Learning Interpretable Models

Dissertation

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

der Universität Dortmund
am Fachbereich Informatik
von

Stefan Rüping

Dortmund

2006

Tag der mündlichen Prüfung:	8. August 2006
Dekan:	Prof. Dr. Bernhard Steffen
Gutachter/Gutachterinnen:	Prof. Dr. Katharina Morik
	Prof. Dr. Heinrich Müller

Danksagung

Ich möchte mich an dieser Stelle bei allen Leuten bedanken, die mich unterstützt und mir bei dieser Arbeit geholfen haben, allen voran bei meiner Freundin Diana und meinen Eltern. Bei Katharina Morik bedanke ich mich für die ausgezeichnete Betreuung und für das fruchtbare und gleichzeitig so freundliche Arbeitsumfeld am LS8. Dazu haben natürlich auch alle jetzigen und ehemaligen Kolleginnen und Kollegen beigetragen, nämlich Andreas, Bülent, Hanna, Harald, Heike, Helge, Ingo, Iris, Oliver, Martin, Michael, Peter, Ralf, Sammy, Stefan, Thorsten und Timm. Ebenfalls bedanken möchte ich mich bei den netten Kolleginnen und Kollegen vom SFB475, insbesondere bei Conny, Karsten, Thomas, Thorsten, Uschi, Uwe und bei Prof. Weihs. Vielen Dank auch an Sigrid Dany und die ganze Supervisionsgruppe. Herrn Prof. Müller danke ich für die Bereitschaft, meine Dissertation zu begutachten. Und für die kleinen Freuden im Leben eines Wimis bedanke ich mich bei Döner-Divan, dem Bioladen "Fruchtbare Erde", Meyerbeer-Kaffee am Campus Nord und der Cafe-Bar in der Südmensa.

There is always an easy solution to every human problem – neat, plausible, and wrong.

Henry Louis Mencken

Contents

List of Figures

Chapter 1

Introduction

More and more data is collected everywhere and sizes of data sets available for knowledge discovery increase steadily. Currently, standard data sets contain several thousands of examples [Murphy and Aha, 1994], while data mining problems with sizes of 80 million examples have been described in the literature [Neiling and Lenz, 2004]. Furthermore, the world's largest data warehouse in 2005 has been reported to have a size of 100 TB [Winter Corporation, 2005] and technologies like the Grid [Foster and Kesselman, 2004] propose a massive increase of data sizes in the near future.

From the data mining perspective, on the one hand this development is good, because learning with high-dimensional data and complex dependencies needs a large number of examples to obtain accurate results. On the other hand, there are several learning problems which cannot be thoroughly solved by simply applying an off-the-shelf standard solution. A main problem is the fact that there are several other performance criteria for a knowledge discovery system than predictive accuracy [Giraud-Carrier, 1998], which is usually the main point of concern when evaluating a machine learning algorithm. A large part of the problem is due to the fact, that currently databases and knowledge discovery systems are meant to operate with less and less costly human interaction.

The handling of data in modern databases is being more and more automated, starting from data input over data analysis up to automatically reacting to new data. For data input, it is seldom the case that a well-trained human operator enters the data into the system. Usually data is entered automatically, for example by scanning incoming and outgoing deliveries in a goods management system, monitoring a patient in an intensive care unit, handling incoming emails or web server requests or taking orders in an online shop. While this reduces the risk of human errors such as typos (6.000 instead of $6,000$) or errors when copying the data, it increases the risk of systematic errors that possibly go undiscovered over a long period of time, because there is too much data for a proper validation or because the system is used in a situation that was not thought of at the time it was designed. For example, the intensive care patient might move around, causing a shift in the ECG readings or some sensor may give incorrect readings under special circumstances. When one is not looking for such problematic data instances, because one does not know that these thing can happen in the first place, it is problematic to discover, explain and correct these errors later.

Data analysis and reacting to incoming data is usually a domain of human experts. However, in several applications this is being automatised as well. For example, in an intensive care unit doctors and nurses cannot monitor a patient all the time, because there are simply too many patients. Instead, they rely on monitoring systems to analyze the state of the patient and give an alarm if a critical condition arises. Similarly, in credit card systems, warehouses, industrial processes management or online stores there are simply too much decisions to be made, decisions are too time critical or involve too much pieces of data to rely on a human. Nevertheless, one is still interested in checking the decisions made and being able to give guarantees about the systems performance. For example, a system used in medical applications should be guaranteed to correctly identify and react to well-known life threatening situations.

Psychology has long since mapped out the limits of the human capability to work with large masses of complex data and several of the discovered human limits are surprisingly small. Using a meta-analysis of several psychological experiments designed to investigate the human capability of information processing, Miller [Miller, 1956] has established the fact that humans can simultaneously deal with only about seven (plus or minus two) cognitive entities. The cognitive entities can of course be highly aggregated and complex and should not be confused with simple observations or attributes used to represent data in typical machine learning applications. For example, a medical practitioner may subsume a whole set of symptoms, measures of vital signs, visual information and theoretical knowledge under the name of one illness and only reason with this description. However, once cognitive entities cannot be combined into a more abstract concept, the bound of seven plus or minus two still holds. It has also been found that humans are seriously limited in estimating the degree of relatedness of more than two variables [Jennings et al., 1982]. An optimal solution of a high-dimensional, large-scale learning task, however, may lead to a very large level of complexity in the optimal solution and in this case, humans can only reason in terms of abstraction, approximation or splitting up problems into independent parts in order to reduce complexity.

This thesis deals with the problem of interpretability of classification models. Interpretability is an important, yet often ignored criterion when applying machine learning algorithms to real-world tasks. In particular, it deals not only with the question how a learning task can be solved in an interpretable way – which is a very subjective question – but also in how far the often conflicting goals of accuracy and interpretability can be integrated into a single framework. In order to lay the ground for the approach taken here, we first have to discuss the concept of interpretability.

1.1 Interpretability

"To understand", in the sense it is used here, is defined as "to grasp the meaning of" something or "to grasp the reasonableness of" someone or something [Merriam Webster, 2004]. In the formal logic sense, an interpretation is a mapping of a formal construct to the entities and their relations it represents. In this sense, someone can be said to understand a formal construct, if he can relate it to the corresponding real-world entities and propositions and reason about the implications. It is important to distinguish the understandability of a model from the understandability of why the model is true or how the model was induced from data, which would by the question of the validity of the model and the understandability of the learning algorithm. For example, the sentence “rhinoceroses can fly” is very easy to understand, as it is easy to relate its words to the properties of certain wild animals (Figure 1.1) and one can easily reason about its implications to, say, aircraft safety, while hardly somebody will hold the sentence for true and it is hard to see how somebody could come up with such a proposition.

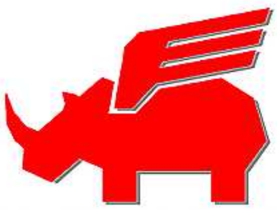


Figure 1.1: Flying Rhinoceros

Interpretability is hard to formalize, as it is a very subjective concept. While one person likes to see his data as plotted in various shapes and colors, another one may better understand a concise textual description of the important properties of the data. One person may like a formal model of the data while the other better understands with single, representative examples. Some person may be more patient when looking at various different plots and properties of the data, carrying on the inspection of the data where others already have given up. Finally, different users have different levels of education and exposure to concepts such as box plots, vector spaces, probability distributions and formal logic, making such formulations more accessible to them.

This explains why the problem of interpretability, important as it is, has received only little attention in the field of machine learning. Most work in this direction either states that one hypothesis language is “naturally” more understandable than another (e.g. because it can easily be translated into natural language sentences), or give empirical evidence by letting domain experts judge the understandability of hypotheses. But as the access to experts is usually very limited and their time expensive, the empirical evidence is mostly not too convincing. Often, a statement about interpretability is only available from a single expert, who has been involved in the project from the start, which can hardly be seen as convincing evidence.

1.1.1 Interpretability as Part of the Data Mining Process

The important point about interpretability is to view data mining as a process and not only as the application of a learning algorithm. Data mining is an iterative, interactive process that contains steps such as understanding the domain the data comes from, understanding the data itself and preparing the data before a learning algorithm can be applied. After the model has been found, it has to be evaluated, the results have to be reported and possibly the data mining chain has to be deployed for practical use. These steps have been formalized in the CRISP data mining process model [Chapman et al., 1999], see Figure 1.2. The single data mining steps are interconnected with each other and one usually has to return to previous steps from time to time, for example when analyzing the model shows that some important information from the application domain is missing in the data.

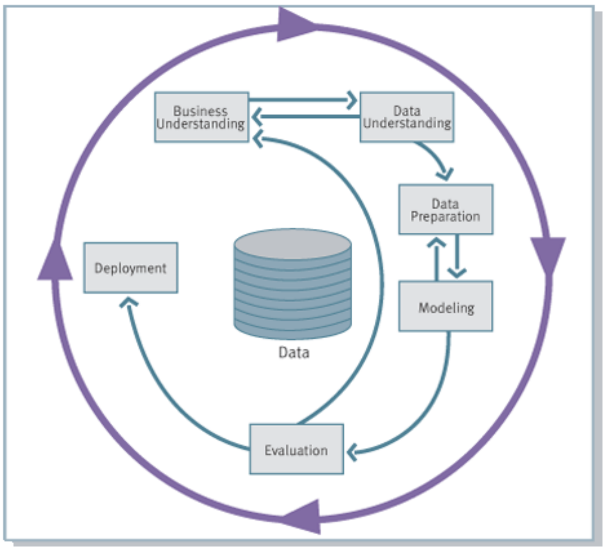


Figure 1.2: The CRISP Data Mining Process Model

Obviously, a successful data mining process needs the cooperation of both data miners and domain experts. This practical insight has been theoretically backed up by the No Free Lunch Theorems [Wolpert and Macready, 1995, Wolpert and Macready, 1997], which state that on the average over all problem instances over a given examples space, all learning algorithms perform equally well. It follows the only chance to get better than random performance in learning lies in finding an adequate representation, such that the important structure behind the data can be found by the learner at hand. To define which structure found in the data is or is not meaningful, one needs the domain expert and to reason about the meaning of structures found by a learning algorithm, these structures have to be presented in an interpretable way.

1.1.2 Interpretability Heuristics

Despite all the problems defining interpretability, there are some heuristics which seem to work in practice and can be used as a guideline to interpretability.

The first heuristic is to use a small number of features and parameters. Following the afore mentioned results from psychology, the number of features and parameters used in a model is a heuristic measure of complexity, if one assumes that every parameter and feature corresponds to one cognitive entity to be reasoned with when thinking about the model. While this assumption is not strictly true, as the user may understand even a high number of features if he can relate the feature values to existing mental concepts and prior knowledge

(e. g., a doctor explaining a large number of symptoms by one disease), this heuristic has often been used in practice. For example, in learning in logic, the number of variables and constants in a formula or set of formulas, the depth of the formula, its structure and the number of formulas can be used to define a measure of model complexity [Sommer, 1996]. For decision trees, the number of nodes is the number of tests that form up the decision boundaries while the depth of the tree is the maximum number of tests that have to be made for a single example to be classified. While it is not clear how big a tree may become to be still called understandable, it is obvious that the smaller a tree is, the more understandable it will be. On the other hand, it has also been reported that in the presence of prior knowledge experts said that shorter trees are "unnatural" and do not "tell enough", despite the fact that the measured accuracy of the trees was significantly higher than the experts accuracy [Bratko, 1996].

The second heuristic is that splitting up a problem into several independent sub-problems reduces the complexity and may still give reasonable results, even if the sub-problems are actually not completely independent. In machine learning, the Naive Bayes algorithm, which treats all features as independent, has been reported to give good results in cases like text categorization, where the features – occurrences of single words – are actually not independent.

