Department of Information and Computer Science

Overfitting in Feature Selection: Pitfalls and Solutions

Juha Reunanen

Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 19/2012

Overfitting in Feature Selection: Pitfalls and Solutions

Juha Reunanen

Doctoral dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the School of Science for public examination and debate in Auditorium T2 at the Aalto University School of Science (Espoo, Finland) on the 30th of March 2012 at 12 noon.

Aalto University
School of Science
Department of Information and Computer Science

Supervisor
Prof Olli Simula

Instructor
Dr Francesco Corona

Preliminary examiners
Dr Pawel Smialowski, Technische Universität München, Germany
Dr Kari Torkkola, Amazon.com, USA

Opponent
Prof Christophe Ambroise, Université d'Évry Val d'Essonne, France

Aalto University publication series
DOCTORAL DISSERTATIONS 19/2012

ISBN 978-952-60-4515-3 (printed)
ISBN 978-952-60-4516-0 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)

Unigrafia Oy
Helsinki 2012

Finland

The dissertation can be read at http://lib.tkk.fi/Diss/

441 697
Printed matter

Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author
Juha Reunanen

Name of the doctoral dissertation
Overfitting in Feature Selection: Pitfalls and Solutions

Publisher School of Science

Unit Department of Information and Computer Science

Series Aalto University publication series DOCTORAL DISSERTATIONS 19/2012

Field of research Computer and Information Science

Manuscript submitted 20 September 2011 **Manuscript revised** 28 November 2011

Date of the defence 30 March 2012 **Language** English

☐ **Monograph** ☒ **Article dissertation (summary + original articles)**

Abstract

When facing a typical pattern recognition task, one usually comes up with a number of so-called features: properties that describe the objects to be recognised. Based on these features, the task of the classifier building algorithm is to find useful rules that are suitable for the recognition of new objects.

Feature selection is a process where one tries to identify the useful features from among a potentially large set of candidates. The task is notoriously hard, and researchers have been tackling it already for decades. Solving the problem properly might today be more important than ever before, because in many applications, dataset sizes seem to grow faster than does the processing power of computers. For example, in the domain of genetic microarray data, there can easily be thousands of features.

Several research groups have published comparisons aiming to identify the feature selection method that is universally the best. Unfortunately, too often the way that such comparisons are done is just plain wrong. Based on the results of such studies, the computationally intensive search algorithms seem to perform much better than the simple approaches. However, it is shown in this thesis that when the comparison is done properly, it very often turns out that the simple and fast algorithms give results that are just as good, if not even better.

In addition, many studies suggest that excluding some of the features is much more useful than it actually is. This observation is relevant in practice, because the selection process typically takes a lot of time and computing resources — therefore, it would be very convenient not to have to carry it out at all. This thesis shows that the benefits obtained may be negligible compared to what has been presented previously, provided that they are measured correctly.

Moreover, the thesis presents a better-performing approach for accuracy estimation in case the amount of data is small. Further, extensions are discussed from feature selection to generic model selection, from the selection of individual features to that of sensors measuring the features, and from classification to regression. Finally, an industrial application is pointed out where the methods discussed prove useful.

Keywords machine learning, feature selection, variable selection, overfitting, search algorithms, comparison of algorithms, model selection, classification, regression

ISBN (printed) 978-952-60-4515-3 **ISBN (pdf)** 978-952-60-4516-0

ISSN-L 1799-4934 **ISSN (printed)** 1799-4934 **ISSN (pdf)** 1799-4942

Location of publisher Espoo **Location of printing** Helsinki **Year** 2012

Pages 136 **The dissertation can be read at** http://lib.tkk.fi/Diss/

A? Aalto-yliopisto

Tiivistelmä

Aalto-yliopisto, PL 11000, 00076 Aalto www.aalto.fi

Tekijä
Juha Reunanen

Väitöskirjan nimi
Ylisovittuminen piirrevalinnassa: sudenkuoppia ja ratkaisuja

Julkaisija Perustieteiden korkeakoulu

Yksikkö Tietojenkäsittelytieteen laitos

Sarja Aalto University publication series DOCTORAL DISSERTATIONS 19/2012

Tutkimusala Informaatiotekniikka

Käsikirjoituksen pvm 20.09.2011 **Korjatun käsikirjoituksen pvm** 28.11.2011

Väitöspäivä 30.03.2012 **Kieli** Englanti

☐ **Monografia** ☒ **Yhdistelmäväitöskirja (yhteenveto-osa + erillisartikkelit)**

Tiivistelmä

Hahmontunnistustehtävää ratkaistaessa määritetään yleensä joukko piirteitä eli ominaisuuksia, jotka kuvaavat tunnistettavia kohteita. Luokittelijan rakentavan algoritmin tehtävä on löytää näihin piirteisiin perustuvat säännöt, jotka soveltuvat uusien kohteiden luokitteluun.

Piirrevalinnassa pyritään löytämään käyttökelpoiset piirteet mahdollisesti suuresta ehdokkaiden joukosta. Tehtävä on vaikea: sitä on yritetty ratkaista jo 1960-luvulta lähtien. Aihe on edelleen ajankohtainen, sillä aineistojen koot kasvavat monissa sovelluksissa jopa nopeammin kuin tietokoneiden laskentakapasiteetti. Esimerkiksi geneettisissä tutkimuksissa käytettävissä olevien piirteiden määrä nousee helposti tuhansiin.

Monissa vertailevissa tutkimuksissa on pyritty löytämään paras piirrevalintamenetelmä, mutta vertailu on usein tehty väärin. Tällöin on saatu tuloksia, joiden mukaan raskaat hakualgoritmit tuottavat selvästi parempia tuloksia kuin mitä yksinkertaisilla ja nopeilla menetelmillä saadaan. Tässä väitöskirjassa osoitetaan, että raskaat ja hitaat algoritmit eivät välttämättä anna lainkaan parempia tuloksia, kunhan vertailu yksinkertaisiin menetelmiin tehdään oikeaoppisesti.

Kirjallisuudessa annetaan myös ymmärtää piirrevalinnan olevan hyödyllisempää kuin mitä se todellisuudessa on. Tämä on käytännön sovellusten kannalta tärkeä havainto, sillä usein valintaprosessi vie paljon aikaa ja laskentaresursseja, joten olisi helpotus, jos sitä ei tarvitsisi lainkaan tehdä. Tässä työssä tuodaan esille, että piirrevalinnalla saavutetut hyödyt voivat olla suorastaan olemattomia kirjallisuudessa aiemmin esitettyyn verrattuna, mikäli mittaaminen suoritetaan oikein.

Väitöskirjassa esitetään myös uusi tarkkuuden estimointiin soveltuva menettely, joka suoriutuu aikaisemmin tunnettuja menetelmiä paremmin silloin, kun dataa on vain vähän. Lisäksi käydään läpi laajennuksia piirrevalinnasta yleisempään mallinvalintaan, yksittäisten piirteiden valinnasta sensorien valintaan, sekä luokittelusta regressioon. Lopuksi käsitellään teollista sovellusta, jonka yhteydessä työssä esitellyt menetelmät osoittautuvat käyttökelpoisiksi.

Avainsanat koneoppiminen, piirrevalinta, muuttujien valinta, ylisovittuminen, hakualgoritmit, algoritmien vertailu, mallin valinta, luokittelu, regressio

ISBN (painettu) 978-952-60-4515-3 **ISBN (pdf)** 978-952-60-4516-0

ISSN-L 1799-4934 **ISSN (painettu)** 1799-4934 **ISSN (pdf)** 1799-4942

Julkaisupaikka Espoo **Painopaikka** Helsinki **Vuosi** 2012

Sivumäärä 136 **Luettavissa verkossa osoitteessa** http://lib.tkk.fi/Diss/

Preface

I wish to sincerely thank Professor Olli Simula for encouraging and supervising me throughout my postgraduate studies. I am also grateful to Dr Francesco Corona for his feedback regarding this dissertation.

Next, I thank Mr Mika Mononen, Professor Marko Vauhkonen, Mr Anssi Lehikoinen and Professor Jari P. Kaipio for coauthoring the last publication of this thesis.

Moreover, I am grateful to the pre-examiners, Dr Pawel Smialowski and Dr Kari Torkkola, for their valuable comments. Further, I would like to thank Professor Christophe Ambroise for kindly agreeing to take the role of the opponent.

I am also thankful to my parents for their support regarding my studies. Lastly, I thank Pirjo for her love and, well, everything.

Juha Reunanen
Helsinki, 31 January 2011

Contents

1. About the thesis

This chapter provides some meta information about this thesis. The thesis consists of five publications that are listed in Section 1.3, plus this compendium, whose structure is described in Section 1.5.

1.1 Background

The research reported in this thesis started with a Master's thesis project funded by ABB Industry Oy (today part of ABB Oy) in 2000–2001. The topic was feature selection for a specific industrial pattern recognition problem, namely the image-based classification of metal strip surface defects. During that work, results were obtained suggesting that the state-of-the-art literature on feature selection may contain some rather serious mistakes. However, as that was not the topic of the Master's thesis, the generality of the observation remained unknown.

After finishing the Master's thesis, I was employed by ABB to work on matters mostly unrelated to feature selection. However, having been misled by the literature for months, I wanted to know whether the same could happen to someone else with some completely different data. From a theoretical point of view, the possibility for that seemed obvious, but it was not evident whether it would happen in practice. Hence, an attempt to repeat the results of one particular study was made, and soon it was obvious that some of the results reported in it were just plain invalid. Therefore, a decision was made to put in some more time to proceed with a more thorough research, and to report the results. One of the outcomes of this decision was my Licentiate's thesis in 2004.

The said Licentiate's thesis focused on pitfalls and misunderstandings in the literature of the time. It basically comprised the results reported in

Publications[1] I and II of the present thesis — essentially, problems. It did not make a lot of effort to mitigate the issues caused by those problems. Therefore, the obvious next step was to work on methodology that would provide reasonable recommendations as to what a practitioner can do, instead of just pointing out that what was rather commonly done did not really make a whole lot of sense. To that end, the method presented in Publication III saw light. Finally, in Publications IV and V, extensions were provided to describe how to work with the methods when faced with research problems that were not too obviously catered for by the previous work.

1.2 Scope

First and foremost, the thesis addresses automatic and algorithmic *feature selection* (also known as variable selection), the art of being able to point out the important features from among a typically large set of candidates. The problems and benefits related to such selection have lately drawn attention from a myriad of researchers; the research topic, although investigated by scientists already for decades, seems to be more important today than ever before. Examples of relevant areas with recent real-world applications include document categorisation, prosthesis control, cardiac arrhythmia classification, fMRI analysis, gene selection from microarray data, real-time polymer identification, and credit card fraud detection (Rodriguez-Lujan et al., 2010).

The increased interest in feature selection is probably caused by more and more data being available all the time. While computers are also getting faster, the combinatorial nature of many machine learning algorithms implies that if the amount of data grew linearly, the computational power would have to increase *exponentially* in order to let us continue using the old algorithms. Although exponential growth in microprocessor power has been observed and even thought of as a "law" (Moore, 1965; Tuomi, 2002), the amount of data also grows superlinearly in many domains, such as the World Wide Web (Huberman and Adamic, 1999).

