tr>
<td>disgust</td>
<td>0.01</td>
<td>0.15</td>
<td>0.01</td>
<td>0.67</td>
<td>0.01</td>
<td>0.15</td>
<td>81</td>
</tr>
<tr>
<td>sadness</td>
<td>0.01</td>
<td>0.08</td>
<td>0.02</td>
<td>0.02</td>
<td>0.88</td>
<td>0.01</td>
<td>81</td>
</tr>
<tr>
<td>fear</td>
<td>0.01</td>
<td>0.01</td>
<td>0.14</td>
<td>0.27</td>
<td>0.01</td>
<td>0.56</td>
<td>25</td>
</tr>
</tbody>
</table>

Table 2.2: For comparison purposes, the confusion matrix of human majority vote (rows) against the final verdict of the best classifier architecture (Details see Table 2.4). Database: Cohn-Kanade

<table>
<thead>
<tr>
<th>maj. \ classif.</th>
<th>hap.</th>
<th>ang.</th>
<th>sur.</th>
<th>disg.</th>
<th>sad.</th>
<th>fear</th>
<th>samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>happiness</td>
<td>0.95</td>
<td>0.01</td>
<td>0</td>
<td>0.04</td>
<td>0</td>
<td>0</td>
<td>105</td>
</tr>
<tr>
<td>anger</td>
<td>0</td>
<td>0.80</td>
<td>0</td>
<td>0.10</td>
<td>0.10</td>
<td>0</td>
<td>49</td>
</tr>
<tr>
<td>surprise</td>
<td>0.01</td>
<td>0</td>
<td>0.97</td>
<td>0</td>
<td>0.01</td>
<td>0.01</td>
<td>91</td>
</tr>
<tr>
<td>disgust</td>
<td>0.17</td>
<td>0.12</td>
<td>0.01</td>
<td>0.63</td>
<td>0.01</td>
<td>0.05</td>
<td>81</td>
</tr>
<tr>
<td>sadness</td>
<td>0.01</td>
<td>0.05</td>
<td>0.04</td>
<td>0.01</td>
<td>0.89</td>
<td>0</td>
<td>81</td>
</tr>
<tr>
<td>fear</td>
<td>0.08</td>
<td>0</td>
<td>0.20</td>
<td>0.32</td>
<td>0.04</td>
<td>0.36</td>
<td>25</td>
</tr>
</tbody>
</table>

For more details on the labelling tools and preprocessing procedures, see [217].

2.1.4 Features

The classification was based on three sorts of features: PCA, orientation histograms, and optical flow.

In the primary component analysis (PCA) approach (see [25]), the data is transformed to a different basis, with the dimensions being orthogonal. The new basis is found by identifying the eigenvectors of the covariance matrix of the data, where the vectors associated with the largest eigenvalue are the directions with the highest variance in the data, the primary components. We transformed our face regions to standard dimensions (40x40 pixels for the face, 10x20 for the mouth, 20x20 for each eye) and projected this image data onto the first 150 primary components, resulting in 150-dimensional feature vectors for each area.

Another feature, proven to be successful in [73], are orientation histograms. The idea is to detect the edges in the pictures, then to produce statistics on the main
2.1. Emotion Recognition from Facial Expressions

Figure 2.5: The Sobel edge detection filter applied to an image from the Cohn-Kanade database.

directions. For edge detection, we used the Sobel operator \[91\] (see Figure 2.5), which first convolves the image using a 3x3 mask, separately for each dimension:

\[
S_x = \frac{1}{8} \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \quad S_y = \frac{1}{8} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}
\]

The direction of an edge at a specific location can then be calculated by applying the \[\text{arctan}\] function on the derivations just computed with the Sobel operators. We subdivided each face area (mouth, eyes...) into 4 subregions, and for each of these produced a histogram based on those 10% of the edges with the highest norm. There being 12 bins, we ended up with a 48-dimensional feature vector for each area. The within-class distances compared with the inter-class distances of this feature are promising (see Figure 2.6).

A very powerful feature is the optical flow \[115\]. The motivation for this is to analyse the motion field of a sequence, that is the projection of the motion onto a 2D image plane. The optical flow measures shifts in grey values, and can serve as an estimation for the motion field. To determine the optical flow, we employed a biologically inspired framework developed in the Vision and Perception Science Lab of the Institute of Neural Information Processing at the University of Ulm [17, 18]. It is based on maintaining plausible motion hypotheses, and implements a feedback between the modelled visual areas \(V1\) and \(MT\): an initial motion estimation is calculated in \(V1\), which is rather location specific, but has a more vague speed prediction. The area of \(MT\) effectuates a sampling of the \(V1\) values it received, generating motion hypotheses of higher quality on a coarser locational scale. The hypotheses selected in \(MT\) generate a feedback signal to \(V1\), in the next iteration strengthening those \(V1\) hypotheses that are compatible with the
Figure 2.6: Showing the edge orientation histograms for two examples of a surprised and one of a disgusted face. The last row shows more generally the mean differences between histograms of different classes, and those within the same class.

MT estimation. Of course, since we are dealing with moving objects, the location of each hypothesis has to be adjusted for the next timestep (predictive shift), using the velocity estimation available. The number of different hypotheses for a single location is capped. The (initial) motion estimation is based on a census transform of the input images, which yields a compact representation \( c \) of the local structure around each pixel [294]. For each pair of frames, corresponding census values are identified (within a certain pixel radius), and based on their location a motion hypothesis calculated. The census value \( c \) itself is calculated over the neighbourhood of the respective current pixel \( b \) (and its value \( b \)):

\[
\begin{array}{ccc}
  a_8 & a_9 & a_{10} \\
  a_0 & a_1 & a_2 \\
  a_{15} & b & a_3 \\
  a_6 & a_5 & a_4 \\
  a_{14} & a_{13} & a_{12}
\end{array}
\]

\[
c = \sum_{i=0}^{15} 3^i \cdot \begin{cases} 
  0 & \text{if } a_i < b \\
  1 & \text{if } a_i \approx b \\
  2 & \text{if } a_i > b
\end{cases}
\]

The basis for the flow feature we use are the motion estimations in MT, which are, as can be seen in Figure 2.7, better spatially integrated than those in V1.
2.1. Emotion Recognition from Facial Expressions

Figure 2.7: Visualisation of the optical flow for one pair of images, calculated as a mean over the location-specific hypotheses. (a) Optical flow in V1. (b) Optical flow in MT (sampled). (c) Colour code for the hypotheses directions. Light green for example means a motion towards the top of the image.

From the 30% of the hypotheses with the highest norm of each face area, again histograms are created, as described for the orientation histograms above.

2.1.5 Classification Architecture, Setup

The emotion recognition system is built in a multiple classifier architecture with temporal fusion. Each emotional video sequence contains a stream of pictures. From each of these images (or image pairs), several features are extracted, as described above. In fact, this extraction is done for four different regions of the face. That is, for each emotional sequence, we now have as many feature streams as there are different feature/face-region combinations. Each element of each stream is then classified, yielding streams of classification labels\(^\text{13}\). For each stream, these are combined into one fuzzy decision using simple averaging. These decisions are then combined into a final answer using advanced classifier fusion techniques (as described in detail in Chapter 6.1), a comparison with our baseline labels yields the classification accuracy rate. All results are based on 8-fold cross validation experiments.

As classifiers, we used k-nearest neighbour, support vector machines, and RBF

\(^{13}\text{In fact, for the classification phase, the stream is cut and only the answers from the second half are used. This was employed as a crude technique to overcome feature outliers at the beginning of the sequences. A more refined treatment of this issue is described below in Chapter 2.1.6, the techniques described there were so far applied to RBF networks only.}\)
networks (for details on all of these, see Chapter 4). For the KNN, the number of neighbours could be fixed to \( k = 5 \) for best performance. For the SVMs, we employed a RBF kernel for both the One-vs-One and One-vs-Rest architectures, and determined the optimal free parameter \( C \) out of a range of \([10^{-4}, 10^4]\), albeit operating only on one eighth of the available data to limit the search time required\(^\text{14}\). The RBF networks had 25 hidden neurons, which were placed randomly, each network was trained in 200 batch epochs. A further increase of the number of epochs or neurons did not lead to better results.

<table>
<thead>
<tr>
<th>feature</th>
<th>region</th>
<th>classifier</th>
<th>accuracy %</th>
</tr>
</thead>
<tbody>
<tr>
<td>edge orient. histogr.</td>
<td>mouth</td>
<td>SVM (One-vs-One)</td>
<td>74.0</td>
</tr>
<tr>
<td>edge orient. histogr.</td>
<td>mouth</td>
<td>RBF (energy weighting)</td>
<td>70.3</td>
</tr>
<tr>
<td>optical flow</td>
<td>face</td>
<td>RBF (energy selection)</td>
<td>67.1</td>
</tr>
<tr>
<td>optical flow</td>
<td>mouth</td>
<td>RBF (energy selection)</td>
<td>67.1</td>
</tr>
<tr>
<td>PCA</td>
<td>mouth</td>
<td>RBF</td>
<td>65.9</td>
</tr>
<tr>
<td>PCA</td>
<td>face</td>
<td>RBF</td>
<td>62.7</td>
</tr>
<tr>
<td>edge orient. histogr.</td>
<td>face</td>
<td>SVM (One-vs-Rest)</td>
<td>54.3</td>
</tr>
<tr>
<td>edge orient. histogr.</td>
<td>face</td>
<td>RBF (energy weighting)</td>
<td>52.3</td>
</tr>
<tr>
<td>optical flow</td>
<td>right eye</td>
<td>RBF (energy selection)</td>
<td>51.1</td>
</tr>
<tr>
<td>optical flow</td>
<td>left eye</td>
<td>RBF (energy selection)</td>
<td>45.8</td>
</tr>
<tr>
<td>edge orient. histogr.</td>
<td>right eye</td>
<td>RBF (energy weighting)</td>
<td>42.3</td>
</tr>
<tr>
<td>edge orient. histogr.</td>
<td>left eye</td>
<td>RBF (energy weighting)</td>
<td>41.4</td>
</tr>
<tr>
<td>PCA</td>
<td>left eye</td>
<td>RBF</td>
<td>37.9</td>
</tr>
<tr>
<td>PCA</td>
<td>right eye</td>
<td>RBF</td>
<td>35.1</td>
</tr>
</tbody>
</table>

\( \text{Table 2.3: The optimal classifiers for each feature-region combination, on the Cohn-Kanade dataset, with two supplemental SVMs. The recognition rate is given in percent.} \)

There are three different features and four face regions. For each of these 12 different combinations, the classifier with the optimal classification accuracy was determined\(^\text{15}\): on edge orientation histograms, RBF networks with weighting of the frames based on the motion energy (see Chapter 2.1.6 for the different RBF stream fusion variants); on optical flow based features, RBF networks with frame selection based on the motion energy; on PCA features, RBF networks. SVMs had very high recognition rates on the edge orientation histograms of the mouth and face area\(^\text{16}\), so we included included these classifiers, yielding a total number of 14 optimal classifiers (see overview in Table 2.3). This selection was all effectuated on the Cohn-Kanade database, for our JACFEE-Morph experiments, we employed

---

\(^\text{14}\)For a justification of this simplification, compare our experiments summed up in Table 4.5.  
\(^\text{15}\)The optimum sought included a subjective bias for selecting a uniform classifier type for each feature type across all face areas.  
\(^\text{16}\)Mouth: One-vs-One architecture, \( C = 1 \). Face: One-vs-Rest architecture, \( C = 1000 \). RBF kernels, as mentioned above.
2.1. Emotion Recognition from Facial Expressions

the same feature-facial area-classifier combinations, of course with the classifiers being trained on the JACFEE-Morph data.

The responses of the fourteen classifiers were then combined using the following fusion schemes (for details see Chapter 6): minimum, maximum and probabilistic fusion; linear transformation based on minimum mean square error; decision templates; Dempster-Shafer fusion; For each fusion scheme, we made an exhaustive cross-validation experiment to determine the best combination of the 14 features, searching through all $2^{14} = 16384$ possible groupings. The highest accuracy (83.1%) was achieved using probabilistic product fusion on the eight features of orientation histogram mouth (RBF and SVM), optical flow mouth, optical flow left eye/right eye, PCA mouth, PCA face, and orientation histogram face (SVM).

It is interesting to see that the best 200 combinations all yielded an accuracy of at least 80%. To reduce the computational burden of an exhaustive search for the best combination, we tried the approach of sequential forward selection (SFS). This algorithm starts with the best single classifier, then searches for the next classifier to add that will yield the biggest improvement, and so on. An experiment for the linear transformation/pseudoinverse fusion yielded an accuracy of 81.0%, just 0.4 percent points short of the best solution obtained via exhaustive search for the same cross-validation partitioning. Thus, for this application there seems to be no need for expensive feature selection algorithms.

2.1.6 Using Motion Energy Estimation to Improve the Classification Accuracy

In most of the sequences, the subject did not or barely move its face muscles for the first few frames. Our initial time fusion approach gave equal weight to each frame, resulting in bad recognition rates because the neutral start frames diluted the training data and answers. The heuristic approach of using only the second half of each sequence improved the recognition rates, but in many sequences we were as a result disregarding important information. A more nuanced technique could be based on the motion energy of sequence, under the assumption that only those images where motion occurs are relevant. Plotting the motion energy, that is the sum of motion vectors (see Chapter 2.1.4), for each segment, we obtained the curves that are exemplarily shown in Figure 2.8. To get rid of the high starting values in the first frames (that are due to the motion extraction algorithm needing some frames to take the proper feedback into account), all frames before the first motion energy minimum were discarded (as a constraint preserving at least half of

---

17 Using a pseudoinverse solution.
18 Here the DS variant with discounting based on the Gini-index, attributing belief to the universal hypothesis $m(\Theta)$ [261].
19 For an introduction to SFS and other feature selection algorithms like the promising floating search approach, see [275], Chapter 9.2.3, “Suboptimal search algorithms”. 

Figure 2.8: The motion energy (y-axis) for each frame pair (x-axis) of a sequence from the Cohn-Kanade database (for four different face regions). The peak at the start, which is nicely visible in these examples, has to be especially treated.

the sequence), what we call “frame selection based on motion energy” or “energy cutting”. As an alternative approach, in the temporal fusion a frame is weighted with the sum of motion energy of all its pixels, normalised by the energy sum over all pixels of all frames of the sequence, which amounts to a data-driven weighting of the frame-classification answers.

We did an experiment on the Kohn-Canade database using RBF classifiers (for details see Chapter 6.3 in [217]). Compared to the method that used only the second half of the frames for all features, depending on the feature, one of the two methods, energy cutting and frame weighting, could significantly improve the classification accuracy. Flow-based features are best fused over time using the energy cutting technique, so this technique seems to be able to eliminate the confusing starting frames. PCA and edge orientation based features all benefit from the frame weighting technique, the motion energy seems to be a suitable indicator here to emphasise the frames with emotion-typical facial expressions.

2.1.7 Experimental Results

For the optimal combination of features, classifiers and fusion scheme, we obtain a classification accuracy of 83% on the Cohn-Kanade database (see Table 2.4; We used 432 sequences for our experiments;). Comparing this to a human recognition rate of 81.9% (compare end of Chapter 2.1.3), this is quite respectable.

It is desirable to compare the classification performance of our systems to others (all operating on the Cohn-Kanade set), which we will attempt to do in the following. Two hindrances must be kept in mind here: as detailed in Footnote 11, the database size of the Cohn-Kanade dataset, and with it the problem’s difficulty.

---

20 Except for the optical flow and PCA based features of the left eye area, which are best classified using a combination of the two advanced techniques.

21 Found via exhaustive search, as mentioned in Chapter 2.1.5.
2.1. Emotion Recognition from Facial Expressions

<table>
<thead>
<tr>
<th></th>
<th>happiness</th>
<th>anger</th>
<th>surprise</th>
<th>disgust</th>
<th>sadness</th>
<th>fear</th>
</tr>
</thead>
<tbody>
<tr>
<td>happiness</td>
<td>100</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>anger</td>
<td>0</td>
<td>39</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>surprise</td>
<td>1</td>
<td>0</td>
<td>88</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>disgust</td>
<td>14</td>
<td>10</td>
<td>1</td>
<td>51</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>sadness</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>72</td>
<td>0</td>
</tr>
<tr>
<td>fear</td>
<td>2</td>
<td>0</td>
<td>5</td>
<td>8</td>
<td>1</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 2.4: Confusion matrix of the emotion-recognition experiment (8-fold cross validated) on the Cohn-Kanade database, using product classifier fusion. Baseline emotions given on the rows. Example: happiness is recognised with a rate of 100/105.

...
Chapter 2. Applications

within a control-theoretic framework modeling the facial muscles, is very promising. Later works of Pentland focused on detecting facial expressions in natural settings, but not in general the emotions associated.

Our experiments with the JACFEE-Morph database yielded a classification accuracy of 79.1%, using least-squares pseudoinverse fusion\(^{23}\). This is surprisingly high, given that we simply used the feature-classifier combinations deemed optimal in our Cohn-Kanade experiments. (Of course, the classifiers were trained on the new database.)

Measuring the classification power of an emotion classifier based on crisp answers against a crisp baseline is not undisputed. In fact, it would be desirable to base it on the fuzzy labels and answers. But, beyond the simple calculation of the Euclidean distance, no generally accepted procedures exist here. As Klaus Scherer puts it in [220] (p. 249), “there is no established method for analyzing mixed or blended emotions, both with respect to the production and to the inference from expressive material”. One interesting approach to solve this is detailed in the article Of all things the measure is man [250]. Based on the Shannon entropy, it compares the answers of the average human voter to the machine answers. In our own experiments [217] we found that the main factor that led to large differences between humans and machines in these comparisons was the type of classifier fusion function employed. See also Chapter 3.3 dealing with the representation of uncertainty or labels as distributions of (people’s) opinions.

In our experiments, we made some interesting observations concerning the interplay of face areas, features and emotions [217]. For example, happiness and sadness are best classified using orientation histograms on the mouth area. Anger is best detected using flow on the left eye\(^{24}\), for surprise the most important area is the mouth, with PCA as best single feature. Fear can not be detected well by any classifier. Disgust can be told by looking at the orientation histogram of the mouth, but as classifiers SVMs must be used\(^{25}\).

\(^{23}\)Best result achieved with a combination of two features. The next-best rate of 77% could be obtained with classifier combinations based on 3 to 7 features.

\(^{24}\)The recognition rate was 64% here. Surprisingly, the performance on the right eye was 4 percent points lower. With many experiments, we could observe that left- and right-eye-features yielded quite different answers.

\(^{25}\)With an accuracy of 62%, the SVMs outperformed the other classifiers by more than 20 percent points.
2.2 Emotion Recognition from Speech

A short introduction into the importance and foundations of emotion recognition has already been given at the start of Chapter 2.1. In this section the aspect of emotion recognition from speech will be treated. Voice (or speech) is just one of many traits, and can be embedded within a multi-modal recognition system (see Chapter 2.1.2). But, there are also scenarios where only the voice is available, the most famous example here being a call centre [186] where the caller is treated or routed differently depending on his automatically detected emotional state (angry customers might be immediately transferred to an experienced human agent). The technique has also interesting applications in the area of computer games, to make the game experience more emotionally responsive, to include affect, for example by changing the behaviour of the in-game character depending on the player’s mood [121]. The automotive industry wants on-board computers to be aware of the driver’s emotional state [120]. However, not all of the applications of emotion detection in voice are desireable, for example the Harrow Council in London uses a variant to monitor phone calls to their public benefit centre, to detect if the applicant is lying [19].

There has been extensive research into emotion recognition from speech (see [53, 220] for summaries and [14] for a key article). We will restrict our experiments and applications to use only paralinguistic information such as the pitch. Linguistic information is computationally hard to extract and even harder to process automatically, and using word lists representing categories like aroused is clearly not enough. Also, paralinguistic information is much more universal, with studies showing that humans are able to correctly pinpoint emotions in foreign language speech [185, 219].

Some databases exist that contain emotional speech, a few of them are labelled with the according emotions. However, the databases are generally labelled with distinct basic emotions, a minority is labelled within a dimensional model of affective dimensions like the activation-evaluation space [273]. But, human emotional states rarely correspond purely to one of the archetypical emotions, as exemplified by the low agreement between human labellers (see Table 2.6). Thus, we had to produce our own database with fuzzy labels, that associates each recording with different emotions to different degrees.

The speech corpus we used has been recorded within the project ‘Perception and Interaction in Multi-User Environments’ at the competence centre Perception and Interactive Technologies (PIT) [251]. It consists of 413 recorded sentences, each

---

26In this application however, linguistic information is also used to detect the intent or emotion of the caller. For example, from personal experience I know that when calling the helpdesk of the Deutsche Telekom, uttering the word Kündigung (German for cancellation of contract), regardless of where you are in the voice computer menus, will immediately get you transferred to a human operator.
spoken in one of the emotions of hot-anger, happiness, fear, sadness, disgust, boredom or neutral (see [222] for details). The basis are 15 sentences taken from everyday situations with semantically neutral context, for example *Und wohin wollen wir gehen?* or *Was du immer für Ideen hast.*<sup>27</sup> Then, 4 different people were asked to read each of the sentences in the different emotions listed above. To get into the right frame of mind for each emotion, little plays, which were introduced with a small screenplay, were performed before recording. The recording itself was effectuated in an office environment with background noise to ensure practical applicability, using an external USB sound card<sup>28</sup>.

To obtain the features for our classification algorithms, each sentence of the corpus was analysed based on a time window of a length of 30ms with an overlap of 20ms. The pitch was computed using the scale-invariant feature transform algorithm (SIFT) [200] which performed satisfactory. Based on results of [14, 53], we decided to include the following statistical measurements in our 37-dimensional feature vector: The minimum (min), maximum (max), median (med), mean and standard deviation (std) of the pitch<sup>29</sup> and of the first derivative of the pitch. The fundamental frequency was scanned for rising and falling slopes, that is positive or negative first derivatives, and statistics (mean, med, std) calculated on both. Also included were the first and third quartiles of the pitch. The fraction of voiced elements in the speech is also an important clue, so we included the speaking rate<sup>30</sup>, and the mean height of peaks and valleys in the voiced part. Based on the energy in the spectrogram, further statistics (min, max, med, std) have been extracted for the first, second and third formants.

As mentioned above, of all the databases with emotional speech, none associated multiple emotions with one recording, and we had to produce our own samples. The way we chose to do this was to have 10 persons label each of our recordings as belonging to exactly one emotion. They would not all agree on the same emotion, the resulting distribution thus providing us with an estimate for the emotions a human would perceive as being expressed in the recording. The setup was a desktop computer with headphones. Using a simple graphical user interface the test persons could vote on each recording, playing it repeatedly if desired. This procedure was not supervised. The fuzzy labels for each recording were obtained by taking the mean of all 10 votes, and normalising, as a result corresponding to the vagueness of Definition 3.1.

Apart from providing us with fuzzy labels, the procedure also served as a basic human emotion recognition performance test, yielding a recognition baseline

---

<sup>27</sup>These two German sentences roughly translate to *And where do we want to go?* and *The ideas you have.*

<sup>28</sup>To be precise, a Creative Labs Audigy 2 external USB sound card at 16 kHz, 16 bit resolution, mono recording.

<sup>29</sup>In speech processing, *pitch* and *fundamental frequency* are used synonymously.

<sup>30</sup>The speaking rate is the fraction of voiced elements per second.
2.2. Emotion Recognition from Speech

(72%) against which to compare our automatic system. In the following, we will present some interesting findings about the fuzzy labels obtained.

For 113 of the 413 recordings, all 10 test persons agreed on the emotion contained, that is about a quarter of the labels still remained hard. Figure 2.9 gives a closer view of the fuzzy label distribution. It is evident that on the largest portion of the dataset, there was no clear consensus among the human labellers.

\[
\begin{array}{|c|c|c|c|c|c|c|}
\hline
\text{int. \ maj.} & \text{anger} & \text{sadn.} & \text{neutral} & \text{bored.} & \text{happin.} & \text{disgust} & \text{fear} \\
\hline
\text{anger} & 0.85 & 0 & 0.08 & 0 & 0.07 & 0 & 0 \\
\text{sadness} & 0 & 0.63 & 0.10 & 0.19 & 0 & 0.05 & 0.03 \\
\text{neutral} & 0 & 0.03 & 0.90 & 0.05 & 0 & 0.02 & 0 \\
\text{boredom} & 0.03 & 0 & 0.17 & 0.76 & 0 & 0.03 & 0 \\
\text{happiness} & 0.03 & 0 & 0.07 & 0 & 0.88 & 0 & 0.02 \\
\text{disgust} & 0.08 & 0.12 & 0.08 & 0.03 & 0.07 & 0.58 & 0.03 \\
\text{fear} & 0.03 & 0.14 & 0.14 & 0.02 & 0.03 & 0.14 & 0.51 \\
\hline
\end{array}
\]

\textbf{Table 2.5:} Confusion matrix of intended emotions (down) against the human majority vote (to the right), given as fractions.

In our experiments, we used the fuzzy human labels as a baseline, but it is interesting to briefly compare the human majority opinion with the intended emotions (see Table 2.5). Overall, the defuzzified soft labels coincide only in 73\%\textsuperscript{31} of the cases with the original hard labels. Rather few human labellers recognise correctly the intended emotions of fear, disgust and sadness. Strikingly,

\textsuperscript{31}The value rises to 79.4\% when omitting the emotions of sadness and fear.
when we compare the accuracy of the average human labeller against the human
majority opinion (see Table 2.6), it is again exactly these emotions where humans
do bad. One could therefore hypothesise that it was not bad voice-acting that
caused the recognition-problems on the intended emotions, but general human
insufficiency to recognise the three emotions from speech only. This is partly
supported by the literature-review in [220], where disgust is also reported as
being badly identified by humans, while sadness on the contrary is described as
one of the two best-recognised emotions. Based on the human majority vote
as baseline, our subjects did exhibit a quite high recognition accuracy of 72.1%
(74.8% if we omit sadness and fear, as we did in some applications), while the
average rate reported in [220] is only 62%, highlighting differences in the data.

<table>
<thead>
<tr>
<th></th>
<th>anger</th>
<th>sadn.</th>
<th>neutral</th>
<th>bored.</th>
<th>happin.</th>
<th>disgust</th>
<th>fear</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.80</td>
<td>0.58</td>
<td>0.75</td>
<td>0.74</td>
<td>0.81</td>
<td>0.64</td>
<td>0.73</td>
</tr>
</tbody>
</table>

Table 2.6: For each emotion, the accuracy of the average human. Ground truth is the
human majority vote. Average: 72.1% (74.8% without sadness and fear).

The classifier that yielded the best classification rate in our experiments was the
fuzzy-input fuzzy-output support vector machine (see Chapter 4.5). We used 10
times 10 fold cross validation, the SVMs were trained using the SMO algorithm
by Platt [187]. The fuzzy output from the One-vs-Rest SVMs (polynomial kernel,
\textit{degree} = 3) was obtained by feeding the distance output to sigmoid functions
optimised by batch gradient descent [260]. Using cross validation experiments,
we determined the optimal values for \(C\) to be \(10^{-2}\), out of a range of \([10^{-4}; 10^{3}]\).
The emotions of sadness and fear were omitted from the experiment, since they
were of less interest in our application setting of restaurant recommendation.

<table>
<thead>
<tr>
<th></th>
<th>anger</th>
<th>neutral</th>
<th>bored.</th>
<th>happin.</th>
<th>disgust</th>
</tr>
</thead>
<tbody>
<tr>
<td>basel. res.</td>
<td>0.52</td>
<td>0.20</td>
<td>0.04</td>
<td>0.24</td>
<td>0</td>
</tr>
<tr>
<td>anger</td>
<td>0.03</td>
<td>0.71</td>
<td>0.11</td>
<td>0.11</td>
<td>0.03</td>
</tr>
<tr>
<td>neutral</td>
<td>0</td>
<td>0.35</td>
<td>0.56</td>
<td>0.03</td>
<td>0.06</td>
</tr>
<tr>
<td>boredom</td>
<td>0.30</td>
<td>0.09</td>
<td>0</td>
<td>0.51</td>
<td>0.09</td>
</tr>
<tr>
<td>happiness</td>
<td>0.10</td>
<td>0.30</td>
<td>0.10</td>
<td>0.20</td>
<td>0.30</td>
</tr>
</tbody>
</table>

Table 2.7: Confusion matrix of the recognition experiment, for one run with not
equally distributed emotions. Baseline emotions are given in the rows, values are frac-
tions, the overall accuracy for the run is 54.4%.

The classification accuracy obtained was 55.6%, the confusion matrix for one run
is shown in Table 2.7. For a single classifier, this is respectable. When more features
(based on \textit{linear predictive coding (LPC)} and \textit{mel frequency cepstral coefficients (MFCC)}) are added to form a multiple-classifier system, reportedly [222]
recognition rates of 75.0%\textsuperscript{32} can be achieved, which is better than the human baseline (compare Table 2.6). There is no consensus as to how emotions should be categorised. Also, our own emotional speech database labelled fuzzily is quite unique. That makes it futile to compare the accuracy of our experimental results with other numbers published. Just as rough indication, [153] reports an accuracy of around 76% for their experiments, based on their database with the four emotions of anger, happiness, sadness and fear.

If the goal is not the comparison of algorithms, but to have an application with the highest recognition rates possible, more features would have to be added. In addition to employing LPC and MFCC, and albeit being very difficult to get hold of, voice quality looks very promising and can be extracted for example from spectral properties \cite{96}. Also based on the latter, very good results have been obtained using a modulation spectrum feature \cite{162}: First, frequency statistics are extracted from the recordings via fast Fourier transformation (FFT). To these, a mel-scale transformation (using 8 filter banks) is applied, modeling the perception of the human auditory system. Then, the modulations of the signal for each band are computed via the FFT algorithm. The median values of the resulting modulation vectors inside each band then form the final feature, which represents the rate of change of the frequencies. As for architectures, good results were reported in \cite{50} for a two-layer Bayesian network architecture, where the outcome of the first layer decided which network would then process the first-layer-answers to provide the final answer. The features extracted were basic, the training data recorded from TV and labelled with the usual six emotions.

Both the intended emotions and the human votes are available for our data. So, it is possible to compare the answers from the fuzzy-trained system with both. And although the human majority and the intended emotion coincide only in 79% of the cases, the recognition rate of the fuzzy-trained SVMs is about the same, when taking either of them as baseline. That means, the classifier has never seen the originally intended hard labels, which are quite different to the voted labels, but delivers an equal accuracy on them. This is a strong indication that the fuzzy labels really help the SVMs in capturing the emotion distribution.

\textsuperscript{32}Measured not against human majority vote, but the originally intended emotions. But the results based on the two different baselines should not differ too far, see the remark below in this chapter. Broadening the scope to seven emotions, their recognition rate drops to 59.7%, ours more gently to 48.4%.
2.3 Fruit Recognition

Object recognition from images is a line of research that has numerous real-world applications, for example in industrial production processes. The use case we chose stems from a robotic environment, where the robot is given (voice) orders to locate objects, using his camera eye, and then fetch them. This task was solved within the EU project MirrorBot (EU-IST-FET, [279]), and we could use the image data produced there to improve on the object recognition performance, as a testbed for our classification endeavours.

![Figure 2.10: Example cutouts from the images employed, for each of the 7 fruit classes. The colour variations of the white background and table are due to bad auto-white-detection on the part of the camera, a fact that introduces quite some difficulty into the classification task.](image)

The dataset consists of 840 colour images (resolution: 384x288 pixels), exactly 120 for each of the 7 classes of lemon, red apple, green apple, tangerine, orange, yellow plum and red plum. In a preprocessing step, a square region of interest is detected containing the object, which then has to be classified as belonging to one of the classes listed. In her dissertation, Fay made a performance comparison of different features extracted from the dataset. Based on her ranking, and only selecting diverse feature types to obtain independent clues, we decided to use the following four features: colour histograms, Sobel edge histograms, Canny edge histograms, and edge histograms in the black-white opponent colour space. Each feature is extracted from multiple regions of the image, the resulting vectors are concatenated to form the final feature. The optimal value for the $m \times m$ region grid covering the image depends on each feature. An overlap of 20% between the regions was determined to work well, and all the histograms are set to consist of 8 bins.

The colour histograms operate on the RGB representation of the images, and for each of the three channels individually calculate the frequency of the different values occurring in the image. Since a value of $m = 3$ was chosen, with 8 bins and 3 channels, we obtain $3 \times 3 \times 8 \times 3 = 216$-dimensional colour histogram feature vectors for each image. For the Sobel edge orientation histograms, the image is
folded to detect the edges occurring in it, then the frequency of the edge directions encoded in a histogram (for details see Chapter 2.1.4; \( m = 4 \), feature dimensionality is 128). The Canny edge orientation histograms add some processing steps to the Sobel procedure, to obtain more concise edges [46]: First, the image is blurred with a Gaussian mask to reduce noise. Then, the edges are detected using the Sobel operator. In the non-maxima-suppression step, only those local maxima that are in the direction of the gradient survive. In the final hysteresis step, thresholds discard further edges (\( m = 3 \), feature dimensionality is 72). The last feature operates in the black-white opponent colour space (APQBW), which is based on the theory that different opponent colours share the same channel in the perception of the image [101, 42]. Using the Sobel filter, it again produces edge direction histograms (\( m = 2 \), feature dimensionality is 32).
2.4 Land Cover Mapping in Remote Sensing

Remote sensing is the science and engineering task of using sensors to measure the electromagnetic radiation reflected from a target\textsuperscript{33}. In earth observation, satellite-borne or airborne sensors are employed to gather information about the surface of our world. Land cover mapping then is the process of using the registered information to produce maps of structures on the ground, for example street maps or emergency maps of areas struck by natural disaster\textsuperscript{34}. Different structures, like buildings or vegetated land, can be thought of as classes, leading to a classification task. This is why we chose land cover mapping as one of the example applications of our fuzzy classification techniques.

A special focus was lent to so-called mixed pixels. These are areas on the ground which can not be subdivided into smaller parts due to main sensor resolution, but which comprise several distinct land cover classes, for example water and pine wood. Alternatively it may happen that classes gradually overlap, with several of the basic classes being present in the same geographic location to different degrees, which particularly occurs near imprecise boundaries. These mixed pixels are abundant in remote sensing applications and present a challenge because assigning them to one class in the map leads to misinterpretations [290]. In the remote sensing community, there is a renewed interest to deal with this problem, and for example a future edition of the DFTC Data Fusion Contest (see below in this chapter) shall for the first time include and focus on mixed pixels. Thus, our classification algorithms that can deal with fuzzy class information are especially useful here.

The area of interest (see Figure 2.11) is a coastal plain in the southern part of Italy, located in the alluvial plain of the Salerno Gulf. The area is densely inhabited for the fertility of the land since Greco-Roman times. Land use is primarily agricultural, but during the last sixty years an urbanisation phenomenon occurred, giving rise to a very indented and complex landscape. Consequently, the principal types of land cover are agricultural fields (both fallow fields and crop covered ones), rural fabrics (greenhouses), sea water, a coniferous wood strip along the coastline, and small urban areas made up of discontinuous fabric mixed with vegetation. For our study, we hence defined the following seven classes: vegetated agricultural fields, buildings, pine forest, urban green, sea shore, not vegetated agricultural fields, and water.

It was decided to use two sources of multi-spectral satellite imagery: data captured by the advanced spaceborne thermal emission and reflection radiometer\textsuperscript{33}.

\textsuperscript{33}\textsuperscript{34}For an introduction to remote sensing, turn to [167].

For applications see the Zentrum für satellitengestützte Kriseninformation (ZKI)/ Center for satellite based crisis information of the Deutsches Zentrum für Luft- und Raumfahrt (DLR) at http://www.zki.caf.dlr.de.
Figure 2.11: The area in the gulf of Salerno that is the subject of our experiments. The image is a false colour composite, based on the ASTER bands of near infrared (red), red (green) and green (blue), the pixel size is 15m \[85\].

(ASTER)\(^{35}\) on NASA’s Terra satellite, and high resolution imagery captured by IKONOS 2\(^{36}\), a commercial earth observation satellite. All image data was processed using the geographic information system (GIS) IDRISI\(^{37}\), and rectified to the UTM 33N projection system based on the WGS84 ellipsoid.

The Terra satellite, formerly known as EOS AM-1, has a weight of 5190kg, and a sun-synchronous 705km orbit. One of the five instruments on board is the ASTER sensor, which provides images in 15 bands ranging from visible to thermal infrared. Our image data was taken in winter 2004. We chose a total of 9 bands, going from the visible to the short wave infrared region of the electromagnetic spectrum, from each band one scalar value was extracted. The bands 1-4 have a resolution of 15 m/pixel, and bands 5-9 of 30 m/pixel. All bands were


\(^{36}\)IKONOS 2 is operated by GeoEye, [http://www.geoeye.com](http://www.geoeye.com).

georeferenced for this study and resized to a resolution of 15 m/pixel. The satellites sun-synchronous orbit ensures that the surface illumination angle is nearly the same for every pass.