While the first two heuristics are more directed at describing the complexity of the model independent of the user, the following two approaches take the prior knowledge of the user into account. One of these approaches is to use examples and basic features instead of formal models and constructed features to explain the learning result to the user. The user can often explain single examples very well, because there is usually more information to a real-world case than what is encoded in the representation of the examples, either because someone forgot to include this data in the examples representation or because the data is too unstructured to work with. For example, in information retrieval a text is usually represented as a vector of word counts, because it is more convenient, while of course reading the text gives much more information about its content. Also, domain experts often have very detailed procedural knowledge, that is they know how to interpret and react to certain situations, but this knowledge is not explicit, they can hardly put this knowledge into words (as a practical example, even an experienced driver can hardly set up a complete model for how to drive a car. Instead, the driving instructor will correct the students' actions in specific situations, letting the student build up his own model). A formal model that characterizes sets of examples, e.g. a linear decision rule, cannot be checked using implicit, procedural knowledge and is thus less interpretable than prototypical examples (examples, to which many examples are similar) and discriminating examples (examples, which are very similar to examples from a different class). Similarly, basic features have usually been chosen, because they characterize a specific, important, and well-defined property in the application domain. The domain expert can reason about this property, while mathematical combination may not make sense in the application (e.g. constructed features like $0.5 * \text{blood pressure} - 1.2 * \text{heart rate}$). Of course, other constructed features (distance per time, length times width) may be perfectly sensible.

The final heuristic seems to be very trivial, but its importance is usually very much underestimated. The heuristic says that people tend to find those things understandable, that they already know. Hence, if a user has much experience with a specific learning algorithm, it may be favorable to keep using this learner, regardless of its accuracy. Even if this learning algorithm is not optimal for the problem at hand, the user may have much experience reasoning with the results and working around the shortcomings of the method. For example, if the users preferred method is susceptible to outliers in the data, the user may have developed a high skill in using his domain knowledge for detecting spurious data points and odd algorithmic results. In this case, applying a theoretically more convenient learning method will lead to worse results, if a non-intuitive hypothesis language hinders the user from using his domain knowledge to improve the results. Hence, the decision to switch to a different learning algorithm should not only take the increase in algorithmic accuracy into account, but also the possible loss of additional information the domain expert may offer to analyze and improve the results. There are also cases, for example in medicine, where a learning algorithm or statistic has become a standard in an application domain for historical reasons, such that one cannot expect the users to change to another algorithm without serious effort even if the new algorithms results are superior to the standard procedure.

But of course, successful as the heuristics are in certain situations, one has to acknowledge that in the end there has yet not been discovered a general way to define and optimize interpretability, other than by shifting the whole responsibility to the user.

1.1.3 Measuring Interpretability

Given the importance of interpretability the obvious question is how the interpretability of a model can be rated. Are there performance criteria that can quantify this interpretability?

Due to the informal nature of the concept of interpretability, a survey over human experts is the most promising measure. In particular in medicine, where human control of the process of treating patients is particularly stressed, this is a popular performance measure of hypotheses [Lavrac, 1998, Zelic et al., 1997, Morik et al., 2002] However, the significance of these tests often suffer from a small number of observations – expert time is precious and a large set of experts is hard to find. Also, in many cases the expert that is judging the results is a part of the data mining team and was involved in the whole mining process, such that his opinion can be severely biased. The opinion of this expert may say more about the quality of the whole process of having a staff of data miners work with you than about the quality of the final model. There are also a number of practical problems constructing a meaningful survey. For example, it does not suffice to simply ask the question “do you understand this model”, but one has to come up with a test to see whether the proband only claims to have understood the model, got it wrong, or actually understood it. Further, it may be hard to separate the aspect of interpretability from the aspect of the personal belief about the validity of the model: a formal model may be more clear when its meaning conforms with what a person already knows. In particular, there can be cases where different schools of expert knowledge exist [Scholz, 2002]. In conclusion, although they are most direct with the least number of assumptions necessary, surveys are not practicable as a measure of interpretability.

A more practical way of measuring interpretability is the use of formal complexity measures for hypotheses. A number of complexity measures have been proposed, for example the number of nodes in a decision tree or its depth, the number of rules in a rule base, the depth of rule, the number of attributes used in a hypothesis, the number of prototypes generated in instance-based learning, the norm of the vector of parameters in a parametric model, measure of the expressibility of the hypothesis space like the Vapnik-Chervonenkis-dimension [Vapnik, 1998] and the Minimum Description Length criterion [Rissanen, 1983], and different statistical information criteria like the Akaike Information Criterion [Akaike, 1973] and the Bayesian Information Criterion [Schwartz, 1979].

The obvious question is how to prove that a formal measure of computational complexity is a valid indicator of such an imprecise concept as interpretability. An interesting study has been performed in [Morik and Muehlenbrock, 1999]. In developmental psychology, the problem of how children explain astronomical phenomena (e.g. answer the question "where does the sun go at night") and how the transitions between different explanations (e.g. from "the sun moves down on the ground" to "the sun revolves around the earth" or "the earth revolves around the sun") has been intensively studied [Baxter, 1989, Nussbaum, 1989, Vosniadou and Brewer, 1992]. Morik and Mühlenbrock modeled these explanations in an inductive logic programming framework and showed that the formal complexity of an explanation rule and the difficulty of a transition from one explanation to another as measured by the necessary formal revisions of the knowledge base is related to the frequency of the children's explanations. This shows that formal complexity measures can be empirically justified in psychological investigations.

One drawback of formal complexity measures is that they are either only applicable for a specific class of models, e.g. the depth of a decision tree, the norm of a parameter vector or the number of prototypes, or provide only a very coarse measure of complexity, like the number of attributes used. Hence, they provide little help in choosing the right hypothesis space in the first place. People sometimes argue that one hypothesis language is "obviously" more intuitive than others, but this is of course highly subjective and hence useless.

On the downside, different definitions of formal measures of complexity have revealed that the problem of minimizing complexity is hard. For example, investigations of the problem of minimizing the number of features used showed that finding a subset of n features, such that no two examples have identical feature values and different class value, is NP-hard [Davies and Russell, 1994]. Even in the case of the seemingly more simple linear classifiers, minimizing the number of features of a separating linear classifier is NP-hard [Amaldi and Kann, 1998]. For learning in logical hypothesis space it has been shown that covering a number of examples with a minimal set of rules is an instance of the set covering problem and hence NP-hard [Breuer, 1970]. In the special case of decision trees it has been shown that minimal decision trees are hard to approximate up to any constant factor [Sieling, 2003].

1.2 Local Models

How can we solve the interpretability problem? Experience shows that one can often find a simple model which provides not an optimal solution, but a reasonably good approximation. The hard work usually lies in improving an already good model. The idea is to separate the entire model into two parts: a more understandable, but less accurate global model, which serves as an approximation to the entire model, plus additional local models which improve the global model on certain, well defined regions of the spaces. The regions are called local patterns and in order to justify the term “local” these regions will have to be reasonably small. In short, the idea can be summarized in the intuitive equation

$$Data = Global\ Model + Local\ Models + Noise$$

proposed by [Hand, 2002].

The main idea is that there are two ways to look at the combined model. In order to get a high-quality prediction, one first finds out if the example falls into a local pattern; in this case the corresponding local model is used, else the global model is used. Seen in this way, learning with local models is a multi-classifier system with the local pattern working on the top level to switch between the single classifiers. On the other hand, if one is interested in interpretability, the global model serves as an approximation to the combined model. Of course, an approximation is only meaningful if guarantees about its quality can be given, and in order to make the approximation quality most transparent to the user, the local patterns are required to be understandable to the user as well. Hence, the user can not only see a simplified version of the real model, but also a description of where the approximative model coincides with the real model and where it may differ. The left side of Figure 1.3 shows the interpretability view on the local models, while the right side shows the accuracy view on the local models.

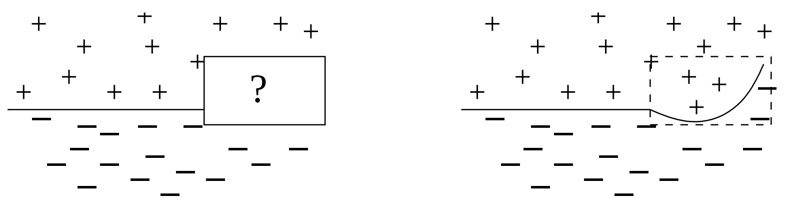


Figure 1.3: The local model idea

Formally, learning with local models can be defined as follows.

Definition 1.2.1 (Global-plus-Local Model)**.** Given two hypothesis spaces $\mathcal{H}_G$, $\mathcal{H}_L$ of classification functions $f : X \to \{-1, 1\}$, a hypothesis space $\mathcal{H}_P$ of clusters in $C \subset X$, a natural number k, a real number $\tau \in [0, 0.5[$ and a probability distribution $P(X)$, a global-plus-local model $(f_G, f_{i,L}, C_{i,L})$ consists of hypotheses $f_G \in \mathcal{H}_G$, $f_{i,L} \in \mathcal{H}_L, i = 1 \ldots k$ and $C_i \in \mathcal{H}_P, i = 1 \ldots k$, such that the probability of drawing a local example is constrained by τ:

$$P(\bigcup_{i=1}^{k} C_i) < \tau$$

$\square$

f_G is called the global model, the $f_{i,L}$ are the local models and C_i are called the local patterns. The requirement $P(C) < \tau$ will be called the τ-constraint.

The idea is that the user defines hypothesis spaces $\mathcal{H}_G$ and $\mathcal{H}_P$ (i.e. learning algorithms) that he finds understandable and also gives the amount of slack τ he is willing to accept in order to improve the results, while the hypothesis space $\mathcal{H}_L$ is selected to optimize accuracy.

There are two possible learning tasks for global-plus-local models, either to optimize accuracy under the interpretability restriction implied by the hypothesis spaces $\mathcal{H}_G$ and $\mathcal{H}_P$ or to explicitely minimize a complexity measure under the restriction that the error is less than a given threshold ϵ. Formally:

Definition 1.2.2 (Prediction-optimal Global-plus-Local Model)**.** Given examples $(x_i, y_i)_{i=1,\ldots,n} \subset X \times \{-1, 1\}$ drawn from a probability measure $P(X, Y)$, hypothesis spaces $\mathcal{H}_G, \mathcal{H}_L$ and $\mathcal{H}_P$, a natural number k and a real number $\tau \in [0, 0.5[$ as in the definition of the global-plus-local model, the problem of finding a prediction-optimal global-plus-local model consists of finding a global-plus-local model, such that the probability $P(X)$ used in the definition of the global-plus-local model is the marginal probability $P(x) = \sum_y P(x, y)$ and the prediction error on new examples drawn from $P(X, Y)$ is minimized. $\square$

Definition 1.2.3 (Interpretability-optimal Global-plus-Local Model)**.** Given examples $(x_i, y_i)_{i=1,\ldots,n} \subset X \times \{-1, 1\}$ drawn from a probability measure $P(X, Y)$, hypothesis spaces $\mathcal{H}_G, \mathcal{H}_L$ and $\mathcal{H}_P$, a complexity measure *comp* on $\mathcal{H}_G$, a real number $\epsilon \in]0, 1[$, a natural number k and a real number $\tau \in [0, 0.5[$ as in the definition of the global-plus-local model, the problem of finding an interpretability-optimal global-plus-local model consists of finding a global-plus-local model, such that the probability $P(X)$ used in the definition of the global-plus-local model is the marginal probability $P(x) = \sum_y P(x, y)$, the prediction error on new examples drawn from $P(X, Y)$ is less than ϵ and the complexity $comp(f_G)$ is minimal over all such models. $\square$

These approaches reduce complexity in two ways. First, a less than optimal hypothesis language can be used for the global model, because errors can still be corrected by the local models. This leaves room for choosing a hypothesis language that optimizes criteria other than the prediction error, namely the interpretability of the global model. Second, for the aspect of discovering new knowledge, it may happen that the global model finds only the obvious patterns in the data that domain experts are already aware of. Patterns are more informative, if they contradict what is already known [Guyon et al., 1996]. Hence, it may also be the case that the local models actually contain the interesting cases.