In addition to speed issues, machine learning algorithms may also have theoretical problems when given too many possibilities but not enough grounds for making valid decisions. For example in the analysis of genetic

[1]The publications will be enumerated in Section 1.3.

microarray data, the number of candidate features may be overwhelmingly large compared to the number of training samples available (Xing et al., 2001; Guyon et al., 2002; Raudys, 2006).

Unfortunately, the research reported in this thesis shows that some advances in feature selection that used to be widely accepted were probably not as useful as had been claimed. Moreover, it is shown that feature selection with such advanced methods might in general not have been as beneficial as was commonly believed at the time when Publications I and II were written. The key to arriving at these conclusions is in measuring the advantages rigorously — not only seemingly correctly. The conclusions definitely have a negative impact on the state of feature selection, and they point out clearly that more research is needed before these kinds of techniques can be used as universal black-box systems.

The results also prove that the phenomena observed first in the context of a particular industrial problem are in no way limited to that specific domain. Instead, similar behaviour is witnessed in fields varying from the identification of unsolicited e-mail to medical diagnoses, and from artificially generated problems to very specific and even esoteric ones. Moreover, there seems to be no reason to believe that similar results would not be obtained also for many other domains where feature selection has found widespread use. However, in Publication V, a counter-example is also discussed.

In addition to the different domains, the thesis also shows that the results are not incidental for a certain type of classifier architecture. On the contrary, similar behaviour can be observed in the contexts of three different pattern recognition paradigms: nearest neighbours, decision trees, and neural networks with the multilayer perceptron architecture (Section 2.2).

Finally, even though the thesis mainly deals with feature selection, the more generic model selection problem is also addressed. And even though the emphasis clearly is on classification, a particular regression problem is also looked at briefly.

1.3 List of publications

This thesis is based on the research reported in the following five publications:

Publication I — Juha Reunanen (2003). Overfitting in Making Comparisons Between Variable Selection Methods. *Journal of Machine Learning Research*, volume 3, pp. 1371–1382.

This paper highlights the problems that emerge when feature selection search algorithms of different complexity and execution time are compared using the cross-validation scores obtained during the search. Because of overfitting, the more exhaustive search algorithm seems to perform much better. However, this often turns out not to be the case when the subsets selected by the algorithms are compared using independent test data.

Publication II — Juha Reunanen (2004). A Pitfall in Determining the Optimal Feature Subset Size. In A. Fred (Ed.), *Proceedings of the Fourth International Workshop on Pattern Recognition in Information Systems* (PRIS 2004), Porto, Portugal, April 13–14, pp. 176–185.

This paper shows that when only cross-validation scores found during the search are examined, the exclusion of some of the features often seems to be highly beneficial. Again, proper tests with fresh data can result in opposite conclusions: excluding the features may actually turn out to *decrease* the classification performance.

Publication III — Juha Reunanen (2006). Less Biased Measurement of Feature Selection Benefits. In C. Saunders, M. Grobelnik, S. Gunn, J. Shawe-Taylor (Eds.), *Subspace, Latent Structure and Feature Selection: Statistical and Optimization Perspectives Workshop* (SLSFS 2005), Revised Selected Papers (LNCS 3940), pp. 198–208.

This paper suggests two algorithms to remedy the problem introduced in Publication II, namely the difficulty of measuring the accuracy improvement obtained by performing feature selection.

Publication IV — Juha Reunanen (2007). Model Selection and Assessment Using Cross-indexing. In J. Si, R. Sun, D. Brown, I. King, N. Kasabov (Eds.), *Proceedings of the Twentieth International Joint Conference on Neural Networks* (IJCNN 2007), Orlando, Florida, USA, August 12–17, pp. 2581–2585.

This paper describes how to use the algorithms presented in Publication III for the selection and evaluation of model hyperparameters. In addition, a generalisation is outlined that includes the previous two algorithms as special cases.

Publication V — Juha Reunanen, Mika Mononen, Marko Vauhkonen, Anssi Lehikoinen, and Jari P. Kaipio (2011). Machine Learning Approach for Locating Phase Interfaces Using Conductivity Probes. *Inverse Problems in Science & Engineering*, volume 19, issue 6, pp. 879–902.

This is an application paper dealing with a certain regression problem that provides an extension to sensor selection. The results effectively compose a counter-example to how a reader in hurry might have interpreted the results of Publication I.

While not containing revolutionarily new theoretical ideas, Publications I and II address common faults made by researchers introducing new feature selection methods or those assessing the performance of previous ones. Because even distinguished pattern recognition researchers had clearly committed these mistakes, it seems that the findings were non-trivial with respect to the state of the art of feature selection at that time.

Next, Publication III suggests a novel method, which is shown to perform better than the obvious alternative. Then, in Publication IV, this method is extended to a more generic model selection setting, where its efficacy is demonstrated in an open competition. Finally, in Publication V, some extensions are introduced — together with results that rule out a possible misunderstanding of the implications presented in Publication I.

In this compendium, the contents of the publications are predominantly referred to as presented in Table 1.1.

Table 1.1. The sections of this compendium essentially addressing the contents of the different publications.

Publication	Section(s)
I	5.1
II	5.2
III	6.1
IV	1.4, 6.1 and 6.2
V	5.3, 6.3 and 6.4

1.3.1 Contribution of the author

The author of the thesis at hand is the sole author of Publications I–IV. As far as Publication V is concerned, the said author is responsible for applying machine learning methodology to the problem described in the

publication, which basically rules out (1) the description of the problem itself, and (2) the solution to the so-called forward problem. In other words: the said author is the principal responsible for Sections 3, 4 and 5.2–5.5, whereas Sections 1, 5.1 and 6 were written jointly with the other authors, and Section 2 was written entirely by the other authors.

1.4 Impact

Publications I and II pointed out specific pitfalls that researchers keep falling into. A more recent example of this is the study related to music classification by Fiebrink et al. (2005), where feature selection indeed appears highly beneficial. However, the authors later realised having fallen into the very pitfalls addressed in this thesis. Consequently, in a later study (Fiebrink and Fujinaga, 2006), they cited Publications I and II and carried out new tests using sound methodology. The results of these new tests — together with the eventual conclusions — strongly support the claims made in Publications I and II.

In spite of Publications I and II, together with some independent work pointing out largely the same thing (e.g., Ambroise and McLachlan, 2002), it unfortunately appears that many researchers still fall into these same pitfalls (see, e.g., Smialowski et al., 2010).

The improvement suggested in Publication III was successfully applied in three open competitions. The results were reported in Publication IV, and they can be summarised as follows:

- In the *WCCI 2006 Performance Prediction Challenge*, the error estimate obtained was the second best out of the 28 valid entries competing for the final ranking (Guyon et al., 2006b).
- In the *NIPS 2006 Model Selection Game*, the approach was used to win the game (Guyon et al., 2007) — but there were only three participants.
- In the *IJCNN 2007 Agnostic Learning vs. Prior Knowledge Challenge*, the method ranked third out of thirteen (3/13) on the agnostic learning track, and second out of seven (2/7) on the prior knowledge track (Guyon et al., 2008).[2]

[2]These are notable achievements in the sense that unlike the winners and almost all other participants, the entries were structurally eligible to participate

1.5 Structure of the compendium

First, Chapter 2 of this compendium (introductory part) describes supervised machine learning in general, and in particular a couple of techniques that are used in this thesis.

Next, Chapter 3 covers the feature selection problem, and the basic building blocks required for solving an instance of it.

What follows is Chapter 4 discussing *overfitting*: a highly important phenomenon that is critical for a notably varied class of problems, including feature selection.

Then, Chapter 5 describes the first results of the thesis: two pitfalls that an ignorant practitioner might not be able to avoid when doing feature selection.

After that, Chapter 6 goes through the suggested remedies to the problems introduced in Chapter 5, and also discusses solutions to some related issues.

Finally, Chapter 7 concludes the thesis by repeating the main results.

in the NIPS 2006 Model Selection Game, too — in other words, they were automatically picked from among a model family defined by the organisers, whereas the winning entries used their own top-performing models.

2. Introduction to supervised learning

Machine learning is an approach that can be used for saving labour in many tedious tasks, or to help in some difficult ones. Machine learning is usually categorised into supervised and unsupervised learning. This thesis deals solely with supervised learning, where the user tells the machine that it should learn a mapping from x to y — in other words, it should learn how y depends on x.[1]

Supervised learning is typically employed in areas such as *classification*, where the machine learns to discriminate certain kinds of objects from the rest, or *regression*, where the objective is to learn how a variable or several variables of interest depend on some other variables. The emphasis of this thesis is on classification; discussion related to regression is deferred until Section 6.4.

In automatic classification, the objective is to build a classifier: an entity that is able to automatically sort any objects with respect to the properties of the objects. Automatic classification can be roughly divided into two approaches: expert rules and trainable classifiers. Expert rules do not represent supervised learning, but they are nevertheless discussed here to serve as an introduction to the challenges that drive practitioners to rely on supervised learning methods.

2.1 Expert rules

Expert rules are condensed information: often rather straightforward ways of determining the class of the object. For instance, someone might

[1] On the contrary, in unsupervised learning there is no y: for instance, the goal may be to find clusters in — or alternative representations of — the data at hand, x (see, e.g., Chapter 14 of Hastie et al., 2008).

suggest that bananas are yellow and apples are red. Given a way to measure the colour of an object, an automatic classifier embodying such a rule is trivial to come up with for anyone who knows programming at even the most basic level.

However, the trouble with expert rules is that while it can be easy to devise some, it is often very hard to come up with correct ones for the classification. For instance, bananas and apples can both be green; which class label should we then assign to a green fruit? In this case, the colour is not enough: instead, information for example about the shape of the object is needed. For instance, in contrast to apples, bananas typically are elongated and have a degree of curvature. Except, perhaps, if they are viewed from certain angles. Or, maybe, when peeled and partially eaten.

While it ought to be possible to come up with reasonable rules based on colour and shape for discriminating between bananas and apples, the task gets increasingly difficult when the number of classes is increased. What about lemon, grapefruit, plum, peach, kiwifruit or papaya? Cherries, strawberries, pears? Is it an orange, a mandarin, or perhaps a kumquat? One is likely to need more and more properties such as size, or some more special ones, like the smoothness of the surface of the fruit. However, when you add more properties, it soon gets quite hard to verify that the rules you specify are meaningful and correct.

2.2 Trainable classifiers

This is where supervised machine learning comes in. In a typical setting, instead of specifying the rules, the expert manually labels a number of examples. He tries to find different kinds of bananas and apples, and assigns each of them the correct classification result. Then, he initiates a supervised training process, which is usually mere computation done by a computer program. In this process, the program tries to find suitable rules that are capable of repeating the results of the expert. The idea is that such rules which can correctly classify the manually labelled instances are likely to be useful for the classification of completely new samples as well.