The IKONOS 2 satellite has a polar, circular, sun-synchronous 681km orbit, its weight is 725kg. Both its sensors have a swath width of 11 km, providing panchromatic and multispectral readings. It has a very high resolution of up to 1 m/pixel. One method to combine its data with the coarser ASTER images is to extract information from larger ground areas (for example 15 m × 15 m), and enrich the corresponding ASTER pixel features with this inter-pixel information: The features we extracted are textural characteristics, which in urban settings were found to improve the classification accuracy over spectral features alone, if the scale of the texture is proper [232]. They are based on the statistical means of the grey-level co-occurrence matrix (GLCM) [97], which is widely used in land cover mapping [282]. The GLCM measures the probability of the co-occurrence of grey-levels in the image area under consideration, co-occurrence in our application being restricted to directly adjoining neighbours (d = 1). The data is typically scaled to a smaller range of integers, [0, . . . , 7] here. Among the several statistical measures which can be extracted from the GLCM to describe specific textural characteristics of the image [167], we chose energy and correlation, which provided the best performance for our dataset:

\[
\text{energy} = \sum_{i,j=1}^{8} GLCM_{i,j}^2
\]

\[
\text{correlation} = \sum_{i,j=1}^{8} \frac{\text{cov}(i, j)}{\text{stdev}(i) \text{stdev}(j)}
\]

Here, covariance and standard deviation are of course calculated on the entries of the 8 × 8 GLCM. The basis for the energy function was a GLCM calculated on the band-ratio between near-infrared and red (4 m/pixel resolution, resized to 1 m/pixel), which in the remote sensing literature is considered as being a reasonable way to avoid effects caused by shadows[38]. The correlation was calculated on GLCMs based on the panchromatic band, which offers a black and white imagery of 1 m/pixel resolution, sensitive to all visible radiation. To eliminate directional relationships, a GLCM is calculated for the horizontal, vertical, and both diagonal directions. The textural features are extracted for each of the directions separately, then averaged over them.

Summing up, our data vectors are made up of 11 components, the first 9 standing for the spectral information taken from ASTER bands, and the last 2 representing textural measures extracted from IKONOS images.

---

38See the chapter “Image division and vegetation indices” in [167] for an introduction to this topic.
2.4. Land Cover Mapping

<table>
<thead>
<tr>
<th>land cover class</th>
<th>training set</th>
<th>test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>vegetated land</td>
<td>145</td>
<td>104</td>
</tr>
<tr>
<td>built-up area</td>
<td>90</td>
<td>98</td>
</tr>
<tr>
<td>pine wood</td>
<td>162</td>
<td>64</td>
</tr>
<tr>
<td>urban green</td>
<td>159</td>
<td>92</td>
</tr>
<tr>
<td>greenhouse</td>
<td>166</td>
<td>106</td>
</tr>
<tr>
<td>not vegetated land</td>
<td>144</td>
<td>89</td>
</tr>
<tr>
<td>water</td>
<td>163</td>
<td>75</td>
</tr>
<tr>
<td>total</td>
<td>1029</td>
<td>628</td>
</tr>
</tbody>
</table>

*Table 2.8: For the Salerno dataset, the distribution of the different land cover classes among the test and training set pixels, given in absolute numbers.*

Out of all the 236985 pixels, expert photointerpreters labelled two spatially separate sets of pixels with their correct land cover class, the training set containing 1029 pixels, the test set 629 (see Table 2.8 for the distribution of classes). Note that because the test set has to be from a different geographic area, the usual technique of random sampling for cross-validation can not be employed.

These numbers already illustrate a core problem of land cover mapping: the limited availability of labelled pixels, which require expensive human labour in the acquisition. To deal with limited training data, we employ self organising maps that can take into account unlabelled data (SOMs, see Chapter 4.4), and support vector machines that can generalise well from few samples (SVMs, see Chapter 4.5). Given the limited amount of labelled test data, the problem of judging the performance of a classifier remains. For example, the confusion matrices of experiments in [86] show near-perfect performance for the greenhouse class. However, we know that this is one of the hardest classes for the classifiers to identify correctly. Thus, all performance measures based on (fuzzy) confusion matrices, as for example promoted in [23] or employed in [290], ultimately operate on a data basis that is too small\(^{39}\). A possible solution, which we pursued in the aforementioned paper, is to have a powerful algorithm (SVMs in this case, with an accuracy of 95.4% on the test set) label the entire dataset, and take this as the groundtruth. This approach impressively singles out greenhouses as the class with the worst SOM recognition rate, contrary to the results on the limited test set. A visual analysis of the land cover mapping can also serve in judging the quality of the algorithm, but can hardly provide concrete accuracy numbers.

\(^{39}\)For an extensive overview of accuracy measures employed in land cover mapping, see the work by Foody [80].
IEEE GRSS Data Fusion Contest

This section presents an application where the fuzzy-output support vector machines have been used very successfully. The work was done in close collaboration with Ferdinando Giacco from the University of Salerno, Italy, during his half-year long stay at our institute in 2008. The occasion for our efforts was the 2008 edition of the DFTC data fusion contest\[^{40}\] organised by the Data Fusion Technical Committee (DFTC)\[^{41}\] of the Geoscience and Remote Sensing Society (GRSS)\[^{42}\] of the International Institute of Electrical and Electronic Engineers (IEEE)\[^{43}\]. The goal of the contest was to establish a benchmark for researchers in this field. Being among the contest winners (we placed 4th), our approach is also detailed in a joint paper with the contest organisers and the other winning groups [156].

The very high resolution hyperspectral data was captured by an airborne ROSIS-03 (reflective optics system imaging spectrometer) optical sensor, the flight on the 8th of July 2002 was operated by the Deutsches Zentrum für Luft- und Raumfahrt (DLR, the German Aerospace Center) in the framework of the HySens project. From the 115 spectral bands, ranging from 430nm to 860nm, 13 were excluded due to noise, leading to 102 bands being available to the contestants, with a resolution of 1.3m per pixel. The total number of pixels was 716800. The area under consideration was a part of the town of Pavia, Italy (45.11N, 9.09E), including a large part of the Ticino river. The distinction of five classes was requested: buildings, roads, shadow, vegetation, and water.

In the following, the architecture we developed to obtain the overall accuracy\[^{44}\] of 96.41% will be presented. In the first stage of our architecture, a 29-dimensional feature vector was employed. We performed a principal component analysis on all the 102 ROSIS spectral bands, and selected those 26 bands with the most significant contribution to the first principal component\[^{45}\]. In addition, we introduced two spatial information measures, namely the standard deviation calculated on the data projected onto the first principal component, and the standard deviation on the ratio of the near-infrared to red bands (bands 102/66, serving to emphasise vegetation by detecting shadows). The 29th feature is the energy calculated on

\[^{40}\]The 2008 edition of the DFTC data fusion contest can be found at http://tlclab.unipv.it/dftc/home.do.
\[^{41}\]DFTC: http://www.grss-ieee.org/DFC/DFCMain
\[^{42}\]GRSS: http://www.grss-ieee.org/
\[^{43}\]IEEE: http://www.ieee.org
\[^{44}\]The accuracy is measured against the baseline labelling the organisers created. It is interesting to note that this baseline labelling is still undisclosed, hence the winning approaches demonstrated that they have a good generalisation ability. The accuracy of a map, together with a confusion matrix and a low-resolution classification map, was shown to the contestant immediately after he uploaded his newest classification effort.
\[^{45}\]The 26 ROSIS spectral bands selected based on the Principal Components Analysis are 1, 2, 3, 5, 6, 10, 13, 26, 27, 41, 46, 61, 62, 63, 64, 65, 72, 78, 80, 81, 82, 84, 89, 90, 100 and 102.
2.4. Land Cover Mapping

the GLCM (mean of two diagonal directions, see detailed explanation above in this chapter). Each textural feature is computed based on a moving window of $3 \times 3$ pixels. We had labels available for a set of 2133 pixels, which were split into two subsets for training (882) and test (1241) during the parameter optimisation phase. As classifier, a One-vs-One fuzzy-output support vector machine (for details see Chapter 4.5) was used, with Bradley-Terry coupling to generate the soft classifier answers. Surprisingly, RBF- and polynomial kernels did not perform well, but were clearly outclassed by a linear kernel, for which we determined an optimal $C$ of 1 using cross-validation experiments.

In the classification results, there was much confusion between the classes of buildings and streets, attributed to the presence of flat, grey roofs and also courts surrounded by roofs. Hence we applied a second classification step to address this problem, which looked only at those pixels that were classified as street or building in the first step. Additional pixels were labelled to capture the temporarily introduced class of grey buildings, leading to a total number of 1614 labelled samples. Also, additional features were introduced as the ones hitherto employed did not provide sufficient information: the one GLCM measure was extended to six, measuring energy, correlation and contrast \cite{85} of GLCMs calculated in the two diagonal directions. For another four new features based on composition coefficients, we employed the $\text{HYPERUSP}$ algorithm implemented in the IDRISI software (see Footnote 37), which first makes use of an unsupervised stage in which a predetermined number of hyperspectral signatures is identified by looking at the whole ROSIS spectral data set. Then, every pixel of the image is considered being a combination of all the signatures computed in the first stage. The coefficients of the four most representative components\footnote{IDRISI HYPERUSP coefficients number 2,4,5 and 6 were selected.} of this hyperspectral decomposition were selected. The classifier was again a One-vs-One fuzzy-output SVM as described for the first stage, the overall accuracy improved from 96.05 to 96.41 percent.

In the final third stage, a simple filter was used to avoid isolated pixels which are classified differently then their neighbours. Using a window of $3 \times 3$ pixels, if the majority of the 8 border pixels agreed on a label, then the decision about the pixel in the middle was adjusted accordingly. The accuracy which the contest system reported for our mapping remained at 96.41%, but the visual impression improved.

The land cover mapping we produced can be seen in Figure 2.12, alongside with a false colour image of the area for visual comparison. Many of the streets were in the shadow, accounting for the prominent appearance of white pixels. The problem of courts being classified as roof/building and flat grey roofs being classified as street is still not solved, which is even more obvious when looking at the confusion matrix in Table 2.9. The other winning groups made their major
Chapter 2. Applications

Figure 2.12: (left) An RGB false colour composite (bands 66, 26 and 12) of the ROSIS data. (right) The mapping achieved by our architecture. Most of the streets are (correctly) classified as shadow.

confusions in different classes, so it is our hope that by intelligently combining the results a near-perfect system could be built\textsuperscript{47}.

No groundtruth labels were provided to the contestants, so each group had to create their own training and test set mappings. The two of the other four groups that report on their labelled pixels give numbers upwards from 200000, which is roughly 100 times the size of our labelled set. Our very competitive setup does thus decrease the need for hours of expensive expert groundtruth-labelling.

I would like to thank Ferdiando Giacco from the Department of Physics, University of Salerno, Italy, for the close and fruitful collaboration, and his supervisors in Italy, especially Silvia Scarpetta from the Computational Neuroscience and

\textsuperscript{47}Combing the five best groups’ results by simple majority vote, in fact the organisers of the contest were able to achieve an accuracy of 99.21%, with the best single approach exhibiting an accuracy of 98.84%.
2.4. Land Cover Mapping

<table>
<thead>
<tr>
<th>groundtr. \ res.</th>
<th>buildings</th>
<th>roads</th>
<th>shadow</th>
<th>veget.</th>
<th>water</th>
</tr>
</thead>
<tbody>
<tr>
<td>buildings</td>
<td>98.9%</td>
<td>211804</td>
<td>1906</td>
<td>262</td>
<td>136</td>
</tr>
<tr>
<td>roads</td>
<td>86.8%</td>
<td>1403</td>
<td>9327</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>shadow</td>
<td>99.3%</td>
<td>29</td>
<td>57</td>
<td>16306</td>
<td>1</td>
</tr>
<tr>
<td>vegetation</td>
<td>99.9%</td>
<td>31</td>
<td>0</td>
<td>1</td>
<td>24456</td>
</tr>
<tr>
<td>water</td>
<td>100.0%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>99.3%</td>
<td>82.6%</td>
<td>98.4%</td>
<td>99.4%</td>
<td>99.8%</td>
</tr>
</tbody>
</table>

Table 2.9: The confusion matrix of our final mapping, as presented by the contest system after the upload of our winning contribution. The percent numbers give the accuracy on the groundtruth classes (down) and predicted classes (to the right).

Artificial Neural Network group of his faculty\(^48\), for sharing their knowledge in the remote-sensing domain, which enabled us to find the most suitable features for our task.

---

\(^48\)Silvia Scarpetta’s other affiliations include the INFN Gruppo Collegato di Salerno, the INFM CNISM Unità di Salerno, and the IASS International Institute for Advanced Scientific Studies, Vietri sul Mare, Italy.
Information or data can be certain (“This is a car.”) or tainted by uncertainties (“The car has a blueish-red colour.”). The same goes for decisions experts make (“I think this patient has the measles.”). The need to represent and work with such uncertain information or decisions led to the evolution of formal uncertainty theories (like Bayes’ theory), some of which will be presented in this chapter. For a comparison of even more uncertainty theories, turn to [246] or [235]. Being in a classification context, this chapter will primarily concentrate on the certainty of decisions.

In one of the earlier approaches, Klir introduces the concepts of vagueness and ambiguity. Vagueness being “associated with the difficulty of making sharp or precise distinctions in the world” (ibid., p. 141), and ambiguity arising from multiple possibilities in an evidence, perhaps even dissonant or conflicting ones. Dubois and Prade compare the information measures of imprecision, dissonance and confusion. Schürmann talks about a set of alternatives, and a residual error rate for the probability that the correct label is in the set of alternatives. In their 2001 work, Borgelt and Kruse introduce three concepts which, after a specialisation to the classification context, are very close to what we are going to use in this thesis: Impräzision provides sets of alternatives. Unsicherheit assesses if a statement is necessarily true. Vagheit refers to inherently vague concepts like “big”, where the authors postulate a Penumbra inside which values can not be attributed unambiguously (Wittgenstein uses a similar concept here, the Familienähnlichkeit).

1 Even though the book is no longer adequate as an introduction to uncertainty in artificial intelligence, as the author notes on his website www.glennshafer.com, it still is a good introduction to the several uncertainty theories.

2 A more formal framework is presented by the two authors in their 1993 context model which is presented in Chapter 3.10.1.

3 Borgelt and Kruse take this as an indicator if the options given in the statement are exhaustive, or if another is possible. We do not use the notion of Certainty in their set-specialised context.
Chapter 3. Uncertainty Theories

We are now going to define the concepts of uncertainty more clearly in the scope of classification, for use in this work. In classification, a sample point (or object) \( x_\mu \) is coupled with a class label \( y_\mu \). In the most basic setting, often dubbed the *hard* case, we have \( y_\mu \in [1, \ldots, L] \), the sample is associated with exactly one of the \( L \) classes, meaning there is no uncertainty at all. We now propose to use the following concepts of uncertainty:\(^4\):

**Definition 3.1.** (Kinds of uncertainty in classification)

- **Vagueness** means that the sample point is associated with different classes to a different degree. Also called the *fuzzy* or *soft* case.
  
  Example: \( y_\mu \in [0, 1]^L \)

- **Imprecision** means that as class label of the sample, a set of classes is given. The true class is contained in the set, but there is no further information that would allow to identify this class.
  
  Example: \( y_\mu = \{y_{\mu,1}, \ldots, y_{\mu,n}\}, y_{\mu,i} \in [1, \ldots, L], n \leq L \)

- **Certainty** quantifies how certain the label is, that means how much the expert or classifier trusts his decision about the sample. This is accomplished by adding a certainty factor \( c \) to a label. This kind of uncertainty can be used together with a traditional hard label as well as the two other cases described above.
  
  Example: label \((y_\mu, c_\mu)\), \( c_\mu \in [0, 1]\)

In some certainty theories it is not strictly necessary to confine the vagueness or certainty to \([0, 1]\), but for convenience reasons we do. Also, when referring to fuzzy or vague labels, we assume that \( \sum_{i=1}^L y_{\mu,i} = 1 \).

In the following, various frameworks to deal with uncertainty are introduced. At the end of this chapter, we show how the classification-specific notions of uncertainty defined above relate to the theories presented, and briefly mention formal approaches to compare uncertainty theories. We can of course not claim to have included all existing frameworks, but tried to capture the most important or distinctive ones. Among the omitted is the subjective probability theory \([110]\) as for example put forward by Savage or Kyburg \([149]\), where a 20% value for a certain event is defined to mean that the current expert would accept a 1:4 bet on this event. The paradigm of *granular computing* \([15]\) shall also be mentioned, which is a technique and framework that borrows from many of the uncertainty theories detailed below, and draws on the fact that data can be viewed at different levels of detail, revealing different knowledge.

---

\(^4\)In German, the following translations are suggested to be used: Vagueness ↔ Vagheit, Imprecision ↔ Ungenauigkeit, Certainty ↔ Sicherheit.
3.1 Probability theory

Probability theory is one of the key concepts in the area of pattern recognition to deal with uncertainty, the foundations for its modern form were laid in 1933 by Kolmogorov in his work *Grundbegriffe der Wahrscheinlichkeitsrechnung* [138]. A short introduction can be found in [25]. In the following, the basic concepts of probability theory will be presented briefly.

Suppose that two boxes, one coloured green and the other brown, are standing on a table. They contain a mixture apples and citrons. Now we randomly pick one of the boxes, take a random fruit out of it, note it, and put the fruit back. Repeating this process, for example ten times, we have statistics about the choices made, for example we know we had chosen the green box in 30% of the cases and the brown one in 70%. The identity of the box chosen can now be considered to be a random variable, say \( B \), which can take on the values \( g \) or \( b \). Similarly for the identity of the chosen fruit, where the random variable \( F \) can take the values \( a \) or \( c \). Then, the probability of an event is defined as the fraction of the times this event occurs in the total number of trials (with the number of trials going to infinity). The identity of the box chosen can now be considered to be a random variable, say \( B \), which can take on the values \( g \) or \( b \). Similarly for the identity of the chosen fruit, where the random variable \( F \) can take the values \( a \) or \( c \). Then, the probability of an event is defined as the fraction of the times this event occurs in the total number of trials (with the number of trials going to infinity). The probability of selecting a brown box would be \( \frac{7}{10} \), and be denoted \( p(b) = 0.7 \). By this definition, probabilities will always be in the range of \([0, 1]\), and if events are mutually exclusive, the sum over all events will always be one. With this knowledge, it is now possible to answer questions like “What is the probability that the citron I had randomly chosen came from the brown box?”, formally written as \( p(B = b | F = c) \), or “What is the probability that when choosing, I end up with a citron from the brown box?”, formally \( p(B = b, F = c) \). To make the formulas easier to handle, the value of the probability variable is not given if it is clear from the context, that is \( p(F = c) \) becomes \( p(F) \). This notation is already enough to formulate the two fundamental rules of the probability theory:

\[
\text{sum rule : } p(B) = \sum_F p(B, F) \tag{3.1}
\]

\[
\text{product rule : } p(B, F) = p(F|B) \cdot p(B) \tag{3.2}
\]

Using the product rule and the symmetry of \( p \), we obtain the famous and useful

\[
\text{Bayes’ theorem : } p(B|F) = \frac{p(F|B) \cdot p(B)}{p(F)} \tag{3.3}
\]

which allows to compute the conditional probability of \( B \) given that we know the value of \( F \).

If we had been asked “How often do you choose the green box”, we would only know about the probability \( p(B = g) = 0.3 \) before having chosen a fruit. In fact, \( p(B) \) is called prior probability. If we now know what type of fruit we have chosen, we can compute the probability \( p(B|F) \) which is called posterior probability. This
Chapter 3. Uncertainty Theories

makes it possible to include knowledge, for example from other classifiers, into what is known about the probability of a sample.

So far, we have only dealt with discrete sets of events, that is the random variables could only take values from a limited set, for example \( \{ \text{Apple, Citron} \} \). This is the most usual case in pattern classification. If a random variables \( x \) now can take real values, and the probability of \( x \) falling into the interval \([x, x + \delta x]\) is given by \( p(x)\delta x \) for \( \delta x \to 0 \), then \( p(x) \) is called a probability distribution over \( x \). The probability to draw an \( x \) with a value in the interval \([a, b]\) is then

\[
p(x \in [a, b]) = \int_a^b p(x) \, dx.
\]

(3.4)

Still, probability densities have to satisfy the following two conditions

\[
p(x) \geq 0 \quad (3.5)
\]

\[
\int_{-\infty}^{\infty} p(x) \, dx = 1 \quad (3.6)
\]

If the variable only takes discrete values, it can be called a probability mass function, with the meaning that all probability is concentrated in places with permitted values of the variable.

The explanations so far were based on the frequentist interpretation of probability, with probabilities being calculated from existing data. The second well-established approach is called the Bayesian view. It allows to assign probabilities to events about which not much is known, for example the question when the internet will collapse. Additional knowledge can be incorporated later using Bayes’ theorem, as given above in formula 3.3. But, the nature of the process has a different meaning. Suppose there is a set of samples \( S \), that has to be fitted to a model using the parameter \( w \), then the uncertainty in the choice of parameter given the observed data is:

\[
p(w|S) = \frac{p(S|w)p(w)}{p(S)} \quad (3.7)
\]

Here assumptions, or prior knowledge, about the distribution of \( w \) are incorporated using the prior distribution \( p(w) \). A difference to the frequentist approach is the quantity \( p(S|w) \), expressing how likely the current samples are given the choice of parameter. Being a function of a so-called likelihood function, it is not a probability distribution over \( w \), nor does its integral need to be equal to one.

The propagation of (new) knowledge is possible using just the methods above, but also for the sake of a better visualisation, graphical models with vertices and edges can be used. The most prominent examples are directed graphs (Bayesian networks [183, 116]) and undirected graphs (Markov random fields [133]).
Probability theory nowadays is soundly anchored in most peoples’ pool of knowledge, and we use it to decide in plenty of daily life situations. However, as pointed out long ago by Tversky and Kahneman [265], these judgments made under uncertainty regularly violate Bayes’ theory, because of the prevalence of heuristics and biases even with more experienced researchers.

### 3.2 Logic

The first approaches to a formal logic date back to the fourth century BC, with Aristotle’s *Analytika protera* [13]. Logic provides formal languages to make precise and unambiguous statements, that is to encode knowledge. A set of rules allows to draw conclusions from the set of statements given, through a process called inference.

Using disjunctions, it is easily possible to model a set of alternatives (imprecision). But, certainty or vagueness can not be represented, and hence an extension called modal logic was brought forward. Using its three modes, or modalities, namely possibility, probability and necessity, it was possible to mark the certainty of statements. But still, no preference among statements could be encoded. This lead to the development of multi-valued logic, with \( n \) different truth values, pioneered by Jan Łukasiewicz. The concept of fuzzy logic (see Chapter 3.6) can be regarded as a multi-valued logic with \( n \to \infty \). Likewise, the concept of certainty factors, which was popular in the artificial intelligence community in the early 1970s can be regarded as multi-valued logic with values between -1 and 1. The approach was used mainly in the expert system *MYCIN* [238], designed to identify bacteria causing severe infections. The concept did not transfer well to other application areas, and an analysis [99] revealed that it made unwarranted implicit independence assumptions.

A newer approach is not to quantify the uncertainty exactly, but to use non-monotonic logic. These formal frameworks allow defeasible inference, that is to draw conclusions which can be later reversed when new contradicting information is obtained. One example is the theory of default logic [202], which is modeled as a pair \((W, D)\). The sure knowledge is encoded in the background theory \( W \), the default rules in \( D \) in the following form:

\[
\frac{P(x) : J_1(x) \ldots J_n(x)}{C(x)}
\]  

(3.8)

If the prerequisite \( P \) is believed to be true, and we do not find evidence in the database that any of the justifications \( J_i \) is false, we infer that the conclusion \( C \) is true. This allows to model for example the following: we know that birds normally fly, and that a penguin is a bird. If we do not explicitly know (and encode) the
contrary, we can conclude that a penguin flies. The set of all conclusions that can be drawn from the pair \((W, D)\) is then called an extension.

In their DUCK-approach (*deduction of uncertain knowledge*, explained for example in [263]), the group of Kießling decided to use an interval-mapping of probabilities to logic values. That means that an event would be true if the associated variable was in a certain range. The inference rules for this special view were also specified in detail.

Romeyn regarded the Bayesian probability approach (see Chapter 3.1) in the light of a logic framework [206]. Observations are encoded in a cylindrical algebra, and beliefs as probability distribution over this algebra.

Current research on the topic is also collected in the proceedings of the European Conferences on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, (*ECSQARU*[^5]).

### 3.3 Uncertainty as Distribution of Opinions

In their 1997 paper [110], Hummel and Manevitz proposed a framework for dealing with uncertainty, where uncertainty is not seen as Bayesian probability, but is measured by the range of opinions offered by multiple experts. This viewpoint allows to gain important insight concerning the nature of uncertainty and also the relation of different uncertainty calculi. For example, they succeed in expressing Dempster-Shafer theory (see Chapter 3.7) and Kalman filtering [126] in terms of their model. Note that in his 1991 work [266], Tzeng proposed a quite similar theory. In the following, the basic idea of uncertainty as variance of opinions is presented, followed by a brief description of the design options in the framework.

When thinking in the terms established in Chapter 3.1, the authors’ view of uncertainty is more of a frequentist than a Bayesian one, albeit with frequencies related to the opinions, not the events. That is, if the diagnosis says *appendicitis* with a 70% measure, this does not mean that out of 100 patients with these symptoms, 70 have appendicitis, but that given these symptoms, 70% of the doctors would give this diagnosis. Hence, the basic uncertainty does not come from probabilistic uncertainty arising from a certain piece of evidence, but the spread of opinions experts have when presented with this evidence.

The model devised to formalise the treatment of this kind of uncertainty provides choices in four categories: the representation of expert opinions, the statistics to retain, methods to combine opinions, and methods to combine pools or sets of experts.

3.3. Distribution of Opinions

The straightforward way to encode an opinion is to give probabilities. So, being presented with a piece of evidence, each expert $\omega$ gives an opinion $x_\omega(l_i)$, with $i \in [1, \ldots, L]$; for each of the $L$ classes\(^6\) he states his estimate of the probability that, given the information, the current object under consideration belongs to class $i$. Each opinion is bounded by $0 \leq x_\omega(l_i) \leq 1$, but it is not necessary that the sum over the $L$ opinions for all classes equals to one, since each single opinion for a class is based on a separate evaluation of the expert. However, the experts might be asked to normalise their opinions to fulfil the sum criterion. As the authors say, “in practice, this is the most likely scenario.” When combining probability opinions, often the prior probability $p(l_i)$ of each class has to be known.

The use of log-probabilities offers then a way to encode the prior probability of a class into the opinions:

$$
x_\omega(l_i) = \log\left(\frac{\text{Probability of } l_i \text{ according to expert } \omega, \text{ based on the evidence}}{\text{prior probability of } l_i \text{ as estimated by the expert } \omega}\right)
= \log\left(\frac{p(l_i|\text{evidence } s)}{p(l_i)}\right)
$$

It indicates the influence of evidence $s$ on the probability of the label $l_i$, and of course the conditional probabilities from the option above can be recovered if the prior probabilities $p(l_i)$ are known.

In cases with only two classes $l_1$ and $l_2$, an odds formulation is possible:

$$
x_\omega = \frac{p(l_1|\text{evidence } s)}{p(l_2|\text{evidence } s)}
$$

To measure the uncertainty in combined answers, one has to maintain knowledge or statistics about all the opinions. The most detailed and expensive choice is to retain representations of all opinions, an approach employed in the Dempster-Shafer theory (see Chapter 3.7). If it is necessary and desired to reduce the opinion data to statistics, maintaining mean and standard deviations or mean and covariance are the options given, where covariance is meant to be computed among the $L$ labels, not among the experts. In the cases just mentioned, a narrower distribution means more certainty in the combined opinion.

There are also several more or less relevant approaches given for combining sets of experts $E_i$, which each contain many opinions. In the field of classification, such a distinction between combining sets of experts and combining opinions seems artificial, since one could directly combine the opinions, the more since one assumes that the sets of experts to be combined have all seen the same evidence.

\(^6\)The authors do not speak of classes, but labels. In the literature, words with a similar meaning are: hypotheses, outcome or frame of discernment. Note that in this context, only exactly one class can be true in a specific situation.
In the *set union* option, the sets are only put together, that is not combined but pooled. With the *set product* $E_1 \times E_2$, combining produces pairs of experts, with all possible combinations that include one expert from $E_1$ and one from $E_2$. This is is the approached used in the Dempster-Shafer theory. Here it is also possible to use a *product measure*, that is to assign each individual expert a weight in the combination. The following possible combination schemes will only be applicable in special cases. With *pairwise matching*, it is assumed that the experts in the two sets to be combined have the same order, and the experts on the same position in the sets will be combined. It is necessary that the errors in the opinions are orthogonal here. As the authors demonstrate, it is possible to see Kalman filtering \[126\] as a combination of measurements using pairwise matching. In the method *non-commutative update by means*, each expert in $E_2$ is combined with the mean of all opinions in $E_1$. There is also a commutative variant.

The core of the framework has to deal with the question how to *combine opinions*, that is the expert probability estimates. The most powerful concept is *functional updating*:

$$r = \pi_{s_1,s_2}(p_1, p_2)$$

with $p_i$ being the full probability vector based on one piece of evidence $s_i$, $p_i = ( p(l_1|s_i), \ldots, p(l_L|s_i) )$. The resulting $r$ has the same form as $p_i$, but represents the probability based on the combined evidences. The combination function $\pi$ can model arbitrarily complex combination behaviour, with the drawback that the desired behaviour, conditioned on different evidences, is not known.

To simplify, several assumptions can be made, for example one might take for given the *conditional independence* between $s_1$ and $s_2$:

$$p(s_2|s_1, l_i) = p(s_2|l_i) \quad \forall l_i$$

This leads to a combination function with normalising constant $K$:

$$p(l_i|s_1, s_2) = \frac{1}{K} \frac{p(l_i|s_1) p(l_i|s_2)}{p(l_i)}$$

If additionally *unconditional independence*, $p(s_1, s_2) = p(s_1) p(s_2)$, is postulated, $K$ reduces to 1, but in order to retain valid probability distributions, constraints would have to be placed on $p(l_i|s_2)$. This problem vanishes once the elements to be combined are viewed not as probabilistic but as opinions which, being estimates as explained above, need not be normalised.

As the independence assumptions rarely hold in practical applications, the authors introduced the notion of *alpha dependence* \[111\]:

$$p(s_2|s_1, l_i) = \left[ p(s_2|l_i) \right]^\alpha(s_1,s_2), \quad \forall l_i$$
3.4 Possibility Theory and Fuzzy Sets

With \( \alpha(s_1, s_2) \) being independent of the class \( l_i \), the combination formula remains simple.

The conjunctive updating using the \( \min \) rule is grounded in fuzzy set theory (see Chapter 3.4) and only to be used in special cases, for example a 2-class setting.

For each combination of the choices they offer in their framework, Hummel and Manevitz give tables detailing how to update the statistics retained. For example, when using probabilistic opinions, product combination of experts and conditional and unconditional independence assumptions in opinion combination, the formulas for updating the mean \( \mu \) and covariance \( C \) are:

\[
\mu = \mu_1 \mu_2 \gamma^{-1} \\
C = (\gamma^{-1})^2 [C_1 C_2 + \mu_1^2 C_2 + \mu_2^2 C_1]
\]

Here \( \gamma \) is the probability vector of the respective class, and the multiplications are intended as termwise. Note, however, that for the functional combination no tables are given, since \( \pi \) can be a nearly arbitrary function.

For the area of classification, sadly no practical solutions are offered for the combination of opinions required. An assumption of conditional independence leads to combination via Bayes’ theorem (see Formula 3.3), while the less strict alpha-dependence can not be assured in most applications.

The achievements of this framework are two-fold: the authors offer a view of uncertainty as spread of opinions, which seems well suited in practical applications like medical diagnosis or emotion recognition. Secondly, by presenting the associated opinion combination framework as construction kit with lists of choices, they enable the reader to decompose this complex problem.

3.4 Possibility Theory and Fuzzy Sets

The term possibility theory was coined by Zahdeh in his 1978 paper *Fuzzy Sets as a Basis for a Theory of Possibility* [296], emphasising already a close tie with the theory of fuzzy sets. The framework offers a way to deal with uncertainty that is different from and in some sense complementary to probability theory. It was extended later by Dubois and Prade, on whose works (especially [67] and [68]) the following introduction is based. An introduction to fuzzy sets can also be found in [140].

The authors use the notions of imprecision and uncertainty (also called confidence) in their paper. Speaking in our nomenclature as introduced in the beginning of Chapter 3 (see Definition 3.1), imprecision subsumes our vagueness and imprecision, while uncertainty corresponds to certainty.

As starting point, a measure of confidence \( g(A) \in [0,1] \), \( A \in \Omega \) is assumed,
which takes increasing values with increasing confidence. The reference set \( \Omega \) contains all possible events (or classes in the realm of classification). As minimum requirement, \( g \) has to be monotonic with respect to inclusion:

\[
A \subseteq B \Rightarrow g(A) \leq g(B)
\]

As consequence, the following holds for disjunctions \((A \cup B)\) and conjunctions \((A \cap B)\), for \( \forall A, B \subseteq \Omega \):

\[
\begin{align*}
  g(A \cup B) &\geq \max(g(A), g(B)) \\
  g(A \cap B) &\leq \min(g(A), g(B))
\end{align*}
\] (3.12) (3.13)

For the extreme case of Formula 3.12, we naturally arrive at the definition for the possibility measure \( \Pi \):

\[
\Pi(A \cup B) = \max(\Pi(A), \Pi(B)), \quad \forall A, B \subseteq \Omega
\] (3.14)

One concrete realisation of this confidence measure could be the following, assuming the event \( E \subseteq \Omega \) is sure:

\[
\Pi(A) = \begin{cases} 
  1 & \text{if } A \cap E \neq \emptyset \\
  0 & \text{otherwise}
\end{cases}
\]

Here, the meaning of \( \Pi(A) = 1 \) is intuitively clear: \( A \) is possible.

Finite reference sets \( \Omega \) allows for a different formulation, with the help of the possibility distribution \( \pi(\omega) = \Pi(\{\omega\}) \), which maps from \( \Omega \) into \([0, 1]\):

\[
\Pi(A) = \sup\{\pi(\omega)|\omega \in A\}, \quad \forall A \subseteq \Omega
\]

Analogue to \( \Pi \), a necessity measure can be constructed from the extreme case of Formula 3.13:

\[
N(A \cap B) = \min(N(A), N(B)) \quad \forall A, B \subseteq \Omega
\] (3.15)

An event \( A \) and its complement in \( \Omega \), \( \bar{A} \), can not both be necessary for even small values:

\[
\min(N(A), N(\bar{A})) = 0
\]

As for the relation between necessity and possibility measures, the following holds \( \forall A \subseteq \Omega \):

\[
\begin{align*}
  \Pi(A) &\geq N(A) \\
  N(A) &> 0 \Rightarrow \Pi(A) = 1 \\
  P(A) &< 1 \Rightarrow N(A) = 0
\end{align*}
\]
In probability theory, the probability of an event plus the probability of the
contrary event equal to 1. In possibility theory, only the following relations are
implied:

\[ N(A) + N(\overline{A}) \leq 1 \]
\[ \Pi(A) + \Pi(\overline{A}) \geq 1 \]

That leaves the expert more freedom to model and encode its knowledge, or as
Dubois and Prade put it, “it seems natural not to wish to rigidify the relationship
between indications one has in favour of an event (the degree of necessity) and
those that weigh against it (one minus the degree of possibility)” ([67], p. 11).
Note that, if all the focal elements \( \omega \) of a possibility distribution with \( \pi(\omega) > 0 \)
are elementary (that is disjoint, for example consisting only of one class label
each), the possibility distribution folds down to a probability distribution\(^7\).

Very closely related to possibility theory is the notion of fuzzy sets [295], which
expands ordinary set theory by introducing degrees of memberships for sets. Form-
ally, the degree of membership to a fuzzy set \( S \) is given by a mapping \( \mu_S(\omega) \)
from the reference set \( \Omega \) to \([0, 1]\). This allows for the modelling of vague associa-
tions as possible with natural language. “Bob is somewhat tall” could be written
as \( \mu_{\text{tall}}(\text{Bob}) = 0.7 \), with “Bob” \( \in \Omega \).

The way to obtain crisp ordinary sets from fuzzy sets is called an \( \alpha \) – cut:

\[ S_\alpha = \{ \omega \in \Omega \mid \mu_S(\omega) \geq \alpha \} \quad (3.16) \]

The crisp cut \( S_\alpha \) contains all elements of \( \Omega \) that are compatible with \( S \) on the
level of at least \( \alpha \). Looking from the reverse direction, it is hence possible to
define a fuzzy set by giving a series of (nested!) \( \alpha \) – cuts in the form of crisp sets.

Another view on fuzzy sets is to consider them as the outcome of a possibility
distribution if only looking at singletons. So, for any such possibility distribution
\( \Pi \), there is a fuzzy set \( S \) with \( \mu_S(\omega) = \Pi(\{\omega\}) \). We might say that the fuzzy
set \( S \) is the sure event the possibility distribution focuses on. This highlights
an important difference: in the context of classification, in possibility theory the
reference set \( \Omega \) contains the class labels, while with fuzzy sets, there is one fuzzy
set per class, with \( \Omega \) containing the feature variable values possible (continuous
reference set).

Furthermore, the membership function of a fuzzy set can be interpreted as a
likelihood function in the frequentist sense, that is \( \mu_S(e) = P(S|e) \), giving the
proportion of experiments in which the event \( e \) was found to be compatible with
the concept or label \( S \).