Of course, it is very easily possible that the combined global-plus-local model is much more complex than a single high-performance model. It is important to keep in mind that only the global model and the local patterns are meant to be interpreted, but not the local models. The goal of learning with local models is not to improve interpretability of the whole model, but to structure the model into understandable and high-performant parts.

1.3 Related Approaches

It is instructive to compare local models with related fields of machine learning. A major connection is with local pattern detection [Hand et al., 2002, Morik et al., 2005], which is defined as the unsupervised search for local accumulations of data points with unexpected high density with respect to some background model [Hand, 2002]. The local patterns in the global-plus-local model are exactly patterns in this definition, when one defines the background model as the data distribution defined by the local model. That is, a high accumulation of positive examples where there only should be negative examples forms a local patterns. The definition of patterns in the global-plus-local model specializes the definition of local patterns of Hand in the sense that it focuses them on local patterns with respect to the conditional class probability $P(y|x)$.

The discovery of frequent itemsets [Agrawal and Srikant, 1994] is an examples of an explicitely local data mining problem. The problem stems from the analysis of market basket data with the goal of identifying products that are usually bought together with each other, for example beer and chips or bread and butter. In terms of local patterns, the problem of frequent itemset analysis can be expressed as follows: given a set of transactions, where each transaction is a set of items, find a set of items whose probability of occurrence is significantly higher than the default probability of each item appearing independently of the others. Association rule analysis goes one step further and tries to find rules from frequent itemsets. Frequent itemset mining has received much attention in research and in practice with numerous algorithms developed to tackle the problem of efficiently mining large data sets [Agrawal et al., 1996, Goethals and Zaki, 2003]. Although frequent itemset mining is a very important subfield of local patterns, they will not be discussed in this thesis. The first reason is that frequent itemset mining is very restricted in the type of data it can handle, which is only bit-vectors (one bit for each item indicating the presence of the item in the transaction). This restriction is necessary to deal with the massive amounts of data in a typical frequent itemset analysis (several millions of transactions), but hinders the use of these algorithms for more general problems. The second reason for not using frequent itemset mining is that in terms of interpretability frequent itemset methods perform very poorly. Usually, several thousands of frequent itemsets and association rules are found on a single data set and these results cannot be inspected with a severe amount of post-processing. Thus, while frequent itemsets are an interesting and relevant research field for problems of local patterns and interpretability, solving these problems is a very specific task and lies outside the scope of this thesis.

Besides locality, the combination of different classifiers is a part of learning with local models. The idea of building systems with multiple classifiers to remove some limitations and extend their capabilities of the base classifiers as such is is not new. Among the proposed approaches for combining multiple classifiers are Voting, combination by order statistics [Tumer and Ghosh, 1995], Meta-Level Learning [Chan and Stolfo, 1993], Stacking [Wolpert, 1992], Cascade Generalization [Gama and Brazdil, 2000] and Boosting [Freund and Schapire, 1996]. All these approaches have in common that they employ a more or less complicated rule to combine the predictions of the individual classifiers into a single prediction. The idea is to minimize the bias and variance of the final classifier by aggregating over a large set of models.

Multiple Classifier Systems are a most successful approach in terms of accuracy, but the interpretability of its model is very limited. The problem is that one has to understand each base model plus the combination strategy in order to understand the complete model. Further, one cannot interpret one of the base models on its own to gain insight into the model, but always has to keep in mind all interactions between the base model. If for example the i-th model shows that a certain combination of attribute values is indicative of the positive class, this does not mean that there is a correlation between these attribute values and the positive class in the data, but only means that the rest of the combined classifier for some reason estimates too much influence of these attributes to the negative class. Also, most combinations methods are greedy, i. e. previous models are not corrected once they have been learned, even if it turns out that they are wrong in several parts. In combination, these problems outweigh the advantages of having simple base models.

An understandable combination of classifiers needs some kind of orthogonality, such that the effect of one model is independent of the effect of the other models, to ensure that the problem can be validly split up into smaller independent parts. One way to ensure this orthogonality is to split up the input space and find out which classifier works best in the different regions. Splitting up the input space can be done either beforehand by clustering or inside the learning procedure. Examples of this approach are [Todorovski and Dzeroski, 2000] and [Todorovski and Dzeroski, 1999]. Decision trees also iteratively split up the input space, such that theoretically one could define the first levels of the tree as a partition of the input space and the following levels as separate classifiers for each partition (but this is probably stretching out the idea of local classifiers too far). More advanced, in [Smyth et al., 1995] decision trees and kernel density estimators have been combined to smoothen the posterior class probabilities. However, in conclusion existing approaches are usually either not easily interpretable or limited to a specific class of base learners.

Finally, it is important to notice that the local patterns also distinguish the global plus local model approach from the seemingly more simple idea of independently learning an understandable and a high-performance model. In contrast to two independently learned models, the local patterns assure that there is a strict, well-defined correspondence between the two models – they can only disagree on the local pattern. The philosophy behind this approach is that accuracy and interpretability are two aspects of the same problem and that their solutions should be as independent as necessary, but as close as possible.

1.4 Goals

The goal of this thesis is to investigate several strategies to improving the interpretability of classification models, in particular by using local models. As a result of the introductory discussion, interpretability is pragmatically defined as employing a user-given hypothesis space or minimizing a user-given complexity criterion such as the number of features in order to not restrict the user to any specific criterion and to use as few assumptions about the nature of interpretability as possible.

In order to obtain results as general as possible, different types of learning algorithms and data sets should be investigated, such that in principle the empirical results hold true for almost all kinds of learners. However, in order to keep the amount of experiments manageable, a total of 18 data sets and 4 learners have been selected for investigation. The data sets will be described in Chapter 8. The learners selected are the Support Vector Machine (SVM) [Cortes and Vapnik, 1995, Vapnik, 1998, Schölkopf and Smola, 2002] with the linear kernel as an example of a linear numeric classifier, the Support Vector Machine with a radial basis kernel as an example of a nonlinear classifier, the C4.5 decision tree algorithm [Quinlan, 1993] as an example of a propositional logic learner and the RIPPER algorithm [Cohen, 1995] as an example of a proposition logic learner that returns a rule set.

The following four questions will be tackled:

Black Box Optimization: How can one optimize the interpretability of a classifier if one does not now how the classifier is working?

White Box Optimization: How can knowledge about the internals of the learning algorithm help to increase understandability?

Local Patterns: What is the best way to describe on which examples not to trust the classifier?

Local Models: Given an understandable classifier, how can one add extra additional classification performance without hurting understandability?

This thesis is structured structured along the dimensions of black-box vs. white-box setting, description of the classifiers decision vs. description of the classifiers errors, and description vs. optimization. At the same time, it is structured by the three connected, sometimes competing goals of understandability, accuracy and efficiency (Figure 1.4). Accuracy is targeted at the connection between the selected and the optimal hypothesis, it is important because it is always possible to generate a trivial, easily understandable hypothesis without any connection to the data. Hence, the interpretability of a model is only meaningful in relation to the degree of its validity. At the same time, it is clear that interpretability can only be one goal among several others. This raises the question how much time the user is willing to invest in order to make the learner's output more understandable. The task of increasing interpretability targets at a higher interaction between learning algorithm and user, and hence it is mandatory that interpretability optimization techniques are efficient enough to allow the user to flexibly inspect and optimize the model.

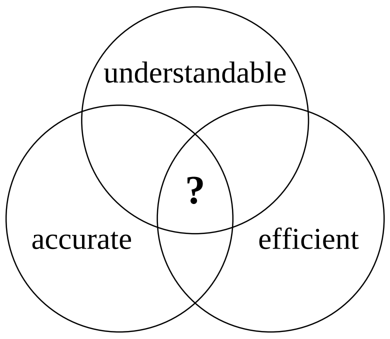


Figure 1.4: Three goals of interpretability

In particular, the following topics are covered in this thesis: The next chapter introduces the theoretical foundations of the thesis, in particular the paradigms of Statistical Learning Theory [Vapnik, 1998] and the Minimum Description Length principle [Rissanen, 1978]. Both formulations of learning employ a concept of complexity of models and are hence well suited to investigate the problem of interpretability. The chapter will also introduce the important statistical concept of robustness [Huber, 1981, Hampel et al., 1986, Rousseeuw and Leroy, 1987, Barnett and Lewis, 1994] which deals with the effects of small sets of outlying observation in statistical analyses. In particular, in Section 2.4 a novel method that improves probabilistic classifiers by integrating robustness concepts will be presented.

Chapter 3 will deal with optimizing the interpretability of models in the black-box scenario, that is without detailed knowledge of the underlying learning algorithm. In particular, Section 3.1.5 will present a method for large-scale non-linear feature selection, Section 3.2 will present a new general approach to instance selection that extracts information about the geometry of the hypothesis space and Section 3.3 will present a method for generating understandable approximations by decomposing complex non-linear classifiers. Finally, Section 3.4 will empirically investigate the feasibility of improving interpretability by optimizing explicit measures of complexity that are classically used to optimize accuracy.

The white-box setting is investigated in Chapter 4. For the important special case of Support Vector Machines, which are a very popular, complex learning algorithm, the chapter will present three novel approaches to improving interpretability by making explicit use of the structure of the SVM hypothesis space. The first approach, sparse models, consists of describing SVMs by a concise formula and a small set of prototypes (Section 4.1), the second approach investigates the possibility of explaining SVMs by possibly more understandable logical formulas (Section 4.2) and the final approach is a method for the visualization of the structure of the SVM hypothesis space (Section 4.3).

Chapter 5 will deal with local patterns and their use for describing the approximation quality of global models. That is, the local patterns will describe the errors the classifier makes instead of the classifier itself. The chapter will discuss both the theoretic complexity of finding local patterns based on a result of Statistical Learning Theory [Vapnik, 1998] and present an expectation-maximization [Dempster et al., 1977] approach for identifying local patterns. Pattern detection will be formulated both as a supervised and as an unsupervised problem and the different consequences for interpretability and effectiveness will be investigated.

The combined description and optimization of a model using local models will be the topic of Chapter 6. It will not only show how to solve local model induction as an extension of local patterns, but also present an approach based on basis pursuit [Chen et al., 1998] that finds local models without the explicit construction of a local pattern while maintaining the interpretability of the original local model idea. Conclusions will be drawn in Chapter 7.

Chapter 2

Machine Learning and Statistics

This chapter offers an introduction into the field of machine learning and the algorithms used in this thesis. It serves both to lay a foundation for the following chapters and to introduce the notation used in this thesis. More detailed introductions into the field of machine learning and the related areas of statistics and data mining can be found in the excellent books [Mitchell, 1997, Hand et al., 2001, Pyle, 1999] and [Hastie et al., 2001].

Machine learning started out as a field of artificial intelligence concerned with the study and computer modeling of learning processes. Three objectives of machine learning can be identified [Carbonell et al., 1983]: the development of learning systems to improve performance in a predetermined task (engineering approach), the investigation and simulation of human thought (cognitive approach) and the theoretical analysis of learning algorithms independent of an application (learning theory). Today, the engineering approach has become the major focus of machine learning, because of the practical need for methods to extract knowledge from the huge amounts of data collected in computer systems everywhere. This has led to the new field of knowledge discovery in databases and data mining at the intersection of machine learning, statistics and database theory. Formerly, these three fields have been mostly disjoint. Database theory deals with the storage and retrieval of data, not with inference. Machine learning and statistics share the same goal of making inferences from data, but traditionally with different methods. As statistics originated in a time where computers did not exist and every calculation had to be done by hand, statisticians developed very sharp but labor-intensive tools to extract the most information from as small as possible data sets. Machine learning, on the other hand, originated in computer science with the general idea to automate as much human work as possible, resulting in efficient algorithms to be applied to very large data sets, which are sometimes ad-hoc solutions without a more general theory in mind. These different origins can be seen in the different languages used by statisticians and computer scientist; for example what is usually called an attribute in computer science is called a variable in statistics, or what statisticians call a model will be called a hypothesis or hypothesis language by machine learners, depending on the context. However, driven by the demand for efficient and effective solutions for real-world problems, these three fields have begun to converge and overlap in large parts.