Methods for trainable automatic classification are typically based on the concept of a *feature*. A feature represents a property or an attribute of the object being classified, often in numeric form. For instance, in the

fruit problem useful features could include the volume of the object, the prominent wavelength of the light reflected by it, or its weight. The task of the algorithm that is supposed to build the automatic classifier is to find useful rules with respect to those features that happen to be available.

In order to define the goal of the training process, one may specify a *loss function* to be minimised. In the domain of classification, the loss typically is the fraction of misclassified instances. If we knew how probable each class is given the feature values, we could derive a so-called *Bayes rule* that minimises the loss (see, e.g., Schalkoff, 1992; Hastie et al., 2008). However, in practice it is an extremely rare occasion that the true probability density functions of the different classes are available; normally, we only have individual points (in the feature space) for which we know the correct class label, and we need to resort to more or less heuristic methods in order to come up with useful rules for the classification of new samples.

The different methods developed by the research community number so many that only some of those used in the publications constituting this thesis are reviewed here shortly. In order to have as much generality in the results as possible, approaches have been picked with somewhat varied backgrounds: statistics, computer science and psychology.

2.2.1 Nearest neighbours

Rooted in statistics, the k nearest neighbours (kNN) rule (Cover and Hart, 1967; for textbook accounts, see, e.g., Fukunaga, 1972; Duda and Hart, 1973; Devijver and Kittler, 1982; Schalkoff, 1992) for the classification of new items is one of the simplest. The algorithm thinks of each training instance as a point in a Euclidean space[2]. This space has as many dimensions as there are features available representing the objects to be classified: each axis corresponds to exactly one feature. On the other hand, each manually labelled instance is represented by exactly one point in this space, such that the position of the point with respect to each axis is determined by the corresponding feature values representing the instance.

The rule for classifying new samples is then easy: Each instance is positioned in the same space using the feature values that should be available, and the k points with a known class that are nearest to the new instance

[2]Strictly speaking, the space does not have to be Euclidean: it is enough that a real-valued distance between any two points can be determined.

are determined. The new instance is then assigned the class label suggested by the majority of these k instances.

An example of a feature space with two dimensions (i.e., features) is given in Figure 2.1. There are three classes, which are illustrated by circles, squares and triangles. Based on the figure, it is possible to devise useful rules for the classification of new samples, but it is impossible to come up with rules that are always correct: these two features do not carry enough information to facilitate that. A new instance — identified by the question mark in the lower right hand corner of the figure — would be classified as a circle using the 1NN rule, but as a triangle with any $k \geq 3$. (Yes, there are more triangles than circles or squares in the figure.)

2.2.2 Decision trees

A different kind of approach is that of decision trees, traditionally studied in computer science. There, the rules are hierarchical, and normally based on just one feature at a time: for instance, a rule might state that if an object is curved, then it is a banana. Otherwise, it is an apple, provided that it is red. If not, then if it is blue and very small, it is a blueberry. Otherwise, we cannot be sure. The form of the rules of a decision tree closely resembles that of the rules that human experts typically come up

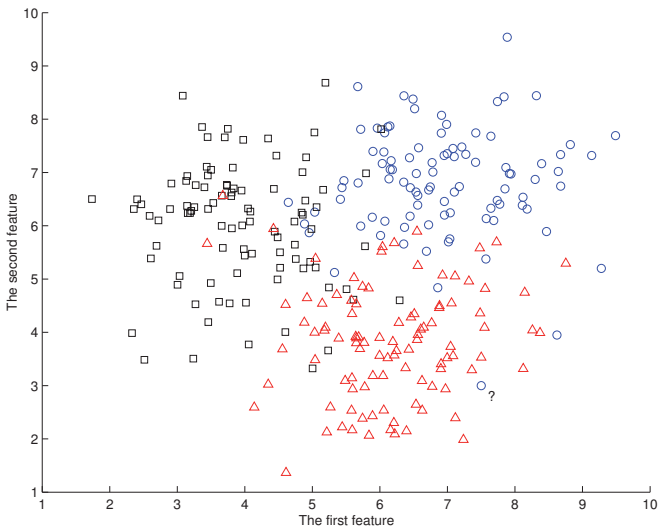

Figure 2.1. An example of a feature space with two features and three classes. The samples labelled by the expert as belonging to one of the three classes are illustrated by circles, squares and triangles. A new sample to be classified is denoted by a question mark (see text).

with.

There are several algorithms for determining the hierarchy and the limits for the individual rule steps. In this thesis, the C4.5 algorithm (Quinlan, 1993) for building the rules will be used as an instance of the decision tree approach.

2.2.3 Multilayer perceptrons

Yet another branch of science contributing to the field of machine learning is psychology: the journal Psychological Review published the first results (Rosenblatt, 1958) that eventually led to the boom of artificial neural networks (ANNs). Neural networks are mathematical structures that have (more or less) been designed based on what we know about how the human brain works. The prominent ANN architecture used in this thesis is the multilayer perceptron (MLP) trained using error backpropagation[3] (Rumelhart et al., 1986; for textbook accounts, see, e.g., Bishop, 1995; Haykin, 1999). In contrast to those of decision trees, the rules found by a backpropagation algorithm can be quite difficult to interpret. However, a couple of such rules in an easy case of two features will be illustrated in Section 4.1.

[3] In this thesis, algorithms such as resilient backpropagation (RPROP) (Riedmiller and Braun, 1993) and Levenberg–Marquardt (Hagan and Menhaj, 1994) are used for the training of the MLPs.

3. Feature selection

In Chapter 2, we discussed a couple of features for the fruit classification problem: colour, volume, curvature, weight. Other useful features could include smell, homogeneity of the colour, glossiness, or other properties describing the shape of the object.

Ever since the 1960s, researchers have been trying to find ways of pointing out those features that are actually important in making the correct decision (Lewis, 1962; Marill and Green, 1963; Elashoff et al., 1967; Swonger, 1968). Obviously, if we do not have to automatically measure the smell of the fruit, our system will be much easier and cheaper to build. On the other hand, discovering that weight alone is actually sufficient to discriminate between different types of fruit would be very surprising and might give some new ideas for the research of the fruit. These reasons — reduced data acquisition costs and an insight into the importance of the features — for selecting only some of the features could be categorised as *external* with respect to the classification algorithm.

In contrast to the external ones, *internal* reasons are such that benefit the classification system itself. One such reason is that it is easier and faster to contemplate all the possible rules for the classification if the number of features is small. On the other hand, if irrelevant features are removed, then the data presented to the algorithm that is looking for the rules will be cleaner. It can be assumed that by cleaning the noise and redundancy in the data, we can obtain a more accurate classifier. An example of a useless feature is given in Figure 3.1.

Approaches for feature selection can be divided into filter methods, wrapper methods, and embedded methods (John et al., 1994; Guyon and Elisseeff, 2003; Guyon et al., 2006a). Nowadays, the computationally less intensive filter techniques are typically used mostly for preprocessing; at

this point, we are not yet concerned with the eventual classification accuracy. Wrappers, on the other hand, plug in to the classifier building algorithm, and try to identify the useful features by training and evaluating actual classifiers that use different feature subsets. Empirical comparisons between wrappers and filters have been provided for example by Aha and Bankert (1996) and Kohavi and John (1997). Finally, embedded methods are classifier training algorithms that contain feature selection as an integral part of the learning process.

This thesis focuses mainly on the wrapper approach. Filters are not studied at all, but the embedded approach is present due to the choice of the classification architectures. The C4.5 decision tree generation algorithm clearly includes feature selection: the algorithm can generate such decision trees that do not use all the features. On the other hand, the backpropagation training of MLPs gives more emphasis on those features that are observed to be useful[1]. However, the kNN classification rule does not have any kind of embedded selection of features.

[1]More explicit embedded feature selection for MLPs is also possible (see, e.g., Cibas et al., 1994, 1996; De et al., 1997; Setiono and Liu, 1997)

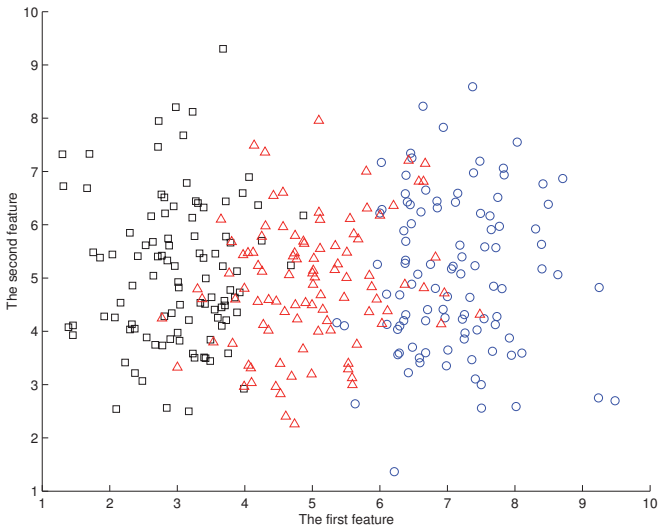

Figure 3.1. An example of a feature space with two features and three classes, where the second feature is largely useless for coming up with any rules for the classification of new samples.

3.1 Subset evaluation

In the wrapper approach, a practitioner needs to make two basic decisions: how are the feature subsets evaluated, and how are the subsets selected for evaluation?

The evaluation of the subsets consists of splitting the dataset somehow into a training set which is used to build the classifier, and a test set used to evaluate its performance. In different evaluation schemes, the exact procedure of this division varies somewhat.

Perhaps the most popular way to do the division is *cross-validation* (CV), where the dataset is divided into a number of folds (e.g., Kohavi, 1995). Then, one fold at a time is chosen as the test set, while the others are merged to constitute the training set. Therefore, each sample appears in exactly one test set. When the correct classifications in the different folds are counted up and divided by the total number of samples, the cross-validation accuracy estimate for the dataset and the classifier is obtained. The special case where the number of folds is equal to the number of samples in the original set is called *leave-one-out cross-validation* (LOOCV).

3.2 Subset selection

Once a scheme for evaluating the different subsets has been decided, one can start the search for the best subset. However, this may not be very fruitful without a reasonable *search strategy*, which determines the order in which the different subsets are evaluated.

During the years, a large number of different search strategies, or selection algorithms, have been proposed. In a way, this thesis questions that development. In Publication I, one of the simplest possible strategies is compared to a more recent, more complex, and seemingly better algorithm. The result is that when the comparison is made properly, the real difference (in the eventual classification performance) is often negligible. On the other hand, Publication II shows that the selection algorithms might in general not be as useful as one might suppose at the first glance.