Inspired by the concept of the \( \alpha \) – cut (see Formula 3.16), the following combi-
nation of fuzzy sets and possibility theory was codified. For a current event \( e \), it

\(^7\)This is similar to Dempster-Shafer theory, see Chapter 3.7.
can be said that a fuzzy set $F$ is $\alpha$-possible if

$$\forall \omega \in \Omega, \quad \pi_\varepsilon(\omega) \geq \min(\mu_F(\omega), \alpha).$$

The combination of information from different sources is handled on the level of possibility distributions. To account for differences in reliability, if the certainty of a certain source $i$ is known, the possibility distribution $\pi_i$ it produced can be adjusted. Let $c_i \in [0, 1]$ be the certainty measure. Then the new measure could be:

$$\pi'_i = \max(\pi_i, 1 - c_i) \quad (3.17)$$

Alternatively, in analogy to the procedure in Dempster-Shafer theory, the technique of discounting could be used to adjust the weight of focal elements.

For the combination of possibility distributions, there is no universal solution. Rather, the mode of combination will depend on assumptions and knowledge about the sources of the distributions. There are two extreme cases: in conjunctive mode ($\cap$), one assumes that all sources are equally reliable, while in disjunctive mode ($\cup$), only one source is supposed to have the right answer. Possible choices for the conjunctive combination $\pi_\cap(\omega) = \pi_1(\omega) \ast \pi_2(\omega)$ of two sources are for example:

$$a \ast b = a \cdot b \quad (3.18)$$
$$a \ast b = \min(a, b) \quad (3.19)$$
$$a \ast b = \max(a + b - 1, 0) \quad (3.20)$$

Note that the product operation is not idempotent. In the min case, the source that gives the least weight to an alternative is assumed to be the best-informed one. This is contrary to the case of disjunctive combination, where each distribution is thought to give a lower bound that can be topped by better-informed sources. Hence some alternatives for the disjunctive combination $\pi_\cup(\omega) = \pi_1(\omega) \perp \pi_2(\omega)$ are:

$$a \perp b = \max(a, b) \quad (3.21)$$
$$a \perp b = \min(a + b, 1) \quad (3.22)$$
$$a \perp b = a + b - a \cdot b \quad (3.23)$$

If not all sources are equally reliable, it is quite natural to partition them into reliability classes $K_i, \quad i = 1 \ldots n$, with those in $K_i$ being more reliable than sources in $K_{i+1}$. In each class $K_i$ the sources would be combined in a symmetric fashion. Across classes, the combination would be iterative pairwise, starting with classes $n$ and $n - 1$, using the following combination rule:

$$\forall \omega \in \Omega, \quad \pi_{1-2}(\omega) = \min(\pi_1(\omega), \max(\pi_2(\omega), 1 - h(\pi_1, \pi_2)))$$
3.5 Rough Sets

The measure $h$ indicates the consistency of the two sources, it is zero when they are entirely contradicting.

As mentioned above, one of the practical applications of possibility distributions is to encode natural language categories. For simple concepts like “tiny”, psychometric experiments can identify the fuzzy values on the chosen scale. The second way to obtain fuzzy sets is from statistical data. Starting with histograms on the relationship of events $\omega$ and nested focal elements, a probability measure can be constructed that is bound by possibility and necessity functions. Nested focal elements can also be produced based on interval-valued sensor readings.

Fuzzy sets are also suitable as a tool to solve problems of multi-criteria choices, as they are able to encode constraints and objectives. For a literature survey on how to do so in practical applications see \[65\].

Furthermore, fuzzy sets are the foundations for the frameworks of fuzzy logic and fuzzy control described in Chapter 3.6, where you will also find practical examples for membership functions.

3.5 Rough Sets

The name rough sets suggests a rather close relationship to the fuzzy sets of Chapter 3.4. This is the sole reason for including a short introduction to rough set theory here, which is not so much concerned with modeling uncertainty, but identifying fuzzy classes in existing data.

Rough set theory was initially introduced by Pawlak [181, 182] and widely studied afterwards. An introductory overview can be found in [139]. The relationship with Dempster-Shafer theory, for example how to produce basic probability assignments from decision tables, is covered in [240].

The basis of rough sets is a collection of data, a so-called information system, which consists of a list of objects with associated attribute values. Formally, all objects are in $U$, the universe, and all the attributes are listed in $A$. Now equivalence classes are introduced, with objects in one class being indistinguishable while only looking at a subset $B \subset A$ of the attributes. The equivalence relation

$$\text{Ind}(B) = \{ (x, \hat{x}) \in U^2 \mid a(x) = a(\hat{x}), \forall a \in B \}$$

(3.24)

is then also called a B-indiscernability relation\(^8\). Most of the times, it is not possible to produce equivalence classes that only contain objects of one class (the class can be considered a special attribute), and the solution of rough sets is to provide upper and lower approximations for the desired data set $X$ in the form

---

\(^8\)The letter $B$ here refers to the name of the subset $B$ of attributes, and of course changes if the name of the subset is changed.
Chapter 3. Uncertainty Theories

of crisp sets:

\[
B:\text{lower} = B_X = \{x \mid [x]_B \subseteq X\} \quad (3.25)
\]

\[
B:\text{upper} = \overline{B}X = \{x \mid [x]_B \cap X \neq \emptyset\} \quad (3.26)
\]

The equivalence class containing the object \(x\) is here denoted by \([x]_B\). Objects in \(B_X\) are in \(X\) for sure, while those in \(\overline{B}X\) might be outside. A rough set is now defined as the family of subsets of \(U\) having the same \(B_X\) and \(\overline{B}X\). The accuracy of this approximation is given by counting the number of agreeing objects:

\[
\alpha_B(X) = \frac{|B_X|}{|\overline{B}X|} \quad (3.27)
\]

One of the most interesting applications of rough sets is to find a subset of the attributes that is sufficient to well represent the entire dataset. This is accomplished using a so-called reduct\(^9\).

About the relationship of rough sets with fuzzy sets, a wide body of literature is available (see for example \([69]\)). Basically, since rough sets provide an approximate description of concepts, they can be employed to model fuzzy sets. Rough sets adhere more to probabilistic principles, as reflected by their rough membership function

\[
\mu_X^B(x) = \frac{|[x]_B \cap X|}{|[x]_B|}, \quad \text{with } \mu_X^B(x) : U \rightarrow [0, 1]
\]

which can be interpreted as a frequency-based probability estimate. Comparing the following property of rough memberships

\[
\mu_X^B(x) \cap \mu_Y^B(x) \leq \min(\mu_X^B(x), \mu_Y^B(x)), \quad \forall x \in U
\]

to its fuzzy counterpart (see Formula 3.15), it can be seen that rough sets are a generalisation of the latter.

3.6 Fuzzy Logic and Fuzzy Control

Fuzzy logic is an extension to Zadeh’s fuzzy sets, which are presented in Chapter 3.4. It has become the common name for a large number of theories and techniques, which can be divided into two main direction: the best-known fuzzy logic in the broad sense or fuzzy control is very close to fuzzy sets, enabling the encoding of vague (natural language) categories, and being widely employed for controlling industry and household hardware automata, like robots and washing

\(^9\)The problem does remain NP-hard [241].
3.6. Fuzzy Logic and Fuzzy Control

machines. Its goal is to provide working solutions that are good enough, but not necessarily to provide deep mathematical foundations. On the contrary, fuzzy logic in the narrow sense or formal fuzzy logic strives to provide a theoretical underpinning for fuzzy sets, and continue the development under the rules of multi-valued logic (compare Chapter 3.2). For a short introduction and a fuzzy logic bibliography see [94]. The relationship between fuzzy logic and probability is explored in [95].

In formal fuzzy logic, the goal is to model the operations possible in standard logic using equivalents that are able to operate on fuzzy sets, to allow inference under vagueness. In this multi-valued logic, the statement \( x \in X \) can take values in \([0, 1]\), and not only be true or false. To facilitate notation, for these statements we will be operating on variables \( a, b \in [0, 1] \). The basis for operations are t-norms, short for triangular norms, that are used in fuzzy set theory to describe conjunctive and disjunctive (t-conorms here) operations (see Formulas 3.18 to 3.23), and here represent the logical equivalents of \( \cap \) (and) and \( \cup \) (or). They are binary operations in the unit interval that have to be commutative, associative and non-decreasing, with 1 being their unit element. The best known formulation for the negation \( \neg \) is the Łukasiewicz negation: \( \neg a = 1 - a \). The logic implication \( \rightarrow \) can be constructed from the already defined conjunction \( * \) and negation operators (S-implication), or by employing for it the residuum-operator\(^{11}\) \( \Rightarrow \) defined for t-norms (R-implication). The second choice leads to \( a \rightarrow b = \max(c | a * c \leq b) \). Another possible choice would be the Zadeh implication \( a \rightarrow b = \max(1 - a, \min(a, b)) \).

Fuzzy control is a way to model the behaviour of controllers for technical applications in cases where it is not possible to do so by traditional means. For example, it might not be possible to specify the input-output behaviour using differential equations, because there is not enough knowledge available about the system. The solution using fuzzy control is to take the partial expert knowledge available, encode it as fuzzy sets, and build the system response on this. One example for such knowledge is the statement “When the temperature is very low, the heating must be running at least at medium level”. A brief introduction into fuzzy control, accompanied by a clear example, can be found in [100], a very thorough one in [171], while [40] provides extensive theoretical background.

In the system to control, there will be a multivariate input variable \( x \), generally consisting of sensor readings, and a possibly multivariate output variable \( y \), which represent the parts of the system that shall be manipulated by the controller. The

---

\(^{10}\)More formally, \( a, b \) will be the value of the possibility distribution \( \pi(\omega) \) for the event \( \omega \in \Omega \) under consideration, as explained in detail in Chapter 3.4.

\(^{11}\)The residuum operator \( \Rightarrow \) is a binary operator defined on left-continuous t-norms * such that \( a \ast b \leq c \iff a \leq (b \Rightarrow c) \forall a, b, c \in [0, 1] \).
basis for the control are fuzzy implications:

\[ r_k : IF \ (x \ is \ A_k^1) \ AND \ (x \ is \ A_k^2) \ \ldots \ \AND \ (x \ is \ A_k^n) \ THEN \ (y \ is \ B_k) \quad (3.28) \]

This rule \( r_k \) splits into the \( n \) antecedents and once consequence. The \( A_k \) represent fuzzy sets modelled on the (multivariate) input variable, while the \( B_k \) are fuzzy sets that model the output. If for example \( x \in X^1 \times X^2 \times \cdots \times X^n \), then for each condition, one can now read the value \( \mu_k^j(x_j) \), with \( j = 1 \ldots n \), which the respective set membership function takes. Since here all the antecedents are combined conjunctively, a t-norm \( t \) (for example the \( \min \) operator) is employed to obtain the combined value \( \alpha_k \):

\[ \alpha_k = t(\mu_k^1, \mu_k^2, \ldots, \mu_k^n) \]

To obtain the output \( \mu^{out}_k \), the membership function of \( B_k \) and \( \alpha_k \) are combined via a t-norm. Again, in the case of the \( \min \) operator, this amounts to cutting the said membership function at the level \( \alpha_k \), taking only the values below\(^{12}\). The \( k \) different rules represent alternative solutions to the control problem, hence a disjunction which will be combined using a co-t-norm \( \omega \), such as the \( \max \) operator:

\[ \mu^{out} = \omega(\mu^{out}_1, \mu^{out}_1, \ldots, \mu^{out}_k) \]

Do note that the output \( \mu^{out} \) so far is still a fuzzy membership function. But, we can not set the output- or control-variable \( y \) of a machine to a fuzzy value. The process that solves this issue is called defuzzification, and the most important approach is the so-called centre of gravity/area method. It works by weighting each possible output value with the value it gets assigned by \( \mu^{out} \), then calculating the normalised average. This ensures a smooth regulation behaviour, and is intuitively correct in the sense that only fuzzy distributions with high membership values will have any significant impact.

Note that for each rule \( r_k \) (see Formula 3.28) it is possible to define a confidence-value\(^{13}\) \( c_k \in [0, 1] \) which expresses the believe in the correctness of the rule. Using a t-norm, \( c_k \) is combined with the outcome \( \alpha_k \), which in the case of the \( \min \) operator chosen limits the membership of the output fuzzy set \( B_k \).

It should be mentioned that fuzzy sets can be defined on discrete events, as introduced with the reference set \( \Omega \) in Chapter 3.4, or on a continuous scale, for example \( x \in \mathbb{R} \), as used in fuzzy control. Most combination schemes are the same for both choices, but for example as co-t-norm \( \omega \) integration would be a possible choice in the continuous case (compare Formula 3.29).

\[^{12}\text{This is in contrast to the } \alpha\text{-cut established in Formula 3.16 where all values above are chosen.}\]

\[^{13}\text{The confidence value would be called certainty value in the nomenclature set forth in Definition 3.1.}\]
To give a better intuition on fuzzy control, a brief example will be presented in the following. Braking with a tradition in the literature, it will not be the “pole balancing” task but about controlling a steam turbine, as presented in [90]. The steam engine to be controlled has two\textsuperscript{14} input variables, pressure and temperature, and one output variable, the turbine throttle setting. An expert states that the temperature is divided into the levels cool, nominal and warm, while the pressure level is distinguished into the categories low, ok and strong.

The membership functions for the associated fuzzy sets \textit{Press} and \textit{Thr} could look like those presented in Figure 3.1. The turbine can be run in reverse mode, so the throttle parameter, which we want to set, can be negative (compare Figure 3.2 for the corresponding fuzzy sets \textit{Thr}).

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig3.png}
\caption{Example fuzzy sets for the temperature and pressure input variables.}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig4.png}
\caption{Example fuzzy sets for the throttle output variable. The machine can be operated in reverse, hence the negative values.}
\end{figure}

\textsuperscript{14}In order to simplify the notation, in the example we have two input variables here, and not one multivariate one as introduced above.
Figure 3.3: Evaluation of the fuzzy system as described in the example. The rules $r_{k_1}$ and $r_{k_2}$ are evaluated and after combination produce a fuzzy set. By finding the centre of it, we learn to which value $thr$ the throttle variable has to be set.
3.6. Fuzzy Logic and Fuzzy Control

The rules for the system are specified in the form presented in Formula 3.28, for example the following two rules could be part of the system:

\[ r_{k1} : IF \ (temperature \ is \ Temp_{cool}) \ AND \ (pressure \ is \ Press_{low}) \ THEN \ (throttle \ is \ Thr_{pos1}) \]
\[ r_{k2} : IF \ (temperature \ is \ Temp_{cool}) \ AND \ (pressure \ is \ Press_{ok}) \ THEN \ (throttle \ is \ Thr_{zero}) \]

The evaluation of the rules \( r_{ki} \) would proceed as given in Figure 3.3. Starting from the current temperature \( t \) and pressure \( p \), for each rule the antecedents are combined using the \( \min \) operator, producing a membership function on the output variable. These memberships are again combined over all rules, this time using the \( \max \) operator. To determine the value \( thr \) to which the throttle output variable has to be set, the centre of area method is applied on the final fuzzy membership function.

In the example above, the fuzzy sets had triangular membership functions. Other function choices are trapezoidal, Gaussian, B-splines [40] or generalised bell-shapes (compare Figure 3.4).

For practical applications, there exist different integrated processes or frameworks that describe how to build a fuzzy system. They have different flavours for example concerning rule generation. Among the most popular are the intuitive Mamdani system [164], and the Sugeno framework [253], the latter of which is implemented in the widely used ANFIS [237] method. In [140], you will find an example where the idle-running speed of a passenger car motor is controlled and regulated using a fuzzy control system. Fuzzy rules are also used to convert black-box neural networks into rule systems that domain experts can understand [48]. Some works even convert rules into artificial neural networks, optimise these, and then re-extract the fuzzy rules [1]. Also, the answers of crisp decision trees can be fuzzified using fuzzy rules [54].
3.7 Dempster-Shafer Theory of Evidence

The Dempster-Shafer (DS) theory of evidence, also known as the theory of belief functions, is a tool for representing and combining evidence. Being a generalisation 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 imprecision concerning a question can be modelled independently.

Dempster-Shafer theory was brought forward by Dempster [58] and Shafer [233], then came to the attention of artificial intelligence researchers in the early 1980s as an approach to adopt probability theory to expert systems [234], and is still applied in this field [93, 192]. We used it to accumulate belief in a classifier hierarchy and thus better compare the different paths through the tree [75].

The 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 \( 2^\Theta \) denote the set of all subsets of \( \Theta \). 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 be assigned to an atomic hypothesis, but some set \( A = \{a_1, ..., a_n\} \subset \Theta \). Hence, our belief in \( m(A) \) represents our ignorance, which can not be subdivided among the subsets of \( A \), modelling the imprecision in our nomenclature. Each element \( B \) with \( m(B) > 0 \) is called a focal element. Now with a bpa \( m \), the belief function \( \text{bel} : 2^\Theta \rightarrow [0, 1] \) is defined as

\[
\text{bel}(B) = \sum_{A \subseteq B} m(A).
\]

It represents the minimum trust we can have in \( B \) because of the supporting subsets \( A \). Looking at the definition, it can be noticed that \( \text{bel}(\{\theta_k\}) = m(\{\theta_k\}) \forall k = 1 \ldots n \). So, if every focal element is an atomic hypothesis, we find ourselves in the situation of standard probability theory. To get an intuitive understanding, one can consider a basic probability assignment a generalisation of a probability density function and a belief function a generalisation of a probability function [205].

Any belief mass assigned to \( \Theta \), the universal hypothesis or proposition [289], represents the belief that the assignment of this mass to other focal elements may be based on evidence that is not legitimate, thus forming an indicator for the certainty of the assignments.

The most interesting part of the theory is the possibility to combine two basic probability assignments \( m_1 \) and \( m_2 \) on \( \Theta \) with the orthogonal sum \( m_{12} = m_1 \oplus m_2 \)
3.7. Dempster-Shafer Theory of Evidence

which is defined as

\[ m_1 \oplus m_2 = m_{12}(C) = K \sum_{A,B:A \cap B = C} m_1(A) \cdot m_2(B) \] (3.32)

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 [100]. 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

\[ \text{plausability}(A) = 1 - \text{bel}(\neg A) \, . \]

It expresses how much we should believe in \( A \) if all currently unknown facts were to support \( A \), thus forming a trust or knowledge interval \([ \text{bel}, \text{plausability}]\)\textsuperscript{15}.

3.7.1 Example

Let there be a frame of discernment with two atomic hypotheses \( \theta_1 \) and \( \theta_2 \), and based on it three basic probability assignments \( m_1, m_2 \) and \( m_3 \) as shown in Table 3.1.

\textsuperscript{15}When new evidence becomes available, it can be incorporated using Dempster’s rule of conditioning, a special case of the orthogonal sum (see Formula 3.32). Pearl claimed in [184] that this combination could lead for the gap between probability and plausibility functions to “mysteriously disappear, thus giving the false impression that \( Bel(\cdot) \) is based on precise probabilistic information, when such information is in fact not available” (p. 378), illustrating the issue with the 3 prisoners example. In a direct response to this claim, Smets [244] scrutinises the underlying assumptions, and (using the TBM model) comes to a different outcome of the example scenario than Pearl.
Then calculating the orthogonal sum $m_{12} = m_1 \oplus m_2$ is straightforward:

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

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

The normalising factor $K$ is calculated as:

$K^{-1} = m_1(\theta_1) \cdot m_2(\theta_1) + m_1(\theta_2) \cdot m_2(\theta_2) = \sum_{\forall \text{ focal elements } f \in m_{12}} m_{12(\text{non-normalised})}(f) = 0.48 + 0.08 = 0.56.$

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 $\theta_1$ is the most likely, but $\theta_2$ is not ruled out altogether, because there was some belief in it, especially in $m_3$.

### 3.7.2 Practical Applications

The Dempster-Shafer theory of evidence has successfully been applied to several real-world problems. Some of these solutions will be presented in the following (compare also our summary in [261]).

As early as 1988, Mandler and Schürmann [165] developed a classifier fusion system in the domain of on-line script recognition, employing a prototype-based approach. 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 Dempster-Shafer theory. The bpas of three experts were then combined, using the orthogonal sum.

Bloch incorporated deep domain knowledge into a classifier for anti-personnel mine detection [172]. The algorithm answered with Dempster-Shafer belief assignments, based on indicators like the ratio between the areas of an object as detected by the metal detector versus by ground penetrating radar. Le Hégarat-Mascle, Richard, and Ottle [152] employed Dempster-Shafer in a land cover mapping application, to incorporate information from sensors with different spatial and temporal resolution. Many aspects of the framework were applied and evaluated by Côme [52] in the setting of detecting defective stretches of railway tracks.
The aim of Denoeux [59] was to build a single classifier producing basic probability assignments. He used prototypes with fuzzy memberships and adapted them using a multi layer perceptron, optimising a performance criterion. The Dempster-Shafer theory was used to merge the information provided by each of the prototypes. An implementation of the algorithm was published, and we employed it in our rejection-study [261] on the fruit dataset, observing a rather high dependency on the prototype-initialisation.

Dempster-Shafer theory has also been used to enhance established machine learning algorithms. One example is the usage of the orthogonal sum to combine the outputs of One-vs-Rest support vector machines [107], another the weighting of association rules [281]. Fay reports success in reading out decision trees using evidence-theoretic rules [73, 75].

Security-related application scenarios with their multitude of sensors lend themselves quite naturally to the application of the Dempster-Shafer theory. A rising political interest here spurred a new wave of research, in the field of internal security (for example airport threat surveillance [194]) as well as for military purposes (for example source integration to obtain a tactical situation picture [4]).

### 3.7.3 Transferable Belief Model

The transferable belief model (TBM) by Smets and Kennes [248] is an “interpretation not of Dempster’s work but of Shafer’s work [...]. It is a ‘purified’ form of Dempster-Shafer’s model in which any connection with probability concept has been deleted” ([242], p. 1, see also there for a comparison of the TBM and DS theory.). Taking up the idea of discounting [233], in the TBM it is possible to introduce a certainty factor for belief distributions [245]. If one thinks that the evidence underlying the belief distribution $\text{bel}_\Omega$ is reliable to a degree $\alpha \in [0, 1]$\(^{16}\), the distribution would be adapted, or discounted: $\text{bel}_{\Omega}(A) = \alpha \text{ bel}_\Omega(A)$. For an application of the TBM to some constructed real-world problems see [247].

### 3.8 Uncertainty in Databases

Talking about databases, the word uncertainty can have many different meanings. In the context of information management, it is concerned with the quality of the data entered into the database [274]. Related to more technical problems, uncertainty means dealing with ever-changing sensor readings [49], cached objects (divergence, [179]) or being related to streams of data with aggregates maintained

\(^{16}\)In [245], the certainty is expressed as $1 - \alpha$, this notational issue has been changed to fit the conventions in this work.
in different granularity [297]. For a literature overview on these issues, see chapters 2.4f in [109].

Of course, database systems were also built as a realisation of specific uncertainty theories, to prove their usefulness and apply them in bigger scenarios. One example is the DUCK system mentioned in Chapter 3.2 on logic, another the ProbView database by Lakshmanan, Leone, Ross, and Subrahmanian [150] implementing probability theory. The authors of the later give a framework for all necessary operations like updates or projections, encoding probabilities in intervals. The operations themselves however are not fixed, but can be chosen by the user in the limits of the restrictions imposed by the framework. With the ProbView system, an example implementation is presented. Specialised approaches that tackle uncertainty in special dimensions, for example time or space [56], also exist.

Traditionally, databases can only handle precise retrieval requests, but not vague linguistic concepts ones like “recently” or “big”. This has spurred many extensions, for example to SQL [28]. Ma and Meng recently proposed [160] an extension to relational databases where experts construct rule sets consisting of fuzzy sets (see Chapter 3.4) to transform vague user queries. The sets are then converted into precise queries employing alpha-cuts (see Formula 3.16), and the results ranked according to the weight of the attributes used\textsuperscript{17}.

On the level of data warehouses that allow the analyses of data stored in single databases, there is a special need for storing uncertain data and making uncertain queries. This is why Kiewra very recently extended [132] the OLAP (online-analytical-processing) approach to directly store probability values. For each fact (called cube cell) a certainty measure can be declared, indicating the trust in it. Also, the attributes (for example “destination”) of a fact don’t have to be precise values, but can be vague, represented by a probability distribution over the possible values for this attribute (for example \{\textit{(Colombia, 0.8)}, \textit{(Peru, 0.1)}\}). On each attribute, it must be possible to define a hierarchy. As a drawback, union and intersection operations are only possible on facts that have the same certainty. The aggregation operator, which is used to compress the results of queries, calculates the expected values of the output attributes using both probabilistic uncertainty and vagueness.

The topic receives more and more consideration in the database community, exemplified by the upcoming special issue\textsuperscript{18} Mining Large Uncertain and Probabilistic Databases of the IEEE Transactions on Knowledge and Data Engineering, which

\textsuperscript{17}The weight of the features is determined using logs of past user queries. The authors assume that an attribute used frequently in queries is important to all users. The granularity of this approach seems to be too coarse, though.

\textsuperscript{18}The IEEE Transactions on Knowledge and Data Engineering special issue on Mining Large Uncertain and Probabilistic Databases is tentatively set to appear in May 2010. The guest editors are Reynold Cheng, Michael Chau, Minos Garofalakis, and Jeffrey Xu Yu. The call for papers can be found at \url{http://i.cs.hku.hk/~ckcheng/tkde-si/cfp.html}. 
strives to gather research in the field, ranging from uncertainty models to query optimisation under uncertainty.

### 3.9 Uncertainty in Artificial Intelligence

In the field of artificial intelligence (AI), the treatment of uncertainty plays an important role, too. Early expert systems include MYCIN [238] and INFERNO [197], who were built to logically reason under uncertainty assumptions (see also Chapter 3.2). These days, artificial intelligence research is not confined to expert systems, but covers a broad, even fragmented, bundle of directions. One testament to this is the conference on Uncertainty in Artificial Intelligence (UAI), organised each year by the Association of Uncertainty in Artificial Intelligence\(^\text{19}\), which in their 24th edition in 2008 had main sessions as diverse as inference, topic models, games, multi-view and reinforcement learning, and modeling and regression. Other topics in AI include agent systems, and the notion of chance discovery [178], which “is a contemporary direction [...] which analyzes events with uncertain information, incomplete past data, chance events, which are typically rare or hard to find. A chance is defined as some event which is significant for decision-making in a specified domain” ([212], p. 951).

### 3.10 Comparison of Uncertainty Theories

The uncertainty theories presented in this chapter offer a vast pool of choices to model uncertainty. There have been some attempts to compare these theories on a formal level [135], and two of them, the context model and the Umkehrer framework are presented briefly at the end of this section (see Chapters 3.10.1 and 3.10.2). Much literature is devoted to comparing the uncertainty theories with respect to a specific example, or in the light of an exemplary application [30, 78]. We will follow a similar path, concentrating on the task of classification, and showing how different uncertainty frameworks embody the classification-related uncertainty notions of vagueness, imprecision and certainty as set forth in Definition 3.1: Vagueness means that the sample point is associated with different classes to a different degree. With Imprecision, the label of a sample consists of a set of classes. Certainty quantifies how certain the label is, by adding a certainty factor \(c\) to it. The Tables 3.2 to 3.4 detail how these concepts can be represented in the various uncertain theories.

As a short reminder, in the area of classification, a sample point \(x\) is coupled with a class label \(y\). In the most basic setting, we have \(y \in [1 \ldots L]\), that is the sample

\(^{19}\text{http://www.auai.org/} \)
is associated with exactly one of the $L$ classes.

While the concept of imprecision is not especially supported in probability theory or fuzzy sets, it can be emulated by artificially defining sets that include more than one class.

Note that the rough sets of Chapter 3.5 are not included in the tables because the theory is not so much concerned with modelling uncertainty as with identifying uncertain concepts in data.

<table>
<thead>
<tr>
<th></th>
<th><strong>Imprecision</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Probability Theory</td>
<td>Not a standard option, probabilities only given for individual classes.</td>
</tr>
<tr>
<td>Possibility Theory</td>
<td>Possibility can be given to subsets of the reference set $\Omega$, that is also to subsets having more than one element.</td>
</tr>
<tr>
<td>Distr. of Opinions</td>
<td>Not an option in the model.</td>
</tr>
<tr>
<td>Fuzzy Sets</td>
<td>Not a standard option. Theory provides only fuzzy membership to clearly defined sets (that is, sets pertaining to one class).</td>
</tr>
<tr>
<td>Logic</td>
<td>It is possible to model sets of alternatives using disjunctions ($\cup$).</td>
</tr>
<tr>
<td>Fuzzy Logic</td>
<td>It is possible to model sets of alternatives using disjunctions ($\cup$).</td>
</tr>
<tr>
<td>Fuzzy Control</td>
<td>The expert can formulate rules that, in the antecedents, are applicable to samples from different classes. However, being based on fuzzy sets, the consequence of a rule will pertain to a precise class.</td>
</tr>
<tr>
<td>Dempster-Shafer</td>
<td>A basic probability assignment can give belief to an entire set $A$ of atomic hypotheses, $A = {a_1, ..., a_n} \subset \Theta$, thus staying at the level of the imprecise set, not caring about individual $a_i$.</td>
</tr>
</tbody>
</table>

*Table 3.2:* Representing the basic classification-related concepts of imprecision in different uncertainty theories.
### 3.10. Comparison of Uncertainty Theories

#### Vagueness

<table>
<thead>
<tr>
<th>Theory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probability Theory</td>
<td>$y = (p_1, \ldots, p_L)$, $p_i \in [0,1]$, $\sum_{i=1}^{L} p_i = 1$. The standard case. The sample belongs to class $i$ with the probability $p_i$.</td>
</tr>
<tr>
<td>Possibility Theory</td>
<td>Possibility can be assigned to focal elements $\omega$ having only one element, hence a specific class. Thus, the current sample (on which the possibility distribution $\pi$ is based) is associated with different classes to different degrees of possibility (or even necessity).</td>
</tr>
<tr>
<td>Distr. of Opinions</td>
<td>For every class $i$, the expert gives his estimate $x(l_i)$ of the probability that the current object belongs to this class.</td>
</tr>
<tr>
<td>Fuzzy Sets</td>
<td>One sensor reading yields degrees of memberships to all fuzzy sets, each of which is bound to a specific class.</td>
</tr>
<tr>
<td>Logic</td>
<td>Only possible with extensions like modal logic or multi-valued logic.</td>
</tr>
<tr>
<td>Fuzzy Logic</td>
<td>Being a multi-valued logic, fuzzy logic allows to have different degrees of true $\in [0,1]$ pertaining to different classes.</td>
</tr>
<tr>
<td>Fuzzy Control</td>
<td>Possible using so-called MIMO fuzzy models (multi-input multi-output, see for example [8]).</td>
</tr>
<tr>
<td>Dempster-Shafer</td>
<td>Each hypothesis $a_i \in \Theta$ (that is, each class) can be associated with a different degree of belief $b$ in the basic probability assignment: $m(a_i) = b_i$.</td>
</tr>
</tbody>
</table>

**Table 3.3:** Representing the basic classification-related concepts of vagueness in different uncertainty theories.
Chapter 3. Uncertainty Theories

Certainty

| Probability Theory | Possible by attaching a certainty measure $c$ to the label, where $c$ is also a probability, for example the fraction of cases where this label was correct. |
| Possibility Theory | If the certainty $c$ of the process that produced the possibility distribution is known, the distribution itself can be adjusted using Formula 3.17. |
| Distr. of Opinions | Not a standard option. However, the rather general functional updating (see Formula 3.9) could be adapted to include certainty measures associated with opinions. |
| Fuzzy Sets | See possibility theory. |
| Logic | Only possible with extensions, for example to multivalued logic. In the MYCIN-system [238], certainty factors $cf \in [-1, 1]$, can be specified for rules. |
| Fuzzy Logic | Not a standard option, but see fuzzy control. |
| Fuzzy Control | A confidence value $c_k$ can be set for each rule. It will be combined with the rule output using a t-norm. |
| Dempster-Shafer | Any belief assigned to $\Theta$, the universal hypothesis, is an indicator for doubt in the basic probability assignment, hence a measure for its certainty (complete trust indicated by $m(\Theta) = 0$). In the transferable belief model variant, the doubt can be induced via discounting. |

Table 3.4: Representing the basic classification-related concepts of certainty in different uncertainty theories.
3.10. Comparison of Uncertainty Theories

3.10.1 Context Model

In their context model [83], Gebhard and Kruse established a framework to model vagueness\(^{20}\) and certainty. Its primary purpose was to allow the formal comparison of Bayes and Dempster-Shafer theory. The centrepiece is the concept of vague characteristics \(\Gamma_c\) that associate a context \(c\) with a set of imprecise characterisations of the object under consideration, the characterisations being in the domain of the object (for example, readings of each of its features). Each context can be associated with a relevance measure \(P_c\), leading to a so-called valuated vague characteristic. In the classification setting (which is not an application where the use of the framework suggests itself), a context can be considered to correspond to the occasion of a real-world sensor reading of the object, while the class would be encoded in the vague characterisation, simply being a variable in the domain. The framework also defines acceptance degrees (corresponding to necessity and possibility measurements) and defines operations on the vague characteristics, like combination or mass distribution. It is briefly mentioned how Bayes and Dempster-Shafer theory can be represented in the framework, but the most interesting part is where the authors compare how the well-known unreliable alarm paradigm is handled by the three frameworks.

3.10.2 Umkehrer Framework for Comparing Numerical Uncertainty Theories

The goal of Umkehrer in her dissertation [267] was to compare different aspects of probability theory, Dempster-Shafer theory and fuzzy set theory (compare Chapters 3.1, 3.7 and 3.4). Coming from a knowledge-based systems perspective, she first analyses how knowledge is presented. As basic element, she finds the notion of distinction\(^{21}\), which divides a space into two parts, establishing borders. A portion of the space that is not subdivided by any border can be called atomar. A distinction system allows borders to be set, and subjects to decide for one side of it or the other. Interestingly, not all of these borders must have a name associated, but those that can and have are called concepts. The representation of knowledge is seen as the sum of messages about the state of a distinction system. A probability value is defined over the cardinality of atomar parts involved.

All these concepts are then codified in an extensive four-layered formal framework. The structures of the three uncertainty theories mentioned above are then presented, together with their current interpretation in the literature, and each

---

\(^{20}\)Their concept of vagueness arises as a mixture of imprecision and conflicting sources.

\(^{21}\)For readability and clarity, the concepts Umkehrer introduced are not referred to with their original German name, but their English equivalents. A rough sketch of the basic ideas, also of the roots of the formalism, was presented years earlier in an article of hers [268] in English language.
reconstructed using the newly created framework and its axioms. This allows to come to numerous new interpretations of the theories, which are summed up relating to the topics of allowed concepts (or facts), of ways to modify the knowledge, and of types of messages. For the last, Umkehrer finds for example that probability theory and fuzzy set theory are both founded on probability statements, while Dempster-Shafer makes attribution statements. As a coarse summary, probability theory is found to be the most natural one, its statements being closest to the structure of facts, while Dempster-Shafer needs a structuring of knowledge first.
Classifiers

In land cover mapping, a single pixel on the ground may contain both a street and forest. In emotion recognition, somebody might be scared and excited at the same time. Human experts do not agree on a subject, or are not sure about their decision. The various aspects of uncertainty arising in these applications can well be expressed using the notions of fuzzy, imprecise and uncertain labels introduced in Chapter 3, or modelled with any of the uncertainty theories presented there. But, these theories do not directly propose any classifiers that can work based on uncertain labels.

In this chapter, the inclusion of uncertainty into the classification process will be discussed, both in the training as well as the output phase. The basic principles of several well-known classifiers will be modified to accommodate the idea of uncertain labels. The most notable example here are the support vector machines, which will be extended to deal with fuzzy inputs and outputs. The behaviour of each new uncertain classifier is evaluated experimentally in at least one real-world application.

Apart from the genre of applications mentioned above, uncertain labels are a cornerstone of the field of semi-supervised learning. The basic assumption there is that data samples are plenty, but labels for them are scarce, since difficult or expensive to obtain. In traditional classification algorithms, samples without labels are disregarded. Not so in semi-supervised learning, where the goal is to use these unlabelled data points to improve classification accuracy, by learning about the distribution of samples in feature space. An early work for example draws from the idea of transductive inference [119], essentially testing if switching a guessed label increases the target function. In another basic approach [176], the certainty (“credit”) of non-expert-labelled data is degraded based on the distance to expert-labelled samples. The approach of co-training [26] entangles two classifiers, each is allowed to label new samples for the training set of the respective other classifier. A combination with support vector machines, and the technique of expectation maximization [57] leads to Co-EM support vector learning [35], raising the weight of the newly labelled samples in the optimisation.
problem with every iteration. Recently, Didaci and Roli justified the usage of co-training in multi classifier systems [60]. One concrete application is the work by Bruzzone and Marconcini [41], where the weight of a guessed label concerning a ground pixel depends on how many iterations it has been stable. Somewhat related, in [86] we use a SVM-produced mapping of the area as groundtruth for testing further algorithms.

Another research field that concerns itself with fuzzy labels is called learning with multiple labels, where each data sample can be associated with multiple classes at once\footnote{Ironically, as an exception, in the publication “Learning with Multiple Labels” [117], it is assumed that there is only one true class for each sample.}. Publications mainly concern the application of text categorisation, a literature review can be found in [298].

The concept of a classifier shall shortly be introduced, to familiarise the reader with the notation employed in this work. A classifier is an algorithm that takes a feature representation $x$ of an object or concept, a so-called sample or data point, and maps it to one or more classes, the mapping result being encoded in the classification answer $y$:

$$\text{classifier} : x \in \mathbb{R}^N \rightarrow y$$

In the case of supervised classification, a classifier generally has to be trained, that is presented with a set of $M$ training samples that each have the correct class label $y_\mu$ associated:

$$\text{training set} : (x_\mu, y_\mu), \  \mu = 1, \ldots, M$$

A trained classifier can then classify a hitherto unseen sample, for example say if the person on an image looks happy.