Two main types of machine learning tasks can be distinguished, supervised and unsupervised learning. In unsupervised learning, the learner is given a set of observations expressed in a formal language and its goal is to extract structure from the data and discover new dependencies. The learner returns one or many hypotheses, expressed in a formal language, that optimize a criterion of data fit or interestingness. In supervised learning, the learner is given a label in addition to each observation (observations together with labels are called examples) and the task is to find a structure that describes the dependency of the label on the observation. Supervised learning tasks are either descriptive, meaning that the goal is to summarize the given data, or predictive, meaning that the label of new observations should be predicted as good as possible.

The rest of this chapter is structured as follows: first, the task of supervised learning will be defined, including the important learning paradigms of Statistical Learning Theory and the Minimum Description Length Principle. This thesis will be mainly concerned with supervised learning. Nevertheless, there will be a short introduction to unsupervised learning in Section 2.2. Very relevant to the idea of local models is also the statistical concept of robust statistics, see Section 2.3. Finally, a more specific task, namely the probabilistic scaling of classifiers will be discussed in Section 2.4. In particular, this section will present a novel algorithm that introduces the concept of robustness into probabilistic scaling.

2.1 Supervised Learning

One formulation of supervised learning is learning as the approximation of a function.

Definition 2.1.1 (Machine Learning as Function Approximation)**.** Given a finite set of examples $(x_i, y_i)_{i=1\ldots n} \subset X \times Y$, the task of finding a function $f : X \to Y$ that predicts the labels y as closely as possible (in some sense that needs to be made precise), is called machine learning as function approximation. $\square$

The idea behind this definition is that the observed data is assumed to be generated by an unknown function t and learning consists of inducing an approximation f of the true function t from the data. Alternatively, in order to account for the fact that in practice some values of x occur more often than others and errors in measuring observations x and labels y frequently occur, one can assume that the data is generated according to a probability distribution $P(X, Y)$ and try to find a function f that approximates the most probable label $\hat{y} = \arg\max_y P(x, y)$ for the observation x.

The learning problem as formulated here is ill-posed in two ways: first, it is not clear what a "close as possible" prediction is, and second it is computationally intractable as the learner would have to search over all possible functions $f : X \to Y$. The latter problem is handled by restricting the set of functions the learner can choose from. It is usually assumed that the set of available functions $\mathcal{F}$ is indexed by a parameter $\theta \in \Lambda$ from a given set Λ. For example, for $X = \Re^d$ and $Y = \Re$ the set of linear functions $\mathcal{F} = \{f | f(x) = \theta_1 x^{(i)} + \ldots + \theta_d x^{(d)} + \theta_0\}$

is indexed by the parameter $\theta = (\theta_0, \dots, \theta_d)$. In general, the parameter θ does not need to be a real vector; more complex parameters like trees, which define a piecewise constant function, are often used.

Notation: The set X is usually called the input space, the set $X \times Y$ is called the example space or example language. The set $\mathcal{F} = \{f_\theta | \theta \in \Lambda\}$ of admissible functions is usually identified with the set of parameters Λ, both are usually called the model space, hypothesis space or hypothesis language. A single parameter θ or function f_θ is called a hypothesis. In particular, the hypothesis returned by a learner on a data set $(x_i, y_i)_{i=1\dots n}$ is called a model of this data[1].

In particular, the following learning tasks have been intensively studied:

Definition 2.1.2 (Learning Tasks)**.** Depending on the set Y, the following learning tasks are defined:

Binary classification: The task of predicting a label from a set of size 2 is called binary classification.

Multiclass classification: The task of predicting a label from a finite set Y of size greater than 2 is called multiclass classification.

Regression: The case of $Y = \Re$ is called regression. □

The term "classification" is used in two different ways. It either refers to the case binary and multiclass classification together, describing the case of predicting a label from a finite set. However, as binary classification is by far the most popular machine learning task, "classification" is often used as a synonym for "binary classification". We will adopt this formulation, as this thesis is mainly concerned with the problem of binary classification.

In binary classification, for the sake of convenience it will be assumed that the set $Y = \{-1, 1\}$ is used. This will be convenient for numerical classifiers, that is functions of the form $g(x) = \text{sign}(f(x))$ with $f : X \to \Re$, because then $g(x) = y$ holds iff $yf(x) > 0$. The function f is called the decision function of the classifier.

2.1.1 Statistical Learning Theory

To make precise what a "good" prediction is, the framework of Statistical Learning Theory [Vapnik, 1998] makes use of a statistical formulation. First, a loss function is defined to formalize the error incurred by predicting the label of an observation x as $f(x)$ instead of y.

Definition 2.1.3 (Loss Function)**.** A loss function L is a nonnegative function $L : X \times Y \times Y \to \Re_{\geq 0}$. Given a loss function L, the loss of a hypothesis f at an example (x, y) is $L(x, f(x), y)$. □

Often, loss functions are used that are independent of x, resulting in a loss function $L : Y \times Y \to \Re_{\geq 0}$. Popular loss functions are:

[1]To make things more complex, in some areas the hypothesis space and not a single hypothesis is called a model.

L_p-loss: The L_p-loss is defined by $L_p(f(x), y) = |f(x) - y|^p$ for $p > 0$. The L_2 loss is also called the squared loss, the L_1 loss is called the absolute loss.

0-1-loss: This loss is defined by $L_{01}(y, y) = 0$ and $L_{01}(f(x), y) = 1$ iff $f(x) \neq y$.

Hinge loss: The hinge loss is defined by $L_{hinge}(f(x), y) = \max\{0, 1 - yf(x)\}$. This loss is only used for binary classification and numerical classifiers with decision functions f, as in this case a hinge loss less than 1 will result in correct prediction of the classifier $\text{sign} f(x)$.

Exponential loss: This loss is defined by $L_{exp}(f(x), y) = \exp(-yf(x))$. Like the hinge loss, this loss is only used for binary classification and numerical classifiers with decision functions f. This loss is derived from logistic regression.

Cross-entropy loss: The cross-entropy loss is defined only for binary classification tasks and for functions f which try to estimate the probability $P(Y = 1|x)$. It is defined by $L_{cre}(f(x), y) = \frac{y+1}{2} \log(1 - f(x)) + \frac{1-y}{2} \log f(x)$. This loss is derived from Maximum Likelihood probability estimation. Notice that if $y = 1$ the loss reduces to $L_{cre}(f(x), 1) = \log(1 - f(x))$ and hence in this case the probability estimate $f(x)$ should be maximized. Correspondingly, $L_{cre}(f(x), -1) = \log f(x)$, resulting in a minimization of the probability estimate for negative examples.

Cost matrix: Let $Y = \{y_1, \ldots, y_k\}$ and $C \in \Re^{k \times k}$ be a nonnegative matrix. This matrix defines a loss function $L_C(y_i, y_j) = C_{ij}$ for $f(x) = y_i$.

The loss is defined for a specific examples (x, y). In order to describe the quality of a function f, a measure of error over the whole example space $X \times Y$ is needed. To account for the probabilistic nature of the data, the examples are assumed to be generated according to a probability distribution $P(X, Y)$.

Definition 2.1.4 (Risk)**.** Given a loss function L, a hypothesis f and a probability distribution $P(X, Y)$, the risk of f is the expected value of the loss of f under P:

$$\mathcal{R}(f) = \int L(x, f(x), y) dP(x, y).$$

□

The risk is also called the expected risk to distinguish it from other forms of risk defined later. Putting these definitions together, we can formally define the statistical learning problem:

Definition 2.1.5 (Statistical Learning Problem [Vapnik, 1998])**.** Given examples $(x_i, y_i)_{i=1\ldots n}$, independently identically drawn from an unknown probability distribution $P(X, Y)$, a set of functions $\mathcal{F} = \{f_\theta | \theta \in \Lambda\}$ and a loss function L, the statistical learning problem consists of finding a function f_{θ^*}, $\theta^* \in \Lambda$, which minimizes the risk $\mathcal{R}$:

$$\theta^* = \arg\min_{\theta \in \Lambda} \mathcal{R}(f_\theta).$$

□

This definition contains the assumption that every example (x, y) is drawn independent from the other examples and all examples are drawn from the same distribution $P(X, Y)$. In practice, this assumption may fail. Failure of the assumption of independence is, for example, found in the field of time series analysis [Schlittgen and Streitberg, 2001, Chatfield, 1984], where each observation depends on the observations recorded earlier. In the extreme case this means that instead of n observations of a function, only one observation of a time series is available, which makes the induction of an appropriate hypothesis much harder. However, in many cases such a problem can be reformulated in a way that removes or minimizes the dependencies [Morik, 2000, Rüping and Morik, 2003, Rüping, 2001b] or it can be shown that the learner tolerates a certain amount of dependency [Fender, 2003]. Failure of the assumption of identically distributed examples can be found in the case of outlier-contaminated data, where a certain amount of observations is assumed to come from a different distribution, or concept drift, where the distribution is assumed to change over time [Klinkenberg and Rüping, 2003]. In both cases, a careful analysis of learning tasks is necessary – what is to be predicted and what not – and the construction of algorithms which identify or tolerate different examples is necessary.

Although the statistical learning problem is a formally well-defined problem, in practice it cannot be solved directly. The problem is that the probability distribution $P(X, Y)$ that is used to define the risk is not known, but only the examples distributed according to P are available. Several ways to solve this problem have been proposed. If it is known that the probability distribution comes from an appropriate parametric family P_ν of distributions, for example the Gaussian distributions $\mathcal{N}(\mu, \Sigma)$, the examples may be used to estimate the parameters of the distribution, and learning methods can be used that are provably optimal for this kind of distribution. This is the approach used in classical statistic modeling. However, in general this problem is stochastically ill-posed ([Vapnik, 1998], Ch. 7.1). The problem is that the distribution $P(X, Y)$ contains much more information than is needed to obtain an optimal classifier and hence is much more difficult to estimate.

The practical alternative is to replace the risk $\mathcal{R}$ in the definition of the learning problem with an approximation that is more easy to compute from data. The easiest way is to estimate the risk on the training data alone:

Definition 2.1.6 (Empirical Risk Minimization)**.** Given a set of examples $(x_i, y_i)_{i=1\ldots n}$, i.i.d. from an unknown probability distribution $P(X, Y)$, a set of functions $\mathcal{F} = \{f_\theta | \theta \in \Lambda\}$ and a loss function L, the statistical learning problem consists of finding a function f_{θ^*}, $\theta^* \in \Lambda$, which minimizes the empirical risk $\mathcal{R}_{emp}$ defined by

$$\mathcal{R}_{emp}(f) = \frac{1}{n} \sum_{i=1}^{n} L(x_i, f(x_i), y_i).$$

□

The empirical risk of a classifier is also called the training error. Notice that the empirical risk minimization principle is a special case of the risk minimization principle if one replaces the distribution P in the definition of the statistical learning problem by the empirical probability distribution from the sample $(x_i, y_i)_{i=1\ldots n}$ (the empirical distribution function of a set of n points gives probability $1/n$ to each of this points and probability 0 to every other point).

The drawback of empirical risk minimization is that it is possible that the induced hypothesis adapts too much to random effects in the training data and not enough to the overall structure defined from the underlying probability distribution, such that it predicts new examples considerably less well. In the extreme case, a learner which simply consists of a list of all training examples and predicts all unknown examples randomly can achieve zero empirical error without the hope of correctly predicting new examples at all. This effect is called "overfitting the training data". See Figure 2.1 for an example of an overfitted model (red) and a possibly more sensible model (blue) of the data.