Because of the nature of this thesis, no complete list is included of all the approaches that have been proposed for doing the search. Instead, only

```
function SFS(n, J)                    Returns a set of feature subsets of different sizes
begin
    S := (0, ..., 0)  [n];            Start with an empty set
    k := 0;
    B := ∅;                           Initialise the set of best feature sets found
    while k < n − 1                   Repeat for as long as there are branches to compare
        R := ∅;                       Initialise the set of evaluations of different branches
        for each {j | S_j = 0}        Repeat for each possible branch
            T := S;                   Copy the feature subset
            T_j := 1;                 Add the jth feature
            R(j) := J(T);             Evaluate the branch
        end;
        k := k + 1;
        j := arg min R(·);            Find the best branch
        S_j := 1;                     Take the best branch
        B(k) := S;                    Store the newly found subset
    end;
    return B;
end;
```

Algorithm 1. Sequential Forward Selection (SFS).

four similar, yet in behaviour drastically different algorithms are presented. More of them have been proposed for example by Chang (1973), Stearns (1976), Narendra and Fukunaga (1977), Kittler (1978), Siedlecki and Sklansky (1988), Somol et al. (1999) and Somol and Pudil (2000). Many of these algorithms were discussed in a survey written by the author of the present thesis (Reunanen, 2006).

The first algorithm discussed here is often referred to as Sequential Forward Selection (SFS), although other names have also been used. The idea, introduced first by Whitney (1971), is that the search starts with an empty set, and at each step, all the features that are not yet part of the set are evaluated. The one that attains the best score is then added to the set, and a new step is started. The approach is summarised in Algorithm 1. In the notation, S and T are arrays of Boolean values representing feature subsets, and S_j is a binary value indicating whether the jth feature has been selected in S. The function $J(\cdot)$ is used as the evaluation measure for the subsets; a larger value means a worse subset.

The second algorithm, Sequential Floating Forward Selection (SFFS) devised by Pudil et al. (1994a,b) is similar, but after a forward step, a backtracking phase takes place. Here, each feature currently included in the set is considered to be excluded. The backtracking goes on for as long as better subsets of the corresponding sizes are found than previously. When the scores do not increase through this pruning, then a new forward step

```
function SFFS(n, J)                    Returns a set of feature subsets of different sizes
begin
   S := (0, ..., 0)  [n];              Start with an empty set
   k := 0;
   B := ∅;                             Initialise the set of best feature sets found
   while k < n                         Repeat until the set of all features is reached
      R := ∅;                          Initialise the set of evaluations of different branches
      for each {j | S_j = 0}           Repeat for each possible branch
         T := S;
         T_j := 1;                     Add the jth feature
         R(j) := J(T);                 Evaluate the branch
      end;
      k := k + 1;
      j := arg min R(·);               Find the best branch
      S_j := 1;                        Take the best branch
      B(k) := S;                       Store the newly found subset
      t := 1;                          This is reset when backtracking is to be stopped
      while k > 2 ∧ t = 1              Backtrack until no better subsets are found
         R := ∅;                       Initialise the set of evaluations of different branches
         for each {j | S_j = 1}        Repeat for each possible branch
            T := S;
            T_j := 0;                  Prune the jth feature
            R(j) := J(T);              Evaluate this branch
         end;
         j := arg min R(·);            Find the best branch
         if R(j) < J(B(k − 1))         Was a better subset of size k − 1 found?
            k := k − 1;                If yes, backtrack
            S_j := 0;
            B(k) := S;                 Store the newly found subset
         else
            t := 0;                    If no, stop backtracking
         end;
      end;
   end;
   return B;
end;
```

Algorithm 2. Sequential Forward Floating Selection (original version).

is taken, and again potentially followed by backtracking. This version of the SFFS algorithm is illustrated as Algorithm 2.

However, in this original version of the algorithm there is a small bug pointed out by Somol et al. (1999): after first doing some backtracking and then one or more forward steps, the current subset can be worse than the best one of that size that was found before backtracking started. This is easy to fix with simple bookkeeping; the corrected version is shown in Algorithm 3.

In practice, SFFS differs from SFS in two respects: First, with typical datasets, it takes a lot more time to run SFFS. Second, with SFFS one tends to get cross-validation estimates that promise a much higher accuracy

```
function SFFS(n, J)                          Returns a set of feature subsets of different sizes
begin
    S := (0, ..., 0)  [n]                    Start with an empty set
    k := 0;
    B := ∅;                                  Initialise the set of best feature sets found
    while k < n                              Repeat until the set of all features is reached
        R := ∅;                              Initialise the set of evaluations of different branches
        for each {j | S_j = 0}               Repeat for each possible branch
            T := S;
            T_j := 1;                        Add the jth feature
            R(j) := J(T);                    Evaluate the branch
        end;
        k := k + 1;
        j := arg min R(·);                   Find the best branch
        if R(j) ≥ J(B(k))                    Was this branch the best of its size found so far?
            S := B(k);                       If no, abruptly switch to the best one
        else
            S_j := 1;                        If yes, take the branch
            B(k) := S;                       Store the newly found subset
            t := 1;                          This is reset when backtracking is to be stopped
            while k > 2 ∧ t = 1              Backtrack until no better subsets are found
                R := ∅;                      Initialise the set of evaluations of different branches
                for each {j | S_j = 1}       Repeat for each possible branch
                    T := S;
                    T_j := 0;                Prune the jth feature
                    R(j) := J(T);            Evaluate the branch
                end;
                j := arg min R(·);           Find the best branch
                if R(j) < J(B(k − 1))        Was a better subset of size k − 1 found?
                    k := k − 1;              If yes, backtrack
                    S_j := 0;
                    B(k) := S;               Store the newly found subset
                else
                    t := 0;                  If no, stop backtracking
                end;
            end;
        end;
    end;
    return B;
end;
```

Algorithm 3. Sequential Forward Floating Selection (corrected version).

```
function SBS(n, J)                  Returns a set of feature subsets of different sizes
begin
  S := (1, ..., 1)  [n];            Start with the full set that includes all the features
  k := n;
  B := ∅;                           Initialise the set of best feature sets found
  while k > 1                       Repeat for as long as there are branches to compare
    R := ∅;                         Initialise the set of evaluations of different branches
    for each {j | S_j = 1}          Repeat for each possible branch
      T := S;                       Copy the feature set
      T_j := 0;                     Prune the jth feature
      R(j) := J(T);                 Evaluate the branch
    end;
    k := k − 1;
    j := arg min R(·);              Find the best branch
    S_j := 0;                       Take the best branch
    B(k) := S;                      Store the newly found subset
  end;
  return B;
end;
```

Algorithm 4. Sequential Backward Selection (SBS).

than do those obtained with SFS.

There are also *backward* versions of both algorithms. In Sequential Backward Selection (SBS), the idea of which was first introduced by Marill and Green (1963), the search starts with a full set comprising all the candidate features. Then, at each step, the feature whose exclusion yields the best results is pruned. The SBS algorithm is shown as Algorithm 4. Likewise, a floating version of backward selection was introduced by Pudil et al. (1994a,b) — the resulting SBFS algorithm is detailed in Algorithm 5.

```
function SBFS(n, J)                    Returns a set of feature subsets of different sizes
begin
  S := (1, ..., 1) [n];                Start with the full set that includes all the features
  k := n;
  B := ∅;                              Initialise the set of best feature sets found
  while k > 1                          Repeat for as long as there are branches to compare
    R := ∅;                            Initialise the set of evaluations of different branches
    for each {j | S_j = 1}             Repeat for each possible branch
      T := S;
      T_j := 0;                        Prune the jth feature
      R(j) := J(T);                    Evaluate the branch
    end;
    k := k − 1;
    j := arg min R(·);                 Find the best branch
    if R(j) ≥ J(B(k))                  Was this branch the best of its size found so far?
      S := B(k);                       If no, abruptly switch to the best one
    else
      S_j := 0;                        If yes, take the branch
      B(k) := S;                       Store the newly found subset
      t := 1;                          This is reset when backtracking is to be stopped
      while k < n − 1 ∧ t = 1          Backtrack until no better subsets are found
        R := ∅;                        Initialise the set of evaluations of different branches
        for each {j | S_j = 0}         Repeat for each possible branch
          T := S;
          T_j := 1;                    Add the jth feature
          R(j) := J(T);                Evaluate the branch
        end;
        j := arg min R(·);             Find the best branch
        if R(j) < J(B(k + 1))          Was a better subset of size k + 1 found?
          k := k + 1;                  If yes, backtrack
          S_j := 1;
          B(k) := S;                   Store the newly found subset
        else
          t := 0;                      If no, stop backtracking
        end;
      end;
    end;
  end;
  return B;
end;
```

Algorithm 5. Sequential Backward Floating Selection — this time, the fix by Somol et al. (1999) is pointed out by underlining the lines to be added.

4. Overfitting

In the training of an automatic classifier, learning the data too well is called *overfitting* or *overlearning*. The phenomenon normally takes place when models with too many parameters are fitted to the data too carefully.

4.1 Overfitting of the first kind

In the most common type of overfitting, a single model is allowed to be too complex. As a consequence, the model ends up taking into account useless aspects of the training data (for example, noise). This can happen when the model has relatively many free parameters with respect to the size and the nature of the dataset, and the parameters are adjusted for as long as a decrease in the cost function of the model is observed.

In Figure 4.1, a single-layer perceptron has been trained based on the data presented already in Figure 2.1. The figure shows the predictions made by the perceptron for new data: the brighter an area, the more certain the classifier is of a new sample that appears in the corresponding position of the coordinate system. A single-layer perceptron is a special case of the multilayer perceptron architecture (see Section 2.2.3) that has no hidden layers at all: therefore, the rules it is able to learn are limited to be very simple.

On the other hand, Figure 4.2 shows the predictions of a genuine multilayer perceptron. The network has a hidden layer of ten neurons, which means that it has many more free parameters than has the single-layer perceptron. Therefore, it is able to adjust so that a few more samples in the training set get correctly classified. As it is able to repeat the results of the expert with more accuracy, it seems to be a better classifier.

Finally, the network whose predictions are illustrated in Figure 4.3 has

two hidden layers, both with ten neurons. The class labels suggested by the expert are taken into account even more rigorously than in Figure 4.2, but the rules start to get rather questionable. One could suspect that

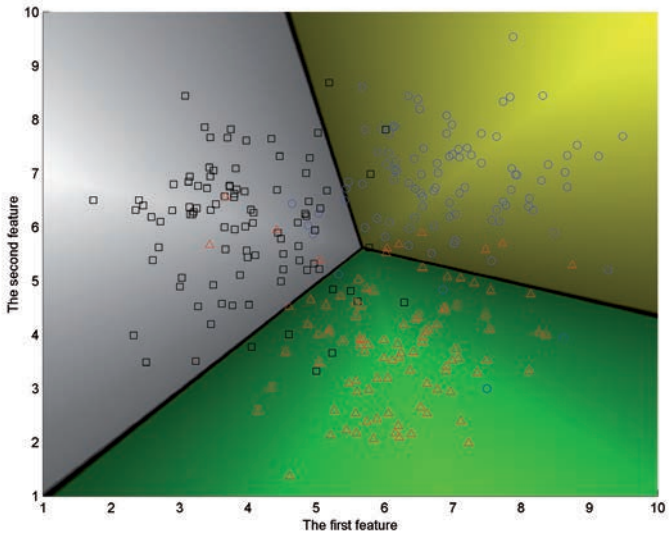

Figure 4.1. The rules for the classification of new samples for a single-layer perceptron. The brighter the color, the more certain the classifier is of a new sample appearing in the corresponding area.