The training label $y$ of a sample, and hence also the classifier’s answer, can take different forms, depending on the type of uncertainty encoded in it, as detailed in the basic Definition 3.1. For example in the hard case, exactly one of the $L$ classes is correct for each sample, so $y \in \{1, \ldots, L\}$. In the fuzzy or soft case, each sample can be associated with different classes to different degrees, represented by $y = (y_1, \ldots, y_L)$, $y_i \in [0, 1]$. If the training set carries soft labels, we speak of a fuzzy-input classifier, and accordingly use the term fuzzy-output for classifiers that answer with fuzzy labels. In these cases, with $y_i$ representing how strong the classifier estimates that the sample belongs to class $i$, a label normalisation is often desireable, ensuring that $y_i \in [0, 1]$ and $\sum_{i=1}^{L} y_i = 1$. The normalised answer $y_{\text{norm}}$ is calculated as

$$y_{i\text{norm}} = \frac{y_i}{\sum_{i=1}^{L} y_i}.$$
Of course, not all classifiers’ answers can be considered being estimates of posterior distributions, and some might even give negative values. These cases, for example occurring in RBF networks, require special treatment.

As mentioned, each sample $x$ is represented by a $N$-dimensional feature vector. To ensure that not one or few dimensions carry an excessive weight in the classification process, each dimension is normalised to have a mean value of 0 and standard deviation of 1. In cross-validation experiments, the normalisation parameters are of course calculated based only on the training set. Ensuring that just relevant features are included in the sample description vector is an art and discipline in itself, known as feature selection. For an introduction, see the chapter on “feature selection and extraction” in [275]. A study on feature ranking in multi classifier settings has recently been conducted by Windeatt and Dias [284]. We also briefly report on experiences with feature search algorithms in Chapter 2.1.5.

This chapter is mainly structured around classifier algorithms and their ability to deal with uncertainty, with experiments directly following the description of each approach. The first classifier described is the established RBF network (Chapter 4.1), which works well for most problems. Included are some notes on optimising the kernel width. Our implementation of the very simple but sometimes surprisingly well performing fuzzy k-nearest neighbour classifier is presented in Chapter 4.2. The prototype-based learning vector quantisation algorithm (Chapter 4.3) is studied more in detail. With the proposed extension, its prototypes now carry fuzzy labels, and in the training phase can change location depending on label-similarities. Adding noise on the training labels, interesting effects could be observed. The fuzzy labelled centres can be used to convey uncertainty in classification results in situations where the application or expert explicitly requires prototypes. Self organizing maps (Chapter 4.4) are also prototype-based, but unsupervised. An approach to produce fuzzy classifications is presented, along with a more detailed analysis of the fuzzy answers arising in the land cover mapping setting, and a comparison with previously published fuzzy-output procedures. The most important contribution is in Chapter 4.5: fuzzy-input fuzzy-output support vector machines ($F^2$-SVMs). SVMs are large-margin classifiers that operate not in feature- but a higher-dimensional kernel-space, and can achieve good classification results on problems where other classifiers fail. The machines were expanded so that they can accept fuzzy training samples, and give fuzzy answers. We studied different possibilities for producing fuzzy outputs in the two major multi-class architectures. Each aspect of the $F^2$-SVM procedure is evaluated through experiments. Further trials with a reduced training set suggest a method to considerably speed up the initial phase. An extension to the special case of one-class SVMs is proposed. Getting more general, in Chapter 4.6 ideas are presented how a certainty value affixed to a training label could be incorporated into various classification algorithms. One of the promises of fuzzy-trained classifiers
is their heightened resilience against noise or errors on the training labels. This property is explored and experimentally validated in Chapter 4.7. A short note on statistical methods for measuring classifier performance concludes (Chapter 4.8).

4.1 Radial Basis Function Networks

The theoretical basis of RBF networks lies in the field of the interpolation of multivariate functions [193]. They are often used for function approximation, performing nonparametric regression, but can also be employed for the task of classification [159, 190]. RBF networks are already able to function as fuzzy-input fuzzy-output classifiers without requiring special changes, and have been used quite extensively as part of the experimental setup in many of our publications [258, 229, 75, 261, 259].

![Figure 4.1: Concept of a RBF network.](image)

Reducing the problem of interpolation to that of the approximation of class probabilities lowers the number of required basis functions and allows the network to be modelled as a neural network [39]. This consists of one hidden layer with $k$ basis functions, or neurons. At the input of each neuron, the distance $d_j$ (Euclidean in most applications, also here) between the neuron centre $c_j$ 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 $\phi$ of the RBF network is formed by a weighted ($w$) sum of the neuron answers:

$$\phi(x) = \sum_{j=1}^{k} w_j h_j(d_j)$$
4.1. Radial Basis Function Networks

For the task of classification, there is one network output unit \( \phi_i(x) \) for each of the \( L \) classes to be learned. The function to be approximated is simply the one that maps each input vector \( x \) to its corresponding \( L \)-dimensional training-label \( y \), where each component of \( y \) is approximated by the corresponding output node \( \phi_i \).

The radial basis function used in our experiments is the commonly employed standard Gaussian\(^2\).

\[
h_j(d_j) = \exp\left(-\frac{d_j^2}{2\sigma_j^2}\right) \frac{1}{\sqrt{2\pi\sigma_j^2}}
\]

A training of the network is accomplished with so-called two-phase learning [230]. In the first phase, the centres \( c_j \) and the scaling or width parameters \( \sigma_j \) are learned and fixed. The second phase aims to find the optimal combination weights \( w_j \).

To obtain centres for the Gaussian function, often unsupervised clustering methods are used [173]. In case of hard training labels, we employ the \( k \)-means clustering algorithm [161] to produce \( k \) prototypes per class by applying it consecutively to training data with samples only from that class, totalling in the \( k \) prototypes (=centres) for the network. As an estimate for the \( \sigma_j \) of each centre, the mean Euclidean distance of the centre to every sample in its cluster\(^3\) can be used. If the training labels are vague, \( k^4 \) cluster centres are determined by applying \( k \)-means to the entire training set. The width of each centre is then optimised using the more elaborate Breiman procedure described in the short extra section below.

In the second phase, the combination weights of each network output unit are determined [102] in one pass using the pseudoinverse solution [39], which is based on the overdetermined least-square criterion: Let \((x_\mu, y_\mu), \mu = 1, \ldots, m\), be the set of \( m \) training samples with feature vector \( x_\mu \) and the corresponding class labels \( y_\mu \in [0,1]^L \). And let \( H_{\mu,j} \) be the outcome of the \( j \)th basis function with the \( \mu \)th feature vector \( x_\mu \) as input, and \( Y_{\mu,i} \) the \( i \)th component of the \( \mu \)th training label vector \( y_\mu \). Then the desired approximation can be formulated as follows (effectively finding parameters for \( L \) functions \( \phi_i(x) \)):

\[
Y = HW
\]

The matrix \( W \) consists of the output weights \( w \), one column with \( k \) weights for each output unit\(^5\). The optimal \( W \) is now the minimum of the error function

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

\(^2\)Other choices for basis functions as suggested by Bishop [24] are the thin-plate spline function \( h(d) = d^2 \ln d \) and \( h(d) = (d^2 + \sigma^2)^{-\alpha}, \ \alpha > 0 \).

\(^3\)That is, those samples nearer to this prototype than any other.

\(^4\)For our implementation, we used a heuristic formula to set the number of clusters: \( k = (m/30) + (5L) \).

\(^5\)The dimensions of the matrices hence are: \( H: m \times k, W: k \times L, Y: m \times L \).
to which the solution is explicitly given [230] in the form of $W = H^+ Y$, where $H^+$ denotes the pseudo inverse matrix of $H$ which is defined as

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

(4.2)

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 $x_{\text{new}}$, it has to be fed to the $k$ basis functions, assembling the answers into a vector

$$A = (h_1(\|x_{\text{new}} - c_1\|), h_2(\|x_{\text{new}} - c_2\|), \ldots, h_k(\|x_{\text{new}} - c_k\|)).$$

The final (soft) classification $y_{\text{new}}$ of the sample is calculated as simple as $y_{\text{new}} = AW$.

The output of a RBF network trained on hard labels can be interpreted as being proportional to the a posteriori probability that the input belongs to the corresponding class ([24], p. 181). Network outputs can be normalised to sum up to one to facilitate integration with other classifiers’ results. The RBF-answer to new samples can in rare cases have negative entries; these are set to 0 by us before normalisation.

RBF networks are well understood classifiers that perform well. In our experiments, their accuracy was somewhat lower than that of SVMs. This can for example be seen when comparing some results from the Cohn-Kanade facial expression experiment given in Table 4.1: RBF clearly beats KNN and MLP, but always lags behind the SVMs. However, these require a more complex treatment of multi-class cases.

<table>
<thead>
<tr>
<th>Face region: mouth</th>
<th>KNN</th>
<th>MLP</th>
<th>RBF</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge orientation histograms</td>
<td>62.2</td>
<td>58.3</td>
<td>68.9</td>
<td>74.0</td>
</tr>
<tr>
<td>Optical flow</td>
<td>54.8</td>
<td>18.2</td>
<td>55.5</td>
<td>60.8</td>
</tr>
</tbody>
</table>

**Table 4.1:** Comparing the accuracy of $k$-nearest neighbour, multi-layer perceptron, RBF network, and support vector machine classifiers on the Cohn-Kanade facial emotions dataset (see Chapter 2.1). Results are given for the two strongest features on the strongest face region, the mouth. The experiment employed 8-fold cross validation, the answers of all classifiers here are based on the second half of each time series only, and improved significantly when using the data driven approach for time series fusion described in Chapter 2.1.6. Detailed results for more classifier/region combinations can be found in [217]. The accuracy is given in percent, measured against the hardened vague training labels.
4.1. Radial Basis Function Networks

Optimising the Kernel Width

Once the number of prototypes or kernels has been fixed, the most critical parameter to be set in the RBF network is the width $h$ of the kernels [24]. If these are too wide, the resulting network is too smooth and the underlying distribution in the data is lost. Being too small, the kernels would model a function that only has sharp peaks at the location of the prototypes, and a flat surface elsewhere. For an overview of methods for estimating the width, see [122, 275], or in more condensed form in [230]. In many of the approaches, the width parameter is the same for all kernels. But in fact, it is often more appropriate to set an own $h_i$ separately for each radial basis function, for example if there are sparsely populated regions in the input space where broad kernels are needed. It is immediately clear why taking the intra-cluster distances in account when setting the individual kernel widths is necessary. Another way to obtain variable kernel widths that is often cited is the one introduced by Breiman in [38]. Because it has only one parameter to be optimised, it is rather fast and requires only a limited amount of additional computation. This is why we adopted it in our implementation of RBF networks.

Breiman worked with radial basis functions $f_i$ of the general form

$$
f_i(x) = \frac{1}{(\alpha_k d_{i,k})^M} K \left( \frac{x - c_i}{\alpha_k d_{i,k}} \right)
$$

with a kernel $K$ and dimensionality $M$ (which we set to 2 in the following). As usual, $c_i$ is the centre of kernel $i$, whereas $d_{i,k}$ is the distance from $c_i$ to its $k$th nearest neighbour in the training set. In his experiments, Breiman ([38], p. 136) finds that “good fits can be produced over a very wide range of values $k$” as long as the free parameter $\alpha_k$ satisfies the relation

$$
\frac{\alpha_k d_k^2}{\sigma(d_k)} = \beta = \text{const.}
$$

where $d_k$ is the mean of the $k$th nearest neighbour distances and $\sigma(d_k)$ their standard deviation. Therefore, it is enough to tune a single parameter $\beta$ for the entire net to get near optimal results. Note that each single kernel $f_i$ will still have an individual width depending on its $d_{i,k}$.

Using $n$ Gaussian kernels, the RBF network $f$ we used looks like this:

$$
f(x) = \sum_{i=1}^{n} \frac{1}{h_i \sqrt{2\pi}} \exp \left\{ \frac{-(x - c_i)^2}{2h_i^2} \right\}, \quad h_i = \beta \frac{\sigma(d_k)}{d_k^2} d_{i,k}
$$

For a very simple optimisation in search of $\beta$, it is sufficient to assess the performance of the network on the labelled training data, trying out appropriate
In our own experiments, good values proved to be $a = 5$ and $b = 80$ across different data sets. The choice of $k$ for the nearest neighbours is not critical, and can be set to any sensible value like $\frac{1}{4}$ of the number of training samples of each kernel-cluster.\footnote{See Footnote 3.}

### 4.2 Fuzzy-Input Fuzzy-Output K-Nearest Neighbour

The *k-nearest neighbour algorithm* (KNN, \cite{82}) is used to classify feature vectors. Being simple, elegant and straightforward, it is among the standard set of classifiers employed in this work. The instance-based classifier operates by identifying the $k$ training samples closest to a new sample, its *neighbours*, and assigning it to the class that most neighbours bear. The Euclidean distance is used to determine neighbourhoods.

A fuzzy-output $k$-nearest neighbour classifier \cite{239} works by giving each of the $k$ neighbours an influence in labelling a new sample, depending on the distance to it. Let $N_j^k(x)$ be the set of those points among the $k$ training samples closest to $x$ that have the hard label $j \in [1, \ldots, L]$. The support for the hypothesis that $j$ is the true class label of $x$ is now:

$$
\delta_j(x) = \sum_{n_i \in N_j^k(x)} \frac{1}{\|x - n_i\|_2}
$$

If $N_j^k(x)$ is empty, then $\delta_j(x)$ is set to zero. Subsequent normalisation of these values produces a standard vague label.

The fuzzy-input fuzzy-output KNN, or *weighted KNN* \cite{256}, is a straightforward extension of the concept. The answer is simply a weighted sum of the labels which the nearest $k$ neighbours carry, the weight $w$ being given by the following Gaussian function:

$$
w_i(x) = \exp \left(-\frac{\|x - n_i\|^2}{\sigma^2}\right), \quad n_i \in N_k(x), \quad i = 1, \ldots, k \tag{4.3}
$$

The spread $\sigma^2$ of the Gaussian is set for each sample in relation to the median of all its $k$ neighbour distances.\footnote{The value of $\sigma^2$ for a specific (test) sample equals 0.1 multiplied by the median distance to the $k$ nearest training samples. For the fuzzy-input fuzzy-output algorithm, all distances are normalised by dividing through the dimensionality of a sample.}

There is no universal rule for the size of $k$ (\cite{275}, p. 104). A larger number increases the robustness of the algorithm, but also computation time. Cross
validation experiments could be used to determine the optimal number. In our experiments, we used $k = 200$ in land cover mapping (Chapter 2.4, Salerno dataset), $k = 5$ in emotion recognition from speech (Chapter 2.2), and $k \in \{1, 3, 5\}$ for cricket song classification [213].

### 4.3 Fuzzy-Input Fuzzy-Output Learning Vector Quantisation

Prototype based classification is popular because an expert familiar with the data can look at the representatives found and understand why they might be typical. We wanted to make this advantage available for classification problems with uncertain labels, as current approaches require the training data to be hard labelled. Thus, we decided to take the well-known learning vector quantisation (LVQ) approach and enhance it with the ability to work with soft labels, both as training data and for the prototypes. The algorithm we devised and our experimental results are also published in [262] and the Diplomarbeit of Britta Sonntag [249].

A related approach is the soft nearest prototype classification [231] with its extension in [271], where prototypes also are assigned fuzzy labels in the training process. However, the training data is still hard labelled. Note that also some variants of fuzzy LVQ exist, for example from Karayiannis and Bezdek [128] or those in [287], that have a hard labelling of the prototypes and a soft neighbourhood function, which is exactly the opposite to our approach in this respect.

In the following, first the basic principles of fuzzy LVQ are given, then the details on how to change the prototype locations and labels. After describing the experimental setup, we present conclusions on the different fuzzy LVQ variants examined.

#### 4.3.1 Basic Principles

Like the original learning vector quantisation approach proposed by Kohonen [137], we employ $r$ prototypes, and learn their locations. However, our fuzzy LVQ does not work with hard labels, but vague ones (see Definition 3.1), that is a training set of pairs $(x_\mu, y_\mu)$, $\mu = 1 \ldots M$ with $M$ samples. We assume that for each of the labels $y_\mu \in [0, 1]^L$ the condition $\sum_{i=1}^L y_{\mu,i} = 1$ holds.

As second major difference to the original LVQ network, each prototype $p^\eta$ (located in the feature space $\mathbb{R}^N$) is associated with a soft class label $l^\eta$, which will influence the degree to which the prototypes (and their labels) are adapted to the data during training.
Note that the techniques presented can be applied to various prototype based learning procedures, for example LVQ3 or OLVQ.

Feeding a hitherto unseen sample \( x_{\text{new}} \) to the trained algorithm, we get a response \( y_{\text{new}} \) estimating a soft class label by identifying the prototype \( p^* \) closest to \( x_{\text{new}} \) (we employed the Euclidean distance) and responding with its learned label \( l^* \). The training of the network is described in the following.

### 4.3.2 Adapting the Prototypes

The training procedure of LVQ adapts the locations of the prototypes. For each training sample, the nearest prototype \( p^* \) is determined. If the label \( l^* \) of the winner \( p^* \) matches the label \( y_\mu \) of the presented sample \( x_\mu \), the prototype is shifted into the direction of the sample. If they do not match, the prototype is shifted away from the sample. This is done for multiple iterations.

Working with vague labels, the similarity between two labels should no longer be measured in a binary manner. Hence, we no longer speak of a match, but of a similarity \( S \) between labels. Presenting a training sample \( x_\mu \), the update rule for our fuzzy LVQ is:

\[
p^* = p^* + \theta \cdot (S(y_\mu, l^*) - e) \cdot (x_\mu - p^*)
\]  \hspace{1cm} (4.4)

Still, the prototype is shifted to or away from the sample. The direction now is determined by the similarity \( S \) of their labels: if \( S \) is higher than the preset threshold \( e \), \( p^* \) is shifted towards \( x_\mu \), otherwise away from it. The learning rate \( \theta > 0 \) controls the influence of each update step.

As similarity measure \( S \) between two fuzzy labels, we opted to use alternatively the simple scalar product (component wise multiplication, then summing up) and the \( S_1 \) measure introduced by Prade in 1980 [63] and made popular again by Kuncheva (for example in [146]):

\[
S_1(l^a, l^b) = \frac{||l^a \cap l^b||}{||l^a \cup l^b||} = \frac{\sum_{i=1}^{L} \min(l^a_i, l^b_i)}{\sum_{i=1}^{L} \max(l^a_i, l^b_i)}
\]  \hspace{1cm} (4.5)

### 4.3.3 Adapting the Labels

The labels of the prototypes do not have to stay fixed. In the learning process, each winner’s label \( l^* \) can be updated according to how close the training sample \( x_\mu \) is. This is accomplished with the following update rule:

\[
l^* = l^* + \theta \cdot (y_\mu - l^*) \cdot \exp\left( -\frac{||x_\mu - p^*||^2}{\sigma} \right)
\]  \hspace{1cm} (4.6)
4.3. \(F^2\) Learning Vector Quantisation

The scaling parameter \(\sigma > 0\) of the exponential function is set to the mean Euclidean distance of all prototypes to all training samples. Again, \(\theta > 0\) is the learning rate.

4.3.4 Experimental Setup

The purpose of our experiments was twofold: Assessing the classification power of fuzzy LVQ, and its ability to stand up to noise added to the labels of its training data (compare Chapter 4.7), all in comparison with standard LVQ1. The results described here were published in [262].

As test bed, we employed the fruit recognition application described in Chapter 2.3, adding the mean colour information in HSV space as fifth feature. The fruits data set originally only has hard labels (a lemon is quite clearly a lemon). As soft labels were required for our experiments, the original ones had to be fuzzified, which we accomplished using two different methods, k-means and Keller. In the fuzzy k-means [161] approach, we clustered the data based on the APQBW feature, then assigned a label to each cluster centre according to the hard labels of the samples associated with it to varying degrees. Then, each sample was assigned a new soft label as a sum of products of its cluster memberships with the centres’ labels. The fuzzifier parameter of the fuzzy k-means algorithm, which controls the smoothness or entropy of the resulting labels, was chosen by hand so that the correspondence between hardened new labels and original ones was around 70%. The number of clusters was set to 35. In the second approach, based on work by Keller [130], the fuzziness of a new soft label \(y^\text{fuzzy}_\mu\) can be controlled very straightforwardly, and using a mixture parameter \(\alpha > 0.5\) it can be ensured that the hardened \(y^\text{fuzzy}_\mu\) is the same as the original class \(c_{\text{orig}}\):

\[
y^\text{fuzzy}_\mu, i = \begin{cases} 
\alpha + (1 - \alpha) \cdot \frac{n_i}{k}, & \text{if } i = c_{\text{orig}} \\
(1 - \alpha) \cdot \frac{n_i}{k}, & \text{otherwise}
\end{cases}
\]

The fraction counts what portion of the \(k\) nearest neighbours are of class \(i\). In our experiments, \(\alpha = 0.51\) and \(k = 5\) were used.

All results were obtained using 5-fold cross validation, our accuracy measure compares the hardened answer label with the original hard one.

Finding a suitable value for the threshold \(e\) in Formula 4.4, which controls at what similarity levels winning prototypes are attracted to or driven away from samples, is not straightforward. We looked at intra- versus inter-class similarity values\(^8\), whose distributions of course overlap, but in our case seemed to allow a good distinction at \(e = 0.5\).

---

\(^8\)Intra- and inter-class determined with respect to the original hard labels.
The 35 starting prototypes for the LVQ algorithms were selected randomly from the training data. With a learning rate of $\theta = 0.1$ we performed 20 iterations. One iteration means that all training samples were presented to the network and prototypes and labels adjusted. The experiments were run in online mode, adjusting after every single presentation of a sample.

As mentioned, we also wanted to study the impact of noise on the training labels on the classification accuracy. To be able to compare the noise level on hard and fuzzy labels, noise was simply added to the hard labels, imparting corresponding noise levels to the soft labels derived from the noised hard ones. The procedure for adding noise to the hard labels consisted simply of randomly choosing a fraction (0% to 100%, the noise level) of the training labels, and randomly flipping their individual label to a different class (see also Chapter 4.7 on the topic of label noise).

### 4.3.5 Results and Observations

The experiments on the five different features give rather clear answers. The original LVQ1 approach performs best (see Figure 4.2 at noise level 0). This had to be expected, since the correspondence between hardened new fuzzy labels and original ones is only around 70%, giving LVQ1 more accurate information to train with. Then, depending on the feature, at a noise level between 10% and 30%, its accuracy drops rather sharply, and one of the fuzzy LVQ approaches wins (see Figure 4.2). Right after LVQ1 drops, the fuzzy approach with the Keller-initialisation and scalar product as similarity measure remains most stable. Adding more noise, the approach initialised with k-means and similarity measure $S_1$ now becomes the clear winner with the highest classification accuracy. This means that once there is a not insignificant level of noise on the (training) labels, a fuzzy LVQ approach is to be preferred over the basic hard LVQ one.

The poor performance of the fuzzy LVQ with Keller fuzzification and $S_1$ similarity measure has a simple explanation: setting the Keller mixing parameter $\alpha$ to 0.5 and $k := 5$, the possible values for intra- [0.34,1] and inter-class similarities [0,0.96] overlap largely. A similar explanation, experimentally obtained this time, holds for k-means fuzzification and the scalar product as distance measure. Initially, the same accuracy as with the $S_1$ measure is achieved, but this does not hold once noise is added.

For higher noise levels, the winning approach is a fuzzy LVQ with k-means label fuzzification, and $S_1$ similarity measure. This shows nicely the effect we were hoping to achieve with the process of fuzzifying the labels; the clustering of the training data, and usage of neighbouring labels for the soft labels, encodes knowledge about the label space into the labels itself. Knowledge, which can then be exploited by the fuzzy LVQ training procedure.
4.3. $F^2$ Learning Vector Quantisation

Figure 4.2: Plotting the performance (accuracy given in %) of the original LVQ1 algorithm and our fuzzy LVQ approach once adding noise (noise level given in %). The fuzzy LVQ is plotted with different fuzzy label initialisation techniques (k-means KM, Keller KE) and distance measures on the labels ($S_1$, scalar product SP). Results given for two features, APQBW and Canny, for the other three see Figure 4.3 in [249].
Figure 4.3: Showing how much the location of the prototypes changes (y-axis) from iteration to iteration (x-axis, 20 training epochs), depending on how much noise is present (0% to 100%). Location change is the sum of squared Euclidean distances between the prototypes’ positions before and after each iteration, logarithmised to base 10. Plots are given for the Canny feature, and two fuzzy LVQ algorithm variants: k-means coupled with $S_1$ and Keller coupled with scalar product.
Examining how the labels of the prototypes change from iteration to iteration of the fuzzy LVQ procedure, we find that they do not move much after the first 10 rounds. All in all, the labels remain more stable than we expected and hoped. Even for the first few iterations, the mean Euclidean distance between the prototype-labels from the previous iteration and the corresponding updated ones is typically below 0.1.

Looking closer at how the locations of the prototypes change across iterations (Figure 4.3), we can make an interesting observation. If no noise is added, the movement of the prototypes goes down continuously with each iteration. But as soon as we add noise to the labels, the situation changes. The tendency of the volume of the movement is not so clear any more, for some algorithms it will even go up after some iterations before going down again. Reaching a noise level of 40 to 60 percent, the trend even reverses, and the movements get bigger with each iteration, not stabilising any more. The only (positive) exception here is the fuzzy LVQ with k-means initialisation and $S_1$ as similarity measure, which explains why it performs best of all variants on high noise levels.

The non-settling of the prototype locations also solves the question why, at a noise level of 100%, the original LVQ has an accuracy of 14% (see Figure 4.2), which is tantamount to a random guess. It turned out that the cloud of prototypes shifts far away from the cloud of samples, in the end forming a circle around the samples. One random but fixed prototype is now the closest to all the samples, leading to the effect described.

### 4.4 Self-Organizing Maps

The self-organizing maps (SOM) of Kohonen [137] are another prototype-based classification technique that originally only gives hard answers. The training stage that builds the prototypes is unsupervised, not needing to rely on training labels. This feature makes it very useful and popular in remote sensing applications ([276, 196], see also Chapter 2.4) where an abundance of data meets a scarcity of labelled data, the ground-truth needing to be produced by expert photointerpreters. Also, the prototypes produced mark a certain point in the feature space, allowing conclusions about the relevance of certain features for certain land cover classes.

Classification using SOM is a two-stage procedure. First, using the algorithm, a clustering of the feature space is produced based on the available (unlabelled) data. The clustering produced by SOMs is special in that the prototypes are tied to a lattice, with neighbouring prototypes on the lattice also being close in the feature space. In the second step, the cluster centres receive a label, using a small amount of labelled data. To classify a new sample, simply the closest prototype
is determined, and its label given as the answer.

The step of obtaining a label of the prototype is where, as a fruit of the collaboration with Ferdinando Giacco, a fuzzy labelling procedure was developed. In the most basic approach, a hard prototype-label is produced by taking the majority vote among the labelled samples in this cluster. This has already been generalised by Li and Eastman [154], producing soft labels proportional to all the labels in the cluster (not just the majority). But still, each sample falling into a certain cluster will obtain the same label.

This issue is healed by our proposed procedure [87]. The soft label $y$ for a new sample $x$, which falls into cluster $k$, is a weighted mean of the labels of all the (labelled) pixels falling into this cluster. Concerning class $l \in \{1, \ldots, L\}$, it is hence given by

$$y_k^l = \frac{\sum_{j \in S(k,l)} w_j^l(x)}{\sum_{d=1}^L \sum_{j \in S(k,d)} w_j^d(x)},$$

where $S(k, l)$ is simply the set of pixels falling into cluster $k$ that belong to class $l$. The weights for each labelled sample $z_j^l \in S(k, l)$ are calculated as

$$w_j^l(x) = \frac{1}{N_l} \exp \left( -\frac{\|x - z_j^l\|^2}{2\sigma_k^2} \right). \quad (4.7)$$

The multiplicative factor adjusts the sums for the fact that there is a different
4.4. Self-Organizing Maps

number \( N_i \) of pixels associated with each class in the labelled data. As spread \( \sigma_i \), we chose the mean squared Euclidean distance between the prototype and all labelled samples in its cluster.

The problem of unclassified nodes in the SOM lattice still persists. The advantage of the unsupervised approach is that only few labelled training samples are necessary. However, it may happen that there remain nodes that can not be associated with any (or too few) labelled samples, hence they have to remain unclassified. For a closer look into this issue, see our conclusions in [86]. An important finding there is that unclassified nodes correspond to areas that contain mixed pixels, that is pixels that can not be associated mainly with a single class.

For ideas on directly integrating fuzzy labels into the SOM training stage, turn to the works of Villmann, for example [272].

We did some experiments to evaluate our algorithm [87], with land cover mapping as example application. The data employed pertains to the gulf of Salerno, and is described in detail in Chapter 2.4. As comparison, we employed support vector machine classifiers, both in the One-vs-Rest (sigmoid output, batch gradient descent optimisation) and One-vs-One case (coupled sigmoid output), for linear and RBF kernels (see Chapter 4.5 for details). Accuracy is measured by hardening the soft classifier answers and comparing them with the (hard) labelled portion of the training set.

![Figure 4.5: Accuracy calculated on not-rejected pixels against an increasing rejection rate. Plotted are four SVM variants and the two SOM alternatives.](image_url)

The central idea of the experiment is to reject pixels for which the associated soft answer label is too fuzzy, that is where the maximum membership value over all classes stays below a certain threshold. The result is shown in Figure 4.5 which plots the behaviour of the accuracy of the not-rejected pixels against an
increasing rejection rate\textsuperscript{9}. Without rejection, the two SOM approaches exhibit very competitive classification accuracy, only beat by the linear One-vs-One SVM. Looking at the rejection curves, it is striking to see that the curve for the fuzzy-output approach of Li and Eastman has such large jumps. This means that their soft labels tend to be rather similar, being distributed in clusters of closely resembling labels. Thus, a slight increase of the threshold leads to the rejection of a whole group of samples. The rather smooth curve for our approach testifies that it answers with well nuanced soft labels.

Figure 4.6: The fraction of retained pixels plotted against an increasing rejection threshold, for each label group separately. Note that the lowest maximum membership which the SOM assigns is 0.48.

Figure 4.6 offers a glimpse at the nature of the fuzzy labels produced. It groups the labels according to their hardened output label, and for each group plots the percentage of rejected pixels against a rising threshold. At first glance, it is obvious that the curves for different classes vary largely\textsuperscript{10}, which in conclusion calls for class-specific rejection thresholds. The curves obviously begin to react at certain values of the threshold, for example the class of not vegetated land has two sharp declines around 0.6 and 0.95, which means that the pixels can be divided into two subgroups, assigned to different nodes, and probably related to two different types of bare soil. The labels of the water class are almost entirely hard, but this is rightfully so, considering the 100% accuracy on water also on the dataset in Chapter 2.4 (see Table 2.9, and Table 2 in \cite{86}). The greenhouse class receives the lowest maxima in its labels, it is rejected very early, which is also correct as the recognition accuracy for this class is by far the worst in our system. Contrary to SVMs, the SOM remains pretty sure in the classification of

\textsuperscript{9}Since the rejection rate is not a direct parameter of the algorithms, we simply raise the rejection threshold from 0 to 1 in steps of 0.01, and note the respective rates.

\textsuperscript{10}This is less so with SVMs, where at least the shape of the curves for all class groups is similar.
4.5. $F^2$ Support Vector Machines

the urban green class, which usually is difficult to characterise. This is because on the SOM output grid, some nodes are especially devoted to mixed pixels, in particular to mixtures of vegetation and urban structures. A visual analysis of high resolution images confirms this, highlighting that the SOM is able to capture mixtures of classes in specific nodes.

To give a visual impression, Figure 4.4 shows a section of the land cover mapping our SOM algorithm produced on the Salerno dataset, with the threshold set such that a portion of 30% of all pixels is rejected.

4.5 Fuzzy-Input Fuzzy-Output Support Vector Machines

Support vector machines (SVMs) have become a popular method in pattern classification and were originally developed for the discrimination of two-class problems. They work by projecting the data into a higher-dimensional feature space using kernel functions, then finding the hyperplane that separates the two classes whilst providing the widest margin in which no sample points lie, hence also the name large margin classifier (see [224] or [275] for an introduction). In terms of their inventors, within the framework of Vapnik-Chervonenkis theory, SVMs create a model with a reduced VC dimension, decreasing error probability and thus exhibiting good generalisation abilities.

For more general applicability, considerable research has later been carried out to find ways to make use of the principle of SVMs in multi-class problems (see recently [10]), and architectures like One-vs-One, DAG-SVMs or One-vs-Rest are widely used nowadays (see [125] for a comparison).

Alas, the previously mentioned approaches work with data that has hard labels. Lin and Wang [157] and Huang and Liu [108] partially addressed this issue, their SVMs taking into account a certainty value associated with each sample. The authors had a more practical goal in mind, though: their certainty values are not given by labels, but constructed from the hard labelled data to solve special problems, here the weighting of samples in time series and the decrease of the impact of (detected) outliers in the data. When switching to data with vague labels, the problem is no longer confined to the binary case, and new solutions are required. These are proposed below, dealing with the input and output aspect of the SVMs separately. Recently, a similar approach was presented by Borasca, Bruzzone, Carlin, and Zusi [27], although their output optimisation is flawed.

We would like to take up their naming and call the fuzzy-input fuzzy-output support vector machine $F^2$-SVM in the following:

---

11 See Footnote 14.
12 This is not to be confused with the “Fuzzy Support Vector Machines” in [113], where graded
Once a SVM system is trained, the classification procedure consists of simple and fast multiplications. For industrial applications, this still seems too slow, so recently there has been research into performing this step in dedicated hardware [92, 77]. Least square SVMs promise to reduce the computational complexity of the training step, but produce a larger number of support vectors (which slows down classification). To minimise this effect, techniques like forward selection on the training data are used [3].

A technique that could improve over the already very good classification accuracy of the SVMs are the conceptually closely related support hyperplanes [177]. They work by calculating a bundle of hyperplanes that have no error on the training data, and for each test sample classification individually choose the one hyperplane from the bundle that is farthest away from it. Of course, this costs additional computation time in the classification phase.

In the following, the basic principles of SVMs will be briefly reviewed, before presenting our new approaches for fuzzy-input and fuzzy-output SVMs. Experiments validate their usefulness in different applications. Lastly, the special case of one-class SVMs is touched upon. It should be mentioned that our initial publication on $F^2$-SVMs [260] received the “Best Paper” award at the KES 2007 conference, out of 411 articles.

4.5.1 Basic Principles

As a foundation for the following introduction of our $F^2$-SVMs, the basic theory behind support vector machines will be reviewed briefly (see also in [224] or [275]).

As mentioned above, SVMs were developed to solve two-class problems. That is, there is a training set $S$ given as

$$S = \{ (x_\mu, l_\mu) | \mu = 1, \ldots, M, x_\mu \in \mathbb{R}^N, l_\mu \in \{-1, +1\} \},$$

which can be divided into two sets by a separating hyperplane, each set only containing samples with the same label. Such a hyperplane is determined by a weight vector $w \in \mathbb{R}^N$ and a bias or threshold $w_0 \in \mathbb{R}$ satisfying the separating constraints

$$l_\mu \ (x_\mu^T w + w_0) \geq 1, \quad \mu = 1, \ldots, M. \quad (4.8)$$

The hyperplane has a margin on both sides within which no training points lie. The idea now is to maximise this margin of width $2/||w||$, which leads to the following minimisation problem that is subject to the constraints just mentioned (4.8):

$$\Theta(w) = w^T w / 2 \rightarrow \min$$

outputs of the SVMs are used to deal with conflicts and outliers, all the while resting with hard classes.
4.5. \textit{F}^2 \textit{Support Vector Machines}

But, if the constraints do not hold for some data points, the problem is not linearly separable, and we have to soften it by introducing slack variables $\xi_\mu$ (this is called the “soft margin” approach). For some data points, it is now permissible to lie within the margin ($0 \leq \xi_\mu < 1$) or even on the wrong side of the hyperplane ($\xi_\mu > 1$). The optimisation problem and the constraints become:

$$\Theta(w, \xi) = w^T w/2 + C \sum_{\mu=1}^{M} \xi_\mu \rightarrow \min$$  \hspace{1cm} (4.9)

$$l_\mu \ (x_\mu^T w + w_0) \geq 1 - \xi_\mu, \ \xi_\mu \geq 0, \ \mu = 1, \ldots, M$$

Note the free parameter $C > 0$ which regulates the amount of margin-violations the SVM has to tolerate in finding the optimal hyperplane. Solving the optimisation problem, as detailed later, reveals that the optimal weight vector $w$ is a linear combination of selected training samples, the so-called support vectors, which lie exactly on the edge of the margin.