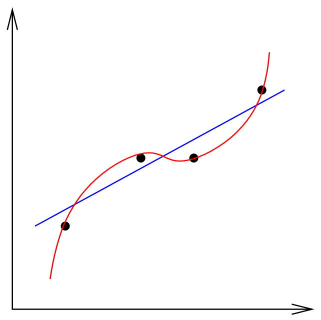


Figure 2.1: The Problem of Overfitting the Data

The possibility of overfitting can be reduced by not using the empirical loss, but an estimation of the expected loss. This is done by splitting the data into a training set and a test, inducing a hypothesis on the training set and choosing the hypothesis with minimal error on the test set. This reduces the chance of overfitting the data, but unfortunately can not prevent it completely.

Behind this problem lies the question of consistency. In principle, when we replace the risk by an empirical approximation we would like to have the property that given enough data we can approximate the actual minimal risk and the function that minimizes it arbitrarily well. For a fixed hypothesis f, it follows from the Law of Large Numbers that the empirical risk of this hypothesis converges against its expected risk as the number of examples goes to infinity. However, given an infinite set of functions it is generally not the case that the sequence of functions f_n which minimize the empirical risk on the examples $x_1, \ldots, x_n$ converges against the function f with minimal expected risk (more precisely, the sequence of risks $\mathcal{R}(f_n)$ needs to converge against the minimal risk $\mathcal{R}(f)$, as the optimal function need not be unique). As an example, consider the trivial learner which keeps a list of all training examples and predicts all unknown examples randomly. The training error of every f_n is zero, but the minimal risk certainly is not.

Definition 2.1.7 (Consistency)**.** Let $\mathcal{L}$ be a learning method which maps a training set to hypotheses $f_\theta \in \Theta$. For a sequence of examples $(x_1, y_1), (x_2, y_2), \ldots$ let $f_n = \mathcal{L}((x_1, y_1), \ldots, (x_n, y_n))$ be the hypothesis induced from the first n examples. The learning method is called consistent, if for any sequence of examples $(x_1, y_1), (x_2, y_2), \ldots$ both the sequence of empirical risk $\mathcal{R}_{emp}(f_i)$ and the sequence of expected risk $\mathcal{R}(f_i)$ converges to the actual minimal risk $\mathcal{R}(f)$.

Actually, this is a simplified definition of consistency. The more formal definition of nontrivial consistency can be found in [Vapnik, 1998] and both contains the more specific concept of convergence in probability and restricts the set of admissible functions to prevent trivial cases of consistency. However, these modifications are of a more technical nature and hence are omitted here.

In general, the empirical risk minimization principle is not consistent if the set of functions is not restricted in some way. On the other hand, if the set of functions is restricted, it is possible that the minimal obtainable risk over this set of functions is significantly higher than the minimal risk over the set of all functions f (this risk is called the Bayes risk and for usual loss functions it is attained by the function $f(x) = \arg\max_y P(y|x)$).

More sophisticated frameworks for choosing a good hypothesis and an adequate hypothesis space take the expressibility of the hypothesis space into account, that is the power to fit different sets of examples. These frameworks are Structural Risk Minimization (Section 2.1.2) and the Minimum Description Length Principle (Section 2.1.4).

2.1.2 Structural Risk Minimization

This section gives a simplified overview over the core theorems and proofs of Statistical Learning Theory, sacrificing mathematical rigorousity for clarity of presentation of the core concepts. The tedious details can be found in [Vapnik, 1998], Chapters 3-5.

The following three theorems by Vapnik and Chervonenkis [Vapnik, 1998, Wapnik and Tscherwonenkis, 1979] are the heart of statistical learning theory, on top of which the structural risk minimization principle is build.

Theorem 2.1.1 (Consistency of Empirical Risk Minimization)**.** *A necessary and sufficient criterion for nontrivial consistency of empirical risk minimization over a set of functions $\mathcal{F}$ is the one-sided uniform convergence in probability of the empirical risk to the expected risk, i.e.:*

$$\lim_{n\to\infty} P\left(\sup_{f\in\mathcal{F}}(\mathcal{R}(f) - \mathcal{R}^{(n)}_{emp}(f)) > \epsilon\right) = 0$$

for all $\epsilon > 0$, where $\mathcal{R}^{(n)}_{emp}(f)$ is the empirical risk on a sample of n examples.

In order to connect the one-sided uniform convergence to the structure of the space of functions $\mathcal{F}$, several steps have to be taken. The first step is the so-called Basic Lemma which states that the probability of the supremum of the difference of the true and the empirical risk in Theorem 2.1.1 can be bounded by a similar expression that depends on the supremum of the difference of the empirical risks on two independent samples of size l (that is, in total one operates on a new sample of size $2l$).

The next step is to remove the supremum. In the case of the 0-1-loss function one can note that although the supremum is taken over infinitely many functions f, it is only possible to distinguish a fixed number of functions on a finite sample of size $2n$. This number corresponds to the number $N^\Lambda((x_1, y_1), \ldots, (x_{2n}, y_{2n}))$

of distinct values the vector $(L(f(x_1), y_1), \dots, L(f(x_{2n}), y_{2n}))$ over all $f = f_\theta, \theta \in \Lambda$. For real-valued loss functions a similar restriction to a finite set of functions is possible if one allows an error of ϵ.

One can now define the random entropy

$$H^\Lambda(Z_{2n}) := \ln N^\Lambda(Z_{2n})$$

with $Z_{2n} := ((x_1, y_1), \dots, (x_{2n}, y_{2n}))$, the entropy

$$H^\Lambda(2n) := EH^\Lambda(Z_{2n}),$$

and the annealed entropy

$$H^\Lambda_{ann}(2n) := \ln EN^\Lambda(Z_{2n})$$

and show that the relation

$$H^\Lambda(2n) \leq H^\Lambda_{ann}(2n)$$

holds. In total, this leads to the annealed entropy bound:

Theorem 2.1.2 (Annealed Entropy Bound)**.** *For a set of functions $\mathcal{F}$ the following bounds holds:*

$$P\left(\sup_{f\in\mathcal{F}}(\mathcal{R}(f) - \mathcal{R}_{emp}(f)) > \epsilon\right) \leq 4\exp\left(H^\Lambda_{ann}(2n) - \frac{n\epsilon^2}{8}\right)$$

Hence, when the annealed entropy does grow sub-linearly in n, the right-hand side of the inequality goes to zero and it is possible to achieve consistency. Using this bound we can further make statements about the expected risk of a hypothesis. By setting the right-hand side of the annealed entropy bound to δ and solving for ϵ one obtains the risk bound.

Theorem 2.1.3 (Risk Bound)**.** *For a set of functions $\mathcal{F}$, with probability $1 - \delta$ the following bounds holds:*

$$\mathcal{R}(f) \leq \mathcal{R}_{emp}(f) + \sqrt{\frac{8}{n}\left(\ln E(N(\mathcal{F}, Z_{2n})) + \ln\frac{4}{\delta}\right)}$$

In other words, the test error can be bounded by the training error plus a confidence interval that depends on the set of functions used.

Finally, it is possible to show that for large enough n the annealed entropy can be bounded by

$$H^\Lambda_{ann}(n) \leq h\left(\ln\frac{n}{h} + 1\right)$$

where h is the Vapnik-Chervonenkis dimension of the set of functions $\mathcal{F}$, if this dimension is finite. The Vapnik-Chervonenkis dimension is a kind of worst-case bound and is defined as follows.

Definition 2.1.8 (Vapnik-Chervonenkis Dimension [Wapnik and Tscherwonenkis, 1979])**.** A finite set of points X is said to be shattered by the set of functions $\mathcal{F}$, if for every subset S of X, there exists a function $f \in \mathcal{F}$ that classifies all points in S as positive and all points in $X\backslash S$ as negative.

The Vapnik-Chervonenkis dimension $VCdim(\mathcal{F})$ of $\mathcal{F}$ is the maximal size of a set X that can be shattered by $\mathcal{F}$. If no such set exists, the Vapnik-Chervonenkis dimension is infinite. □

For example, the Vapnik-Chervonenkis-dimension of linear functions in $\mathbb{R}^d$ is $d+1$. As a proof, one can see that every set of affine independent points in $\mathbb{R}^d$ can be shattered as shown in case (a) of Figure 2.2, while every set X of $d+1$ points either contains two opposing edges of the convex hull of X (case (b)) or one point in the interior of the convex hull of the other points that cannot be separated from the rest (case (c)).

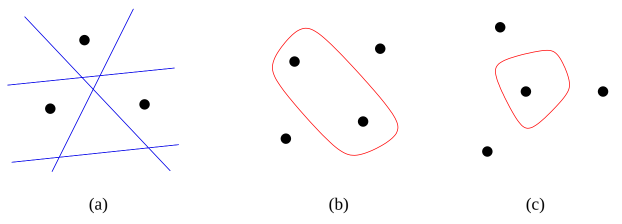


Figure 2.2: Vapnik-Chervonenkis dimension of the linear functions in $\mathbb{R}^d$.

The risk bound of Theorem 2.1.3 and the Vapnik-Chervonenkis dimension provide the ground for a new induction principle that guarantees consistency. The idea is to minimize the risk bound instead of the empirical risk alone and use sets of functions with finite Vapnik-Chervonenkis dimension to guarantee that the confidence interval for the risk is finite.

Definition 2.1.9 (Structural Risk Minimization). Let $\mathcal{F}$ be a set of functions and suppose there exists a structure on $\mathcal{F}$ given by sets of functions $\mathcal{F}_i$ such that

1. $\mathcal{F} = \bigcup_i \mathcal{F}_i$
2. $\mathcal{F}_1 \subseteq \mathcal{F}_2 \subseteq \ldots$
3. $VCdim(\mathcal{F}_1) \leq VCdim(\mathcal{F}_2) \leq \ldots < \infty$

For a set of examples $(x_i, y_i)_{i=1\ldots n}$, the structural risk minimization principle consists of choosing the set $\mathcal{F}_i$ that minimizes the risk bound of Theorem 2.1.3 and then choosing the function $f \in \mathcal{F}_i$ that minimizes the empirical risk. □

Figure 2.3 visualizes the structural risk minimization principle. Notice that although the VC dimension of every set $\mathcal{F}_i$ is finite, the VC dimension of $\mathcal{F}$ can be infinite. A classifier that implements the structural risk minimization principle, the Support Vector Machine, will be presented in Section 2.1.3.

2.1.3 Support Vector Machines

A classifier that estimates a numerical function $f : X \to \mathbb{R}$ and makes predictions of the form

$$y = sign(f(x))$$

is called a numerical classifier. Well-known examples of numerical classifiers are Neural Nets [Bishop, 1995], statistical procedures like Linear Discriminant Analysis [Fisher, 1936] and Support Vector Machines (SVMs, see [Vapnik, 1998,

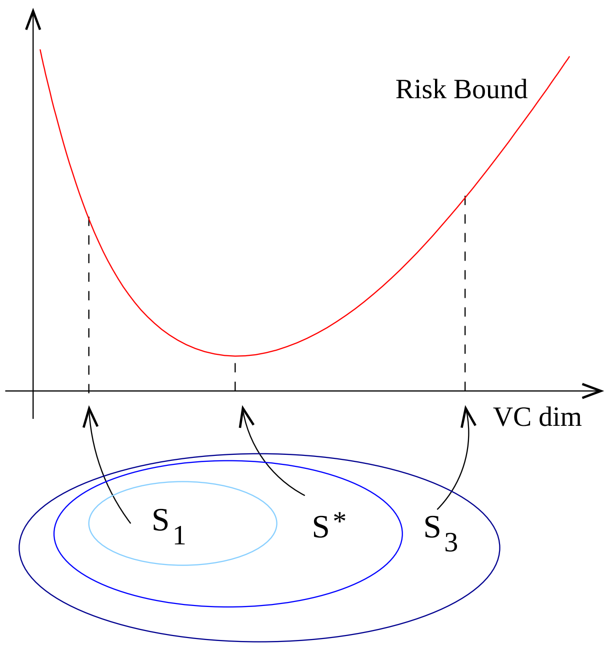


Figure 2.3: The structural risk minimization principle.