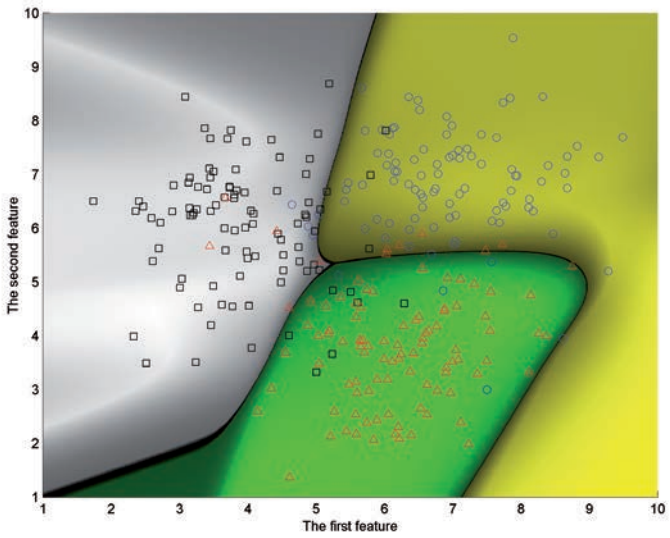

Figure 4.2. The rules for the classification of new samples for a multi-layer perceptron with a single hidden layer.

these rules might not be more useful in the classification of new samples than the rules depicted in, for example, Figure 4.2.

Actually, by looking at Figures 4.1–4.3 only, we cannot tell which classifier is the best. Instead, we should have a fresh set of previously unseen samples, using which we are supposed to *test* each classifier. If each new sample falls very close to some training sample and belongs to the same class, then the classifier whose rules are illustrated in Figure 4.3 is definitely the best one.

However, the samples in the example were generated using three Gaussian distributions, one for each class. In Figure 4.4, a new set of samples generated using the same distributions is shown, plotted on top of the rules of the second classifier. If we applied the rules of all the three classifiers to this so-called *test set*, we would observe that the first network *generalises* better than the others. This means that the rules embodied by it correspond to the underlying distributions more closely than do those of the other networks. In other words, the network in Figure 4.2 is overfitted: it is able to classify the training data accurately, but this capability does not generalise so well to previously unseen data. This applies to the classifier of Figure 4.3 as well, but to a far greater extent. However, this does not mean that the multilayer perceptron architecture is particularly

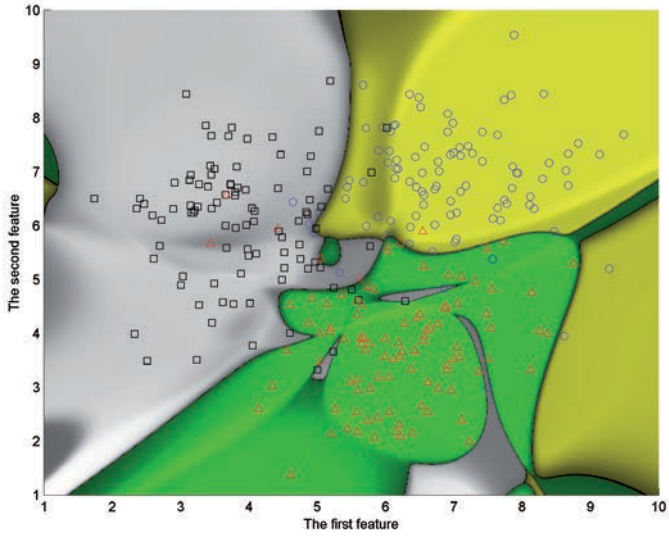

Figure 4.3. The rules for the classification of new samples for a multi-layer perceptron with two hidden layers.

vulnerable to overfitting in general (see, e.g., Lawrence et al., 1997).

The issue described in this section, the overfitting in training a single classifier, is called *overfitting of the first kind* in this thesis.

4.2 Avoiding overfitting

The basic approach for avoiding overfitting of the first kind is to constrain the model: one should not have too many free parameters. On the other hand, a large set of parameters may not be harmful, if their values are somehow controlled. For example, one might want to penalise the model for large parameter (absolute) values — then, the model will rather have some of the difficult training samples misclassified than bend to costly positions in order to take them into account. This sort of an approach is called *regularisation*.

How does one know which number of parameters is the correct one, or how much to penalise for a more complex model? With two features, it is possible to draw something like Figures 4.1–4.3, and to try to figure out visually whether or not there is overfitting. However, with more than two features this approach gets difficult.

What one can still do is to have a test set, like the one shown in Figure 4.4.

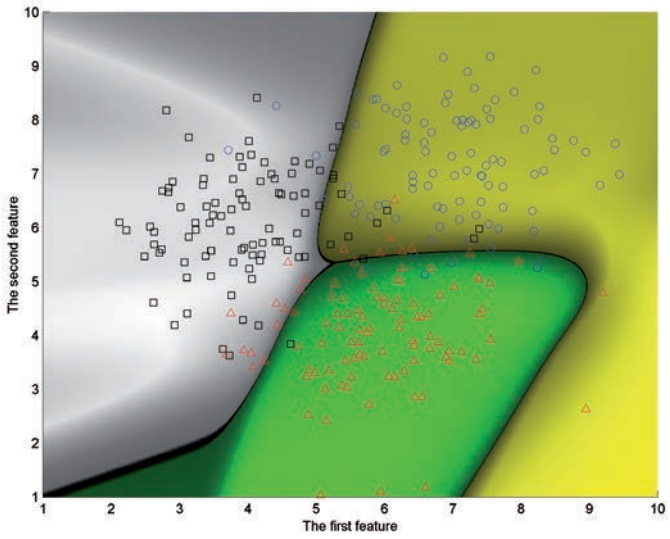

Figure 4.4. New samples plotted on top of the rules of Figure 4.2.

This is the best way to check whether the classifier has overlearnt the data. However, new data might not always be available. In this case, some data can be put aside before training the classifier. Once the classifier is ready, the held-out data can be shown to it, enabling the evaluation of its ability to generalise. If the data is scarce and sacrificing part of it for testing purposes does not sound appealing, then one can take a step further and do cross-validation (see Section 3.1).

When assessing the performance of a classifier, it is thus very important not to use the capability to discriminate between the samples in the training set, but to use either a held-out test set, cross-validation, or something similar — otherwise, the results will be biased. Fortunately, this mistake is not really committed by machine learning researchers anymore, largely thanks to theoretical frameworks such as the bias–variance decomposition (see, e.g., Geman et al., 1992; Hastie et al., 2008) and capacity (Guyon et al., 1992) that have been developed to improve the understanding of the problem.

4.3 Overfitting of the second kind

When one wants to pick the best-performing classifier from among a large number of candidates, a test set can be used to find out which classifier has the best performance on new data. However, just as the discriminatory power demonstrated in Figure 4.3 should not be taken as the accuracy of the corresponding classifier in classifying new data, the accuracy with which the best classifier is able to classify the test set should not be taken as an estimate of its performance on previously unseen data.

This perhaps counter-intuitive fact is vital in any model selection business, and also highly important in understanding the results of this thesis. What follows is an example presented by Jensen and Cohen (2000) that should make the problem obvious:

We are hiring somebody to tell us each day whether the stock market will go up or down the next day. We want to make sure that we hire a capable person, so we come up with a test: the candidate makes predictions for 14 days, and we require 11 of them to be correct in order to believe that he or she is competent. This way, a candidate who is making any equivalent of random guesses will pass the test with a probability as low as 2.87%.

We do not think that the test is going to be an easy one, so we choose ten

candidates to be evaluated. Now, there is a probability of $1-(1-0.0287)^{10}$, or 25.3%, that if everyone is making random guesses, at least one of them will pass the test. Our test is not very useful anymore.

A friend of ours is also evaluating stock market experts. Using the same test, he starts by evaluating 30 candidates. Provided that everyone is really making random guesses, he has a probability of 58.3% of hiring one. Now does this mean that he has a better chance of having a competent expert than we do? No, it does not, because everyone is guessing randomly anyway.

The test only works as intended in predicting a candidate's skill when no selection of candidates has been made based on the results of the test. When a number of candidates is ranked using the test, the scores of those who pass the test should no longer be used for anything. Instead, a new test period of 14 days can be devised in order to estimate their true performance.

This is essentially the same as having two statistical distributions, drawing 10 samples from one and 30 from the other, and comparing the distributions using the maximum value obtained for each distribution. In the case of identical Gaussian distributions, it is probable that the distribution from which 30 samples are drawn scores higher. One might repeat the test several times, each time drawing 10 and 30 samples, and believe that the second distribution produces larger values — even though in reality, the two distributions are very much identical.

In this thesis, the statistical phenomenon taking place when several classifiers are compared is called *overfitting of the second kind*. It can happen not only with a separate test set, but with cross-validation as well[1]. Indeed, this is a phenomenon that one needs to understand even when maximising the Bayesian evidence, or optimising theoretical performance bounds (Cawley and Talbot, 2010).

Overfitting of the second kind is best fought as is that of the first kind. That is, to compare the classifiers chosen, you can have a fresh test set — one that is not used during the training, nor the selection, of the classifiers. Or, if the data is scarce, you can have an outer loop of cross-validation to check which source of the classifiers performs best. How to properly assess the performance of any predictor has been pointed out for

[1]Ng (1997) has shown that under special circumstances, we may even *know* that the model with the best CV score cannot be the best one.

example by Scheffer and Herbrich (1997), and Cawley and Talbot (2010).

4.4 Overfitting in feature selection

During the search for good feature subsets, one gets as a side result the cross-validation scores used for guiding the search. It is tempting to use these scores for other purposes as well, as the CV estimate is known to be a rather unbiased estimate of the ultimate classification performance, especially when the number of folds is large enough (Kohavi, 1995).

However, when several models are compared based on the CV scores, the non-zero variance of the estimates can play a significant role. With a high variance and many models, one may practically end up using random differences for choosing the ultimate model. If the CV scores are then used in making any conclusions, there can be severe overfitting of the second kind.

Kohavi and Sommerfield (1995) mentioned the existence of the overfitting problem in the context of feature selection, but their experimental results suggest that the problem is only a theoretical one. However, the results of this thesis clearly indicate that the CV estimates are next to useless once the search has been finished. As a kind of warning, two specific pitfalls that are waiting for the ignorant practitioner are described in the following.

5. Pitfalls

Chapters 2–4 served as a general introduction to automatic classification, including the subtopics of feature selection and overfitting. This chapter now builds on top of that foundation, and presents the actual results of Publications I and II in a compact form.

5.1 A pitfall in comparing selection algorithms

A straightforward way to compare two feature selection methods is to compare the best evaluation scores found by each algorithm. If this is done with a randomly sampled half of the well-known sonar dataset[1], the results shown in Figure 5.1 can be obtained[2]. Here, it seems obvious that SFFS has found much better subsets than SFS did.