A very important feature of SVMs is the use of so-called kernel functions $K$. A Mercer kernel function [270] implicitly transforms the data points of the input space to a high dimensional Hilbert space, where now it might be possible to find a separating hyperplane, if we do not manage to do so in the original input space. Calculating the dot-product in Hilbert space using the kernel function, is the so-called kernel trick:

$$\langle \phi(x_1), \phi(x_2) \rangle = K(x_1, x_2), \ x_1, x_2 \in \mathbb{R}^N \hspace{1cm} (4.10)$$

The transformation function $\phi$ from input space to Hilbert space does not need to be evaluated, since the dot-product is given implicitly by the chosen kernel function $K$. From the numerous kernel functions available, we found three to work well in our applications: the linear kernel, the radial basis function kernel (see Chapter 4.1, $\sigma = 2$ if not otherwise noted) with each kernel centred on a support vector, and the polynomial kernel $K_{poly}$ of degree $d$,

$$K_{poly}(x_1, x_2) = [(x_1^T x_2) + 1]^d.$$ 

When presented with a new sample $x$, the SVM calculates the distance $d(x)$ in kernel space which the sample has to the hyperplane (defined by $w$ and $w_0$), and thus determines on which side of the hyperplane the sample is situated. It then answers with the class that is associated with this side. As mentioned in the introduction, for applications with more than two classes, special architectures are required. In the One-vs-Rest approach, one SVM is built for each of the $L$ classes, trained to make the decision between $\text{belongs to class } i$ and $\text{belongs to another class}$. The One-vs-One setup is more elaborate, building a separate SVM for each pair of classes (hence $\binom{L}{2} = \frac{L(L-1)}{2}$ machines), each trained to distinguish between class $i$ and $j$ only. The conversion of all their outputs to a meaningful classifier answer will be explained in Chapter 4.5.3 on fuzzy-output SVMs.
4.5.2 Fuzzy-Input Support Vector Machines

So far, the SVMs deal only with samples that are only associated with one class, and not multiple ones as desired. The basic task of the machines being to discriminate between two classes, vague memberships are first introduced for the binary case. In our approach there are two membership values, \( m_{\mu}^{-} \) and \( m_{\mu}^{+} \), associated with each training sample \( x_{\mu} \in S \). These values indicate to what extent the sample data point belongs to each of the two classes \( \{-1,+1\} \). These memberships are incorporated into the minimisation problem (4.9) by weighting the importance of the error-indicating slack variables \( \xi_{\mu} \) accordingly:

\[
\Theta(w, \xi^{+}, \xi^{-}) = w^{T}w/2 + C \sum_{\mu=1}^{M} (\xi_{\mu}^{+} m_{\mu}^{+} + \xi_{\mu}^{-} m_{\mu}^{-}) \rightarrow \min
\]

For the fuzzy-input SVM, a separating hyperplane has to be calculated under the following constraints:

\[
\begin{align*}
    w^{T} x_{\mu} + w_{0} & \geq 1 - \xi_{\mu}^{+}, & \mu = 1, \ldots, M \\
    w^{T} x_{\mu} + w_{0} & \leq -(1 - \xi_{\mu}^{-}), & \mu = 1, \ldots, M \\
    \xi_{\mu}^{+} & \geq 0 \quad \text{and} \quad \xi_{\mu}^{-} \geq 0, & \mu = 1, \ldots, M
\end{align*}
\]

Because this primal problem is very hard to solve with quadratic programming, the Lagrange multipliers \( \alpha^{+}, \alpha^{-} \) and \( \beta^{+}, \beta^{-} \) are introduced for the constraints, so that now the problem becomes finding the saddle point of the Lagrangian \( L \):

\[
L(w, w_{0}, \xi^{+}, \xi^{-}, \alpha^{+}, \alpha^{-}, \beta^{+}, \beta^{-}) = ww^{T}/2 + C \sum_{\mu=1}^{M} (\xi_{\mu}^{+} m_{\mu}^{+} + \xi_{\mu}^{-} m_{\mu}^{-}) - \sum_{\mu=1}^{M} \alpha_{\mu}^{+} ((w^{T} x_{\mu} + w_{0}) - (1 - \xi_{\mu}^{+})) + \sum_{\mu=1}^{M} \alpha_{\mu}^{-} ((w^{T} x_{\mu} + w_{0}) + (1 - \xi_{\mu}^{-})) - \sum_{\mu=1}^{M} \beta_{\mu}^{+} \xi_{\mu}^{+} + \sum_{\mu=1}^{M} \beta_{\mu}^{-} \xi_{\mu}^{-}
\]

Differentiating \( L \) with respect to the variables \( w, w_{0}, \xi^{+}, \xi^{-} \) of the primal optimisation problem, and setting the resulting terms equal to zero, the following necessary conditions are obtained:

\[
\begin{align*}
    \frac{\partial L}{\partial w} & = w - \sum_{\mu=1}^{M} \alpha_{\mu}^{+} x_{\mu} + \sum_{\mu=1}^{M} \alpha_{\mu}^{-} x_{\mu} = 0 \quad \Rightarrow \quad w = \sum_{\mu=1}^{M} (\alpha_{\mu}^{+} - \alpha_{\mu}^{-}) x_{\mu} \\
    \frac{\partial L}{\partial w_{0}} & = -\sum_{\mu=1}^{M} \alpha_{\mu}^{+} + \sum_{\mu=1}^{M} \alpha_{\mu}^{-} = 0 \quad \Rightarrow \quad \sum_{\mu=1}^{M} (\alpha_{\mu}^{+} - \alpha_{\mu}^{-}) = 0 \\
    \frac{\partial L}{\partial \xi_{\mu}^{+}} & = C m_{\mu}^{+} - \alpha_{\mu}^{+} - \beta_{\mu}^{+} = 0, \quad \frac{\partial L}{\partial \xi_{\mu}^{-}} = C m_{\mu}^{-} - \alpha_{\mu}^{-} - \beta_{\mu}^{-} = 0
\end{align*}
\]
4.5. \(F^2\) Support Vector Machines

Inserting (4.13) into (4.12), expanding and reordering yields

\[
L = -\sum_{\mu=1}^{M} \alpha_\mu^+ w_0 + \sum_{\mu=1}^{M} \alpha_\mu^- w_0
\]

\[= 0 \text{ because of (4.14)}\]

\[
+ \sum_{\mu=1}^{M} \alpha_\mu^+(1 - \xi_\mu^+) + \sum_{\mu=1}^{M} \alpha_\mu^-(1 - \xi_\mu^-) + C \sum_{\mu=1}^{M} (\xi_\mu^+ m_\mu^+ + \xi_\mu^- m_\mu^-) - \sum_{\mu=1}^{M} \beta_\mu^+ \xi_\mu^+ - \sum_{\mu=1}^{M} \beta_\mu^- \xi_\mu^-
\]

\[
- \sum_{\mu=1}^{M} \sum_{\nu=1}^{M} \alpha_\mu^+ (\alpha_\nu^- - \alpha_\nu^+) x_\nu^T x_\mu + \sum_{\mu=1}^{M} \sum_{\nu=1}^{M} \alpha_\mu^- (\alpha_\nu^+ - \alpha_\nu^-) x_\nu^T x_\mu
\]

\[+ 1/2 \sum_{\mu=1}^{M} \sum_{\nu=1}^{M} (\alpha_\mu^+ - \alpha_\mu^-)(\alpha_\nu^+ - \alpha_\nu^-) x_\mu^T x_\nu.\]

Simplifying the quadratic parts at the end and reordering yields

\[
L = -1/2 \sum_{\mu=1}^{M} \sum_{\nu=1}^{M} (\alpha_\mu^+ - \alpha_\mu^-)(\alpha_\nu^+ - \alpha_\nu^-) x_\mu^T x_\nu
\]

\[
+ \sum_{\mu=1}^{M} (\alpha_\mu^+ \xi_\mu^+ + C \xi_\mu^+ m_\mu^+ - \beta_\mu^+ \xi_\mu^+) + \sum_{\mu=1}^{M} (\alpha_\mu^- \xi_\mu^- + C \xi_\mu^- m_\mu^- - \beta_\mu^- \xi_\mu^-)
\]

\[= 0 \text{ because of (4.15)}\]

According to the Karush-Kuhn-Tucker theory, with \(\alpha_\mu^+, \alpha_\mu^-, \beta_\mu^+, \beta_\mu^- \geq 0\), the dual problem is now to maximise

\[
L(\alpha) = \sum_{\mu=1}^{M} \alpha_\mu^+ + \sum_{\mu=1}^{M} \alpha_\mu^- - 1/2 \sum_{\mu=1}^{M} \sum_{\nu=1}^{M} (\alpha_\mu^+ - \alpha_\mu^-)(\alpha_\nu^+ - \alpha_\nu^-) x_\mu^T x_\nu
\]

with the product \(x_\mu^T x_\nu\) at the end calculated using a kernel function (4.10), and subject to

\[
\sum_{\mu=1}^{M} (\alpha_\mu^+ - \alpha_\mu^-) = 0 \text{ from (4.14) and}
\]

\[
0 \leq \alpha_\mu^+ \leq C m_\mu^+, \quad 0 \leq \alpha_\mu^- \leq C m_\mu^- \text{ from (4.15).}
\]

(4.16)

The difference to ordinary SVMs is that the number of sample data points \(m_\mu\) doubled, by having each a positive and a negative sample, and that each Lagrange multiplier \(\alpha_\mu\) now is not bounded simply by the fix, a priori set \(C\), but by a function (4.16) that takes into account the membership for each point. For less important samples, \(\alpha_\mu\) has now a smaller range to be selected from.

The Karush-Kuhn-Tucker conditions for the problem now are (with \(\mu = 1, \ldots, M\)):

\[
\alpha_\mu^+ ((w^T x_\mu + w_0) - (1 - \xi_\mu^+)) = 0, \quad \alpha_\mu^- ((w^T x_\mu + w_0) + (1 - \xi_\mu^-)) = 0
\]

(4.17)
Chapter 4. Classifiers

\[ \beta^+ \xi^+ = (C \mu^+ - \alpha^+ \xi^+ = 0, \quad \beta^- \xi^- = (C \mu^- - \alpha^- \xi^- = 0 \quad (4.18) \]

Those samples \( x_{\mu} \) associated with a combined Lagrange multiplier \( \alpha_{\mu} = (\alpha^+_{\mu} - \alpha^-_{\mu}) \neq 0 \) are the important support vectors \( SV \), which determine the separating hyperplane (compare with condition 4.13):

\[ w = \sum_{\mu=1}^{M} (\alpha^+_{\mu} - \alpha^-_{\mu}) x_{\mu} \quad (4.19) \]

Support vectors \( x_{\mu} \) with \( \alpha^+_{\mu} = m^+_{\mu} C \) or \( \alpha^-_{\mu} = m^-_{\mu} C \) will be situated, according to (4.18), within the margin or even beyond the separating hyperplane. With increasing association of samples to multiple classes, this should be expected to happen quite frequently.

Now, using an on-the-margin support vector in (4.17) to obtain \( w_0 \), the final decision function \( f \) to classify a new sample \( z \) is

\[ f(z) = \text{sign}(w^T z + w_0) . \]

Again, all samples are projected into the higher-dimensional Hilbert space \( H \) using a kernel function \( K \). In this case the final decision function is (see (4.19) and (4.10))

\[ f(z) = \text{sign}(\sum_{i \in SV} (\alpha^+_i - \alpha^-_i) K(z, x_i) + w_0^H) \]

So far, the fuzzy-input support vector machines only deal with two classes at a time. To extend this to the multi-class case, the One-vs-Rest and One-vs-One architectures are employed. As training data, we still have a training set \( S = \{(x_{\mu}, y_{\mu}) | \mu = 1, \ldots, M \} \), with the samples \( x_{\mu} \in \mathbb{R}^N \), but their now vague labels \( y_{\mu} \) indicate different simultaneous memberships to each of the \( L \) classes:

\[ y_{\mu} = (y_{1,\mu}, y_{2,\mu}, \ldots, y_{L,\mu}), \quad y_{i,\mu} \in [0,1], \quad \sum_{i=1}^{L} y_{i,\mu} = 1. \]

In the One-vs-Rest case, the machine SVM\(_i \) learns to discriminate between class \( i \) and all other classes. The training set \( S_i \) is hence constructed as

\[ S_i = \{(x_{\mu}, m^+_{i,\mu}) | m^+_{i,\mu} = y_{i,\mu}\} \cup \{(x_{\mu}, m^-_{i,\mu}) | m^-_{i,\mu} = 1 - y_{i,\mu}\}, \quad \mu = 1, \ldots, M . \quad (4.20) \]

That is, each of the \( M \) samples is included twice, once with the weight its label assigns for the current class \( i \), and a second time with the summary weight that is assigned to all the other classes.

The One-vs-One architecture builds one SVM\(_{i,j} \) for each pair of classes \( i \) and \( j \), with the training set

\[ S_{i,j} = \{(x_{\mu}, m^+_{i,\mu}) | m^+_{i,\mu} = y_{i,\mu}\} \cup \{(x_{\mu}, m^-_{j,\mu}) | m^-_{j,\mu} = y_{j,\mu}\} . \quad (4.21) \]
Again, each sample is included twice in the training set $S_{i,j}$. To reduce the complexity of the problem in practical applications, $S_{i,j}$ will only include samples with a membership $m^{+/-} > \frac{1}{2}$. Memberships below the equal distribution indicate that the sample is not important at all for the current machine, and can safely be omitted.

After training the SVMs, they are ready to work as classifiers. How their responses are transformed into vague classifier answers is treated in the following chapter.

As an alternative to setting an individual $C$ for every sample (as manifest in Formula 4.16), the location of each sample in feature space could be altered according to its relevance, as Apolloni and Malchiodi [12] do for the binary case. A highly relevant sample is shifted towards the hyperplane (along its normal direction, $w$), lesser relevant samples are shifted away from it. In their sample applications, relevance is computed based on standard deviation in feature space, through k-means clustering or manual inspection. If the transition from one class to another follows a probability distribution, another approach by Appoloni, Basssis, and Malchiodi could be used [11]. It approximates the probability distribution using sampling, and in the support vector machine then only allows hypotheses which are compatible with that distribution.

### 4.5.3 Fuzzy-Output Support Vector Machines

Presented with a new sample, a SVM will calculate the distance in kernel space which the sample has to the separating hyperplane found in the training phase. Still, with the distance answer, even fuzzy-input SVMs “produce an uncalibrated value that is not a probability” ([188], p. 61). Having observed this, in 1999 Platt developed a method that transforms the distances using a parametric form of a sigmoid function. Platt’s method is widely used today, sometimes with enhancements to address issues like outliers [210]. Parameter optimisation can for example be accomplished using maximum likelihood estimation on the training set.

In the following, the application of Platt’s calibration to One-vs-Rest SVM architectures will be described. There are $L$ classes, hence also $L$ trained $SVM_i$, which upon presentation of a sample $z$ will each calculate a distance $d_i(z) = d_i^z \in \mathbb{R}$. A sigmoid function will be used to transform these distances into fuzzy outputs:

$$o_i(d_i^z) = 1/(1 + \exp(-A_i d_i^z + B_i)). \quad (4.22)$$

The $L$ fuzzy output labels can be normalised to sum up to 1.

The parameters $A_i, B_i \in \mathbb{R}$ are estimated for each $SVM_i$ to minimise the mean squared error on the training data $S_i$ (see Formula 4.20) between the original
Chapter 4. Classifiers

Label and the sigmoid output:

\[
error_i = \frac{1}{M} \sum_{x_\mu \in S_i} (o_i(d_i(x_\mu)) - y_{i,\mu})^2 = \frac{1}{M} \sum_{x_\mu \in S_i} (o_i(d_i^\mu) - y_{i,\mu})^2 \quad (4.23)
\]

Estimation is accomplished via a batch gradient descent technique that stops if there are only slight adjustments of the parameters between iterations. The resulting update rules\(^{13}\), with learning rate \(\eta\) are:

\[
\Delta A_i^\mu = (o_i(d_i^\mu) - y_{i,\mu}) \eta o_i^2(d_i^\mu) \exp(-A_i d_i^\mu + B_i) d_i^\mu \\
\Delta B_i^\mu = (o_i(d_i^\mu) - y_{i,\mu}) \eta o_i^2(d_i^\mu) \exp(-A_i d_i^\mu + B_i) (-1) \quad (4.24)
\]

The solution is not so straightforward in the One-vs-One approach, where a \(SVM_{i,j}\) is built for every pair of classifiers. To get the desired \(L\)-dimensional soft output, the technique employed in most cases today, for example in [88], is as follows: using an indicator function, transform each of the answers \(d_{i,j}\) into a vote for one of the two classes distinguished by the current machine \(SVM_{i,j}\). Then sum these votes per class, and normalise the resulting label to sum up to one. This method does not have a bad performance in terms of hard accuracy, but limitations as to the fuzzy labels produced (detailed in Section 4.5.4 below), since it deliberately does not take into account the distance information provided by the values \(d_{i,j}\). To heal this issue, one can proceed similarly to the One-vs-Rest case and use a sigmoid to transfer the distances to soft answers. Do note that, quite in contrast to the proposition by Borasca et al. [27], it is essential not to optimise the individual transfer functions in the One-vs-One setup\(^{14}\).

\(^{13}\)As usual in batch learning, after presentation of all training samples \(x_\mu \in S_i\), in the update phase the \(\Delta A_i^\mu\) and \(\Delta B_i^\mu\) which have been calculated are summed up and then subtracted from the current \(A_i\) and \(B_i\), respectively.

Note that in our paper [260] in the corresponding formulas, that is (27) through (29), the sample \(z\) or \(x_\mu\) itself is used, when its distance distance \(d_i^\mu\) to the hyperplane is meant. Related, of course \(A_i \in \mathbb{R}\). Also, in the update rules, a factor of \(\exp(-A_i d_i^\mu + B_i)\) is missing: The algorithms used for the experiments were correct, though. On a similar note, the minus-sign of the sigmoid function (7) in the paper by Borasca et al. [27] must be replaced by a plus sign.

\(^{14}\)This is quite a strong claim, but its validity will be demonstrated. To recapitulate, the individual \(SVM_{i,j}\) in the One-vs-One setup output the distance that the current sample has to their hyperplane. These distances shall be transformed into soft labels. The claim is that, other than proposed by Borasca et al. (Chapter 2.3.1. in [27]) the parameters of the transfer functions that transform the distances to soft answers must not be optimised individually for each machine.

Justification: As per the training phase, each machine \(SVM_{i,j}\) is optimised to discriminate between the classes \(i\) and \(j\). But, as is the nature of fuzzy-input fuzzy-output SVMs, the training set includes a high number of samples that only have a low membership in \(i\) or \(j\), but mainly belong to another class, say \(k\). For the optimisation of the sigmoidal transfer function, these samples will and must also be used. Now, let us assume that the sample \(x_a\) is of class \(i\), and has a distance of 2 to the separating hyperplane. Another sample, \(x_b\), be of class \(k\). The current SVM is not trained on class \(k\), so samples of this class may be situated anywhere in the
4.5. \( F^2 \) Support Vector Machines

But even then, the class-pair information provided by the SVM\(_{i,j}\) is not used. This can be addressed by the technique of pairwise coupling, based on the statistical Bradley-Terry model [33]. It uses initial estimations for the pairwise probabilities (derived via sigmoid from the distances \( d_{i,j} \)), and in an iterative process produces as approximation soft labels that take into account the coupled distance information.

Let \( r_{ij} = \text{Prob}(i \mid i \text{ or } j) \) be the probability that the true class is \( i \) if only classes \( i \) and \( j \) are present, and \( p_i = \text{Prob}(i) \) the probability that the true class is \( i \). The Bradley-Terry model now assumes that there exist estimates \( p_i \) and \( p_j \) such that the following \( \mu_{ij} \) is a good estimation for \( r_{ij} \):

\[
\mu_{ij} = \frac{p_i}{p_i + p_j}
\]

(4.25)

Reordering yields that this also fits the logit model

\[
\log \frac{\mu_{ij}}{1 - \mu_{ij}} = \log(p_i) - \log(p_j)
\]

The iterative coupling procedure works as given in Algorithm 1. In each step, the estimates \( \hat{p}_i \) and \( \hat{\mu}_{ij} \) are updated for \( \forall i \) and \( \forall i, j \), respectively.

The final estimation of the class probabilities \( \hat{p}_i \) is then the answer of the One-vs-One SVM. Hastie and Tibshirani [98] prove that the procedure minimises the kernel space, even randomly on both sides of the hyperplane. So, let’s say \( x_b \) has a distance of 10 to the hyperplane. Keep in mind that the sigmoid used (4.22) is constructed so that it transforms higher distances into higher membership values, and vice versa. Now, the sigmoid would have to map the 10 into a low “does not belong to my class” value, all the while assigning a high value to the 2 distance. This target conflict will lead to the sigmoid being parametrised in such a way that even big distances (equals sure classifications) will elicit a low “I don’t know” response. Put another way, there will be many points with the same distance to the hyperplane, the majority being from irrelevant classes with low membership targets, a minority being from classes \( i \) or \( j \) with big membership targets, leading to a grave optimisation conflict. Indeed, in an experiment (with 7 classes, optimising A in the range of [-8, 23] and B in [0, 21]) it could be observed that the error function decreased, but the hard accuracy of the final answers went down to approximately the level of a random guess.

There are several solutions to this dilemma: One might only use voting, resulting in good hard accuracy but yielding unsatisfactory fuzzy labels. Alternatively, the parameters for the sigmoidal transfer function can be optimised globally using cross-validation experiments (we adopted this approach). That means there is only one A and one B for all machines. As third possibility, the transfer function could be replaced by a much more powerful neural net that in addition to the distance gets to know the projection of the sample onto the hyperplane, thereby possibly having enough information to learn that a “sure” distance of 10 can also indicate an untrained-for class.

Do note that the issue does not arise for One-vs-Rest SVMs, since with them every machine has seen all classes.

\(^{15}\)The notation is in accordance with [98], to make the broader results there more accessible to the reader. Do note that their \( n_{ij} \) weights are always 1 in our case and hence omitted.
Algorithm 1 Bradley-Terry coupling

**Input:** Distances $d_{i,j}$ obtained from the SVMs for the current sample.

0: **Initialisation; Calculate the single class probabilities from pairwise values**
   Obtain the pairwise probabilities $r_{ij}$ by passing the $d_{i,j}$ to a sigmoid function (like in the One-vs-Rest architecture, see Formula 4.22).
   Then produce estimates $\hat{p}_i$ for the $p_i$, employing a simple average:
   \[
   \hat{p}_i := \frac{\sum_{j \neq i} r_{ij}}{\frac{1}{2} L (L - 1)}
   \]

1: **Update the pairwise probability estimates**
   As $\hat{\mu}_{ij}$ calculate the estimates for $\mu_{ij}$, based on the $\hat{p}_i$ (see Formula 4.25):
   \[
   \hat{\mu}_{ij} := \frac{\hat{p}_i}{\hat{p}_i + \hat{p}_j}
   \]

2: **Correct the single class probability estimates by the ratio of SVM-pairwise-probabilities to pairwise estimates**
   \[
   \hat{p}_i := \frac{\hat{p}_i \sum_{j \neq i} r_{ij}}{\sum_{j \neq i} \hat{\mu}_{ij}}
   \]

3: **Normalise the single class probabilities**
   \[
   \hat{p}_i := \frac{\hat{p}_i}{\sum_{\forall j} \hat{p}_j}
   \]

4: **Loop until convergence**
   IF change of $\hat{p}_i >$ threshold THEN GOTO 1

**Output:** Normalised class probabilities estimation $\hat{p} = (\hat{p}_1, \ldots, \hat{p}_L)$. 
average (negative) Kullback-Leibler distance\(^\text{16}\) between the \(r_{ij}\) and \(\hat{\mu}_{ij}\) and also converges. Our experiments show that the output of SVMs using Bradley-Terry coupling exhibits a much lower entropy than without coupling, meaning crisper answers. This is in line with the prove in [98] that the initial estimation for the \(\hat{\mu}_i\) in step 0 is closer in Kullback-Leibler distance to the equal distribution vector than the final answer after coupling. But, also shown in [98], if only the probability-order of the classes (and hence hard accuracy) is regarded, there will be no difference between the initial and final \(\hat{\mu}_i\).

Pairwise coupling is also employed by Borasca et al. [27]. Their sigmoidal transfer functions, however, are not symmetric, there are different parameters for \(a_{ij}\) and \(o_{ij}\) (concerning the training procedure employed for their sigmoids, see our critical remarks in Footnote 14). For an introduction to and many interesting theoretical conclusions about pairwise coupling in classification turn to [98].

Having presented the One-vs-Rest and One-vs-One architectures to deal with the multiclass case, the question is which is to be preferred. In their short study [88] based on a land cover mapping application, Anthony, Hulley, and Marwala do find no significant performance difference between the two architectures. And in their experimental assessment of five architectures, Kahsay, Schwenker, and Palm ([125], p. 340) find that, for non-linear kernels, “all the multiclass decomposing schemes show comparably similar performance.” In own comparison experiments based on the application of emotion recognition of faces (see Chapter 2.1), the only statistically significant result was that 2 of the 12 feature/region combinations were significantly\(^\text{17}\) better classified by the One-vs-One approach (see [217], there Chapter 6.7). However, if hard training labels were used, a cross-validation experiment on the feature of optical flow in the mouth area revealed a wide gap between the accuracies of the One-vs-One (accuracy: 0.68) and the One-vs-Rest approach (accuracy: 0.48). This result acts as a reminder that the preference for one architecture or another can well be feature-dependent.

The issue can also be regarded under a technical point of view. Let again \(L\) be the number of classes, and \(M\) the number of fuzzy labelled training samples. Then, the One-vs-Rest approach requires to build \(L\) SVMs, each based on \(2M\) training samples (since each sample belongs to some degree to the class under consideration or the Rest, compare Formula 4.20). In the One-vs-One approach, a much higher number of \(L(L - 1)/2\) pairwise machines has to be build. If, in the worst case, each sample is associated with all classes to some degree, each of

\[^{16}\text{The Kullback-Leibler distance } l \text{ between two probability distributions } f_1 \text{ and } f_2 \text{ is defined as: } l(f_1, f_2) = \int f_1(y) \log \frac{f_1(y)}{f_2(y)} \, dy. \text{ In the current case, it is calculated on symmetric probabilities: } l(r, \hat{\mu}) = \sum_{i<j} r_{ij} \log \frac{r_{ij}}{\hat{\mu}_{ij}} + (1 - r_{ij}) \log \frac{1 - r_{ij}}{1 - \hat{\mu}_{ij}}.\]

\[^{17}\text{The other performance advantages for the One-vs-One approach were not significant (paired } t\text{-test, } \alpha = 0.05). \text{ But, for only three of the weakest feature/region combinations, the mean performance of the One-vs-Rest architecture was higher than of the One-vs-One one.}\]
the pairwise machines also has to be trained on $2M$ samples (compare Formula 4.21), making this architecture much more expensive computationally and in terms of memory consumption. However, as mentioned above, in the One-vs-One setting, all samples in the machines’ training set with a membership below equal distribution will be omitted. In practice, in part also depending on the fuzziness of the labels, this will drastically reduce the training set size. In all of our applications, the One-vs-One architecture completed much faster than the One-vs-Rest approach. All things considered, we would express a preference for the One-vs-One decomposition scheme in the fuzzy case.

An interesting extension to the One-vs-One approach can be found in the theory behind Tri-Class SVMs. These do not only look at the current two classes $i$ and $j$ in the training phase, but all samples. This is for example achieved by confining all samples not belonging to class $i$ or $j$ to a narrow tube around the separating hyperplane. For an introduction and some ideas for new directions, turn to [10].

From a technical point of view, support vector regression could also be employed to produce uncertain answers, thinking of class memberships as continuous-valued functions. In doing so, one would abandon the idea that classes correspond to clouds in (kernelised) feature space that have to be modeled, with each cloud being mainly populated with samples carrying the same class label.

### 4.5.4 Experimental Results

This section serves to experimentally evaluate the fuzzy-input fuzzy-output support vector machines presented above, and to explore some of their properties in more detail.

The very good performance of SVMs is well established in various disciplines today. As an example, three of the five winning teams of the IEEE GRS-S Data Fusion Contest (see in Chapter 2.4) employed SVM classifiers. Exhibiting a high classification accuracy, SVM classification answers can even serve as a baseline in scenarios where no ground truth is available, or only in a crisp form. We employed this approach for example in our experiments with self-organizing maps [86].

#### Comparison with other classifiers

Based on the application of land cover mapping in the Salerno area, as described in Chapter 2.4, we experimentally compared several fuzzy-output classification algorithms (details published in [259]): a multi layer perceptron (MLP, [106]) with a hidden layer of 70 neurons, the neurons with hyperbolic tangent sigmoid transfer functions in the output layer producing probability estimates. Optimisation was accomplished using backpropagation [209] with a maximum of 20 epochs. The RBF network classifier (described in Chapter 4.1) had 69 kernels. For the
4.5. $F^2$ Support Vector Machines

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBF</td>
<td>0.88</td>
</tr>
<tr>
<td>MLP</td>
<td>0.90</td>
</tr>
<tr>
<td>KNN</td>
<td>0.92</td>
</tr>
<tr>
<td>SVM, One-vs-Rest (RBF)</td>
<td>0.92</td>
</tr>
<tr>
<td>SVM, One-vs-Rest (linear)</td>
<td>0.93</td>
</tr>
<tr>
<td>SVM, One-vs-One (RBF), coupled sigmoids</td>
<td>0.91</td>
</tr>
<tr>
<td>SVM, One-vs-One (linear), coupled sigmoids</td>
<td>0.95</td>
</tr>
<tr>
<td>SVM, One-vs-One (RBF), coupled votes</td>
<td>0.93</td>
</tr>
<tr>
<td>SVM, One-vs-One (linear), coupled votes</td>
<td>0.95</td>
</tr>
</tbody>
</table>

Table 4.2: Classification accuracy of supervised learning algorithms, SVM kernels given in parenthesis. As mentioned in the application chapter, cross-validation should not be employed in the land cover mapping scenario, hence the results here are based directly on the spatially separate test set.

weighted k-nearest neighbour classifier (KNN, compare Chapter 4.2), $K$ was set to 200. On the SVM side, One-vs-One and One-vs-Rest fuzzy-output architectures were employed, based on linear and RBF kernels, with voting or (coupled) sigmoid output generation in the One-vs-One case. The results can be consulted in Table 4.2. Support vector machines with a linear kernel outperform the other approaches.

The single best classifier in the application of emotion recognition from facial expressions was a support vector machine (compare Table 2.3, for full accuracy tables see Appendix A.1 in [217]): the One-vs-One architecture on the edge orientation histogram feature in the mouth area achieved a classification accuracy of 74.0%\(^{18}\). The closest competitor on this feature, with 70.3\% accuracy, is a RBF network employing the data driven approach presented in Chapter 2.1.6, which allows to incorporate sequence information. This additional sequence and motion energy treatment could also be the reason that for all other features (except for orientation histograms on the face area, where a One-vs-Rest SVM wins) a RBF approach provides the best performance\(^ {19}\).

**Hard-trained vs. soft-trained**

A big question is whether a soft-trained SVM has better performance than one trained on hard labels. This question was investigated exemplarily on the very difficult task of emotion recognition from speech (see Chapter 2.1 for applica-

\(^{18}\)Result 8-fold cross validated. See Footnote 16 in Chapter 2.1.5 for setting details.

\(^{19}\)First tentative experiments showed that the successful approaches presented in Chapter 2.1.6 to include motion energy into the time fusion process of RBF answers do not work well for SVM answers.
Chapter 4. Classifiers

Table 4.3: Performance of hard SVMs versus fuzzy-input fuzzy-output SVMs (application: emotion recognition from speech). Values for the badly performing MLP and KNN classifiers given as comparison. Best values in bold.

<table>
<thead>
<tr>
<th></th>
<th>hard SVM</th>
<th>fuzzy MLP</th>
<th>fuzzy KNN</th>
<th>F\textsuperscript{2}-SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (in %)</td>
<td>54.1</td>
<td>33.1</td>
<td>29.5</td>
<td>55.6</td>
</tr>
<tr>
<td>Euclidean distance</td>
<td>0.566</td>
<td>2.132</td>
<td>0.724</td>
<td>0.541</td>
</tr>
<tr>
<td>S\textsubscript{1} Min/Max</td>
<td>0.361</td>
<td>0.141</td>
<td>0.248</td>
<td>0.395</td>
</tr>
</tbody>
</table>

Table 4.3: Performance of hard SVMs versus fuzzy-input fuzzy-output SVMs (application: emotion recognition from speech). Values for the badly performing MLP and KNN classifiers given as comparison. Best values in bold.

ation details, and our publication [260] for a broader discussion): MLP and KNN classifiers\textsuperscript{20} yielded meagre classification accuracy on this dataset, only the more powerful SVM approaches could generate good results. In the setup, we trained our F\textsuperscript{2} support vector machines with the fuzzy labels obtained from the labelling experiment, while the training labels of the hard SVM\textsuperscript{21} consisted of hardened fuzzy labels. The SVM architecture was One-vs-Rest in both cases, with a polynomial kernel of degree 3. The kernel choice was straightforward, as the RBF and linear kernels did never reach a comparable performance, and the RBF kernel here was also very sensitive to changes of its width parameter. The results were obtained by 10 times 10-fold cross validation, with coupled runs, that is the soft and hard SVMs operated on exactly the same training and test sets. Besides the accuracy, that is the percentage of correct hard decisions, the mean Euclidean distance between training labels and answers and the mean S\textsubscript{1} measure (see Formula 4.5) between these are given in Table 4.3. As can be seen, the soft F\textsuperscript{2}-SVMs beat all other classifiers, for all performance measures. Even if a hard output is necessary, it is beneficial to use the soft-trained SVMs, as in the experiments they have an accuracy that is on average 1.55 percent higher than that of their hard-trained counterparts. Looking at the individual pairwise runs (100 in total), F\textsuperscript{2}-SVMs lost only in 30 and reached a tie in 16 of the runs.

In our experiment described below on varying training set sizes, using the performance measure S\textsubscript{1}, we find that soft-trained F\textsuperscript{2}-SVMs statistically significantly outperform the hard-trained ones (detailed results see Table 4.5).

Fuzzy output

On the land cover mapping setup again, we compared the behaviour of the classifiers once they were allowed to reject an (increasing) portion of the test samples

\textsuperscript{20}The MLP had 30 hidden neurons, optimisation was accomplished by backpropagation over 20 epochs, minimising the mean square error between network output and the fuzzy labels. The fuzzy-output KNN operated with \( k = 5 \).

\textsuperscript{21}In fact, we used the same program code for the hard-trained as well as for the F\textsuperscript{2}-SVMs, as hard training input automatically transforms a F\textsuperscript{2}-SVM into a standard one (albeit with fuzzy output).
4.5. $F^2$ Support Vector Machines

Figure 4.7: Accuracy of different classifiers, once a portion of the samples was allowed to be rejected. For the SVMs, only the best architecture (One-vs-One, linear kernel, coupled sigmoid output) is shown.

(results also published in [259]). Rejection of a fuzzy classifier output $o$ concerning the $L$ classes followed the simple rule\footnote{For a justification for the usage of this rule, compare the experiments summarised in Table 5.1.} of:

$$\text{reject if } (\max_i o_i < \text{threshold}) \quad , i \in \{1, \ldots, L\} \quad \quad (4.26)$$

This serves to identify the so-called mixed pixels, which are associated to low degrees with many classes. As the rejection rate is not a direct parameter of the algorithms, we simply raised the rejection threshold from 0 to 1 in steps of 0.01, and noted the respective rejection rates. Is is evident from the plot in Figure 4.7 that the SVM steadily provides the highest accuracy on the test set. The RBF has the worst initial performance, and also the slowest increase of accuracy. Interestingly, the relatively simple KNN algorithm has a good initial accuracy, and raises very fast, so that at a level of rejection of 15% it reaches the performance of the champion SVM. Hence, being much easier to implement, the KNN solution might to be preferred in some applications. Not shown in the graph, the first algorithm to reach a perfect accuracy of 1 is the MLP, at the price of rejecting 43% of the samples.

Since the availability of test data is limited in the land cover mapping domain, each analysis must entail a visual inspection and assessment of the resulting
Figure 4.8: A section of the mappings produced by a One-vs-One SVM (bottom left) and a One-vs-Rest SVM (bottom right), each with a linear kernel. For comparison, a false colour composition of the area, based on 3 ASTER bands, is given (upper right). Clearly, the One-vs-One approach produces more compact regions of mixed pixels. In the One-vs-Rest case, one can also observe an overestimation of the greenhouses to the detriment of non-vegetated land, an error which we found, despite good overall accuracies on the test set, common to all the classifiers except for the One-vs-One SVM architecture.

Compare also with the SOM-results for the same area in Figure 4.4.
maps, classifier accuracy alone is not a sufficient indicator. On the other hand, such visual analysis also paints a clearer picture for the layman. To assess the distribution of mixed pixels, which perhaps cover multiple classes on the ground, we fixed a reject threshold so that a 30% portion of the pixels was flagged as mixed. The visual analysis now essentially confirms the effectiveness of the One-vs-One SVM approach: here, the mixed pixels are nicely gathered in compact regions, facilitating further analysis by photointerpreters. For the RBF, MLP, KNN, and One-vs-Rest SVMs, the rejected pixels have a rather sparse and indented distribution. These findings are illustrated in Figure 4.8 which contrasts the results obtained by a One-vs-One and a One-vs-Rest linear SVM.