Cortes and Vapnik, 1995, Schölkopf and Smola, 2002]). Support Vector Machines are a popular learning method that is based on the structural risk minimization principle. Let us first introduce the linear version of Support Vector Machines.

Let $X = (x_i, y_i)_{i=1\ldots n}$ be a set of examples in $\mathbb{R}^d \times \{-1, 1\}$. X is said to be linearly separable if there exists a linear hyperplane $H = \{x | w * x + b = 0\}$ that separates the positive from the negative points. Equivalently, there exists a linear function $f(x) = w * x + b$ such that for all i the inequality $y_i f(x_i) > 0$ holds. As we are only interested in the classification function $\text{sign}(f(x))$, any function $\alpha f(x)$ with $\alpha > 0$ will be equivalent and hence we can normalize f such that $y_i f(x_i) \geq 1$ and there exists at least one x_i such that $y_i f(x_i) = 1$. Given a linearly separable set of points there still will be more than one function that separates the points, so which one should one use? Assuming the observations x_i are measured with some error, it seems favorable that points lying very close to a training point x_i should be predicted to lie in the same class as x_i. This is equivalent to require the separating hyperplane to be as far away from the training points as possible.

Definition 2.1.10 (Optimal Separating Hyperplane)**.** Given a set of points $(x_i, y_i)_{i=1\ldots n}$, the optimal separating hyperplane is the hyperplane that separates the points and maximizes $\min_i d(x_i, H)$.

Theorem 2.1.4 (Existence and Uniqueness of the Optimal Separating Hyperplane)**.** *Given a linearly separable set of examples $(x_i, y_i)_{i=1\ldots n}$, the optimal separating hyperplane exists and is uniquely determined. It can be found by* *minimizing* $||w||^2$ *subject to* $y_i(w * x_i + b) \geq 1$ *for all* $i = 1 \ldots n$ *and setting*

$$H = \{x | w * x + b = 0\} \tag{2.1}$$

Its distance to a point x *is then given by* $d(x, H) = |w * x + b|/||w||$.

Proof. As the set of points is separable, the set of feasible points (w, b) for the minimization problem is not empty. The target function $||w||^2$ is strictly convex, from which follows that the solution of the optimization problem exists and is uniquely determined. Hence it suffices to show that the solution of the optimization problem is identical to the optimal separating hyperplane. A basic fact from linear algebra states that the distance of a point x to a hyperplane given by (w, b) is $|w*x+b|/||w||$. As the term $|w*x+b|$ can be fixed to be greater or equal to 1 without changing the hyperplane, the distance is maximized by minimizing $||w||$, which is equivalent to minimizing $||w||^2$. □

The connection between optimal separating hyperplanes and structural risk minimization is given by the following theorem:

Theorem 2.1.5 (VC Dimension of the Optimal Separating Hyperplane [Vapnik, 1998])**.** *Let* $X = (x_i)_{i=1 \ldots n}$ *be a set of points in* $\mathbb{R}^d$ *and* R *be the radius of the smallest ball containing all points. Consider the set of all function* $f(x) = w * x + b$ *such that* $min_i |f(x_i)| = 1$ *and* $||w|| \leq D$. *The VC dimension of this set of functions is bounded by*

$$VCdim \leq \min\{R^2 D^2, d\} + 1$$

In practice, due to noise a given set of data might not be linearly separable, even if the underlying probability distribution suggests a linear classifier. The solution most coherent with the idea of the VC dimension would be to remove a minimal set of points such that the data becomes separable. However, this would lead to a combinatorial optimization problem which would not be efficiently solvable. Instead, in [Cortes and Vapnik, 1995] the use of slack variables to allow for errors is suggested.

Definition 2.1.11 (Soft Margin Hyperplane)**.** Let $X = (x_i, y_i)_{i=1 \ldots n}$ be a set of examples and $C > 0$ be a constant. The soft margin hyperplane is defined as the unique solution to the following optimization problem: Minimize

$$||w||^2 + C \sum_{i=1}^{n} \xi_i$$

subject to

$$y_i(w * x_i + b) \geq 1 - \xi_i \text{ for all } i = 1, \ldots, n$$

□

Notice that ξ_i is identical to the hinge loss of $L_{hinge}(f(x_i), y_i)$. The soft margin hyperplane classifier is also called the linear Support Vector Machine classifier.

Definition 2.1.12 (Regularized Risk)**.** The weighted sum of the empirical risk $R_{emp}(f)$ and a complexity term $comp(f)$ is called the regularized risk

$$R_{reg}(f) = R_{emp}(f) + \lambda comp(f)$$

The Soft Margin Hyperplane is equivalent to minimizing the regularized risk for $f(x) = w * x + b$ with $comp(f) = ||w||^2$, empirical error measured by the hinge loss and $\lambda = 1/C$.

It turns out that the soft margin hyperplane / support vector machine problem can be better solved in its dual form.

Theorem 2.1.6 (Dual Form of the Support Vector Machine)**.** *The solution of the soft margin hyperplane problem is given by*

$$w = \sum_{i=1}^{n} \alpha_i x_i$$

where the α_i are the solution of the optimization problem

$$\begin{aligned} W(\alpha) = \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i * x_j - \sum_{i=1}^{n} \alpha_i \quad &\rightarrow \quad \min \\ \textit{subject to} \qquad & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & \forall_{i=1}^{n} 0 \leq \alpha_i \leq C \end{aligned}$$

The threshold b can be computed as $b = y_i - w * x_i$ for every i with $0 < \alpha_i < C$.

The proof can be found in [Vapnik, 1998, Schölkopf and Smola, 2002]. This problem is a quadratic optimization problem and can be efficiently solved by standard optimizers or specialized SVM algorithms [Joachims, 1999]. The points x_i with nonzero Lagrangian multiplier α_i are called support vectors, as they support the hyperplane w. It can be shown that the same hyperplane is found when all non-support vectors are removed from the training set.

With the complexity control by the optimal separating hyperplane and the soft margin idea we have already defined two of the crucial ingredients of Support Vector Machines. The final part is the kernel trick. Suppose we want to convert the SVM into a nonlinear classifier. The easiest approach would be to use some function $\Phi : X \rightarrow \mathcal{X}$ to map the training points x from the input space X into a feature space $\mathcal{X}$ and run the SVM algorithm in the feature space. Looking at the dual SVM problem it is obvious that the SVM depends on the observations x only via the inner product $x * x'$, in particular because the function term $w * x + b$ can be written as $\sum_{i=1}^{n} \alpha_i x_i * x + b$. Hence, it suffices to know the value of the inner product $\Phi(x) * \Phi(x')$ in feature space to run the SVM algorithm. The function $K(x, x') = \Phi(x) * \Phi(x')$ is called a kernel function and there are cases where the kernel function can be evaluated much easier than by explicitly mapping the data into the feature space and calculating the inner product there. Kernels are more intensively discussed in the next section.

Definition 2.1.13 (Support Vector Machine [Cortes and Vapnik, 1995])**.** Given examples $(x_i, y_i)_{i=1,\ldots,n}$, a parameter $C > 0$ and a kernel function K, the Support Vector Machine classifier is defined by

$$f(x) = \operatorname{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b \right)$$

where the α_i are the solution of the optimization problem

$$W(\alpha) = \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) - \sum_{i=1}^{n} \alpha_i \quad \to \quad \min$$

$$\text{subject to} \qquad \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\forall_{i=1}^{n} 0 \leq \alpha_i \leq C$$

and the threshold b is defined as $b = y_i - \sum_{j=1}^{n} \alpha_j y_j K(x_j, x_i)$ for any i with $0 < \alpha_i < C$.

General Loss Functions for Support Vector Machines

By definition of the constraints $y_i f(x_i) \geq 1 - \xi_i$ in the soft margin hyperplane, the SVM implements the hinge loss with an additional factor of C. There exists some generalizations

Asymmetric Loss: By using different constants C_+ and C_- for the positive and negative examples, it is possible to use an asymmetric loss function [Rüping, 1999, Morik et al., 1999].

Example-dependent Loss: It is also possible to introduce a different constant C_i for each example (x_i, y_i). This is important when different examples have different importance, for example in incremental learning [Rüping, 2001a] or when dealing with temporal dependencies in concept drift [Klinkenberg and Rüping, 2003].

Squared Loss: Instead of the the standard target function $||w||^2 + C \sum \xi_i$ one can also use the target function $||w||^2 + C \sum \xi_i^2$, implementing a squared loss function. In the dual form this translates to setting the upper bound on α to infinity.

General loss functions: It is also possible to define the primal SVM problem with an arbitrary loss function L. The problem is that in general the SVM problem will then no longer be a quadratic programming problem, but a general optimization problem. This problem will be much harder to solve and the uniqueness of the solution is not clear.

One situation where the problem is still efficiently solvable with an unique solution is the case of the exponential loss, which leads to Kernel Logistic Regression [Wahba, 1999, Keerthi et al., 2002].

General complexity functions: It is also possible to define the primal SVM problem with a general complexity function instead of $||w||^2$. However, care must be taken here because the quadratic term ensures the connection of the SVM to the VC-dimension (see Theorem 2.1.5). Of course, other complexity functions may also make sense, but this remains to be shown in each special case.

One example of a different complexity function is the L1 norm $||w||_1 = \sum |w_i|$, which has the desirable statistical property that there is a higher penalty on values of w_i close to zero than with the quadratic norm. This generally results in sparser vectors w. Also, the implementation of a SVM with the L1 norm reduces to a linear programming problem instead of the more complex quadratic problem [Mangasarian, 2000].

Kernel Machines

In this section, we take a closer look on kernels and the kernel trick employed in Support Vector Machines.

Definition 2.1.14 (Kernel Function)**.** A kernel function $K : X \times X \to \Re$ is a function such that for every finite set of vectors $(x_i)_{i=1,\dots,n}$ the matrix $K_{mat} = (K(x_i, x_j))_{i,j=1,\dots,n}$ is positive definite.

Here a matrix M is called positive if for every $\alpha \in \mathbb{R}^d$ it holds that $\alpha^T M_{mat} \alpha \geq 0$. This condition is usually called positive semi-definiteness, with the term "positive definite" reserved for the case $\alpha^T M_{mat} \alpha > 0$ if $\alpha \neq 0$. However, in kernel theory only the case ≥ 0 is considered and hence the prefix "semi" is usually dropped.

The Mercer Theorem shows that it is not necessary to explicitely construct the feature map Φ in order to define a kernel function.

Theorem 2.1.7 (Mercer Theorem [Mercer, 1909])**.** *Suppose $K \in L_\infty(X^2)$ is a symmetric real-valued function such that for all $f \in L_2(X)$ the inequality*

$$\int_{X \times X} K(x, x') f(x) f(x') dx dx' \geq 0$$

holds. Then K is a kernel function.

The most popular kernel functions are the

Linear Kernel: $K(x, x') = x * x'$

Polynomial Kernel: $K(x', x') = (x * x')^p$ for some integer $p > 0$

Radial Basis Function (RBF) Kernel: $K(x, x') = \exp(-\gamma ||x - x'||^2)$ for some $\gamma > 0$

Sigmoid Kernel: $K(x, x') = \tanh(ax * x' + b)$. This function is only a kernel for specific choices of $a, b \in \mathbb{R}$.

In particular, for every map $\Phi : X \to \mathcal{X}$ into a linear space $\mathcal{X}$ with inner product $< \cdot, \cdot >$, the function $K(x, x') =< \Phi(x), \Phi(x') >$ is a kernel function, as the corresponding matrix is the Gram matrix of the feature space vectors. This poses the question whether the converse also holds: can we find for every kernel function a linear space $\mathcal{X}$, an inner product in this space, and a map Φ such that $K(x, x') =< \Phi(x), \Phi(x') >$ holds? One can indeed construct such a space, which turns out to be a Reproducing Kernel Hilbert Space.