However, these are the overfitted cross-validation scores found by the search algorithms that are plotted in Figure 5.1. When the same subsets are used to train new classifiers, and the other half of the original dataset is used to test these classifiers, the results look quite different. Shown in Figure 5.2, the curves now indicate not only that the scores found by both algorithms are overfitted, but also that the LOOCV accuracy estimates for SFFS are much more biased (optimistically) than are those for SFS. Indeed, when the study is repeated with different holdout sets, it becomes obvious that the true error level is essentially the same for both algorithms.

Both algorithms select the feature subsets properly using cross-validation, so there should be no overfitting of the first kind. However, SFFS evaluates many more candidates than SFS does, which means that a direct

[1] Available at the UCI ML Repository, http://archive.ics.uci.edu/ml/
[2] Using the 1NN classification rule and leave-one-out cross-validation

comparison between the CV scores of the best candidates introduces overfitting of the second kind, effects of which are clearly visible when Figures 5.1 and 5.2 are compared.

Publication I points out that comparative studies even by distinguished scientists have only shown curves corresponding to Figure 5.1, omitting anything like those in Figure 5.2. It is shown in the publication that this results in completely invalid conclusions with respect to the expected performance of the feature selection algorithms. Moreover, it is shown that the rather widely accepted claim of SFFS outperforming SFS in almost any task may not have been studied closely enough.

Independently of the present work, Ambroise and McLachlan (2002) pointed out very much the same thing as Publication I did. What might appear surprising however is that the issue continues to be relevant — more recently, it has been emphasised by Smialowski et al. (2010), for example.

5.2 A pitfall in choosing the optimal number of features

Another question that practitioners have been answering using the CV scores is one about the optimal number of features to choose. Once again,

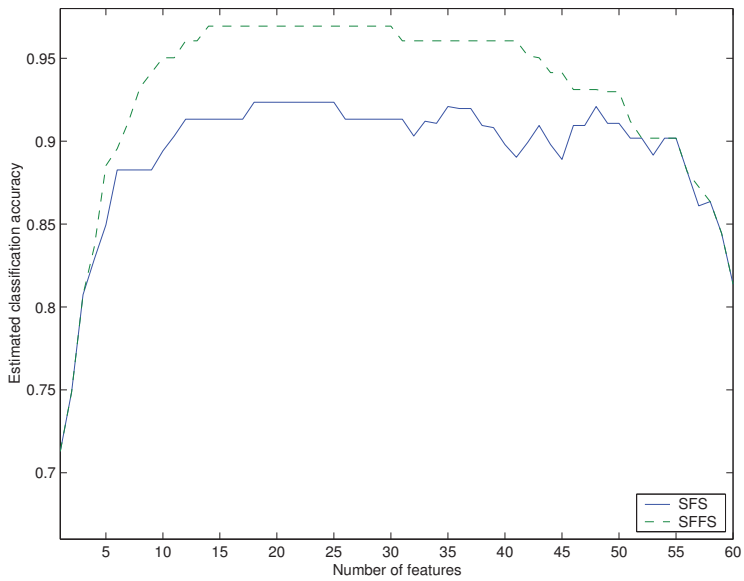

Figure 5.1. The benefits of the feature subsets of different sizes as estimated by LOOCV during the search.

noncautious reasoning leads to picking the subset size with which the maximum CV score has been obtained. (In case of a draw, one might prefer to choose the subset with the smallest number of features.) In Figure 5.1, the winner would be the subset with 18 features for SFS, and that with 14 features for SFFS. When comparing the results of the winners to the score obtained for the full set with all the 60 features, it seems that the exclusion of some of the features makes it possible to build a significantly more accurate classifier.

Again, Figure 5.2 rapidly reveals the truth: First, with SFFS, the optimal number of features based on these subsets seems to be at least 35 if not even 55 — or, if a very small number of features is desired, maybe 2. In particular, the subset with 14 features does not seem to be optimal in any sense. On the other hand, SFS seems to have found a zone of well-performing subsets with sizes around 15 features. However, Figure 5.1 suggests that they are not as good as those having around 20 features — therefore, without independent test data, one has no choice but to believe that the optimal subset size is 18. Second, the subsets suggested by the CV scores are no better in classifying the test data than is the full set of all the candidate features. On the contrary, in this case the so-called winners actually happen to be worse in this respect — therefore, the pruning of the features is certainly not as easy a path to increasing the classification

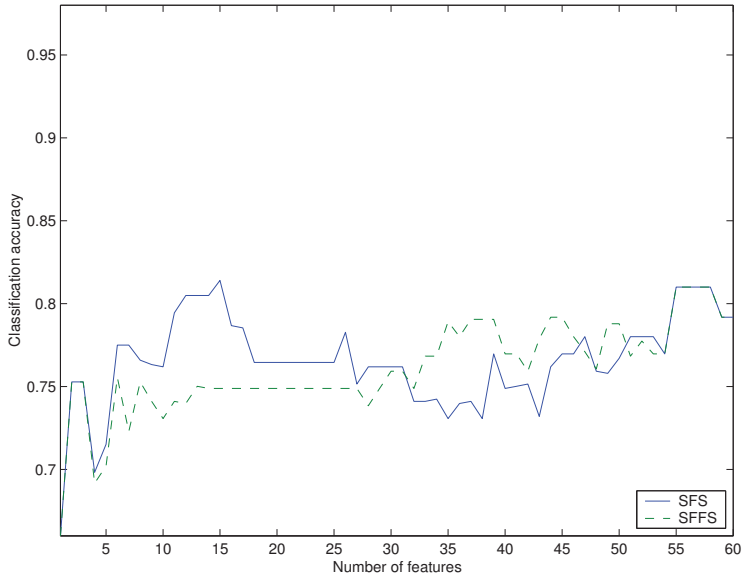

Figure 5.2. The benefits of the feature subsets of different sizes, measured using an independent test set.

accuracy as it might seem based on Figure 5.1.

Why this actually happens is again due to overfitting of the second kind. Simply because both search algorithms evaluate more subsets for some subset sizes, the estimates for these sizes get more overfitted than do those associated to the other sizes. In particular, only one set with all the features is evaluated, which means that the estimate related to it is not really overfitted at all. When a highly optimistic estimate overfitted for some subset size by an intensive search is compared to this realistic estimate for the full set, it often happens that the overfitted estimate predicts lower error, although the situation could look completely different if similarly biased estimates were used.

Publication II shows that this happens surprisingly often, and with different kinds of classifier training algorithms, subset selection methods, and datasets. Thus, the many examples in the literature about feature selection helping in increasing the classification accuracy become suspect to a closer investigation, as it can be that only the overfitted CV scores have actually been used to verify this.

But wait — there is even more to it. It has been suggested (see Section 4.3) that an outer loop of cross-validation should be used for a non-biased estimate. Such an outer loop can be implemented as outlined in Algorithm 6: first, properly cross-validated estimates are obtained for the performance of each subset size (steps 2–4); second, the best-performing subset size is chosen (step 5). Now, the corresponding performance estimate (step 6) is a result of overfitting on yet another level: the results still are likely to suggest that feature selection improves performance more than it really does. This effect is studied in more detail in Publication III.

5.3 Pitfalls do not lurk everywhere

The phenomena described in Sections 5.1 and 5.2 are very common, as the results in Publications I and II show. However, already in Publication I a single counter-example was shown: for the `mushroom` data[3], the SFFS algorithm indeed finds feature subsets that perform better on independent test data than do those due to SFS.

Further, in Publication V a situation is described where SBFS finds bet-

[3]Also available at the UCI ML Repository

ter subsets than SBS does, *and* the subsets found indeed perform better than does the full set comprising all the candidate features. In other words, neither of the pitfalls described in Sections 5.1 and 5.2 seem to exist. These counter-examples prove that the existence of at least some intensive search algorithms (such as SBFS) is justified; indeed, the opposite was never claimed in Publications I or II. Instead, the point is that the testing methodology must be sound — otherwise, we will never know whether or not we actually fell into a pitfall (or two).

1.	Choose a feature selection algorithm, such as SFS or SFFS.
2.	Divide all the data available into K folds.
3.	**for** k := 1 **to** K, (The evaluation loop.)
3.1	Create a (training) set $T^{(-k)}$, which includes the samples of all the folds, except those in the kth.
3.2	Using $T^{(-k)}$, perform a single run of the feature selection algorithm. An inner loop of cross-validation may divide $T^{(-k)}$ further.
3.3	Train classifiers using $T^{(-k)}$ and the obtained subsets of different sizes.
3.4	Test these classifiers using the samples in the kth fold, and record the performance. In what follows, these estimates are referred to as $x_i^{(k)}$, $i = 1, 2, \ldots, D$, where D is the total number of candidate variables.
	end;
4.	For each subset size, determine the average of the K estimates obtained in step 3.4: $\bar{x}_i = \frac{1}{K} \sum_{k=1}^{K} x_i^{(k)}$.
5.	Out of all the subset sizes, find the one that minimises the average error: $\hat{d} = \arg\min_i(\bar{x}_i)$. This is the estimate for the optimal subset size.
6.	The corresponding mean value, $\hat{x} = \min_i(\bar{x}_i)$, is the estimate for the minimum error level that can be reached.
7.	Perform a single run of the feature selection algorithm having all the data available.
8.	Choose the winner from among the subsets having the size defined in step 5.

Algorithm 6. Determining the optimal subset size and the corresponding performance using an outer loop of cross-validation.

6. Solutions

In the previous chapter, two pitfalls related to overfitting in feature selection were described. Now in this chapter, a method suggested in Publication III to overcome these issues is presented. In addition, extensions to generic model selection and feature group selection are discussed. Finally, the use of some of the methods in a regression context is covered.

6.1 Cross-indexing

Suggested in Publication III and then generalised in Publication IV, *cross-indexing* is a method for avoiding the pitfall described in the end of Section 5.2 — namely the issue of properly estimating the very benefits of selecting features in the first place. For clarity, the original (non-generalised) cross-indexing A and B algorithms are presented first.

Cross-indexing A is delineated in Algorithm 7. In step 4.1, averaged estimates for each subset size are computed much like in Algorithm 6 (step 4) but separately for each fold k, always ignoring the results obtained during the kth iteration of the evaluation loop. These estimates are then used in step 4.2 to determine the optimal subset size $\hat{d}^{(-k)}$. The corresponding performance estimate is obtained by recalling the performance estimated during iteration k of the evaluation loop for the subset having size $\hat{d}^{(-k)}$ (step 4.3). In the end, the results for the k iterations are averaged to produce the final estimates (steps 5 and 6). In Publication III, it was shown that the positive bias of this approach, if any, is bounded by that of outer-loop CV.