As detailed in Chapter 4.5.3, the fuzzy output of a One-vs-One SVM can either be based on votes or on distances passed through a Fermi function, in both cases followed by Bradley-Terry coupling. We noted already that concerning the hard classification accuracy, there is no difference between the two approaches. Looking at the fuzzy labels produced, voting yields very hilly answers. But, in the sample-rejection scenario, even using Bradley-Terry coupling, we observed a strange behaviour in the machines with coupled votes: the rejection rate would go from nearly 0 to around 80% at a certain threshold, which makes the voting approach useless in this context. Examining the answers to the 628 test samples closer, we found that there are only 79 different variants of the vote matrix, leading to 77 different coupled answers, with only 13 different maxima. As ten of these maxima even lie in the narrow range of [0.74, 0.76], it is immediately clear that a smooth tradeoff scaling between accuracy and rejection rate is not possible. In contrast, One-vs-One classifier outputs based on coupled transformed distances are completely individual. What is more, by changing the parameter $A$ of the Fermi transfer function (4.22), it is directly possible to influence the fuzziness of the answers, as manifest in the results presented in Table 4.4.

<table>
<thead>
<tr>
<th>$A$ parameter</th>
<th>0.5</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean max. of answers</td>
<td>0.27</td>
<td>0.37</td>
<td>0.53</td>
<td>0.66</td>
<td>0.72</td>
<td>0.75</td>
</tr>
</tbody>
</table>

*Table 4.4:* Changing the $A$ parameter of the sigmoid transfer function for One-vs-One $F^2$-SVMs (see Formula 4.22). Given is the mean maximum of the resulting labels. Application: Facial emotion recognition, optical flow in the face area as feature, RBF kernel, results were calculated before time-fusion of the frames.

**Reduced training set size**

Besides the kernel type and parameters, the SVMs have one more free parameter: the $C$ factor, determining the impact of each sample missclassification on the total
Table 4.5: Performance of the $F^2$-SVM if only a fraction (rows) of the full 378 training samples is used in the training phase, with different values for the free $C$-parameter (columns). The values are a mean over the cross-validation runs, performance measure is $S_1$ for which, like for accuracy, higher values indicate better performance. Best values in bold.

<table>
<thead>
<tr>
<th>Frac([C])</th>
<th>1 e-4</th>
<th>1 e-3</th>
<th>1 e-2</th>
<th>1 e-1</th>
<th>1</th>
<th>10</th>
<th>100</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>0.1554</td>
<td>0.1555</td>
<td>0.1573</td>
<td>0.1779</td>
<td><strong>0.1836</strong></td>
<td>0.1830</td>
<td>0.1829</td>
<td>0.1830</td>
</tr>
<tr>
<td>0.10</td>
<td>0.1586</td>
<td>0.1589</td>
<td>0.1628</td>
<td>0.1927</td>
<td>0.2018</td>
<td>0.2015</td>
<td><strong>0.2023</strong></td>
<td><strong>0.2023</strong></td>
</tr>
<tr>
<td>0.15</td>
<td>0.1614</td>
<td>0.1617</td>
<td>0.1642</td>
<td>0.1987</td>
<td><strong>0.2070</strong></td>
<td>0.2053</td>
<td>0.2050</td>
<td>0.2049</td>
</tr>
<tr>
<td>0.20</td>
<td>0.1466</td>
<td>0.1472</td>
<td>0.1531</td>
<td>0.1988</td>
<td>0.2095</td>
<td>0.2085</td>
<td>0.2097</td>
<td><strong>0.2098</strong></td>
</tr>
<tr>
<td>0.30</td>
<td>0.1649</td>
<td>0.1656</td>
<td>0.1729</td>
<td>0.2182</td>
<td><strong>0.2246</strong></td>
<td>0.2178</td>
<td>0.2150</td>
<td>0.2148</td>
</tr>
<tr>
<td>0.40</td>
<td>0.1577</td>
<td>0.1586</td>
<td>0.1675</td>
<td>0.2190</td>
<td><strong>0.2314</strong></td>
<td>0.2248</td>
<td>0.2226</td>
<td>0.2221</td>
</tr>
<tr>
<td>0.50</td>
<td>0.1578</td>
<td>0.1591</td>
<td>0.1744</td>
<td>0.2327</td>
<td><strong>0.2394</strong></td>
<td>0.2256</td>
<td>0.2231</td>
<td>0.2230</td>
</tr>
<tr>
<td>0.60</td>
<td>0.1553</td>
<td>0.1567</td>
<td>0.1732</td>
<td>0.2389</td>
<td><strong>0.2471</strong></td>
<td>0.2318</td>
<td>0.2259</td>
<td>0.2256</td>
</tr>
<tr>
<td>0.70</td>
<td>0.1557</td>
<td>0.1573</td>
<td>0.1773</td>
<td>0.2354</td>
<td><strong>0.2475</strong></td>
<td>0.2376</td>
<td>0.2287</td>
<td>0.2280</td>
</tr>
<tr>
<td>0.80</td>
<td>0.1553</td>
<td>0.1571</td>
<td>0.1795</td>
<td>0.2444</td>
<td><strong>0.2571</strong></td>
<td>0.2421</td>
<td>0.2365</td>
<td>0.2341</td>
</tr>
<tr>
<td>0.90</td>
<td>0.1557</td>
<td>0.1578</td>
<td>0.1858</td>
<td>0.2494</td>
<td><strong>0.2579</strong></td>
<td>0.2416</td>
<td>0.2341</td>
<td>0.2316</td>
</tr>
<tr>
<td>1.00</td>
<td>0.1563</td>
<td>0.1586</td>
<td>0.1888</td>
<td>0.2532</td>
<td><strong>0.2638</strong></td>
<td>0.2478</td>
<td>0.2407</td>
<td>0.2391</td>
</tr>
</tbody>
</table>

In our experiments we found that the choice of $C$ was crucial for the performance of the classifier. This is a positive result, since if it had no influence it would have meant that the memberships in the labels do not change the optimal decision surface at all. So far, for setting $C$ we have not found a general rule that is applicable to all data sets. However, the optimal value was always in the range of [1e-4; 1000] in our experiments. To choose the optimal setting here, a standard approach is to make $k$-fold cross-validation experiments, trying out values in the range given above. We found that it is enough to start with $C = 1e-4$ and then increase $C$ by one order of magnitude for each subsequent trial until the upper limit of the range is reached. However, with $k = 8$ we still have to compute 64 complete runs, which is quite expensive. A possible remedy would be to use only a portion of the available training data for this parameter optimisation. This can decrease computation time considerably (in our case, minutes instead of days), but the problem is that we are not aware of any result that states if the optimal value for $C$ depends on the number of training samples. Random trials on different data sets seemed to indicate that training set size is not an issue. Thus we ran expansive experiments on a strong feature of the face emotion recognition application, namely edge orientation histograms on the whole face area. The classifier was a One-vs-One $F^2$-SVM with RBF-kernel ($\sigma = 2$). For cross-validation, the set was divided into 8

\footnote{Note that, at least for basic two-class problems, new experiments by Tegnér [255] suggest that the choice of $C$ is not important for very high-dimensional problems.}
parts. Using different fractions \( f \in \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\} \)
of the 378 training samples in the training phase, we determined the optimal choice (in terms of classification accuracy) of \( C \) for each fraction. It turned out to be \( C = 0.1 \) across all of them.

The experiment was repeated (with 2 runs of the cross-validation procedure) for another performance measure, \( S_1 \), the results can be consulted in Table 4.5 in terms of mean performance over 16 runs. The tabulated values nicely demonstrate some points: There clearly is a setting of \( C \) that finds the optimal balance in terms of the influence of the slack variables, yielding best performance. Also, the optimal setting depends on the performance measure, since here for \( S_1 \) it is \( C = 1 \), and not \( C = 0.1 \) as for classification accuracy. Also, the optimal value for \( C \) is the same regardless of the fraction of training samples used (if you look closely, the two exceptions for low number of training samples in the beginning are a maximum of 0.0005 away from the overall winner). With these observations, we feel confident that, as considered above, doing the parameter search based on a much lower number of training samples is indeed a valid solution to reduce computation time.

<table>
<thead>
<tr>
<th>Fraction</th>
<th>0.05</th>
<th>0.1</th>
<th>0.15</th>
<th>0.20</th>
<th>0.30</th>
<th>0.40</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soft</td>
<td>0.1836</td>
<td>0.2018</td>
<td>0.2070</td>
<td>0.2095</td>
<td>0.2246</td>
<td>0.2314</td>
</tr>
<tr>
<td>Hard</td>
<td>0.1792</td>
<td>0.1934</td>
<td>0.1979</td>
<td>0.1976</td>
<td>0.2116</td>
<td>0.2190</td>
</tr>
<tr>
<td>Soft - Hard</td>
<td>0.0044</td>
<td>0.0085</td>
<td>0.0090</td>
<td><strong>0.0118</strong></td>
<td><strong>0.0131</strong></td>
<td><strong>0.0124</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fraction</th>
<th>0.50</th>
<th>0.60</th>
<th>0.70</th>
<th>0.80</th>
<th>0.90</th>
<th>1.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soft</td>
<td>0.2394</td>
<td>0.2471</td>
<td>0.2475</td>
<td>0.2571</td>
<td>0.2579</td>
<td>0.2638</td>
</tr>
<tr>
<td>Hard</td>
<td>0.2252</td>
<td>0.2341</td>
<td>0.2353</td>
<td>0.2445</td>
<td>0.2477</td>
<td>0.2526</td>
</tr>
<tr>
<td>Soft - Hard</td>
<td><strong>0.0142</strong></td>
<td><strong>0.0131</strong></td>
<td><strong>0.0122</strong></td>
<td><strong>0.0126</strong></td>
<td><strong>0.0101</strong></td>
<td><strong>0.0112</strong></td>
</tr>
</tbody>
</table>

**Table 4.6:** Comparing the performance (\( S_1 \) measure) of a soft-trained versus a hard-trained SVM, if only a fraction of the whole training set is used. Statistically significant differences in bold (soft wins).

Using hard training and test labels, the same experiment was run, with similar results. The optimal setting was also \( C = 1 \) for most fractions, with the few exceptions slightly favouring a lower \( C \). The performances of the hard-trained and soft-trained SVMs are contrasted in Table 4.6, showing that a training with soft labels always yields a higher \( S_1 \) performance! From the fraction \( f = 0.2 \) upwards, this is even statistically significant on a level of \( \alpha = 0.5 \) according to the corrected\(^{24}\) resampled paired t-test (see Chapter 4.8). When an oracle was allowed to always choose the optimal \( C \) for each fraction\(^{25}\) on hard and soft

\(^{24}\)If not the corrected t-test (Formula 4.29) is used but the standard one (Formula 4.28), the difference is significant for all fraction levels (also applies to the oracle experiment). This demonstrates the importance of employing the corrected version.

\(^{25}\)Choosing based on the cross-validation results, as shown in Table 4.5 for the soft case.
training labels, the outcome of the experiment was the same, soft still being superior in all cases, except that this was not significant any more for the $f = 0.3$ fraction.

| Train fract. | 0.10 | 0.20 | 0.30 | 0.40 | 0.50 | 0.60 | 0.70 | 0.80 | 0.90 | 1.00 |
| Face accu.   | 29.3 | 34.2 | 38.1 | 40.3 | 41.7 | 43.3 | 43.6 | 44.8 | 45.2 | 45.4 |
| Satellite accu. | 82.5 | 92.3 | 94.6 | 94.7 | 95.4 | 95.5 | 95.2 | 95.4 | 95.3 | 95.4 |

**Table 4.7:** Behaviour of the classification accuracy (in percent) for an increasing number of training samples, and two applications. Application face emotion recognition: feature is edge orientation histograms on the face area, One-vs-One $F^2$-SVM (RBF kernel with $\sigma = 2$, optimal $C = 0.1$); Best classifier for this feature would have been One-vs-Rest SVM.), 6 classes, total number of training samples is 378; Application land cover mapping (satellite): standard feature as described in Chapter 2.4, One-vs-One SVM (RBF kernel with $\sigma = 2$, optimal $C = 0.1$), 7 classes, total number of training samples is 1029;

The experiments so far document an increasing classifier performance with an increasing number of available training samples, which seems logical. All the more interesting are the results presented in Table 4.7, tabulating for two applications the accuracy when the training set size is increased. The accuracy in the face application monotonically increases with more training samples, while for the satellite it reaches a plateau once being presented with at least half of the training samples (around 500). In the latter case, it is obviously possible to scale down the training set to reduce training time, without sacrificing classification accuracy.

### 4.5.5 One-Class Support Vector Machines

An interesting variant of SVMs are the one-class support vector machines, a technique which is also sometimes dubbed support vector data description [254]. It uses only data from one specific class to construct a model of its distribution (in the form of a hypersphere), which then allows to classify hitherto unseen samples. In this it is quite similar to unsupervised learning algorithms like Gaussian mixture models or k-means clustering. The basic idea of one-class SVMs is to find a hypersphere with centre $a$ and radius $R$ encompassing the training data, as illustrated in Figure 4.9.

Assuming the hypersphere covers the whole set of training samples $x_\mu$, the empirical error is equal to zero. In close relation to the standard SVM approach the structural error is defined by

$$\Theta(R, a) = R^2. \quad (4.27)$$

The primal optimisation problem is thus to minimise the (squared) radius of the hypersphere, subject to one constraint for each of the $M$ training samples...
4.5. \( F^2 \) Support Vector Machines

Samples from target class

Figure 4.9: Samples from a single class (left) are encompassed by the hypercircle with centre \( a \) and radius \( R \) (right). This hypersphere defines a boundary separating the one-class dataset from the rest of the input space.

\[ x_{\mu} \in \mathbb{R}^N: \quad ||x_{\mu} - a||^2 \leq R^2 \quad \mu = 1, \ldots, M \]

The solution of this problem employs the Karush-Kuhn-Tucker theory and works mostly along the same lines as for the basic SVMs\(^{26}\). Of course, slack variables \( \xi_{\mu} \) can be introduced to soften the problem (compare Formula 4.9).

As we mentioned \(^{260}\), one-class SVMs can be adapted to handle a fuzzy class membership via the introduction of weights for the slack variables. With \( m_{\mu} \) being the class membership of sample \( x_{\mu} \), the function to optimise (4.27) becomes

\[ \Theta(R, \xi, a) = R^2 + C \sum_{\mu=1}^{M} \xi_{\mu} m_{\mu} \rightarrow \min . \]

As usual, the parameter \( C \) controls the influence of the slack variables and hence constraint violations. Constructing the Lagrangian and differentiating it with respect to \( \xi_{\mu} \) yields the following condition, analogue to Equation 4.15:

\[ C m_{\mu} - \alpha_{\mu} - \beta_{\mu} = 0 \]

This is a nice result and means that the one-class SVMs can really be adapted to use fuzzy class labels, hence capturing fuzzy classes, while still allowing optimisation via the SMO algorithm.

Extending the approach to multiple classes, with one model trained for each class, will inevitably lead to conflict- and outlier-situations, if a sample falls into

\(^{26}\)For details see for example our study on the application of one-class SVMs to the classification of bioacoustic time series [213].
multiple or none of the models created. We presented two possible venues to deal
with this issue [213], called nearest-centre strategy and nearest-support-vector
strategy.

An alternative approach to one-class SVMs has been proposed [223], where the
goal is to find a hyperplane that separates most of the training samples from the
origin point.

4.6 Learning with Certainty Weighted Labels

Training data consists of samples associated with class labels. Additionally, each
label can carry a certainty value $c_\mu$, indicating the reliability of the labelling for
this sample (see also Definition 3.1). These certainty values are useful in different
application settings, for example if experts asked to label a dataset have to rate
their discriminatory ability for each sample (they might be very specialised). Or,
labellers can indicate poor sensor data, for example if they can hardly follow a
conversation due to excessive background noise. In a setting where only hard
labels are allowed, certainty values are a means to indicate that a sample does
not entirely belong to one class. More related to technical reasons, certainty
factors can be used to model smooth boundaries between samples\textsuperscript{27}, as employed
for phoneme recognition [277].

How uncertainty theories could directly deal with label certainty is detailed in
Table 3.4. Incorporating the additional training label certainty information into
classifiers is generally a rather straightforward process, and in the following will
briefly be detailed for the classifiers already presented above. Formally, each of
the $M$ labelled samples will have a label certainty weight $c_\mu$, associated with it,
$c_\mu \in (0, 1]$, these weights being collected in $C = (c_1, \ldots, c_M)^T$. For an attempt
at a formalisation of the different methods to include label certainty into existing
classifiers, see [163].

For the LVQ approach (Chapter 4.3), which we already adapted for fuzzy labels,
the inclusion of certainty weights consists in adapting the similarity measure $S$
in Formula 4.4, for hard labels for example to:

$$S(y_\mu, l^*) = \begin{cases} 
  c_\mu & \text{if } y_\mu = l^* \\
  0 & \text{otherwise}
\end{cases}$$

Of course, the similarity-threshold $e$ that controls the direction of the update
(towards or away from the training sample) has to be tuned again. If set too low,
the majority of samples will be repellent, driving the prototype far away from the
sample point clusters in feature space. As for the learning of prototypes’ labels
(see Formula 4.6), the $c_\mu$ would simply multiplicatively alter the learning rate $\theta$.

\textsuperscript{27}See also Chapter 3.6 on Fuzzy Control.
Fuzzy-input support vector machines (see Chapter 4.5.2) already feature a good training label certainty weighting mechanism, in that each sample $x_\mu$ has two associated membership values $m^+_{\mu}$ and $m^-_{\mu}$, which indicate to what extent it belongs to the + or − class. These values can simply be updated via $\hat{m}^+_{\mu} = m^+_{\mu} / c_\mu$. For SVMs, it is especially important to note that adding a large number of low-certainty training samples will significantly increase the optimisation time needed, but not necessarily improve the classification accuracy.

For the self-organizing maps (Chapter 4.4) and k-nearest neighbour (Chapter 4.2) classifiers, the solution is very similar and simple. The sample-dependent weights $w^j$ in Formula 4.7 and $w_i$ in Formula 4.3 simply have to be multiplied with the corresponding $c_\mu$.

With RBF networks (Chapter 4.1), the certainty values in $C$ can already be incorporated into the error function (Equation 4.1), weighting the error for each of the $M$ samples:

$$E(W) = ||\hat{C}H W - \hat{C}Y||^2$$

Here, $\hat{C}$ is the $M \times M$ diagonal matrix whose diagonal entries are the square-rooted entries of $C$. For this new formulation, the pseudinverse operation + (as specified in Formula 4.2) does again provide a straightforward solution with

$$W = \hat{H}^+ \hat{Y}, \text{ where } \hat{H} = \hat{C}H \text{ and } \hat{Y} = \hat{C}Y.$$  

Alternatively, if the RBF-weights $w$ are to be determined with a gradient descent algorithm, the certainty weights $c_\mu$ show up in the derivation of the error function simply as multiplicative factor, as would have to be expected intuitively:

$$\frac{\Delta E_\mu}{\Delta w_{ji}} = -c_\mu 2 H_{\mu,j} (H_{\mu,i}W_i - Y_{\mu,i})$$

For each training sample $x_\mu$ the value $w_{ji}$ to be updated is the weight of the connection between hidden neuron $h_j$ and output unit $\phi_i$.

### 4.7 Robustness Against Label Noise

In supervised classification, we naturally can not work without labels that are associated with our training data. Obtaining labels, hard or soft, is prone to errors, human or otherwise. That means that the training set for a classification algorithm can contain falsely labelled samples, which in extreme cases might render it useless. What we will show is that classifiers that have been trained on fuzzy labels are less affected by labelling errors than their hard-trained counterparts, and that even in the presence of label noise, a combining of classifiers increases the final accuracy.
Chapter 4. Classifiers

For experiments, erroneous labels are simulated by so-called label noise, that is noise imparted on the original labels. There is not much literature available on how label noise should be modeled and dealt with. One exception is Anluin and Laird’s paper [9] that details how to embed training data with flipped labels into the framework of probably approximately correct (PAC) learning [269], even allowing malicious noise. They give a lower bound on the number of training samples necessary. The analysis is only applicable to a “flip” kind of noise and does not hold for more than two classes. Then, some works exist that do present algorithms that can deal with noise, but mostly with the restrictions just mentioned. One example is [151], where the approach is to learn the parameters of the model that generates the noise. Closely related, in [7] the label flip probabilities are incorporated into the training target criterion. The thrust of that paper is different, however, as the setting is semi-supervised learning, where no noise is actively added, but only a faction of the training data has labels associated. Being an extension of the methods proposed by McLachlan in [170], the algorithm changes the labels attributed to the unlabelled training data with each iteration, minimising the modified classification maximum likelihood criterion.

### 4.7.1 Modelling label noise

Label noise needs to be artificially added according to a specific model. In a two-class case, a given portion of the training data would get randomly selected and the associated label flipped to the opposite class. This methods extends to the multi-class case, with the label being flipped to one of the other classes in a random manner (as employed in [71] and for the experiments in Chapter 4.3.4). But since we are dealing with fuzzy labels, where not only one class is given as the label, these noise models are not applicable. Thus, we employ the procedure described in Algorithm 2.

**Algorithm 2** Adding noise to labels

**Input:** Normalised fuzzy labels $l_\mu$, desired noise level

**for all** $l_\mu$ **do**

% Generate a random label

$l_{\text{rand}} = \text{for each class, draw from uniform distribution in}[0, 1]$

$l_{\text{rand}} = \text{normalise label}(l_{\text{rand}})$

% Mix original and random label

$l_\mu = l_\mu \ast (1 - \text{noise}) + l_{\text{rand}} \ast \text{noise}$

**end for**

**Output:** Normalised labels $l_\mu$ with added noise
This approach models the noise as it could appear in real-world scenarios, and can be understood intuitively. Normalisation in our case simply consists in ensuring that the class memberships in each fuzzy label sum up to one. Note that it is important to first normalise the random label and then combine it with the original one using a weighted sum. Normalising only in the end would seriously flatten the fuzzy labels and decrease their variance. It is obviously essential that the original label and the random label be normalised in the same way, or their combination would yield unpredictable results.

For certain applications, it might be useful to choose a different noise model than the uniform one we employed. For example, one could treat each class label differently, either by assigning a special variance for the randomisation, or using a separate noise level for each. A totally different approach to generate noised labels would be to see each label as a point on the hyperplane of possible (regarding normalisation) labels. To add noise, one would advance on this hyperplane into a random direction, with the length of this vector being determined by the desired noise level.

4.7.2 Experimental Results

The goal is to evaluate the impact that noise on the training labels has on the accuracy of single classifiers and multiple classifier architectures. To this end, different levels of artificial noise are added to the labels, according to Algorithm 2. For each level, four basic classifiers, based on different features, are trained, and their decisions combined. Essentially, we are interested in the behaviour of classification accuracy as we increase the level of label noise. The entire experiment is run twice, once based on fuzzy labels, once on hard labels that have been derived from the fuzzy labels (via simple hardening). This allows to see whether it is beneficial to use soft labels, or if hard labels are to be preferred.

A self-contained description of this experiments and our conclusions has also been published in [258].

The data for the experiment comes from the fruit recognition application as described in Chapter 2.3. Since the labels here are originally hard, they had to be fuzzified using the fuzzy k-means algorithm. The basic classification algorithms used were radial basis function networks (for the Sobel and colour histogram

\footnotesize{We performed an experiment on 700 fuzzy labels. Without noise, 428 of them had a class membership with a value above 0.5. After adding 30\% noise (\textit{noise} = 0.3), this dropped to 250, flattening the labels a bit. Normalising only at the end, this value dropped sharply to 38. Looking at the mean variance of the labels, the original ones had 5.4e-2, the noised ones 2.7e-2, and on the end-normalised ones the value went down to 1.2e-2.}

\footnotesize{Fuzzification as described in Chapter 4.3.4 for the fuzzy LVQ experiment. In the current setup, the fuzzifier was chosen such that hardened fuzzy labels agree with the original hard labels in about 80\% of the cases.}
features) and fuzzy-input fuzzy-output support vector machines (for Canny and APQBW). These two classifier types were selected for generally showing a reliable classification performance. The number of kernels in the RBF network was set to 47 using a simple heuristic formula, optimisation accomplished as detailed in Chapter 4.1. The $F^2$-SVMs (compare Chapter 4.5) had a polynomial kernel of degree 3, the overall architecture was One-vs-One. The accuracy of the fuzzy-output classifiers is calculated from their hardened output, all results are based on 5-fold cross validation experiments.

The combination of the classifiers’ decisions is accomplished using several established multiple classifier system architectures in parallel. We tested the following schemes (see Chapter 6.1 for details): minimum, median, average, Dempster-Shafer orthogonal sum rule, decision templates (DT) using measure $S_1$, simple probabilistic product, and an optimal least squares solution calculated using the pseudoinverse.

The most important findings of our experiments can be seen in Figure 4.10: Even the individual classifiers hold up to noise incredibly well. The classifier fusion step (well performing method plotted) is always able to improve over the individual results. Most importantly, classifiers trained on the soft labels have higher accuracy.

![Figure 4.10](image)

**Figure 4.10:** Behaviour of classification accuracy once adding more and more noise. Shown are plots for the four basic classifiers and their fused answer (fusion method given in brackets). (left) Classifiers trained on soft labels. (right) Only hard labels were provided for training.

A more detailed analysis of the performance of the soft-trained versus the hard-trained classifiers is shown in Figure 4.11. The individual soft classifiers have, in most cases, a higher accuracy (shown as gain in percent points) than their hard counterpart. Taking the mean value, soft wins. But this advantage gets less and less noticeable once more and more noise is added. The better performance of
4.7. Label Noise

Soft-trained classifiers stems, in our opinion, from the ability of the classifiers to take advantage of interdependencies that are encoded in the fuzzy labels.

Concerning the combination of several classifiers to obtain a final answer, it can be seen rather clearly in Figure 4.10 that the extra fusion step is really worth to undertake. The accuracy of the combined answer is always higher than of even the best single classifier. A detailed analysis of this experiment’s results concerning the power of fusion algorithms is available in Chapter 6.2.

One observation strikes as particularly surprising: the high resilience of even single classifiers, trained on hard or soft labels, to added noise. Revisiting Figure 4.10, we see that despite adding 80% noise on the training labels, the best single classifier still has the very high accuracy of 68%. To put this into perspective, we shall look at the effects of noise on the labels from another angle. If we do not treat them as fuzzy labels, but are only interested in the class with the highest probability, the behaviour shown in Figure 4.12 comes up. The hardened noisy labels agree pretty much with the hardened not noised labels. For example, at the above-mentioned noise level of 80%, there still is agreement of 56.4%, meaning more than half of the samples associated (via hardening) with one class are originally from that class. This seems to still be enough for the classifiers to train reasonably well. So, in our classification setting, the fuzzy labels can take quite some amount of such noise before performance problems occur.

An important note on fuzzy versus hard in this context, to keep the observations above in perspective: the finding that the classifiers are quite resilient to high label noise levels is only valid for the soft labels. As shown, the noise added does deteriorate a fuzzy label gracefully, and it will take high noise levels until the winning class with the highest probability changes. As the hard labels in this experiment have been derived by hardening the available soft labels, they share this property. Any noise added directly on a hard label, which is only possible
using the “flip” rule, would instantly change its winning class. Unfortunately, a direct comparison of such noise and our model is not possible. Some clues can be glimpsed from our experiments in Chapter 4.3.5 (particularly Figure 4.2) based on the same dataset. There, noise is added to the hard labels, and the accuracy drops already at lower levels of noise, albeit not as hard as to be expected in a strictly hard setting, since a fuzzy re-labelling based on clusters introduces a fuzzy component there.

The most important conclusion from this experiment is that fuzzy labels can take much noise before the accuracy of classifiers trained on them is affected. Also, soft-trained algorithms have a higher accuracy than their hard-trained counterparts.

4.8 Statistical Methods for Measuring Classifier Performance

This section is a short excursion that discusses what statistical methods should be used to judge the performance of classification algorithms.

If the data available as basis for the classifier is limited, as is the case in all of our application experiments, the technique of cross-validation can be employed to use as much of this data as possible for training while still correctly judging the performance. Cross-validation is based on the idea to randomly split the available data in to $k$ subsets of the same size. Then, $k$ classifiers are built, each using another of the subsets for testing and the union of the respective remaining
4.8. Measuring Classifier Performance

others as training set. The mean of the \( k \) classifier performances obtained is then taken as the final performance. The advantage of this procedure is that its answer has a reduced variability compared to a single trained classifier, hence it is also a more reliable indicator for the generalisation ability of the algorithm tested.

Concerning cross-validation, a number of special terms has been established. The name of \( k \)-fold cross validation simply means the standard way, that is the data has been split into \( k \) subsets. In leave-one-out cross-validation, each individual sample represents one subset in itself. If it is ensured that in each subset the distribution of the class labels is about the same, the term is stratified cross validation. Building the \( k \) classifiers and obtaining the results is called a run, and often several runs are made, resulting for example in 5 times 10-fold cross-validation, with \( r = 5 \) runs and \( k = 10 \) folds.

To determine if a classification algorithm \( A \) has a better performance than algorithm \( B \), based on a certain performance measure, the algorithms have to be trained and tested several times to accurately capture their generalisation ability. The standard technique is to make several runs of cross-validation, and on every training-test partition generated, both algorithms are trained and their respective performance noted. On these measurements, a statistical test can then be computed which can determine if a difference observed between the two algorithms’ performances is only a chance incident or a true difference. The test will have a null hypothesis \( H_0 \) and an alternative hypothesis \( H_1 \), where \( H_0 \) states that there is no difference between the algorithms, and \( H_1 \) postulates that one of them is better. Given a chosen significance level \( \alpha \), here \( \alpha = 5\% \), the test will indicate whether it is, with a probability of error below \( \alpha \), possible to reject the null hypothesis \( H_0 \), meaning that indeed one of the two algorithms is better\(^30\).

A common choice of test for the task of performance comparison is the parametric \( t \)-test, which requires that the measured performances follow a normal distribution. Since with cross-validation the algorithms run on the same data subsets, the additional independence assumption is violated and the coupled or paired \( t \)-test has to be used. First, for each pair of classifiers tested in the cross-validation phase, the difference \( d_i, i = 1 \ldots rk \) of their performances is calculated, then the mean \( m \) and variance \( \hat{\sigma}^2 \) over these \( d_i \). The test statistic \( t \) is then given by

\[
t = \frac{m}{\sqrt{\frac{1}{rk}\hat{\sigma}^2}}
\]  

(4.28)

and can be compared with the tabulated values for the \( t \)-test with \( df = kr - 1 \) degrees of freedom to determine if \( H_0 \) can be rejected.

\(^30\)This is called a two-sided test. If it has to be verified if algorithm \( A \) is better than \( B \) (and not vice-versa), as usual the difference of the performances \( A - B \) would be calculated as basis for the test, then a right-sided test (also called right tail test) will be applied, which does not split the allowed error \( \alpha \) between both sides of the distribution, leading to an easier rejection of the null hypothesis.
However, Nadeau and Bengio noted [175] that the variance is underestimated, because the \( r_k \) readings are based on possibly overlapping training and test-sets and are therefore not independent. The solution proposed is to adjust the variance based on the ratio \( \frac{n_2}{n_1} \) of the number of test samples \( n_2 \) to the number of training samples \( n_1 \), leading to a corrected statistic \( \tilde{t} \) of

\[
\tilde{t} = \frac{m}{\sqrt{(\frac{1}{r_k} + \frac{n_2}{n_1})\hat{\sigma}^2}}.
\]  

This is called the corrected resampled t-test. Bouckaert and Frank looked [32] at different tests and concluded that the one just presented provides the best replicability, with 10 times 10-fold cross validation even being superior to 100 runs of random subsampling. More experiments than the 10 times 10 will not yield [31] an increased power of the test, which is its probability of correctly rejecting a false null hypothesis.

As a result of these findings, we strive to use 10 times 10-fold cross-validation in our experiments, only lowering this requirement if the computational burden is too high. Where statistical tests are applied, the corrected resampled paired t-test (Formula 4.29) is used.
The theoretical foundations and issues of encoding and dealing with uncertainty have already been dealt with in Chapter 3. In the current chapter, the more practical side of uncertain classifier answers is explored. Measuring the fuzzy accuracy of a classifier or the uncertainty of its answers is a requirement for a later fusion in the context of multiple classifier systems. Thus, the first part of this chapter deals with extracting uncertainty measures directly from classifier outputs. Then some approaches at judging uncertain classification performance are presented. Lastly, possibilities are shown to have the classifier itself produce certainty assessments of its answers.

5.1 Calculate Uncertainty Directly from Fuzzy Answers

For this chapter, it is assumed that there is a classifier given, but nothing is known about its inner workings. Also, there be no extensive validation set. Yet it is required to judge the uncertainty associated with each individual decision of the classifier, for example to couple an appropriate weight with the decision for a later decision fusion process. The basic assumption is that a classifier is more uncertain about its verdict if it is more fuzzy. If the answer given is hard, obviously no further analysis is possible. If the classifier answers imprecisely, with a bag of labels, the number of classes in the bag can serve as a basis to gauge the uncertainty of the decision. A vague or fuzzy answers offers the richest information, and the distribution of allotted probabilities can be analysed. Two measures are especially suited for this purpose: Shannon entropy and the Gini function.

The Shannon entropy measure comes from the field of information theory. It was introduced by Claude Shannon in [236] as a means to determine the capacity of
a channel required to transmit a message encoded as binary digits over telephone lines, hence as a measure for the amount of information contained in a message. Employing the classification notation, the entropy $H$ of a decision $y$ is defined as

$$H(y) = -\sum_{i=1}^{L} y_i \log_2 y_i, \quad y \in [0, 1]^L, \quad \sum_{i=1}^{L} y_i = 1.$$ 

The value of $H$ is in the range between 0 and $\log_2 L$. Its maximum value is attained iff the elements $y_i$ of the classifier answer all equal $\frac{1}{L}$. If the classifier is really sure, with one of the $y_i$ being 1, the entropy is 0 [225]. To make the entropy measure comparable across different classification problems, it can be normalised to have a maximum value of 1 by dividing through $\log_2 L$. The use of entropy is not limited to fuzzy answers, for example Hutchinson and Kak [112] employed it to measure the ambiguity of Dempster-Shafer hypotheses. Their algorithm automatically decided which real-world sensing operation should be performed next, simply by predicting the hypotheses sets that might occur if a particular sensing operation was applied, and looking at the resulting changes of ambiguity.

In machine learning in general, entropy has a wide range of applications, most notably in the area of decision trees [198, 76].

The **Gini function** is a measure of inequality or impurity, commonly used in the fields of economics and data mining. In economics, the so-called Gini index shows the inequality of the distribution of wealth or income [84]. In data mining, the Gini function is used in decision tree algorithms to assess the impurity of regions [37]. Applied to classifier answers, the Gini function $G$ is defined as

$$G(y) = 1 - \sum_{i=1}^{L} y_i^2 = \sum_{i=1}^{L} \sum_{j=1, j \neq i}^{L} y_i y_j, \quad y \in [0, 1]^L, \quad \sum_{i=1}^{L} y_i = 1.$$ 

As with the entropy, the minimum of 0 is reached if the classifier replies with a hard $y$, the maximum value is attained for an uniformly distributed answer. Normalisation is accomplished by dividing the resulting Gini value by $\frac{L-1}{L}$. Own experiments [257] showed that certainty measures based on the Gini function or entropy are not much able to boost the accuracy in a multiple classifier setting. The distribution of certainty values for the two cases of “correctly classified” and “falsely classified” samples overlap too much (see Figure 5.1). Shannon entropy produced promising results when used as an indicator to allow the rejection of samples a multi classifier system is not sure about [261]. See Chapter 6.1 for a note on how to incorporate certainty measures into a Dempster-Shafer classifier fusion process.

If the classifier answer is not a simple label, but a more elaborate construction grounded in one of the uncertainty theories presented in Chapter 3, other measures are required. For example, Dubois and Prade [64] extend the so-called
5.1. Calculated from Fuzzy Answers

Figure 5.1: Distribution of the entropy certainty measure calculated on the outputs of a classification experiment, contrasting the certainty on correctly classified test samples with the certainty on falsely classified samples. In this example, there is a sizeable gap between the mean values (black line), but as usual the distributions overlap largely. The certainty measure in this case is the normalised, inverted \((1 - H)\) entropy. As dataset, hand-written numerals were used, the classifier was a Gaussian RBF network operating on a PCA-based feature. All details on this experiment are available in \([257]\).
Chapter 5. Certainty Measures for Classification

**U-uncertainty** of Higashi and Klir [103], which operates using \( \alpha \)-cuts, so that it is not only applicable to possibilistic fuzzy sets but also to Dempster-Shafer basic probability assignments \( m \):

\[
U(m) = \sum_{A \subseteq \Theta} m(A) \log_2(|A|)
\]

The normalised index of fuzziness \( IF \) (Formula (8) in \[23\]) can be used to measure the distance of a fuzzy set \( R \) to the nearest crisp set \( C \):

\[
IF(R) = \frac{\sum_{\omega \in \Omega} |\mu_R(\omega) - \mu_C(\omega)|}{\sum_{\omega \in \Omega} \mu_R(\omega)} \tag{5.1}
\]

In case the classification result is a rough set, Formula 3.27 can be applied to measure how accurate the lower and upper approximations \( B \)-upper and \( B \)-lower are able to capture the fuzzy set.

For an overview of uncertainty measures for fuzzy sets, possibility and probability theory and the Dempster-Shafer framework, turn to [134, 66].

### 5.2 Judging the Fuzzy Accuracy of a Classifier

If a classifier is intended to produce uncertain answers, with an uncertain ground-truth, the problem is how the accuracy concerning this fuzzy goal is to be measured. This issue is addressed in the following.

The elementary question is how uncertain classifiers answers can be compared with the desired output. If the answer is to be understood as stemming from a specific uncertainty framework, of course the measures from this framework should be used, for fuzzy sets for example the difference employed in Formula 5.1. If those measures do not exist in the uncertainty framework itself, numerous enhancements are documented in the literature to this end.