Definition 2.1.15 (Hilbert Space)**.** A Hilbert space $\mathcal{H}$ is a linear space with an inner product $< \cdot, \cdot >$ such that $\mathcal{H}$ is complete with respect to the norm $||x|| = \sqrt{< x, x >}$.

Definition 2.1.16 (Reproducing Kernel Hilbert Space)**.** A Reproducing Kernel Hilbert Space is a Hilbert space $\mathcal{H}$ of functions $X \to \Re$ such that there exists a kernel K such that all basis function $K(x, \cdot)$ (that is, $x' \mapsto K(x, x')$) are contained in $\mathcal{H}$ and the reproducing property

$$< f, K(x, \cdot) >= f(x)$$

holds for all $f \in \mathcal{H}$ and $x \in X$.

Theorem 2.1.8 (Existence of a RKHS [Aronszajin, 1950])**.** *For every kernel function K there exists a corresponding Reproducing Kernel Hilbert Space $\mathcal{H}_K$.*

Proof. Given a kernel function K we define the space $\mathcal{H}_0$ to be the space of all finite linear combinations

$$f(x) \quad = \quad \sum_i \alpha_i K(x_i, x)$$

of basis functions $K(x_i, \cdot)$. We also define the function $\Phi : X \to \mathcal{H}_0$ which maps x to its basis function $K(x, \cdot)$.

To define an inner product $< \cdot, \cdot >$ on $\mathcal{H}_0$, let

$$< K(a, \cdot), K(b, \cdot) > \quad := \quad K(a, b)$$

for any two basis functions. For every f, g which are finite linear combinations of basis functions we define

$$\begin{aligned} < f, g > \quad &= \quad < \sum \alpha_i K(x_i, \cdot), \sum \beta_j K(x'_j, \cdot) > \\ &:= \quad \sum_{i,j} \alpha_i \beta_j < K(x_i, \cdot), K(x'_j, \cdot) > \\ &= \quad \sum_{i,j} \alpha_i \beta_j K(x_i, x'_j) \end{aligned}$$

It is easy to see that this product is linear in both of its arguments and symmetric. Its positive definiteness was postulated in the definition of a kernel K. It also follows that

$$\begin{aligned} < f, g > \quad &= \quad \sum_{i,j} \alpha_i \beta_j K(x_i, x'_j) \\ &= \quad \sum_i \alpha_i \sum_j \beta_j K(x_i, x'_j) \\ &= \quad \sum_i \alpha_i g(x_i) \\ &= \quad \sum_j \beta_j f(x_j) \end{aligned}$$

which shows that with this definition the inner product does not depend on the particular representation of f and g in terms of a linear combination of basis function, which does not need to be unique. It is obvious that for every $f \in \mathcal{H}_0$ the reproducing property

$$< f, K(x, \cdot) > \quad = \quad f(x)$$

holds. In particular,

$$\begin{aligned} < \Phi(x'), \Phi(x) > &= < K(x', \cdot), K(x, \cdot) > \\ &= K(x', x') \end{aligned}$$

which was the desired property of the space $\mathcal{H}_0$.

The inner product in $\mathcal{H}_0$ also defines a norm

$$||f||_{\mathcal{H}_0} \quad := \quad \sqrt{< f, f >}$$

Together with this norm, $\mathcal{H}_0$ forms a pre-Hilbert space, that is, it is a linear space with an inner product, but it is not necessarily complete. We will now construct a completion $\mathcal{H}_K$ of $\mathcal{H}_0$. Assume that $(f_n)_{n \in \mathbb{N}}$ is a Cauchy sequence in $\mathcal{H}_0$. One can show that for every $x \in X$

$$\begin{aligned} |f_m(x) - f_n(x)| &= | < f_m - f_n, K(x, \cdot) > | \\ &\leq ||f_m - f_n||_{\mathcal{H}_0} K(x, x) \end{aligned}$$

where the inequality is a result of the positive definiteness of the 2×2 Gram matrix of $(f_m - f_n)$ and $K(x, \cdot)$. It follows that for every x the sequence of real numbers $(f_n(x))$ is a Cauchy sequence and hence converges in $\mathbb{R}$. Hence, the sequence of functions (f_n) has a pointwise limit. Adding these pointwise limits to $\mathcal{H}_0$ and extending the inner product appropriately gives a complete space $\mathcal{H}_K$ which can be shown to be the Reproducing Kernel Hilbert Space corresponding to K. $\square$

Similar to the construction of the nonlinear SVM, any algorithm that is defined solely in terms of the inner product of the input vectors can be cast into a nonlinear algorithm using kernel functions. This is the so-called kernel trick. For example, there exist kernelized versions of principal component analysis [Schölkopf et al., 1999] or logistic regression [Wahba, 1999].

2.1.4 Minimum Description Length

This section provides a short introduction to the Minimum Description Length (MDL) Principle [Rissanen, 1978] and is largely based on [Grünwald, 2005].

The main idea of Minimum Description Length is to view learning as data compression. If a learning algorithm has extracted some structure from examples $z_1, \ldots, z_n$, this structure can be used to give a more concise description of the data by encoding only the information of the data that is not specified by the structure. For example, if we know that the examples $z = (x, y)$ are structured such that there exists a function f with $y = f(x)$, we do not need to explicitely encode the values y if we instead encode the function f. In MDL, the data is assumed to come from a countable set, which can always be assured in practice (e. g. by rounding).

Theoretically, the Minimum Description Length Principle could be implemented by using the Kolmogorov complexity [Kolmogorov, 1965], which is defined as the length of the shortest computer program that outputs the sequence $z^n = (z_1, \ldots, z_n)$. However, the Kolmogorov complexity is not computable and is independent of the language used to encode the computer program only for $n \to \infty$ [Li and Vitanyi, 1997]. Hence, in practice one has to settle for an approximation. Most important, the set of admissible models, i. e. the hypothesis language, has to be restricted. Note that in MDL theory a model of a set of data is always a probability distribution that describes this data.

The original version of Minimum Description Length employs a two-part code. The first part of the code describes the model used and the second part of the code describes the data given the model.

Definition 2.1.17 (Crude Minimum Description Length)**.** Let $L(H)$ be the length of the encoded model H in bits and $L(Z|H)$ be the length of the data Z encoded using the model H. The MDL principle consists of choosing the model H that minimizes

$$L(H) + L(Z|H).$$

□

One can immediately see the connection of MDL to regularized risk minimization and Statistical Learning Theory: The term $L(H)$ gives the complexity of the hypothesis (the more complex, the longer its description) and the term $L(Z|H)$ gives the error of model on the data (the more the data adheres to the model H, the less additional information has to be encoded).

The MDL principle does not specify which code to use. For encoding the data this is relatively straight-forward because of a connection between code lengths and probabilities known as Kraft's inequality

Theorem 2.1.9 (Kraft's Inequality)**.** *There exists an uniquely decodable code of code length l_i if and only if*

$$\sum_i 2^{-l_i} \leq 1$$

The proof can for example be found in [Cover, 1991]. This allows to identify code lengths l_i with probabilities $p_i = 2^{-l_i}$. For given probabilities p_i, such a code can be found using Huffman coding, which produces minimal codes when the probabilities are powers of 2. As a consequence, in the following we will replace the code length of the data by the term $-\log P(Z|H)$. Minimum Description Length is usually interested in compressing the complete example z. But for supervised learning we are only interested in predicting the label y, not the complete example $z = (x, y)$. This goal can be addressed by using only the conditional probabilities $P(Y|X, H)$ to describe the data, which is equivalent to assuming that the observations x will be compressed by a default code.

The problem of MDL with two-part codes is the code length for the hypothesis $L(H)$. MDL does not specify which code to use and for hypotheses there is no such obvious choice as for data. But different code length functions lead to different MDL solutions, in particular for small data sets. For each fixed hypothesis H one can always define a code which assigns to H code length 1. This leads to the consequence that the MDL solution becomes arbitrary.

A way out of this dilemma lies in the observation that data and hypotheses are not independent in MDL. The trick is that as soon as we know the MDL-optimal hypothesis H, this also restricts the choices for Z: only those values of Z are possible for which H provides a minimum description length encoding. This lead to the development of refined MDL [Rissanen, 1984, Rissanen, 1996], which uses a one-part code for both model and data, that only depends on the class of models $\mathcal{M}$ instead on the single hypothesis H that describes the data.

The formal idea is to employ a universal code, which is only slightly worse than the best code for the data Z in terms of a minimax regret.

Definition 2.1.18 (Regret)**.** Let Z^n be the space of samples $z^n = (z_1, \dots, z_n)$ from Z of size n. For a class $\mathcal{M}$ of probabilistic models on Z^n and a probability distribution $\bar{P}$ on Z^n, the regret of $\bar{P}$ relative to $\mathcal{M}$ on $z^n \in Z^n$ is

$$\mathcal{R}(\bar{P}, z^n) = -\log \bar{P}(z^n) - \min_{P \in M} \{-\log(P(z^n)\} .$$

The worst-case regret of $\bar{P}$ relative to $\mathcal{M}$ is

$$\mathcal{R}_{max}(\bar{P}) = \max_{z^n \in Z^n} \mathcal{R}(\bar{P}, z^n)$$

□

In other words, the regret is the additional number of bits needed to encode the data z^n with $\bar{P}$ instead of the model from $\mathcal{M}$ that encodes z^n best.

To construct a probability distribution with minimax regret, the following definition is needed:

Definition 2.1.19 (MDL Model Complexity)**.** For a class $\mathcal{M}$ of probabilistic models P, let

$$\widehat{\mathcal{M}}(z^n) = \arg\min_{P \in \mathcal{M}} \{-\log(P(z^n)\} .$$

The complexity $\mathbf{COMP}_n$ of the model class $\mathcal{M}$ is defined as

$$\mathbf{COMP}_n(\mathcal{M}) = \log \sum_{z^n \in Z^n} P(z^n | \widehat{\mathcal{M}}(z^n)).$$

□

That is, the more data sequences z^n can be fit well by a model from $\mathcal{M}$, the higher the complexity of $\mathcal{M}$.

Theorem 2.1.10 (Minimax Regret and the Normalized Maximum Likelihood Distribution [Shtarkov, 1987])**.** *Suppose that* $\mathbf{COMP}_n(\mathcal{M})$ *is finite. Then the minimax regret is uniquely achieved by the normalized maximum likelihood distribution* P_{nml} *given by*

$$P_{nml}(z^n) = \frac{P(z^n | \widehat{\mathcal{M}}(z^n))}{\sum_{z'_n \in Z^n} P(z'^n | \widehat{\mathcal{M}}(z'^n))}.$$

Definition 2.1.20 (Refined Minimum Description Length)**.** Given data z^n and a finite set of model classes $(\mathcal{M}^{(i)})_{i=1 \dots n}$, refined MDL consists of selecting the model class $\mathcal{M}^{(i)}$ that maximized the normalized maximum likelihood. This is equivalent to minimizing

$$-\log P_{nml}(z^n) = -\log P(z^n | \widehat{\mathcal{M}}^{(i)}(z^n)) + \mathbf{COMP}_n(\mathcal{M}^{(i)}).$$

The MDL model for the data z^n is then $\widehat{\mathcal{M}}^{(i)}(z^n)$. □

Minimum Description Length can also be defined for infinite sets of models and model classes with infinite $\mathbf{COMP}_n$. See [Grünwald, 2005] for details.

The refined MDL principle again tells us to select the model class by simultaneously minimizing the complexity of the model class and the error of the optimal model in this class (in terms of a maximal likelihood). But different from crude MDL, model complexity is no longer defined in terms of an arbitrary code length function, but in terms of the model classes capacity to fit different sets of data, similar to Statistical Learning Theory (see Section 2.1.1).