On the other hand, in cross-indexing B outlined in Algorithm 8, the estimate number k for the optimal subset size, $\hat{d}^{(k)}$, is determined in step 4.1 using the estimates obtained during the kth iteration of the evaluation

1–3.	Perform steps 1–3 of Algorithm 6, including the substeps.
4.	**for** k := 1 **to** K, (The indexing loop.)
4.1	For each subset size, take the average of the $K-1$ estimates obtained for the *other* folds: $\bar{x}_i^{(-k)} = \frac{1}{K-1}\left(\sum_{j=1}^{k-1} x_i^{(j)} + \sum_{j=k+1}^{K} x_i^{(j)}\right)$.
4.2	Find the subset size using which minimum error is reached: $\hat{d}^{(-k)} = \arg\min_i \left(\bar{x}_i^{(-k)}\right)$.
4.3	Record the performance for the best subset of size $\hat{d}^{(-k)}$ that was obtained during the kth iteration: $x_{\hat{d}^{(-k)}}^{(k)}$.
4.4	For comparison purposes, you can also record the performance for the *full* feature set on the kth iteration: $x_D^{(k)}$.
	end;
5.	Average all the K subset sizes obtained during the different executions of step 4.2. This average is the estimate for the optimal subset size.
6.	Average all the K performance estimates obtained in step 4.3. This average is the estimate for the performance of the best subset having the size discovered during step 5.
7–8.	Perform steps 7–8 of Algorithm 6.

Algorithm 7. The cross-indexing A algorithm. Median or other statistical descriptors could also be used instead of the average in steps 4.1, 5 and 6, if applicable.

loop. This estimate is then used to look up the performances for the same subset size, but for the other iterations (step 4.3).

In Publication III, it is shown that a straightforward implementation of the outer-loop CV approach (Algorithm 6) tends to come up with an optimistically biased estimate for the performance of the best subset. Then, it is pointed out that using cross-indexing (A or B) significantly decreases the magnitude of this bias.

In cross-indexing A, one uses — during each of the K iterations of the indexing loop — as many as $K-1$ estimates to pick the optimal complexity for the model, but only one estimate in order to evaluate it. Normally, this results in a good selection, but perhaps noisy estimation. On the other hand, in cross-indexing B, a single estimate is used to select the model complexity or hyperparameter values, whereas several are employed during the assessment. This may lead to an accurate evaluation of an inferior choice.

What we want is to be able to find a suitable compromise between model selection and assessment. This is possible when using the generalised $(N, K-N)$-fold cross-indexing introduced in Publication IV and reproduced here as Algorithm 9. By choosing a large N, the compromise leans towards the selection of the models, whereas selecting N close to 1 em-

1–3.	Perform steps 1–3 of Algorithm 6, including the substeps.
4.	**for** k := 1 **to** K, (The indexing loop.)
4.1	Pick the subset size using which minimum error was obtained on the kth execution of step 3.4: $\hat{d}^{(k)} = \arg\min_i \left(x_i^{(k)} \right)$.
4.2	Record the performance for this very subset size on all the other iterations except the kth.
4.3	Compute the average of the $K-1$ estimates obtained in step 4.2: $\tilde{x}_{\hat{d}^{(k)}}^{(-k)} = \frac{1}{K-1} \left(\sum_{\ell=1}^{k-1} x_{\hat{d}^{(k)}}^{(\ell)} + \sum_{\ell=k+1}^{K} x_{\hat{d}^{(k)}}^{(\ell)} \right)$.
4.4	For comparison purposes, you can also record the performance for the full set on all the other $K-1$ iterations.
	end;
5.	Average all the K subset sizes obtained during the different executions of step 4.1. This average is the estimate for the optimal subset size.
6.	Average all the K performance estimates obtained in step 4.3. This average is the estimate for the performance of the best subset having the size discovered during step 5.
7–8.	Perform steps 7–8 of Algorithm 6.

Algorithm 8. The cross-indexing B algorithm. Again, the statistical measure computed in steps 4.3, 5 and 6 need not be the average.

phasises the evaluation of those models that are selected. At the two extremes, we get Algorithms 7 and 8: cross-indexing A can now be replaced with $(K-1,1)$-fold cross-indexing, whereas cross-indexing B is directly obtained as the special case $N = 1$.

According to preliminary results, a choice of N greater than 1 but less than $K-1$ produces favourable results when compared to the previous, non-generalised versions.

6.2 Generic model selection

Wrapper-based feature selection is a form of model selection where models having (usually) similar architecture but different sets of input features are compared to each other. Another form is the act of choosing good values for the hyperparameters of a given model, such as the optimal value of k for a kNN classifier (see Section 2.2.1). On a more general level however, models of completely different kinds can be compared to each other (Guyon et al., 2010).

Model selection can be done in different ways, but it seems that almost every approach boils down to cross-validation at some point: CV "seems to be the universally accepted ultimate remedy" (Guyon et al., 2010). How-

1-3.	Perform steps 1–3 of Algorithm 6, including the substeps.
4.	**for** $k := 1$ **to** K, (The indexing loop.)
4.1	For each subset size i, compute the average of N estimates: $\dot{x}_i^{[k]} = \frac{1}{N}\left(\sum_{j=\alpha}^{K} x_i^{(j)} + \sum_{j=\beta}^{k} x_i^{(j)}\right)$, where $\alpha = K + k - N + 1$, and $\beta = \max(1, k - N + 1)$.
4.2	Use them to select the optimal subset size: $\hat{d}^{[k]} = \arg\min_i \left(\dot{x}_i^{[k]}\right)$.
4.3	Compute the average for the other $K - N$ folds: $\ddot{x}_{\hat{d}^{[k]}}^{[k]} = \frac{1}{K-N}\left(\sum_{\ell=1}^{k-N} x_{\hat{d}^{[k]}}^{(\ell)} + \sum_{\ell=k+1}^{\gamma} x_{\hat{d}^{[k]}}^{(\ell)}\right)$, where $\gamma = \min(K, K + k - N)$.
4.4	For comparison purposes, you can also record the performance for the full set on the other $K - N$ folds: $\frac{1}{K-N}\left(\sum_{\ell=1}^{k-N} x_D^{(\ell)} + \sum_{\ell=k+1}^{\gamma} x_D^{(\ell)}\right)$,
	end;
5.	Average all the K subset sizes obtained during the different executions of step 4.2. This average is the estimate for the optimal subset size.
6.	Average all the K performance estimates obtained in step 4.3. This average is the estimate for the performance of the best subset having the size discovered during step 5.
7–8.	Perform steps 7–8 of Algorithm 6.

Algorithm 9. Generalised $(N, K - N)$-fold cross-indexing. (When $a > b$, $\sum_a^b(\cdot) = 0$.)

ever, intensive comparisons are still prone to overfitting (Cawley and Talbot, 2010), very much like described in Chapter 5 in the context of feature selection.

In Section 6.1, the cross-indexing method was introduced for overcoming the issue of optimistically biased performance estimates in the context of feature selection. As a rather natural extension, Publication IV delineates the modifications that are necessary to be able to use the cross-indexing approach in a generic model selection task: selecting the best model structure from a number of candidates, and at the same time finding the optimal values for the hyperparameters of that model. The approach indeed proves useful, because it can estimate the performance of the selected models rather well[1] while at the same time being able to select highly competitive models[2].

[1]The second best entry in this sense out of the 28 in the WCCI 2006 Performance Prediction Challenge (Guyon et al., 2006b)

[2]Very good performance in the NIPS 2006 Model Selection Game (rank: 1/3) and the IJCNN 2007 Agnostic Learning vs. Prior Knowledge Challenge (ranks: 3/13 and 2/7) (Guyon et al., 2007, 2008)

6.3 Sensor selection

Sometimes, it is possible that the features available for feature selection are somehow *grouped*: for instance, there can be several sensors such that each of them outputs a certain number of features. Then, it might be more meaningful to select *sensors* or *feature groups* rather than individual features (e.g., Subrahmanya and Shin, 2008).

In Publication V, an application is discussed where a number of *electrodes* are used to inject alternating current into a target volume, such that the voltage potentials on the surface are then measured. Then, these potentials are used to infer the conductivity distribution inside the target volume. Depending on the exact setup, with just a dozen or two electrodes one can generate hundreds or thousands of voltage measurements (features).

The aim in the publication is to decrease the number of electrodes to be as small as possible. Now, it is clear that if all the measurements requiring a certain electrode (either for injection or for measurement) are pruned, then the electrode can be dropped from the device. However, there are two issues: First, with hundreds or thousands of measurements, the dimension of the feature subset space (i.e., the search space) is very large. Second, it may be that one ends up with a few dozen measurements that still require almost all the electrodes for injection, measurement, or both — in this case, one would not be able to drop any significant number of electrodes, which however was the primary aim of the study.

To those ends, standard feature selection algorithms were modified in Publication V to select electrodes instead of raw features. Because a single electrode is never useful in this context, one would have difficulties getting started with forward selection. Therefore, backward selection methods SBS and SBFS (see Section 3.2, Algorithms 4 and 5) were employed in this study. When an electrode is pruned from the set being evaluated, then all the injections and measurements due to it have to be removed as well. This means that the number of features presented to the learning machine falls rather quickly.

6.4 Regression

So far, this thesis has addressed classification only. However, a related problem is that of regression: given a set of input features (or input variables), predict the value of a dependent variable (or set of variables). Usually, the dependent variable is continuous: it can have any real value in the given domain, whereas in classification the number of possible outcomes is limited and depends on the number of classes available.

In Publication V, the dependent variables being estimated are the model parameters using which the unknown conductivity distribution inside the target volume can be reconstructed.

In this application, one does not really seem to fall into the pitfalls described in Sections 5.1 and 5.2; in other words, here the floating search (SBFS) outperforms the non-floating version (SBS), and the exclusion of at least one electrode is clearly beneficial. However, without measuring the benefits properly, there would be no way of knowing whether or not there was a pitfall waiting in the first place.

It remains unknown and perhaps a relevant topic for further research whether the apparent inexistence of the pitfalls is a result of doing regression instead of classification, electrode selection instead of raw feature selection, or backward selection instead of forward selection — or the combination of some of these, or something completely different. Nevertheless: based on Publications I, II and V, it still appears that more often than not, the pitfalls are there waiting for the ignorant researcher.

7. Summary and conclusions

This thesis points out a number of shortcomings in existing feature selection literature. Basically all of them are due to ignorance regarding the overfitting phenomenon. In practice, the following more or less widely accepted claims are much weakened by the empirical results of the thesis:

1. More exhaustive search algorithms, in particular floating selection methods, are able to find feature subsets that are much better than those discovered by straightforward sequential techniques.
2. The pruning of the feature set is often very useful with respect to the ultimate classification accuracy.
3. The optimal subset size due to feature selection is usually much smaller than the total number of candidate features.
4. Current feature selection techniques provide an easy remedy to the problems caused by an increase in the dimensionality.

No claim is made that these propositions are always or even typically false — only that many of the studies stating them seem to have been carried out in an incorrect fashion, which obviously casts a shadow of doubt over the actual results.

The principal aim of this thesis has been to warn that it is surprisingly easy even for an experienced pattern recognition researcher to obtain invalid results, if the methodology used is not thoroughly sound. Hence, the reader of research reports on feature selection should be extremely alert and try to verify, if possible, that the studies have been conducted as they should have been. In the event that this cannot be assured, one should take a highly suspicious stance towards any results.

Fortunately, it seems that the warnings given (especially in Publications I and II) have not gone unnoticed by the researchers of machine learning (Cano et al., 2003; Bahamonde et al., 2004; Statnikov et al., 2004; Fiebrink and Fujinaga, 2006).