Especially the Dempster-Shafer theory of evidence itself does not dispose of a means to measure the distance between two basic probability assignments (bpa). A very straightforward solution would be to apply the *pignistic transformation* [248] that Smets proposed in the framework of the transferable belief model, which breaks down the belief allotted in a bpa \( m \) such that all belief mass is finally only assigned to atomic hypotheses \( \theta_k \):

\[
pig(\theta_k, m) = \sum_{\theta_k \subseteq A} \frac{m(A)}{|A|}
\]

\(^1\)In the crisp set \( C \), there is only one non-zero assignment, \( \mu_C(\omega_i) = 1 \).
Thus, for each set \( A \) that contains the current atomic hypothesis \( \theta_k \), the belief is simply evenly distributed between the \(|A|\) atomic hypotheses in \( A \). These pignistic probabilities can then easily be compared, for example calculating the mean square error. Zouhal and Denoeux employed this technique \([301]\) to compare the pignistic probabilities produced by their evidence-theoretic KNN classifier with the desired output, and tune the parameters of their classifier accordingly.

A more involved distance measure was proposed by Jousselme, Grenier, and Bossé \([123]\), with the basic assumption that if the atomic hypotheses in \( \Theta \) “are indiscernible and unorderable, the only measure distinguishing any two subsets of is their cardinality” (p. 94). The representation of the beliefs in a bpa \( m \) is given as a \( 2^q \)-dimensional row vector \( \vec{m} \), noting for each element of the powerset of the frame of discernment \( \text{Pow}(\Theta) \) the belief mass assigned. The distance measure is now defined as

\[
d_{\text{bpa}}(m_1, m_2) = \sqrt{\frac{1}{2}((\vec{m}_1 - \vec{m}_2)^T D(\vec{m}_1 - \vec{m}_2))}
\]

As can be expected, the number of elements in combined or intersected sets plays a prominent role in the construction of the distance-matrix \( D \):

\[
D(A, B) = \frac{|A \cap B|}{|A \cup B|}, \quad D \in [0, 1]^{2^q} \times [0, 1]^{2^q}, \quad A, B \in \text{Pow}(\Theta)
\]

The sets \( A \) and \( B \) are identified and matched via the respective position in the \( 2^q \)-dimensional belief vector \( \vec{m} \). This distance measure can also easily be applied to random sets \([79]\).

A very general form to measure the distance between two fuzzy labels \( y \) and \( z \) is the Minkowski distance of order \( p \):

\[
d_{\text{Mink}}^p(y, z) = \left( \sum_{i=1}^{L} |y_i - z_i|^p \right)^{\frac{1}{p}}
\]

For \( p = 2 \), this is the famous Euclidean distance.

For fuzzy sets and fuzzy labels, Prade’s \( S_1 \) distance measure is a good choice (see Formula 4.5 for details):

\[
S_1(y, z) = \frac{\sum_{i=1}^{L} \min(y_i, z_i)}{\sum_{i=1}^{L} \max(y_i, z_i)}
\]

A cruder way to compare fuzzy labels is to consider only the rank order of the classes. For example with \( L = 3 \) classes, label \( y = (0.1, 0.7, 0.3) \) would have the

\(^2\text{In classification applications, the number } q = |\Theta| \text{ of possible hypotheses of course equals the number of classes, } L.\)
associated ranks $ra^y = (1, 3, 2)^3$. One way to compare the ranks of two labels $y$ and $z$ is Spearman’s rank correlation coefficient, $r_s$:

$$r_s(ra^y, ra^z) = 1 - \frac{6 \sum_{i=1}^{L} (ra^y_i - ra^z_i)^2}{L(L^2 - 1)} \quad (5.2)$$

If the rank order in both labels is the same, $r_s = 1$. Another possible approach to calculate rank order consistency is based on inverting the ranks of one of the labels (that is $ra^{y_{inv}} = L - ra^y + 1$):

$$\text{rank order consistency}(y, z) = \sum_{i=1}^{L} \frac{ra^{y_{inv}}_i}{ra^y_i} \left( \sum_{j=1}^{L} \frac{j}{L - j + 1} \right)^{-1}$$

The normalising factor ensures that a perfect match has a rank order consistency of 1, and produces lower values for differing labels.

Despite the wealth of distance measures just presented, the method most heavily used in classification to judge the power of a fuzzy classifier is still the accuracy. For this, first the labels in the test set and the respective classifier answers are hardened, transforming them to hard labels in a winner-takes-all manner. Formally, a fuzzy label $y = (y_1, \ldots, y_L)$ will be hardened by

$$j := \arg\max_{i=1 \ldots L} y_i, \quad y^\text{hardened}_i = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

This procedure is also called the maximum membership rule [146]. Based on this hard label, the percentage of correctly classified samples in the test set is determined. The link of this accuracy to the fuzzy labels is of course considerable weakened in the process. Nonetheless, it is still a standard basic performance measure in fuzzy classification (compare our remarks in the application chapters).

To gain more insight into the classification process and its difficulties, a more nuanced measure than the hard accuracy would be desired. One candidate for this is the $S_1$ vector, derived from the $S_1$ distance measure presented above. The $S_1$ value will be calculated as a mean over all $m$ test samples, but independently for each class in the label:

$$\vec{S}_1 = [S_1^1, \ldots, S_1^L], \quad S_1^i = \frac{1}{m} \sum_{\mu=1}^{m} \frac{\min(y^i_{\mu}, o^i_{\mu})}{\max(y^i_{\mu}, o^i_{\mu})}$$

The $y_{\mu}$ represent the groundtruth labels of the training set, $o_{\mu}$ the corresponding classifier answers, with $y_{\mu} = (y^1_{\mu}, \ldots, y^L_{\mu})$ and $o_{\mu} = (o^1_{\mu}, \ldots, o^L_{\mu})$ both $\in [0, 1]^L$, and $\mu = 1, \ldots, m$.

---

3This is only one way of encoding ranks. The zero might also be included, or the most probable class might receive the lowest rank number.
5.2. Fuzzy Accuracy

For an even more detailed picture of the fuzzy classification performance, something similar to the hard confusion matrix can be of use, for example the fuzzy error matrix proposed by Binaghi, Brivio, Ghezzi, and Rampini [23]. It replaces the hard count of the number of times the sample has been classified as object $i$ while in reality it is object $j$ by a sum over an intersection-operator derived from fuzzy set theory. In their case, $\min$ is used, but other choices are also reasonable. An entry in the $L \times L$ fuzzy error matrix $\tilde{M}$ is defined

\[ \tilde{M}_{i,j} = \sum_{\mu=1}^{m} \min(o^\mu_i, y^\mu_j) . \]

Based on $\tilde{M}$, more specific accuracy measures like producer’s accuracy or overall accuracy can be computed, the latter one analogously to the way it is done for the hard confusion matrix: dividing the sum over the diagonal by all the belief in the training set [2]. But, Binaghi et al. had to conclude that still “the best course of action to obtain the accuracy information is to support the interpretation of the descriptive measures with a detailed inspection of the full fuzzy error matrix” (p. 947).

So, as in most cases we require a scalar accuracy measure for the optimisation of our algorithms, we decided to use the hard accuracy, and sometimes the $S_1$ measure, underpinning the end results with more detailed metrics like confusion matrices.

Note that if the accuracy computed serves as basis for the optimisation of parameters, model selection or the like, it is quite possible that over-fitting occurs, that is the classifier will be fine-tuned to the specifics of the test set, as a consequence generalising badly. The remedy is to split the data into 3 sets, one for training, one test set to compute the final accuracy on, and one validation set that solely serves to gauge the classification performance in the parameter tuning phase. If labelled data is scarce, it might be necessary to have an overlap between the training and validation set [61].

Most accuracy measures can be adapted to take different misclassification costs into account, that is give more weight to an error if the class confusion involved incurs higher costs in the application. An example would be that judging an ill patient as healthy is much more undesirable than the inverse. Related, if the classifier gives an estimate about the correctness of its answer, as presented in the following, this value can directly be integrated into an accuracy measure.

---

4 Note that when confusion matrices are presented in this work, their meaning is transposed with respect to the fuzzy error matrix $\tilde{M}$, with the groundtruth classes given in the lines.

5 The statement of Binaghi et al. in [23] (p. 939) that the an element “in the fuzzy error matrix denotes a crisp number, due to the fact that […] scalar cardinality is applied to the fuzzy set” seems to make no sense, as from their definitions and even examples later it is clear that the entries in the fuzzy error matrix are $\in \mathbb{R}$.

4 If the training labels are normalised to sum up to one, the divisor will be $m$. 
Chapter 5. Certainty Measures for Classification

5.3 Uncertainty as Additional Classifier Answer

Classifiers can be built such that along with each individual sample classification, they also output a certainty value indicating how much this decision should be trusted. This value can then be used for the purpose of weighting in a later decision combination phase, and is much more fine-grained than giving one accuracy weight to the whole classifier (Chapter 5.2), and presumably much more accurate than deriving the accuracy measure solely from the structure of each answer (Chapter 5.1). The classifiers have two sources based on which they can judge their accuracy: the quality of the data, and the specific classifier model they built.

Based on the data quality, quite some classifiers documented in the literature calculate their accuracy for each sample. In a recent paper [191], Poh, Heusch, and Kittler report on a face verification system that, for use in a classifier combination step, quantifies the quality of input images using indicators like illumination and contrast. Also in the biometric domain, Maurer and Baker [169] calculate local quality indicators for their features, to be incorporated into their Bayesian belief network architecture that will not only fuse the classifiers’ answers, but also provide a global accuracy measure. Their quality metrics were a secret score stemming from a US governmental database in the fingerprint application, and based on the NIST speech quality assurance algorithm\(^\text{7}\) for the voice data. Our own approaches presented in Chapter 2.1.6 that use weighting based on the motion energy of the sample can also be thought of as utilising data-dependent certainty factors. In his dissertation in the domain of text categorisation, Bennett [20] lists and analyses the usefulness of 70 different, as he calls it, reliability indicators. Some examples for those are the document length, the number of unique words or the percentage of hitherto unseen words in a test sample.

But, not all the 70 certainty measures he uses are based on the data. The majority in fact is specific to the classifier used, detailed for naive Bayes, KNN, decision trees and support vector machines. For the k-nearest neighbour algorithm, one reliability indicator given is the neighborhood radius, that is the distance from the query sample to the furthest sample from the training phase that is included in the neighbourhood. One idea for the SVM classifier is to measure how much the output \(o\) of the SVM changes when the current sample \(x\) is shifted by \(d_i = \beta_i (\nu_i - x)\) into the direction of each of the support vectors \(\nu_i\). The factor \(\beta_i\) is chosen in such a way that each shift has the same length, and the shift towards the closest support vector is exactly half of the distance. The ideas have later been expanded in work on neighborhood-based local sensitivity [21].

\(^{7}\)The speech quality assurance package of the US National Institute of Standards and Technology (NIST), including documentation, is available for download at \url{http://www.itl.nist.gov/iad/mig/tools/}.
Apart from Bennett’s somewhat systematical approach to the topic, there are plenty of application-specific solutions. For example, very recently Barna [16] used a “measure of uncertainty, expressed as a difference between the similarity with the closest prototype and the dissimilarity from the second closest prototype” (p. 617f) in a camera system.

RBF classifiers can be trained with labels having values in [0,1], however they can give answers well outside this range. Instead of cutting the output, one might also convert too high or too low values into certainty values.

In their semi-supervised learning procedure, Bruzzone and Marconcini ([41], p. 63650Y-6) make the assumption that “unlabeled samples closest to the margin bounds have the highest probability to be correctly classified.”

A certainty value can also be calculated based on statistics about the classifier’s accuracy, collected during the training phase or on a separate validation set. An overview of possible accuracy measures is presented just above in Chapter 5.2. One early application of this principle is the confidence mapping of Schürmann ([226], p. 151). He used eigen- and fremd-histograms to visualise the answer-behaviour of single-class classifiers, and then produced a mapping based on the probabilities observed.

In their influential 1992 paper on methods for combining multiple classifiers [289], Xu, Krzyzak, and Suen also employed statistical information, representing the uncertainty via Dempster-Shafer belief assignments. Based on the crisp classifier answers on a validation set, they estimated the recognition rate $\epsilon^r_k$ and substitution rate $\epsilon^s_k$ for each of the $K$ classifiers. A basic probability assignment function was then constructed by defining

$$m_k(\theta_l) = \epsilon^k_r, \quad m_k(-\theta_l) = \epsilon^k_s, \quad m_k(\Theta) = 1 - \epsilon^k_r - \epsilon^k_s, \quad k = 1 \ldots K, \quad l = 1 \ldots L$$

with $\theta_l \in \Theta$ referring to one of the $L$ possible classes. The introduction of $m_k(\Theta)$ is necessary as the classifiers were allowed to reject samples. So, rejection rate and classifier accuracy of one classifier $k$ are all encoded in one basic probability assignment $m_k$. For more practical applications of the Dempster-Shafer theory, see Chapter 3.7.2.

Own experiments were conducted using a clustering approach: the answers of a classifier to all training samples were clustered using k-means, and for each cluster the (training) accuracy was calculated as certainty-factor to be used for all answers that fall into this cluster. A classifier fusion experiment based only on this principle could not, however, demonstrate the usefulness of this approach.

One experiment especially illustrates the problem of class-dependent classifier behaviour, not only related to the correctness: on the land cover mapping dataset (see Chapter 2.4), the classifiers were allowed to reject mixed pixels, that is pixels that were not predominantly in favour of one class. The rejection threshold was
successively raised. Two classifier architectures were under consideration here, One-vs-One and One-vs-Rest $F^2$ SVMs, each with a linear kernel. Figure 5.2 shows for every class the fraction of pixels retained. Basis is the whole picture, the classifier answers are hardened to determine the class membership. The water class is the one the classifiers are most sure about, giving more unambiguous answers, which also corresponds to the fact that this class has the highest recognition rate. On the other end, pixels classified as urban green are rejected very early, which is not surprising as they are per definition a mix of vegetation and urban structures, hence the membership in a label will be distributed among multiple classes. Also, it is interesting to observe in which range the maximum membership value assigned by a classifier architecture is situated. In the One-vs-One case, it is $[0.25, 0.78]$, and $[0.48, 1]$ for One-vs-Rest SVMs. This means

**Figure 5.2:** In the land cover mapping application, mixed pixels are allowed to be rejected if the maximum membership is below the threshold (x-axis). The class-wise rejection rates (y-axis) are plotted for One-vs-One (top) and One-vs-Rest (bottom) SVM classifiers with linear kernels.
that in the first case, membership is never primarily attributed to one class, a fact which can in part be attributed to the coupling procedure (see Algorithm 1). But, unlike with the SOM classifier (compare Figure 4.6), the rejection curves behave smoothly, which easily allows for setting a rejection threshold separately for each class. For a study into the relation of rejection rate and overall accuracy, see the experiments summed up in Figure 4.7.

<table>
<thead>
<tr>
<th></th>
<th>sum rule</th>
<th>max rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>overall accuracy</td>
<td>91.4%</td>
<td>91.2%</td>
</tr>
<tr>
<td>accuracy on not-rejected</td>
<td>92.2%</td>
<td><strong>99.3%</strong></td>
</tr>
<tr>
<td>accuracy on rejected</td>
<td>89.7%</td>
<td>75.5%</td>
</tr>
<tr>
<td>not-rejected pixels</td>
<td>425</td>
<td>416</td>
</tr>
<tr>
<td>rejected-pixels</td>
<td>203</td>
<td>212</td>
</tr>
</tbody>
</table>

Table 5.1: Results of the sample-rejection experiment using two different rules, sum and max. The accuracy is calculated over the whole test set and also separately for those pixels that have been rejected and those that were not.

The rejection experiments carried out so far work on labels that are normalised to sum up to one, and the rejection criterion is based on the entry having the maximum membership. However, this choice is not the only possible one. As mentioned above, the non-normalised classifier answers could carry important certainty information. Thus, on the land cover mapping data set we experimentally compared two non-normalised approaches: rejection based on the sum of memberships in a label (sum rule), and rejection based on the maximum membership entry (max rule). In both cases, pixels with values below or equalling the threshold were rejected. The classifier for this experiment was a One-vs-Rest SVM (RBF-kernel, \( C = 100 \)). We chose a rejection threshold level where about the same number of samples was rejected for both approaches\(^8\). The results are summarised in Table 5.1, detailed confusion matrices are available in Tables 5.2 and 5.3. It can be very clearly seen that the max rule is much better for detecting exactly which classifier outputs are erroneous, with the accuracy on the not-rejected samples attaining 99.3%. Looking at the confusion matrices for not-rejected pixels, it is evident that the max rule rejected all (but one) pixels belonging to the urban green class, which in itself is a mixed class. The sum rule rejected only a portion of these pixels, but unfortunately kept nearly all those urban green pixels which were missclassified.

So, at least for this One-vs-Rest SVM classifier, the rejection based on the max rule has to be preferred. The additional normalising we generally employ is just a linear scaling and thus our standard rejection method (see Formula 4.26) is advocated by the outcome of this experiment.

\(^8\)The number of pixels rejected differed by 9, which can maximally account for a performance difference of 2 percent points.
Table 5.2: Confusion matrices for rejected pixels. (left) sum rule, (right) max rule;
The groundtruth is given on the lines. Classes are (1) vegetated land, (2) built up area, (3) pine wood, (4) urban green, (5) greenhouse, (6) not vegetated land and (7) water.

Table 5.3: Confusion matrices for not rejected pixels. (left) sum rule, (right) max rule;
The groundtruth is given on the lines. Classes are (1) vegetated land, (2) built up area, (3) pine wood, (4) urban green, (5) greenhouse, (6) not vegetated land and (7) water.

Now that it is possible, with the techniques presented above, to measure the uncertainty associated with a classifier answer, be it through statistical means, knowledge of the inner workings of the classifier or solely based on the structure of the answer, the output of several classifiers can be combined. The combination process can take the uncertainties into account, and produce a final answer that has a higher (fuzzy) accuracy. The techniques and considerations for performing this classifier combination are presented in the following chapter.
6 Multiple Classifier Fusion

The basic idea of multiple classifier systems (MCS) is to train various diverse classifiers for the same task, then fuse their answers to create a more precise and more robust final classification decision. The research field of MCS is well established, with the annual International Workshop on Multiple Classifier Systems, established in 2000, as a focus for the community, the proceedings\footnote{The proceedings of the MCS are published within the LNCS series at Springer.} reflecting the current research trends.

There are several reasons why multiple classifier systems are used. One is the need for integration, as the features available for the current application may be presented in diverse forms that are impossible to treat by the same classifier, a point much emphasised in [289]. Prime example is the existence of various time scales. 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 [195]. Variance reduction of the answers is a consequence of such a system, but also a goal in itself. Perhaps the most obvious motivation for the combination, and also central for this work, is the desire to boost classification accuracy: combining classifiers with different errors [6], or combining local experts, will have that effect.

For anyone interested in the field, Ludmilla Kuncheva’s book “Combining Pattern Classifiers - Methods and Algorithms” [148] is heartily recommended, giving a well-founded introduction and covering most relevant topics in depth.

As already mentioned in the introduction to Chapter 4, and emphasised by Didaci and Roli [60], multi classifier systems are central to the field of semi-supervised learning, especially in co-training. The basic idea there is to make use of the unlabelled data that abounds in some applications to improve the classification accuracy. The procedure is as follows: in a supervised manner, classifiers are trained on different views of the data. Different views most of the times means different feature sets [2], but sometimes also different classifier principles on the same feature set, as in democratic co-learning [300]. Then, unlabelled samples
are presented to the architecture, the classifiers’ decisions on each sample are fused, the final answers then attached to the before unlabelled samples. The new labels with the least uncertainty are then added to the labelled training set\(^2\), thus hopefully boosting the classifier performances in the next iteration of this procedure. Measuring the certainty of the labels can be accomplished using indicators presented in Chapters 3 and 5, for example [2] employs the Dempster-Shafer theory (following [75]), while the fusion in [300] is based on confidence intervals of the individual samples. It should be clear by now that the foundations needed for an involved multi classifier co-training system are all laid out in detail in this thesis. Future research in the co-training area, which is already underway in our institute, can and will be founded on this work.

Another direction of research in multi classifier systems is boosting, with Freund and Schapire’s AdaBoost [81] as the most well-known algorithm. It is based on the notion of weak classifiers, that is classifiers whose accuracy is just slightly above a random guess. By combining these, a stable accurate classification result can be obtained. AdaBoost works by successively building the classifiers, increasing the weight of missclassified samples in the training data for the next iteration. This is different from the technique of bagging [36] (short for bootstrap aggregation) where the bootstrap training samples for the various classifiers are drawn at random from the whole training set. Both approaches work on a single feature set.

The concept of weak classifiers is somewhat counter-intuitive (one would strive for very strong classifiers at first glance), and closely related to the topic of diversity: weakening classifiers on purpose can boost diversity, and such the final accuracy. Diversity, of course, refers to error diversity, with classifiers making independent errors in the optimal case. The central question here is to measure diversity, in such a way that it can guide the classifier creation process in producing a committee of classifiers that complement each other. Many diversity measures have been proposed, among them the interrater agreement \(\kappa\) and Kohavi-Wolpert variance (see [143] for an introduction and comparison). However, there is no consent about the optimal measure yet, and new ones continue to be proposed, for example by Kuncheva in her article “That Elusive Diversity in Classifier Ensembles” [147], drawing on measures established in biology.

A good example that very diverse classifiers can lead to an improved final classification result, even if the accuracy is already very high in the single classifiers, arose in the 2008 IEEE GRS-S data fusion contest (details can be found in Chapter 2.4). Combining the classification verdicts of the five best teams, using simple voting, with 99.21\% the final accuracy increased well over the one of the single best approach (98.84\%).

\(^2\)There are many variants as to which training set the new labels are added, for example to a common pool or only to one classifier that was left out for this purpose.
6.1. Fusion Schemes

There is another challenge in classifier combination that has to be addressed, which arises when fusing classifiers based on different algorithms: the nature of the answers varies depending on the underlying principles. For example, a RBF classifier will have a far more fuzzy answer than a KNN one. This issue is very well dealt with for example by using the fusion function we call pseudoinverse.

In the remainder of this chapter, first various classifier fusion schemes will be presented and compared. Then these schemes will be evaluated experimentally, looking for excellent candidates for hard and soft fusion. In the end, some ideas for future research in classifier fusion will be outlined.

6.1 Classifier Fusion Schemes

A multitude of classifier fusion schemes exists, and while many will be mentioned here and their basic ideas explained, the reader is directed to the literature mentioned in the introduction for details. The wealth of fusion approaches somehow needs to be categorised, and one attempt at this is the hierarchy proposed by Kuncheva \[145\] (see Figure 6.1). The top-level distinction with her is between hard and soft decisions. With soft decisions, when constructing the final answer concerning a certain class, class-conscious schemes will only look at that part of the classifier answer that concern this class, while class-indifferent ones take into account the whole answers. The class-conscious group can be further subdivided into (non-)linear models and order statistics. It is evident that this hierarchy is not sufficient\(^3\), with especially the class-indifferent category needing more subgroups, but it serves well as an overview categorisation.

Another distinction of classifier fusion algorithms is if they only look at the current replies of the classifiers (static) or if they take into account information from the training or validation phase (trainable). One representative of static fusion is the average function depicted in Figure 6.2. There are \(K = 3\) different classifiers \(C_{li}\), and for the current sample each of them gives an estimation \(y_i\) for the true class label, that is a verdict to which of the \(L = 4\) classes the sample belongs to which degree. The final fused classification answer \(z\) is calculated by taking the column-wise average\(^4\), and normalising the result to sum up to one. In terms of the function hierarchy shown in Figure 6.1, average fusion is a class-conscious scheme operating on soft decisions, more precisely a linear model with fixed (even) weights.

The fusion functions of minimum, maximum and median are order statistics,

\(^3\)Being from the Institute of Neural Information Processing, I have to challenge the inclusion of neural network fusion schemes as class-conscious nonlinear model. In fact, neural networks can be used in almost any category, most naturally for example as class-indifferent brute force approach.

\(^4\)The terms average fusion and mean fusion are used synonymously in this work.
class-consciously operating only within the column pertaining to one class. Applying minimum fusion to the example in Figure 6.2 would yield an answer $z = (0.1, 0.1, 0.1, 0.1)$, with $z = (0.25, 0.25, 0.25, 0.25)$ after subsequent normalisation. A representative of a non-linear model is the product function, with the entries of one column simply being multiplied (in the example leading to $z = (0.02, 0.006, 0.012, 0.004)$ before normalisation). In contrast to the fusion functions so far mentioned, voting operates on hard decisions, meaning each classifier is allowed to vote for one class only, and the class that receives the most votes is selected as final answer.

Another breed of fusion functions are the trainable ones, which learn from answers given in the training phase. In the following, four such functions will be presented, before showing that at their heart, they all have the same element: the confusion matrix $W_i$ [229]. To fix the notation, let $\Omega = \{1, \ldots, L\}$ be a set of class labels and $y_{i\mu} = Cl_i(x^\mu)$ the probabilistic classifier output of the $i$th classifier $Cl_i$ concerning the sample point $x^\mu$, with the classifier answer being of the form $y := (y_1, \ldots, y_L) \in [0; 1]^L$, $\sum_{l=1}^L y_l = 1$. The answers of the classifiers to all the $M$ samples in the training set are then recorded in the $L \times M$ matrices $C_i$, where the $\mu$th column of $C_i$ contains the classifier output of classifier $i$ concerning the sample $x^\mu$, hence $(y_{i\mu})^T$, with $T$ being the transposition operation. The desired outputs $\omega^\mu \in \Omega$ for each training sample are given by the $L \times M$ matrix $Y$ defined by the 1 of $L$ encoding scheme for class labels,

$$Y_{i,\mu} = \begin{cases} 1 & \text{if } l = \omega^\mu \\ 0 & \text{otherwise} \end{cases}.$$
That is, corresponding to the \( C_i \), the \( \mu \)th column of \( Y \) contains the binary coded target output vector for the \( \mu \)th training sample. In the following, we will present four different approaches to combine the classifier outputs \( C_i \) into an overall classifier decision, \( z = F(C_1, \ldots, C_K) \).

In all the methods, the fusion mapping \( F \) is realised through \( L \times L \) memory matrices \( V_1, \ldots, V_K \) (calculated by a training algorithm), which all have the confusion matrix \( W_i \) at their core.

The fusion mapping may be realised through an *associative matrix memory* whose error-correcting properties have been shown in several numerical experiments and theoretical investigations \[136\]. In order to calculate the memory matrix \( V_i \) for each classifier \( Cl_i \), the stored classifier outputs \( C_i \) are adapted through a Hebbian learning rule \[204\], and \( V_i \) is given as the product of the classifier outputs and the desired final answers \( Y \):

\[
V_i := Y C_i^T \Bigg\uparrow \hspace{1cm} (6.1)
\]

In the case of crisp classifiers, the matrix \( W_i \) is a standard *confusion matrix* of classifier \( Cl_i \), that is the entry \( W_i(r, k) \) in the matrix reflects how often samples of true class \( r \) have been classified as belonging to class \( k \) by the classifier, as determined in the training phase or on a validation set. For soft classifier decisions, the \( \omega \)th row of \( V_i \) contains the accumulated soft decisions of \( Cl_i \) on samples from class \( \omega \). For ideas on a completely fuzzy-based confusion matrix, see Binaghi’s \[23\] fuzzy error matrix presented in Chapter 5.2.
Chapter 6. Multiple Classifier Fusion

In the classification phase, the memory matrices are then used to obtain a more correct answer from each classifier, 

\[ z_i = V_i(\text{Cl}_i(x^{new}))^T, \]

before all \( K \) classifier decisions are fused by simply aggregating them,

\[ z = \sum_{i=1}^{K} z_i = \sum_{i=1}^{K} V_i(\text{Cl}_i(x^{new}))^T. \]  

(6.2)

The final hard class membership is then determined by applying the maximum membership rule to \( z \).

The concept of decision templates is a simple, intuitive and robust aggregation idea that evolved from the fuzzy template which was introduced by Kuncheva [146, 144]. Basically, a decision template \( T_i^\omega \) is the mean over all classifier outputs, collected by feeding all training samples of class \( \omega \in \Omega \) to the classifiers. For one classifier, this is

\[ T_i^\omega = \frac{1}{|R^\omega|} \sum_{x \in R^\omega} \text{Cl}_i(x). \]

Here, \( R^\omega \) is the set of those training samples that belong to class \( \omega \). The complete decision template \( T^\omega \) over all classifiers, for the class \( \omega \), is then given by the \( K \times L \) matrix

\[ T^\omega := \begin{bmatrix} T_i^\omega \\ \vdots \\ T_K^\omega \end{bmatrix}. \]

In the classification phase (illustrated in Figure 6.3), the idea is now to simply produce a decision profile \( T^{new} \) for the new sample, by feeding the sample to all classifiers, and then to compare \( T^{new} \) to the class-specific decision profiles \( T^\omega \) created in the training phase. The more similar \( T^{new} \) and a \( T^\omega \) are, the more likely they refer to the same class. Of the many similarity metrics available, \( S_1 \) (see Formula 4.5) seems to be preferred (see citations above). In the context of bioacoustic time series classification, but also useful simply to boost the final accuracy, an extension has been suggested that builds several decision templates per class, for example via clustering [62].

To show the common ground between the classifiers, the fusion function will now be reformulated. The basis is again the \( L \times L \) matrix \( V_i \), a decision template for each classifier, based on the class-wise decision templates of each classifier, \( T_i^\omega \):

\[ V_i := \begin{bmatrix} T_i^1 \\ \vdots \\ T_i^L \end{bmatrix}. \]

Similar to the associative memory approach, \( V_i \) can also be computed using the stored classifier outputs, and the desired ones:

\[ V_i = (YY^T)^{-1} \left( YC_i^TF \right)_{W_i} \]  

(6.3)
6.1. Fusion Schemes

Figure 6.3: Illustrating the decision template fusion process. The collected classifier decisions $T^{\text{new}}$ pertaining to the new sample are compared to the class-wise decision profiles $T^i$, the comparison distances $z_i$ then being aggregated into a final output $z$.

Multiplying the confusion matrix $W_i$ by $(YY^T)^{-1}$ from the left is equivalent to the row-wise normalisation of $W_i$ with the number of training patterns per class. Now, assuming the normalised correlation as similarity measure, the decision templates fusion is equivalent to linear associative memory fusion, the final answer $z$ being calculated according to Formula 6.2.

Another fusion mapping can be calculated as an optimal least squares solution of the distance between the confusion matrix from the training phase and the respective training labels. The pseudoinverse approach, discussed in more detail at Formula 4.2, yields such a solution. The linear matrix operator $V_i$ for the fusion is given by

$$V_i = Y \lim_{\alpha \to 0^+} C_i^T (C_iC_i^T + \alpha I)^{-1} = YC_i^+$$

where $I$ here represents the identity matrix, and $C_i^+$ signifies the pseudoinverse of $C_i$. Provided the inverse matrix of $C_iC_i^T$ exists, which is always the case for full rank$^5$ matrices $C_i$, the matrix operator for the pseudoinverse solution is given with

$$V_i = YC_i^T (C_iC_i^T)^{-1} = \underbrace{(YC_i^T)}_{W_i}(C_iC_i^T)^{-1}.$$  \hspace{1cm} (6.4)

As with the decision templates and the associative memory mapping, the final answer, consisting in a linear mapping of the classifier outputs, is obtained by multiplying the outputs with the $V_i$, as detailed in Formula 6.2.

If the classifiers are assumed to be mutually independent and giving hard answers, naive Bayes fusion can be employed. The matrix operator would be calculated

$^5$For the case of hard classifier outputs, with at least one sample attributed to each class, it is easy to see that $C_iC_i^T$ is a diagonal matrix with full rank.
as follows:

\[
V_i = \frac{(Y C_i^T)(C_i C_i^T)^{-1}}{w_i}.
\]

(6.5)

In contrast to Formula 6.3 for decision temples, the confusion matrices \(W_i\) are normalised column-wise, the \(V_i\) being called label matrices. The entry \(V_i(r, s)\) thus represents an estimate for the conditional probability

\[
\hat{P}(\omega = r|C_i(x^\text{new}) = s),
\]

that is the probability that the true class label is \(r\) given that \(C_i\) assigns the crisp class label \(s\) to a new sample \(x^\text{new}\). Using the independence assumption, the output \(z_i\), representing an estimate of the probability that the true class label of \(x^\text{new}\) is \(i\), is derived as

\[
z_i = \alpha \prod_{n=1}^{K} \hat{P}((\omega = i)C_n(x^\text{new}) = y_n),
\]

thus as an element-wise product depending on the \(K\) hard classifier decisions \(y_n\), with the normalising constant \(\alpha\) ensuring that \(\sum_{i=1}^{L} z_i = 1\).

<table>
<thead>
<tr>
<th>Matrix (V_i)</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Associative memory</td>
<td>((YC_i^T))</td>
</tr>
<tr>
<td>Decision templates</td>
<td>((YY^T)^{-1}(YC_i^T))</td>
</tr>
<tr>
<td>Pseudoinverse</td>
<td>((YC_i^T)(C_i C_i^T)^{-1})</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>((YC_i^T)(C_i C_i^T)^{-1})</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.1: Overview of the central matrix \(V_i\) in the different fusion approaches, and what to do in each case to get the classification result \(z\), which is an estimate for the class memberships. The final decision in the hard case is obviously for the class with the highest probability.

The close relationship between the four fusion schemes just presented will be explored more closely, an overview is given in Table 6.1. For an even more in-depth look, see our discussion in [229]. Obviously, all fusion mappings are realised using \(L \times L\) matrices \(V_1, \ldots, V_K\) which were obtained in the training phase, looking at the classifier outputs and the desired outputs. As demonstrated, the confusion matrix \(W_i\) (see Formulas 6.1, 6.3, 6.4 and 6.5) is the main ingredient in all the combination schemes, capturing the class-specific uncertainties of the classifier, or differently put, representing prior knowledge about the classifier [289]. With the exception of naive Bayes fusion, the combination function is a matrix-vector product (compare Formula 6.2). If the training labels are fuzzy, only the pseudoinverse scheme can readily be employed.
6.1. Fusion Schemes

Another approach to classifier fusion, dubbed brute force or stacked generalization [286], is to add a second classifier layer. The answers of the basic classifiers form the feature space for the next-level algorithms, which will learn to map the classifier answers to the training labels. Due to the generic nature of the approach, all classifiers can be employed in the second layer. In the fusion process, the classifiers in the second layer have no concept of classes in their feature space, illustrating how the term class-indifferent (see the fusion hierarchy in Figure 6.1) came into being. In our experiments on emotion recognition from facial expressions, a fusion layer consisting of $P^2$-support vector machines (with RBF kernels) was able to improve somewhat over the best accuracy obtained using other fusion techniques.

So far, the fusion schemes did not take into account explicit certainty information concerning the current sample. Such information could be derived from the structure of the answer, the quality of the data, or any other method detailed in Chapter 5. Kryszczuk and Drygajlo [141] include a quality measure (derived from the known noise level on the samples) into the fusion process, by simply appending it to the classifier answer vector. Using the Bayesian framework, Poh, Heusch, and Kittler [191] include their quality measures $q$ into a log-likelihood test, with the original classifier score $y$ (distinguishing between two classes, $A$ and $B$) being adjusted:

$$y_{\text{norm}} = \log \frac{p(y|A, q)}{p(y|B, q)}$$

Using the brute force fusion approach, any of the numerous classifiers that can take into account certainty weighted labels (see Chapter 4.6) can quite naturally accomplish classifier fusion dependent on certainty information.

Many methods to fuse uncertain classifier decisions have already been presented earlier in this work, with the uncertainty theories in Chapter 3 generally dealing with the combination of uncertain knowledge. Some examples are the conjunctive and disjunctive combination rules in possibility theory and fuzzy logic (Formulas 3.18 through 3.23), the Bayes theorem (Formula 3.3), and the many choices listed in Chapter 3.3 on the distribution of opinions. The last mentioned framework also extensively deals with the issue of uncertainty that has arisen during previous combination steps, showing possibilities how to update appropriate statistics.

Numerous practical applications of the Dempster-Shafer theory to classifier fusion have already been reported in Chapter 3.7.2, so in the following some works that concentrate generally on the fusion part of the process are presented. A very important contribution here is from Rogova [205]. Her idea is to calculate evidences based on the classifiers’ answers, then to transfer this evidences into basic probability assignments. Deriving the evidences is very close to the concept of decision templates (see Figure 6.3). First, for each classifier $c$ the mean classifier
answer to training samples from class $l^6$ is calculated. Then, if a new sample is presented, the classifier answers are compared to the mean templates generated, and the resulting distances $d^l_c$ noted. The distance metrics found to work best are based on the Euclidean distance or the cosine of the angle between the two vectors. The distances $d^l_c$ are transformed into bpas $m$ in such a way that even the information concerning the other $\neg c^7$ classes is taken into account:

$$m^l_c(\theta_c) = d^l_c, \quad m^l_c(\Theta) = 1 - m^l_c(\theta_c)$$

$$m^{l,-c}(-\theta_c) = 1 - \prod_{i \neq c}(1 - d^l_i), \quad m^{l,-c}(\Theta) = 1 - m^{l,-c}(-\theta_c)$$

The bpas concerning each class are combined (using the orthogonal sum), then also $m_c$ and $m_{-c}$, with the final decision being made for the class $c$ with the highest pro-$c$ evidence. Kuncheva, Bezdek, and Duin included this approach in their experimental comparison [144] of decision templates (DT) with other, established classifier fusion 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” (ibid., p. 312).