Bibliography

D. W. Aha and R. L. Bankert. A comparative evaluation of sequential feature selection algorithms. In D. Fisher and J.-H. Lenz, editors, *Artificial Intelligence and Statistics V*, pages 199–206. Springer-Verlag, 1996.

C. Ambroise and G. J. McLachlan. Selection bias in gene extraction on the basis of microarray gene-expression data. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 99, pages 6562–6566, 2002.

A. Bahamonde, G. F. Bayón, J. Díez, J. R. Quevedo, O. Luaces, J. J. del Coz, J. Alonso, and F. Goyache. Feature subset selection for learning preferences: A case study. In *Proceedings of the Twenty-First International Conference on Machine Learning (ICML-2004)*, pages 49–56, 2004.

C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

J. R. Cano, F. Herrera, and M. Lozano. Using evolutionary algorithms as instance selection for data reduction in KDD: An experimental study. *IEEE Transactions on Evolutionary Computation*, 7(6):561–575, 2003.

G. Cawley and N. Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11:2079–2107, 2010.

C. Y. Chang. Dynamic programming as applied to feature subset selection in a pattern recognition system. *IEEE Transactions on Systems, Man and Cybernetics*, 3(2):166–171, 1973.

T. Cibas, F. Soulié, P. Gallinari, and S. Raudys. Variable selection with optimal cell damage. In *Proceedings of the 4th International Conference on Artificial Neural Networks (ICANN'94)*, pages 727–730, 1994.

T. Cibas, F. Soulié, P. Gallinari, and S. Raudys. Variable selection with neural networks. *Neurocomputing*, 12(2–3):223–248, 1996.

T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, January 1967.

R. K. De, N. R. Pal, and S. K. Pal. Feature analysis: Neural network and fuzzy set theoretic approaches. *Pattern Recognition*, 30(10):1579–1590, 1997.

P. A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice–Hall International, 1982.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

J. D. Elashoff, R. M. Elashoff, and G. E. Goldman. On the choice of variables in classification problems with dichotomous variables. *Biometrika*, 54(3/4):668–670, December 1967.

R. Fiebrink and I. Fujinaga. Feature selection pitfalls and music classification. In *Proceedings of the Seventh International Conference on Music Information Retrieval (ISMIR 2006)*, pages 340–341, 2006.

R. Fiebrink, C. McKay, and I. Fujinaga. Combining D2K and JGAP for efficient feature weighting for classification tasks in music information retrieval. In *Proceedings of the Sixth International Conference on Music Information Retrieval (ISMIR 2005)*, pages 510–513, 2005.

K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1972.

S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

I. Guyon, V. Vapnik, B. Boser, L. Bottou, and S. A. Solla. Structural risk minimization for character recognition. In J. E. Moody, S. J. Hanson, and R. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 471–479. Morgan Kaufmann, 1992.

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1–3):389–422, 2002.

I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, editors. *Feature Extraction — Foundations and Applications*. Springer, 2006a.

I. Guyon, A. Saffari, G. Dror, and J. Buhmann. Performance prediction challenge. In *Proc. of the Int. Joint Conf. on Neural Networks (IJCNN06), IEEE World Congress on Computational Intelligence (WCCI06)*, pages 2958–2965, 2006b.

I. Guyon, A. Saffari, G. Dror, and G. Cawley. Agnostic Learning vs. Prior Knowledge Challenge. In *Proc. of the Int. Joint Conf. on Neural Networks (IJCNN07)*, pages 829–834, 2007.

I. Guyon, A. Saffari, G. Dror, and G. Cawley. Analysis of the IJCNN 2007 Agnostic Learning vs. Prior Knowledge Challenge. *Neural Networks*, 21(2–3): 544–550, 2008.

I. Guyon, A. Saffari, G. Dror, and G. Cawley. Model selection: Beyond the Bayesian/frequentist divide. *Journal of Machine Learning Research*, 11:61–87, 2010.

M. T. Hagan and M. B. Menhaj. Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(6):989–993, 1994.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2008.

S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, second edition, 1999.

B. A. Huberman and L. A. Adamic. Growth dynamics of the World-Wide Web. *Nature*, 401(6749):131, 1999.

D. D. Jensen and P. R. Cohen. Multiple comparisons in induction algorithms. *Machine Learning*, 38(3):309–338, 2000.

G. H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the Eleventh International Conference on Machine Learning (ICML-94)*, pages 121–129, 1994.

J. Kittler. Feature set search algorithms. In C. H. Chen, editor, *Pattern Recognition and Signal Processing*, pages 41–60. Sijthoff & Noordhoff, 1978.

R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1137–1143, 1995.

R. Kohavi and G. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1–2):273–324, 1997.

R. Kohavi and D. Sommerfield. Feature subset selection using the wrapper model: Overfitting and dynamic search space topology. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95)*, pages 192–197, 1995.

S. Lawrence, C. L. Giles, and A. C. Tsoi. Lessons in neural network training: Overfitting may be harder than expected. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 540–545, 1997.

P. M. Lewis. The characteristic selection problem in recognition systems. *IRE Transactions on Information Theory*, 8(2):171–178, 1962.

T. Marill and D. M. Green. On the effectiveness of receptors in recognition systems. *IEEE Transactions on Information Theory*, 9(1):11–17, 1963.

G. E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8):114–117, 1965.

P. M. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers*, 26(9):917–922, 1977.

A. Y. Ng. Preventing "overfitting" of cross-validation data. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML-97)*, pages 245–253, 1997.

P. Pudil, F. J. Ferri, J. Novovičová, and J. Kittler. Floating search methods for feature selection with nonmonotonic criterion functions. In *Proceedings of the 12th International Conference on Pattern Recognition (ICPR'94)*, pages 279–283, 1994a.

P. Pudil, J. Novovičová, and J. Kittler. Floating search methods in feature selection. *Pattern Recognition Letters*, 15(11):1119–1125, 1994b.

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

S. Raudys. Feature over-selection. In D.-Y. Yeung, J. Kwok, A. Fred, F. Roli, and D. de Ridder, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, volume 4109 of *Lecture Notes in Computer Science*, pages 622–631. Springer, 2006.

J. Reunanen. Search strategies. In I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, editors, *Feature Extraction — Foundations and Applications*, pages 119–136. Springer, 2006.

M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of the IEEE International Conference on Neural Networks (ICNN93)*, pages 586–591, April 1993.

I. Rodriguez-Lujan, R. Huerta, C. Elkan, and C. Santa Cruz. Quadratic programming feature selection. *Journal of Machine Learning Research*, 11:1491–1516, 2010.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*, pages 318–362. MIT Press, 1986.

R. J. Schalkoff. *Pattern Recognition: Statistical, Structural and Neural Approaches*. John Wiley & Sons, 1992.

T. Scheffer and R. Herbrich. Unbiased assessment of learning algorithms. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 798–803, 1997.

R. Setiono and H. Liu. Neural-network feature selector. *IEEE Transactions on Neural Networks*, 8(3):654–662, 1997.

W. Siedlecki and J. Sklansky. On automatic feature selection. *International Journal of Pattern Recognition and Artificial Intelligence*, 2(2):197–220, 1988.

P. Smialowski, D. Frishman, and S. Kramer. Pitfalls of supervised feature selection. *Bioinformatics*, 26(3):440–443, 2010.

P. Somol and P. Pudil. Oscillating search algorithms for feature selection. In *Proceedings of the 15th International Conference on Pattern Recognition (ICPR'2000)*, pages 406–409, 2000.

P. Somol, P. Pudil, J. Novovičová, and P. Paclík. Adaptive floating search methods in feature selection. *Pattern Recognition Letters*, 20(11–13):1157–1163, 1999.

A. Statnikov, C. F. Aliferis, and I. Tsamardinos. Methods for multi-category cancer diagnosis from gene expression data: A comprehensive evaluation to inform decision support system development. In *Proceedings of the Eleventh World Congress on Medical Informatics (MEDINFO 2004),* to appear, 2004.

S. D. Stearns. On selecting features for pattern classifiers. In *Proceedings of the Third International Joint Conference on Pattern Recognition*, pages 71–75, 1976.

N. Subrahmanya and Y. C. Shin. Automated sensor selection and fusion for monitoring and diagnostics of plunge grinding. *Journal of Manufacturing Science and Engineering*, 130(3):031014: 1–11, 2008.

C. W. Swonger. Property learning in pattern recognition systems using information content measures. In *Proceedings of the IEEE Workshop on Pattern Recognition*, pages 329–347, 1968.

I. Tuomi. The lives and death of Moore's law. *First Monday*, 7(11), 2002.

A. W. Whitney. A direct method of nonparametric measurement selection. *IEEE Transactions on Computers*, 20(9):1100–1103, 1971.

E. P. Xing, M. I. Jordan, and R. M. Karp. Feature selection for high-dimensional genomic microarray data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML-2001)*, pages 601–608, 2001.

DISSERTATIONS IN INFORMATION AND COMPUTER SCIENCE

TKK-ICS-D18 Liitiäinen, Elia.
Advances in the Theory of Nearest Neighbor Distributions. 2010.

TKK-ICS-D19 Lahti, Leo.
Probabilistic Analysis of the Human Transcriptome with Side Information. 2010.

TKK-ICS-D20 Miche, Yoan.
Developing Fast Machine Learning Techniques with Applications to Steganalysis Problems. 2010.

TKK-ICS-D21 Sorjamaa, Antti.
Methodologies for Time Series Prediction and Missing Value Imputation. 2010.

TKK-ICS-D22 Schumacher, André
Distributed Optimization Algorithms for Multihop Wireless Networks. 2010.

Aalto-DD99/2011 Ojala, Markus
Randomization Algorithms for Assessing the Significance of Data Mining Results. 2011.

Aalto-DD111/2011 Dubrovin, Jori
Efficient Symbolic Model Checking of Concurrent Systems. 2011.

Aalto-DD118/2011 Hyvärinen, Antti
Grid Based Propositional Satisfiability Solving. 2011.

Aalto-DD136/2011 Brumley, Billy Bob
Covert Timing Channels, Caching, and Cryptography. 2011.

Aalto-DD11/2012 Vuokko, Niko
Testing the Significance of Patterns with Complex Null Hypotheses. 2012.

Are you building statistical predictors using many input variables, without knowing how important each of them is? Would you like to know if it is possible to simply drop some of the inputs? If so, which inputs should you drop? And how is the eventual accuracy of the predictor expected to change as a result of doing so – will the predictor become more accurate or less accurate, and by how much? The thesis at hand studies these questions in detail. Some research results presented earlier in the literature are refuted, and a novel technique to fill the gap is introduced. The new method is also extended to help estimate accuracy in generic model selection tasks, and results from using it in three open competitions are reported.

Aalto-DD 19/2012

ISBN 978-952-60-4515-3
ISBN 978-952-60-4516-0 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934
ISSN 1799-4942 (pdf)

Aalto University
School of Science
Department of Information and Computer Science
www.aalto.fi

BUSINESS + ECONOMY

ART + DESIGN + ARCHITECTURE

SCIENCE + TECHNOLOGY

CROSSOVER

DOCTORAL DISSERTATIONS