Al-Ani and Deriche came up with a new variant [5]. Their idea was to improve on the way Rogova calculated the evidences and basic probability assignments. The mean per-class-per-classifier answer reference vector is not fixed any more, but adjusted via an iterative gradient descent technique to minimise the mean square error between final fused answers and the training labels. Alongside the reference vectors, the so-called value of ignorance of each classifier is optimised, being the basis for the assignment of belief to the whole frame of discernment $m_c(\Theta)$.

This is quite close to the Dempster-Shafer classifier fusion scheme we proposed in [261] and detailed in [257]. There, $m(\Theta)$ was used to indicate doubt in the final fused bpa $m$. Using discounting, the belief assigned by a classifier was scaled down, and the rest given to $m(\Theta)$, with the strength of the discounting depending on a certainty factor calculated directly from the fuzzy answers, for example based on the Shannon entropy (find more details in Chapter 5.1). The other twist we proposed was to collect a measure for the amount of conflict that arises in the successive bpa fusion steps, assigning it to $m(\emptyset)$. Calculating the conflict was as simple as not normalising the basic probability assignments after a fusion step, that is eliminating factor $K$ in Formula 3.32, then assigning the belief missing to the sum of 1 to the empty set. This procedure is motivated by reasoning of Smets [243] in the framework of the transferable belief model. In our experiments (see references above) we used $m(\Theta)$ and $m(\emptyset)$ as indicators to reject samples that a multi classifier architecture obviously was not sure about, significantly increasing the accuracy on the not-rejected samples.

---

6Rogova used another notation, $k$ for the classes and $n$ for the classifiers, we changed this for the sake of consistency in this work.

7The term $\neg c$ is short for “not $c^*$”, that is all classes that do not have the class number $c$. 
An example where the Dempster-Shafer theory really shines is the collection of evidence in trees of classifiers. Fay showed [73, 75] that in her tree of RBF classifiers, evaluating and fusing all nodes using DS yielded a higher accuracy than the simple decision-tree like approach that follows the most probable path. Sarinnapakorn and Kubat work in the [216] area of text classification. Each of their numerous tree-involving classifiers is based only on a subset of features (that is, words), the tree-answers then being combined using Dempster-Shafer theory, taking into account the class-dependent classifier accuracies. The combination of trees can also be achieved using fuzzy sets. In [54] the crisp nodes are transformed into a tree of fuzzy if-then rules, the combination then accomplished using a fuzzy inference technique.

Another application of the Dempster-Shafer theory is in the combination of association rules [281]. Normally, association rules with “high support” (that is, support higher than a threshold) are treated the same. Improving on that, belief masses are now assigned such that higher support and higher accuracy will lead to higher masses for the rule. Discounting is employed to adjust the basic belief assignments based on the uncertainty and nonspecificity in the rule antecedent.

If the classifier answers consist in ranks, the fusion schemes described so far are of no use. A ranked answer expresses the preferences for different classes as an ordered list of these classes, often encoded as integers, giving the highest number to the class the classifier thinks is most probable. For example, with $L = 3$ classes, label $y = [0.1, 0.7, 0.3]$ would have the associated rank vector $ra^y = [1, 3, 2]$. The Borda count is an established and quite simple method to fuse rank-based decisions, it works by summing up for each class the rank positions assigned in the rank vector. The class with the highest sum then is the most probable one. Based on Friedman’s procedure, Tubbs and Alltop [264] justify the usage of the Borda count (without mentioning it by name), before going further to create a measure of confidence for the combined result: much simplified, the rank answers are converted into random variables, whose distribution is then assessed. Another extension by Ho, Hull, and Srihari [104], based on logistic regression, adds a training-set-optimal linear weighting to the Borda rank summing procedure. An in-depth explanation of the most common rank fusion methods can be found in Saranli’s dissertation [215], where he also details a unifying theory for rank-based multi classifier systems, treating the fusion as a discrete optimisation problem.

If the multi classifier setting deals also with multiple classes, but the base classifiers allow only pairwise comparisons (as is the case for support vector machines), the problem arises how to combine the pairwise results to a class-probability output vector, taking into account all information and not just votes. In this situation, the preferred solution should be Bradley-Terry coupling, as detailed in Algorithm 1.

---

8See also Footnote 3 in Chapter 5.
The classification of time series is another scenario that naturally requires classifier fusion. All the time frames classified separately shall contribute to a final answer. Taking the average decision over the time series is straightforward and rather stable, but fails in cases where there is a class-shift or offset in the time series. The latter issue can be addressed by letting the features control the fusion process, as we proposed in Chapter 2.1.6 using motion energy estimation. Dietrich [61] searched for the right fusion level, for example if the decisions should be combined at the frame level, or afterwards when each feature-specific time-series has already been combined. His application was to determine the subfamily name of orthoptera from recorded songs, concentrating mainly on tettigoniidae (crickets). The research was continued by us [203] and later extended to gryllidae (katydids), which are even harder to classify, requiring to extract features on four different time scales with different frame sizes. In our own (unpublished') experiments, fusion between differing time scales was done as the last step. Another possibility to deal with time series data are hidden Markov models [199], which, as Markert ([166], p. 12) notes, “are almost certainly used by any speech recognition software that one might encounter today.” Recent experiments of ours in the realm of emotion recognition in speech show that the technique of GMM supervectors [45] constructed from maximum a-posteriori-adapted (MAP [22]) Gaussian mixture models works quite well, despite the fact that it ignores the temporal order in the time series.

6.2 Experiments

Of course many of the fusion techniques described above were applied to our application scenarios. Besides striving for a good recognition rate in the application, a number of issues were researched experimentally: Is a fusion of the classifier decisions really beneficiary in all situations, even when adding noise? Which fusion method is the best to choose?

At first, the merit of classifier fusion is evaluated in a situation where noise is

\footnote{To give interested researchers a baseline, our experimental results are briefly summarised here: All data was obtained in collaboration with the database Digitized Orthoptera Specimens Access (DORSA), http://www.dorsa.de, whose samples are available via the SysTax database at http://www.biologie.uni-ulm.de/systax/. Two datasets were created, the first containing 19 different subfamilies recorded solely in Greece, the encompassing bigger consisting of 44 subfamilies with songs recorded in Greece and several other countries. For each subfamily, there were at least 5 song recordings, the results are based on 5-fold cross validation. The simple multi classifier architecture that worked best for us was a CDTD approach, that is classification of each frame (using KNN), then a fusion between those time series with the same timing on a frame basis (using average), then a compression of each resulting time series (using average), followed by a fusion of the final answers of the temporally different time series (using decision templates with $S_1$ measure, average performed worse). The resulting accuracy was 66.3% for the smaller set, and 56.6% for the bigger one.}
6.2. Experiments

<table>
<thead>
<tr>
<th>Fusion \ Noise</th>
<th>0</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum</td>
<td>75.4</td>
<td>74.9</td>
<td>76.9</td>
<td>76.2</td>
<td>76.3</td>
<td>76.9</td>
<td>76.2</td>
<td>76.2</td>
<td>72.9</td>
<td>52.8</td>
<td>15.4</td>
</tr>
<tr>
<td>Minimum</td>
<td>75.4</td>
<td>75.2</td>
<td>76.0</td>
<td>76.1</td>
<td>77.0</td>
<td>75.4</td>
<td>74.8</td>
<td>75.6</td>
<td>71.8</td>
<td>55.2</td>
<td>15.5</td>
</tr>
<tr>
<td>Median</td>
<td>78.9</td>
<td>78.1</td>
<td>78.7</td>
<td>77.9</td>
<td>79.0</td>
<td>79.5</td>
<td>79.2</td>
<td>78.2</td>
<td>74.2</td>
<td>61.8</td>
<td>15.6</td>
</tr>
<tr>
<td>Average</td>
<td>78.2</td>
<td>78.2</td>
<td>78.7</td>
<td>77.5</td>
<td>79.3</td>
<td>79.4</td>
<td>78.6</td>
<td>77.9</td>
<td>74.5</td>
<td>60.0</td>
<td>15.6</td>
</tr>
<tr>
<td>Product</td>
<td>78.1</td>
<td>78.6</td>
<td>78.2</td>
<td>78.3</td>
<td>78.7</td>
<td>78.7</td>
<td>78.3</td>
<td>78.0</td>
<td>74.9</td>
<td>59.8</td>
<td>15.4</td>
</tr>
<tr>
<td>DS-discount.</td>
<td>77.0</td>
<td>77.4</td>
<td>77.6</td>
<td>77.4</td>
<td>78.3</td>
<td>78.3</td>
<td>77.5</td>
<td>77.6</td>
<td>74.3</td>
<td>56.0</td>
<td>16.0</td>
</tr>
<tr>
<td>Dec. Templ.</td>
<td>78.1</td>
<td>78.5</td>
<td>78.9</td>
<td>78.3</td>
<td>78.8</td>
<td>79.1</td>
<td>79.6</td>
<td>78.7</td>
<td>80.0</td>
<td>59.4</td>
<td>15.0</td>
</tr>
<tr>
<td>Pseudoinv.</td>
<td>78.4</td>
<td>77.9</td>
<td>78.0</td>
<td>77.4</td>
<td>77.9</td>
<td>78.2</td>
<td>78.0</td>
<td>76.7</td>
<td>73.3</td>
<td>59.4</td>
<td>15.0</td>
</tr>
<tr>
<td>Best classif.</td>
<td>76.9</td>
<td>75.6</td>
<td>77.0</td>
<td>75.4</td>
<td>76.4</td>
<td>74.5</td>
<td>77.3</td>
<td>74.3</td>
<td>68.9</td>
<td>55.4</td>
<td>17.0</td>
</tr>
</tbody>
</table>

Table 6.2: Classification accuracy of a multi classifier system (5-fold cross-validated) once noise of a level from 0% to 100% is added to the training labels. The fusion function is given for each line. The base classifiers were trained with soft labels.

<table>
<thead>
<tr>
<th>Fusion \ Noise</th>
<th>0</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum</td>
<td>72.7</td>
<td>73.1</td>
<td>73.0</td>
<td>72.5</td>
<td>72.7</td>
<td>74.5</td>
<td>73.2</td>
<td>70.7</td>
<td>71.0</td>
<td>52.1</td>
<td>16.3</td>
</tr>
<tr>
<td>Minimum</td>
<td>73.5</td>
<td>73.1</td>
<td>74.2</td>
<td>74.8</td>
<td>73.4</td>
<td>74.6</td>
<td>75.1</td>
<td>71.4</td>
<td>67.7</td>
<td>53.9</td>
<td>14.4</td>
</tr>
<tr>
<td>Median</td>
<td>73.4</td>
<td>74.0</td>
<td>74.4</td>
<td>74.9</td>
<td>74.4</td>
<td>75.0</td>
<td>73.7</td>
<td>74.3</td>
<td>71.8</td>
<td>60.4</td>
<td>14.8</td>
</tr>
<tr>
<td>Average</td>
<td>73.6</td>
<td>73.6</td>
<td>74.4</td>
<td>74.6</td>
<td>73.6</td>
<td>74.5</td>
<td>74.4</td>
<td>73.3</td>
<td>72.5</td>
<td>59.3</td>
<td>15.0</td>
</tr>
<tr>
<td>Product</td>
<td>73.6</td>
<td>73.8</td>
<td>74.6</td>
<td>75.6</td>
<td>73.1</td>
<td>74.6</td>
<td>74.8</td>
<td>72.6</td>
<td>72.0</td>
<td>58.0</td>
<td>15.1</td>
</tr>
<tr>
<td>DS-discount.</td>
<td>73.6</td>
<td>72.7</td>
<td>73.9</td>
<td>73.3</td>
<td>73.1</td>
<td>75.0</td>
<td>73.4</td>
<td>71.4</td>
<td>71.7</td>
<td>55.8</td>
<td>14.8</td>
</tr>
<tr>
<td>Dec. Templ.</td>
<td>74.2</td>
<td>74.3</td>
<td>75.0</td>
<td>75.4</td>
<td>74.6</td>
<td>75.7</td>
<td>74.5</td>
<td>72.9</td>
<td>74.0</td>
<td>61.6</td>
<td>12.6</td>
</tr>
<tr>
<td>Pseudoinv.</td>
<td>73.6</td>
<td>73.8</td>
<td>75.4</td>
<td>75.5</td>
<td>74.6</td>
<td>75.4</td>
<td>75.1</td>
<td>74.2</td>
<td>71.5</td>
<td>63.7</td>
<td>15.5</td>
</tr>
<tr>
<td>Best classif.</td>
<td>71.8</td>
<td>72.0</td>
<td>72.4</td>
<td>73.1</td>
<td>72.1</td>
<td>74.2</td>
<td>72.7</td>
<td>70.5</td>
<td>68.2</td>
<td>57.4</td>
<td>17.9</td>
</tr>
</tbody>
</table>

Table 6.3: Classification accuracy of a multi classifier system (5-fold cross-validated) once noise of a level from 0% to 100% is added to the training labels. The fusion function is given for each line. The base classifiers were trained with hard labels.

added to the training labels (using Algorithm 2). The four-classifier-setup, operating on the fruits dataset and utilising soft-trained RBF networks and support vector machines, is described in detail in Chapter 4.7.2 and in [258]. Tables 6.2 and 6.3 show the classification accuracy which the system is able to achieve with various classifier fusion techniques, once label noise from 0% to 100% is added\(^{10}\). The mean performance of the best single classifier in the architecture, selected for every noise level, is given in the last row. The “Dempster-Shafer discounting” fusion is taking into account classifier accuracy\(^{11}\).

\(^{10}\)For the fusion architecture analysis, we disregarded the noise levels from 80% to 100%, for these cases are not relevant in practical applications and exhibit very volatile behaviour.

\(^{11}\)The fusion mode of Dempster-Shafer discounting incorporates several ideas. On one side, the answer of each classifier is discounted with its associated (normalised, inverted) Gini un-
Chapter 6. Multiple Classifier Fusion

A quick comparison reveals that the system trained on soft labels achieves a higher top accuracy (78.9%) than the one trained on hard labels (74.2%). This is an important result, as it demonstrates the usefulness of soft or fuzzy labels. This issue has already been explored in Figure 4.11 detailing the soft versus hard accuracy difference on the classifier level. The final accuracy remains high despite great levels of added label noise. This astonishing result is even more evident in the plot of Figure 4.10 in Chapter 4.7.2, where we also look at the underlying causes.

On the issue if it is a good idea to have a MCS with fusion, from the Tables 6.2 and 6.3 it is clear that the classification accuracy which the fusion functions yield is, with few exceptions for the maximum/minimum operations, higher than the accuracy of the best single classifier.

This is even more obvious in experiments on the JACFEE-Morph database (see Chapter 2.1.3), with the task of emotion recognition in facial expressions. The best single classifier\textsuperscript{12} there (see Table A.3 in [217]), a One-vs-One SVM on orientation histograms in the mouth area, had an accuracy of 70.8%, the second best only 56.2%. As mentioned in the application introduction, this bad performance is due to the morphed nature of the database: Since the video sequence was automatically generated based only on the neutral and the full-blown emotional expression, all emotions have the same duration, which gives them an unnatural feel when viewed. But, combining two good classifiers, the best one and a RBF on the optical flow in the face area, using pseudoinverse fusion yields an accuracy of 79.1%, which is a considerable increase. The next-best combination of classifiers, consisting of five contributors, still achieves 77.0 percent, but with a lower variance of 150 (as opposed to \( \sigma^2 = 290 \) for the optimal setup). Interestingly, the second fusion method tried, probabilistic product, could not improve over the single best classifier. On the Cohn-Kanade database, the best single classifier (see Table 2.3), incidentally the same as for the JACFEE-Morph dataset, has an accuracy of 74.0%, while the optimal combination of classifiers achieves 83.1% using probabilistic product fusion!

In the fruit recognition application (see Chapter 2.3, and [261, 257] for experimental details), with RBF networks trained on original hard labels\textsuperscript{13} as base classifiers, the best accuracy obtained is 83.3% on Sobel edge histograms (middle stripe).

\textsuperscript{12}As explained in Chapter 2.1.5, for JACFEE-morph the feature-facial area-classifier combinations deemed optimal for the Cohn-Kanade dataset were used.

\textsuperscript{13}Note that in the results reported in Tables 6.2 and 6.3, both soft and hard classifiers suffer from errors imparted in the label-fuzzification phase. This is why the classification rate reported here can be significantly higher on the same data set.
6.2. Experiments

A fusion of four classifiers using probabilistic product yields 95.4%.

As the experimental results above demonstrate, using classifier fusion to improve the final classification accuracy is a good idea, since a fusion function is always able to improve well over the accuracy of even the best single classifier.

The picture is not so clear when it comes to the issue which fusion function to pick. As just mentioned, on the two facial emotion databases, the product fusion was for one the best fusion method, while on the other set it could not improve over the champion classifier. In the following, further experimental results will be reported, trying to establish some preference order among the fusion schemes.

<table>
<thead>
<tr>
<th>fusion function</th>
<th>accuracy %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum</td>
<td>73.8</td>
</tr>
<tr>
<td>Minimum</td>
<td>77.8</td>
</tr>
<tr>
<td>Average</td>
<td>81.7</td>
</tr>
<tr>
<td>Product</td>
<td>83.1</td>
</tr>
<tr>
<td>Decision templates</td>
<td>75.9</td>
</tr>
<tr>
<td>Pseudoinverse</td>
<td>81.4</td>
</tr>
<tr>
<td>Best classifier</td>
<td>74.0</td>
</tr>
</tbody>
</table>

Table 6.4: Classification accuracy for various classifier fusion schemes. The experiment was run on the soft-trained Cohn-Kanade dataset.

The most exhaustive experiment was run on the Cohn-Kanade dataset, the best results achieved for each fusion function are given in Table 6.4 (see also [218]). As detailed in Chapter 2.1.5, for each fusion scheme every possible combination out of the 14 base classifiers was tested under cross-validation. To examine the variation in accuracy of the combinations, we looked at the product fusion function and found that the 190 best constellations had performance differences of only around 3 percent points. These 190 combinations are only around 1% of the possible ones, but our experiments with sequential forward selection, described in the application chapter, confirm that exhaustive search is not necessary to find a good classifier combination.

To compare classifier fusion schemes, it is essential to make the distinction if the base classifiers have been trained on soft or hard labels. For the soft-trained case, the important results are in Table 6.4 (Cohn-Kanade data) and Table 6.2 (fruits data, at zero noise). Pseudoinverse, average/median and product fusion

\footnote{If a random subsection of the 14 features is chosen, pseudoinverse fusion significantly outperforms product fusion in more than 99% of the possible combinations [218]. The single best classifier is beaten in nearly 50% of the cases.}

\footnote{Average and median fusion have similar performances, hence they are grouped in this analysis.}
are good in both application scenarios. Decision templates fail on the Cohn-Kanade data, possibly because the emotional classes are inherently overlapping. So, despite good performance on the fruits dataset, decision template fusion cannot be recommended for soft-trained classifiers, especially since their logic is based on hard classes. Looking at what happens once noise is added to the training labels in Table 6.2 (the classification task thus becoming more difficult), further insights can be gained: surprisingly, decision templates hold up very well. The performance of the pseudoinverse falls considerably back behind the one of median fusion. As for the product rule, its use has to be discouraged entirely, since in some crossvalidation runs it had drops in accuracy of worse than 10 percent points. So, for the soft case, the safest classifier remains the average/median approach, and product fusion might be used for increased accuracy, with a careful lookout for unpredictable accuracy slumps.

For the hard-trained case, two more relevant experiments have to be reported, based on the crickets and fruits scenarios. On the fruits dataset\(^\text{16}\), four fusion algorithms were compared, the results \(5 \times 5\) cross-validated [229]. The winning fusion scheme was pseudoinverse (95.8% accuracy), followed by decision templates (95.7%) and associative memory (94.9%), the naive Bayes fusion only achieving 89.1%. In our cricket song recognition experiments (see Footnote 9), we made an experiment between average and decision template fusion as last step of the architecture. Decision templates clearly win, with an accuracy of 56.3% versus 55.0% on the big dataset and 66.3% versus 63.0% on the small dataset. Combining these results with the ones from the label noise experiment (Table 6.3), decision templates emerge as a clear winner, closely followed by pseudoinverse fusion. Looking at the results from the noise setup, additionally median and product fusion belong to the top group, with a maximum of only 6 percent points in accuracy difference between the four algorithms for all cross validation runs and noise levels.

6.3 Future Approaches

For an increased classifier fusion performance, several venues are open to be pursued in the future. We present some of our ideas for this, and a promising fusion paradigm.

Our experiments suggested that within each class there are subclasses, to which a classifier reacts differently. Therefore, the decision template fusion could be enhanced: the answer-space would be partitioned by clustering. To couple this with the good performance of pseudoinverse fusion, an optimal mapping would be

\(^{16}\)Four features were used, three Sobel edge orientation histograms calculated on the left, middle and right portion of the images, and the mean HSV colour information. Base classifiers are Gaussian RBF networks. Please see also Footnote 13.
6.3. Future Approaches

calculated for each region identified. Thus, a new sample would be fed to all classifiers, the closest cluster centre determined from their answers, and the answers then mapped to a final common one using the associated linear pseudoinverse-obtained mapping.

Different classifier architectures have a differing answer behaviour. So, measuring the distance between decision templates in a uniform way will lose some information. A remedy would be to weight the distance between the templates depending on the variance in each row.

Apart from dedicated fusion schemes, there is still brute force combination, introducing a second classification layer with the first-level classifier answers as feature space. There is active research in this field [201], among other things addressing the problem of overfitting. Own tentative experiments [218] with support vector machines yielded a slightly increased accuracy in the application of emotion recognition from facial expressions (83.6%), which is encouraging since the traditional fusion architecture implemented there is the result of extensive experiments.

A fusion method can also depend on the location of the current sample in features space. This is explored in the research field that deals with mixtures of experts [114]. In its most basic form, it consists of a linear combination of linear models, where the weight of each model, called expert, is determined by a separate classifier, the gating function. Recently, Kugler, Kuroyanagi, Nugroho, and Iwata [142] proposed an extension to their CombNET-III architecture, with support vector machines as experts, and multiple feature-space clusters affiliated with each SVM. Special attention was given to reducing the computation time of this fusion network.
The inclusion of uncertain class information into the process of classifier fusion with all its stages, as laid out in detail in the introduction, is the topic of this thesis. Having collected and put into context all the approaches and techniques that enable the treatment of uncertainty in multi classifier systems (MCS) is one of the major achievements of this work. This summary will highlight the most important theoretical contributions and the most interesting experimental findings, accompanied by references to my corresponding publications [75, 86, 87, 156, 203, 213, 218, 229, 261, 258, 259, 260, 262].

7.1 Theoretical Contributions

A wealth of formal theories for modelling different kinds of uncertainty has been described, assessing the applicability of each for classifier fusion systems. To what extent the core MCS uncertainty concepts of imprecision, vagueness and certainty are supported by the formalisms is summarised in Tables 3.2 through 3.4. The Dempster-Shafer theory of evidence can be applied most naturally, as it supports the three notions, and in addition excels in hierarchical combination tasks [75]. Due to their capability to model arbitrary concepts, fuzzy sets could be interesting in emotion-aware applications, but unfortunately the set-attribution mechanism is rooted directly in the feature level. This highlights an important difference, which is not self-evident in the literature: when applied to classification, the reference set \( \Omega \) in possibility theory contains the class labels, while with fuzzy sets, there is one fuzzy set per class, with \( \Omega \) spanning over all the possible feature values.

The prototypes produced by learning vector quantisation (LVQ) enable further analysis by domain experts. LVQ was expanded so that it can now learn from fuzzy labelled training data, preserving the prototype idea by associating the centres learned with fuzzy labels [262]. Distances between fuzzy labels can be assessed well using the \( S_1 \) measure. Self-organizing maps (SOMs), a semi-supervised
learning algorithm employed in settings with plenty but unlabelled data, was amended to produce individual fuzzy answers for each sample based on distance-weighting [87], going beyond the one-sample-one-cluster approach.

The extension of the already powerful support vector machine (SVM) classifiers is another major contribution. Based on the idea of duplicating the training samples, SVMs can now learn from fuzzy labelled data [260], while still being able to utilise the established optimisation procedures. The methods for obtaining fuzzy outputs from both the One-vs-Rest and the more accurate One-vs-One multi-class SVM architectures have been described and tested [260, 86, 87]. For One-vs-One SVMs, the voting approach propagated in the current literature as the combination function leads to a loss of a lot of information; instead a pairwise coupling of the machines based on sigmoid-transformed distances is required [86, 87]. On the same note, an isolated output-optimisation of the SVM couples will fail. An extension of one-class support vector machines to be trained on fuzzy labels has been outlined, with remedies for conflict- and outlier-situations [213].

In real-world applications, it is most likely that noise will be present in the training labels. To study its impact, defined levels of noise have to be imparted. Procedures to do so only exist for the case of hard labels, hence an algorithm to add noise to training labels is proposed [258].

For four popular classifier fusion schemes, namely decision templates, naive Bayes, linear associative memory and pseudoinverse, it was possible to show that at their heart, they are all based on the confusion matrix; depending on their focus, the fusion approaches then differ in the normalisation applied and, for naive Bayes, in the method utilised to obtain the final answer [229]. The Dempster-Shafer theory was employed to model the doubt and conflict accumulating in successive fusion steps, by assigning belief mass to Θ and ∅ [261].

7.2 Experimental Results

In the practical applications that served as test environments, very competitive results could be achieved. The land cover mapping produced for the 2008 IEEE GRSS data fusion contest was among the winners of the competition [156]. This is all the more remarkable since other winning groups time-consuming produced labelled training sets that were bigger than ours by two orders of magnitude.

The setup for emotion recognition in videos of facial expressions consists of 14 classifiers; the optimal combination of features and face-areas has been confirmed via an exhaustive search. Fuzzy labels were obtained from test persons, and used to train the system [218]. With 83.1%, its classification accuracy is even slightly higher than that of the average human (81.9%), and seems to be on par with
7.2. Experimental Results

Experimental results on the fuzzy-input fuzzy-output ($F^2$) SVMs are very satisfactory. Compared to a set of established classifiers, a One-vs-One SVM with linear kernel yields the highest accuracy; this also holds true once the classifiers are allowed to reject pixels [259]. Concerning the best multi-class decomposition method for SVMs, One-vs-One is to be preferred over One-vs-Rest, because in many applications the accuracy is higher, and the runtime of the algorithm in practice is much lower. Also, the land cover mapping produced by the pairwise architecture consists of much more compact regions [259], a definitive boost for practical utility.

When the SVMs were provided with only portions of the training set, an interesting observation could be made: the optimal value for the central $C$ parameter was always the same, regardless of the varying training set size! That indicates that the SVM parameter can be optimised on a data set of reduced size, drastically lowering the computation time required. Also, in the land cover mapping application, even the number of training samples for the SVM hyperplane generation itself could be reduced by half and still produce the full accuracy.

In the One-vs-One SVM architecture, the input for the coupling procedure to combine the pairwise machines can be obtained in two ways: either the distance of the current sample to the respective separating hyperplane is converted to a binary vote, or transformed using a sigmoid. Both procedures have a comparable accuracy, however the vote-based answers have a greatly reduced variability [259], posing problems for a later classifier fusion.

Applied to satellite images of the Salerno area, Fuzzy SOMs achieve an accuracy that is only beat by a specific SVM. The mapping produced is much smoother [87] than with other somewhat fuzzy SOM approaches proposed recently in the literature.

When comparing the accuracy of the new $F^2$-LVQ and the original LVQ1, as expected the latter wins. But once noise is added to the training labels, the performance of LVQ1 quickly drops, while the proposed variant can maintain high levels of accuracy [262].

The question if it is beneficial to train with fuzzy labels, versus training with hard labels, can not be considered fully answered. An experiment yields an accuracy advantage of 1.6 percent points for the soft-trained SVM; out of 100 cross-validation runs, it only lost in 30. However, the difference is not statistically significant [260]. When focusing more on the fuzzy nature of the labels, using $S_1$ as distance measure, the soft-trained algorithm has a statistically significant

\footnote{This only as far as a comparison is possible, since even with the seemingly standard Cohn-Kanade database, different portions are used across various publications. See the remarks in Chapter 2.1.7.}
higher fuzzy accuracy; this also holds true if only a fraction of the training samples is used.

When adding noise to the fuzzy training labels, it was very striking to see that the RBF network and SVM classifiers tested were able to retain astonishingly high accuracies, even with a pollution level of 70% \[258\]. This becomes understandable when pondering the relationship with hard noise.

In several experiments, samples were allowed to be rejected by the system, depending on the structure of the answers. For this, the maximum rule proved much more efficient than the sum rule. Rejection is very useful, for example, in the land cover mapping application, aiding photointerpreters to identify ground areas with uncertain classification where more training labels have to be supplied. The class-wise rejection curves observed for the SVMs exhibit a smooth behaviour \[259\], their class-dependently varying characteristics stressing the need for advanced fusion functions.

In the realm of bioacoustic time series classification, short experiments based on RBF classifiers within the application of emotion recognition in video sequences of facial expressions demonstrate accuracy increases when frame selection or frame weighting relying on the motion energy is used. During the course of research on orthoptera species identification from songs \[203\], the best results were achieved by using a simple mean fusion within each time series, followed by a decision templates fusion across the various features.

The fusion experiments could demonstrate the superiority (in terms of final accuracy) of multi classifier systems over single classifiers. On the Cohn-Kanade dataset, the single best classifier had an accuracy of 74.0% while the combined final answer of the MCS there achieved 83.1% \[218\]. The respective figures for JACFEE-Morph are 70.8% and 79.1%, for the fruits dataset 83.3% and 95.4% \[261\]. Adding label noise does not change the situation: an answer obtained by fusing several classifiers will still be more accurate \[258\].

As for which classifier fusion scheme to choose, the distinction still hast to be made whether the base classifiers were trained on hard or soft labels. In the hard-trained case, decision templates are the clear winner, closely followed by pseudoinverse fusion. In the soft case, the best and safest way is to use average fusion. The product rule can sometimes achieve a higher accuracy, but in cross-validation experiments it characteristically exhibits seemingly random drops of several percent points, and therefore has to be handled with extreme care. If hard accuracy is not the goal, but rather fuzzy label likeliness, the use of a pseudoinverse mapping can be justified.

A word of caution: all of the experiments have been conducted in settings with a small number of classes (fewer than ten), and with training labels that were not too fuzzy. In scenarios with many classes or very uncertain or mixed labels,
the methods presented here are partially not longer appropriate, and it might be preferable to think of the task not as of a classification- but a regression problem. The most interesting area for future research, as outlined in Chapter 6.3, are classifier fusion functions. Further advances in classifiers that produce uncertain answers will most likely be tied closely to specific applications, especially if Dempster-Shafer output is desired.


[87] Ferdinando Giacco, Christian Thiel, Luca Pugliese, Silvia Scarpetta, and Maria Marinaro. Uncertainty Analysis for the Classification of Multispectral


Bibliography


[272] Thomas Villmann, Udo Seiffert, Frank-Michael Schleif, Cornelia Brüß, Tina Geweniger, and Barbara Hammer. Fuzzy Labeled Self-Organizing Map


Ich möchte mich bei allen bedanken, die zur Entstehung dieser Doktorarbeit beigetragen haben. Insbesondere danke ich

Dr. Friedhelm Schwenker, der meine Begeisterung für maschinelles Lernen weckte und als Betreuer immer für eine Diskussion zur Verfügung stand, für seine umfassende Unterstützung.

Prof. Dr. Günther Palm, der mit seinem anhaltenden Interesse an Unsicherheits-Kalkülen die Richtung der Arbeit bestimmte, und als Gutachter fungiert. Außerdem dafür, dass er mir so viele Konferenzbesuche ermöglichte, durch die ich mit zahlreichen Wissenschaftlern und Themen in Kontakt kam.

Prof. Dr. Heiko Neumann, der aller Doktorandenhäufung zum Trotz als Gutachter zur Verfügung stand.

Prof. Dr. Barbara Hammer, die kurzfristig zusagen konnte, ein drittes, externes Gutachten anzufertigen.

Birgit Lonsinger, der es gelingt, die Komplexität der Verwaltungsmaschine vor den Mitarbeitern zu verbergen.

For their inspiring talks at our institute, many thanks to Vladimir Vapnik ("transductive inference") and Franz Kurfess ("knowledge management").

Ferdinando Giacco from the University of Salerno, many thanks for a fruitful collaboration.

Allen im Institut für Neuroinformatik, für ein entspanntes, persönliches Arbeitsklima.
Dr. Heiner Markert und Dr. Rebecca Markert, für ihre Freundschaft und guten Ratschläge.

Meinen Büro-Kollegen Guillem Pages, Mohamed F. Abdel Hady, Martin Schels, Martin Clauß und Ludwig Lausser, für spannende Gespräche und Hilfe bei alltäglichen technischen Problemen. Guillem besonders für seine Einführung in den Sport TaeKwonDo.

Ulrich Kaufmann für seine Einblicke in die Welt der Fußball-Roboter.

Stefan Scherer für spannende Gespräche über neue Internet-Dienste und Macs.

Der Vision-Gruppe, für entspannte und anregende Mittagspausen und Frau Bühler für guten Kaffee.

Meiner Familie, die mich bestärkt hat, den wissenschaftlichen Weg weiter zu gehen, und Elisabeth, für vergangenes und zukünftiges Glück.
This index lists only the most authoritative mention of each term.

accuracy, 124
activation-evaluation space, 8
AdaBoost, 132
alpha-cut, 51
artificial intelligence (AI), 65
ASTER satellite, 33

basic probability assignment (bpa), 60
Bayesian view, 44
belief, belief function, 60
black-white opponent colour space, 31
boosting/bagging, 132
Bradley-Terry coupling, 97

Canny, 31
certainty, 42
certainty measures, 119
chance discovery, 65
classifier, 72
classifier fusion schemes, 133
associative memory, 135
average, 133
Borda count, 141
brute force, 139
decision templates, 136
Dempster-Shafer, 139
naive Bayes, 137
product, 134
pseudoinverse, 137
voting, 134
coop-training, 131
Cohn-Kanade, 12
conditional independence, 48
conditional probability, 43
confusion matrix, 135
conjunctive combination, 52
context model, 69
cross-validation, 116
default logic, 45
Dempster-Shafer (DS) theory of evidence, 60
discounting, 63
disjunctive combination, 52
diversity, 132
emotion recognition
  from facial expressions, 8
  from speech, 25
entropy, see Shannon entropy
facial action coding system (FACS), 9
feature selection, 73
focal element, 60
frame of discernment, 60
frequentist, 44
fruit object recognition, 30
functional updating, 48
fuzzy, 42
fuzzy control, 55
fuzzy error matrix, 125
fuzzy set, 51
fuzzy-input, 72
fuzzy-input fuzzy-output support vector machine (F²-SVM), 89
fuzzy-output, 72

Gaussian basis function, 75
geographic information system (GIS), 33
Giacco, Ferdinando, 38
Gini function, 120
GMM supervectors, 142
granular computing, 42
grey-level co-occurrence matrix (GLCM), 34

hard, 42
harden, 124

IKONOS 2, 33
imprecision, 42
indiscernability relation, 53

JACFEE-Morph, 12

k-means clustering, 75
k-nearest neighbour algorithm (KNN), 78

Karush-Kuhn-Tucker conditions, 93
Keller fuzzification, 81
kernel function, 91
Kuncheva, Ludmilla, 131

label noise, 112
label normalisation, 72
label, class label, 72
land cover mapping, 32
large margin classifier, 89
learning vector quantisation (LVQ), 79
learning with multiple labels, 72
local sensitivity, 126
log-probabilities, 47

logic, 45
fuzzy logic, 54
modal logic, 45
non-monotonic logic, 45

margin, 90
mixed pixel, 32
mixtures of experts, 147
modulation spectrum, 29
multi layer perceptron (MLP), 100
multi-modal emotion recognition, 10
multiple classifier system (MCS), 131
MYCIN, 45

necessity measure, 50
noise, see label noise

odds formulation, 47
one of x encoding, 134
one-class support vector machine, 108
One-vs-One, 91
One-vs-Rest, 91
OpenCV library, 13
opinions, distribution of, 46
optical flow, 17
orientation histogram, 16
orthogonal sum, 60
orthoptera, 142
over-fitting, 125

penumbra, 41
pignistic transformation, 122
pitch, 26
Platt calibration, 95
possibility measure, 50
possibility theory, 49
primary component analysis (PCA), 16
probability distribution, 44
probability theory, 43
pseudoinverse solution, 75

radial basis function (RBF) network, 74
rank order consistency, 124
rejection, 103
remote sensing, 32
rough set, 53

S1
measure, 80
vector, 124
satellite data, 32
self-organizing map (SOM), 85
semi-supervised learning, 71
sequential forward selection (SFS), 21
set product, 48
Shannon entropy, 119
SMO algorithm, 28
Sobel, 17
soft, 42
Spearman rank correlation, 124
SQL, OLAP, 64
static/trainable fusion, 133
subjective probability theory, 42
support hyperplanes, 90
support vector, 91
support vector machine (SVM), 89
t-norm, 55
t-test, 117
transferable belief model (TBM), 63

Umkehrer framework, 69
uncertainty
formal theories, 41
in classification, 42

vague characteristics, 69
vagueness, 42
validation set, 125

weak classifier, 132