MDL and Interpretability

There is an obvious connection between the crude MDL principle and interpretability: crude MDL states that the code length of the hypothesis should be minimized as far as possible without accepting too much errors. A short code length means that the hypothesis should be easy to communicate and as the code for the hypotheses is not specified in MDL, we are free to use a code whose code length function models the interpretability of the hypotheses from the view of the user. However, we are stuck with the fundamental problem of crude MDL: as there is no general way to define the degree of interpretability of a model, code length functions are arbitrary and the final hypothesis depends on the implicitly encoded assumptions. Hence, the description length in crude MDL does not address the problems of interpretability, but only hides them in the definition of the code length function.

In refined MDL, codes are given by the complexity of the model class and and not in an explicit complexity measure that could be modified to represent interpretability constraints.

Both MDL approaches do not specify which model classes to investigate in the first place, only how to choose the best model and model class from the given alternatives. However, as discussed in the introduction of this thesis, model classes (i. e. hypothesis languages) play an important role for interpretability. A main problem is the existence of human background knowledge. In principle, the description length of the hypothesis would have to be modified to represent the number of bits it needs to explain the hypothesis given the background knowledge of the user (e.g. for one user a short statistical formula would suffice while another may additionally need to be given an introductory statistics book). Again, the background knowledge of a user cannot be formally measured and additional assumptions would have to be taken.

In particular, Minimum Description Length in general does not support to simplify hypotheses in the sense that if we order model classes $\mathcal{M}^{(i)}$ by their complexity, the optimal hypotheses $h_i = \widehat{\mathcal{M}}(z^n)$ for a fixed data set z^n do not need to exhibit a certain structure that allows to view h_{i+1} as a generalization of h_i. For example, it the model classes consist of decision trees with bounded depth, the optimal tree of depth i does not need to be a subtree of the optimal tree of depth $i + 1$. This is an example of MDL not being directly applicable to the setting of global and local models.

In conclusion, Minimum Description Length with its approach of explicitely modeling complexity has strong connections to the problem of interpretability, but it is not directly applicable to this problem.

2.1.5 Learning in Logic

Learning can also be represented in a logic framework, based on the concept of logical implication. Examples and hypotheses are represented as logical formulas and not as tuples as in the numerical formulation learning. Unique to the logical formulation of learning is the explicit definition of background knowledge, a set of formulas that define what is already known about the data. In logical learning, the problem of classification is also known as concept learning.

Definition 2.1.21 (Concept Learning in Inductive Logic Programming [Muggleton, 1992a])**.** Let there be a set of examples E, background knowledge B and a language of hypotheses L_H, such that the background knowledge is consistent with the examples (no contradiction can be derived from E and B). the goal of concept learning in inductive logic programming is to find an hypothesis $H \in L_H$ that is consistent with the examples and the background knowledge, such each positive example and no negative example can be derived by the hypothesis and the background knowledge.

Unique to logical hypotheses is that an example which is not classified as positive is not automatically classified as negative, as it is possible that neither an observation e nor its negation $\neg e$ can be derived from H and B. As a result, logical hypotheses often contain a default rule to predict every observation that is not covered by another rule or make use of the closed-world-assumption that every observation that can not be shown to be positive is negative.

Depending on the logical formalism used, propositional logic and first-order logic learners are distinguished. In the following, we will discuss two propositional logic learners, namely decision trees and covering rule learners. As discussed in the introduction, we do not include first-order logic learner in the comparison, as their very rich hypothesis space is hard to map and compare to the more restricted space of propositional logic and numerical learners. Accordingly, in the rest of this thesis we only mean propositional logic learners when we talk about learning in logic.

Decision Tree Induction

Decision tree learners like C4.5 [Quinlan, 1986] are divide-and-conquer algorithms, they recursively partition the input space in order to minimize the heterogeneity of the labels in each partition. An inner node of the decision tree corresponds to a decision of the form A rel v where A is an attribute, v is a possible value of this attribute and rel $\in \{=, <, \leq\}$. Each inner node thereby partitions a set of examples into two subsets (see Figure 2.4). A leaf of a decision tree is labeled with a class value.

To build a decision tree from a set of examples, the algorithm chooses the decision that increases a measure of the information about the class value in both subsets the most. If there is no decision left to choose from or should no decision increase the information about the class, the decision tree consists of a leaf labeled with the majority class in the examples. Otherwise, the decision tree algorithm is recursively applied to each of the subsets of examples induced by the selected decision. The decision tree then consists of the node corresponding to the selected decision with the results of the recursion as subtrees. Measures of the information of a split include the entropy, information gain or the gain ratio.

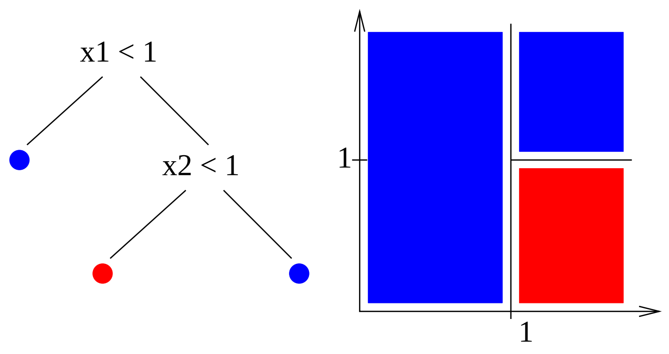


Figure 2.4: Decision Tree and Induced Partition

After the tree is induced, a pruning step is carried out. The tree is traversed depth-first and for each subtree an estimate of the prediction error is calculated. Should the error estimate of the subtree exceed the error estimate of a single leaf corresponding to the same subset of data, the subtree is removed and replaced by this leaf. This step is necessary to avoid overfitting and because the information criterion used to build the tree is only a heuristic for assessing the information content of a set of examples about its labels.

Covering Rule Learning Algorithms

Covering rule learning algorithms are a family of learning algorithms that was first introduced in [Michalski, 1969]. An overview of covering rule learning algorithms can be found in [Fürnkranz, 1999].

Given a set of examples, covering rule algorithms execute a loop in which they first find a logical rule that maximizes a given rule quality criterion and then remove all examples that are covered by this rule from the set of examples. The algorithm stops when all examples are covered or no further rule can be found. At termination, the ordered list of all rules found is returned. Specific instances of covering rule learners differ in the rule quality criterion used and the way candidate rules are generated. Figure 2.5 shows the pseudocode of a simple covering rule learner, see [Fürnkranz, 1999]. Figure 2.6 shows a decision list generated by a covering rule learner and the induced partition of the example space.

One example of rule generation is general-to-specific construction of rules. Construction starts with the most general rule that states that all examples are positive. Iteratively, the rule is specialized by adding a new literal o the rule body until no negative examples are covered by the rules. New literals are selected either based on a full search or on a heuristic as in decision tree induction.

The rule quality criterion has an important influence on the prediction quality of the final classifier. A simple quality criterion is the training accuracy of the rule or the number of positive examples covered. More sophisticated criteria are the precision gain, weighted relative accuracy, correlation and many more [Fürnkranz, 1999]. Rule evaluation metrics can be analyzed via coverage spaces [Fürnkranz and Flach, 2005] which depict which rules obtain identical quality

```
Theory = {}
while Positive(Examples) != {}
  BestRule = {true}
  Rule = BestRule
  while Negative(Cover) != {}
    for Condition in Conditions
      Refinement = Rule + Condition
       if (Quality(Refinement,Examples)
           > Quality(BestRule,Examples))
         BestRule = Refinement
    Rule = BestRule
  Theory = Theory + Rule
  Examples = Examples - Cover
return Theory
```

Figure 2.5: A simple covering rule learner

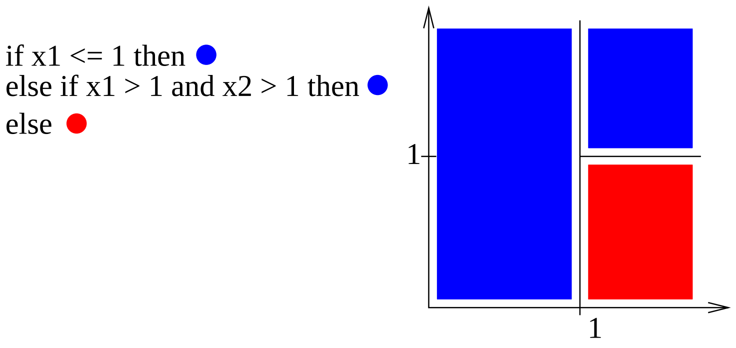


Figure 2.6: Decision List generated by a Covering Rule Learner

values. In this thesis, the covering rule learner Ripper [Cohen, 1995] is used, which differs from the algorithm of Figure 2.5 in details on how to find the best rules and how to post-process rules and theories.

Converting Logical Methods into Numerical Classifiers

In order to compare or combine numerical and propositional logical classifiers these methods must be cast into a single framework. Fortunately, it is easy to interpret logical methods as numerical functions.

In order to apply logical methods to numerical problems, numerical attributes have to be converted into nominal attributes. This task is called discretization and is usually applied as a pre-processing step before running the learning algorithm. Some logical classifiers also have discretization built into the learning step, such that they can be directly applied to numerical tasks.

Given labels $y \in \{-1, 1\}$, a hypothesis returned by the logical classifier then corresponds to the function that first casts the example into nominal values, applies the classifier and returns the predicted y. Some classifiers also produce an estimate of $P(Y = 1|x)$ that can be used as the output instead.

2.2 Unsupervised Learning

The task of unsupervised learning is to extract information about the structure of data $x_1, \ldots, x_n$. As there are possibly many structures that fit the data, the task is not as well defined as supervised learning. Usually, the type of structure to extract is implicitly given in terms of distance and similarity measures. In the context of this thesis we are mainly interested in the tasks of statistical density estimation and clustering.

2.2.1 Density Estimation

A statistical density is a non-negative function $f : X \to \Re_{\geq 0}$ that integrates to 1. A density defines a probability measure by

$$P(x \in A) = \int_A f(x)dx$$

for every measurable $A \subseteq X$. The goal of density estimation is to find a density $\hat{P}$ that approximates the probability P that generated the data, assuming that the x_i are independently identically distributed by P. The typical performance measure for density estimation is the likelihood:

Definition 2.2.1 (Maximum Likelihood Estimate)**.** Given data $x_1, \ldots, x_n$ in X and a set of probability distributions defined by densities $(f_\theta)_{\theta\in\Lambda}$, the maximum likelihood estimate is the density f_θ that maximizes the data likelihood

$$L(x_1, \ldots, x_n; f) = \sum_{i=1}^{n} \log f(x_i)$$

□

Parametric density approximates the data with density functions with a fixed functional form, dependent on a parameter θ, while non-parametric density estimation allows more flexible functions where the form depends more on the data (e. g. a parametric form where the number of parameters depends on the size of the data). For example, Gaussian density estimation uses Gaussian densities with parameter $\theta = (\mu, \Sigma)$:

$$n(\mu, \Sigma)(x) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp(-\frac{1}{2}(x - \mu)^t\Sigma^{-1}(x - \mu)).$$

Gaussian density estimation is particularly easy to compute, because one can show that the maximum likelihood estimate is given by the empirical mean $\hat{\mu}$ and empirical covariance matrix $\hat{\Sigma}$ of the data. A more flexible, but still parametric approach is the Mixture of Gaussians, which assumes that the data is generated by a fixed number k of Gaussian distributions with probabilities $\pi_1, \ldots, \pi_k$, i. e. with the density

$$m = \sum_{i=1}^{k} \pi_i n(\mu_i, \Sigma_i).$$

The parameters can be optimized using the Expectation Maximization (EM) algorithm [Dempster et al., 1977], which iterates between probabilistically assigning each data point to one of the mixture components (E-step) and re-computing the parameters of each component (M-step). See e. g. [Hastie et al., 2001] for details.

The Mixture of Gaussians becomes a non-parametric approach when setting $k = n$. In this case, each data point is taken as a center of one component, the prior probabilities are set to $\pi_i = 1/n$ and the covariance matrices are fixed to a pre-specified value. The general case, in which basis functions other than Gaussian densities are allowed, is known as kernel density estimation [Scott, 1992].

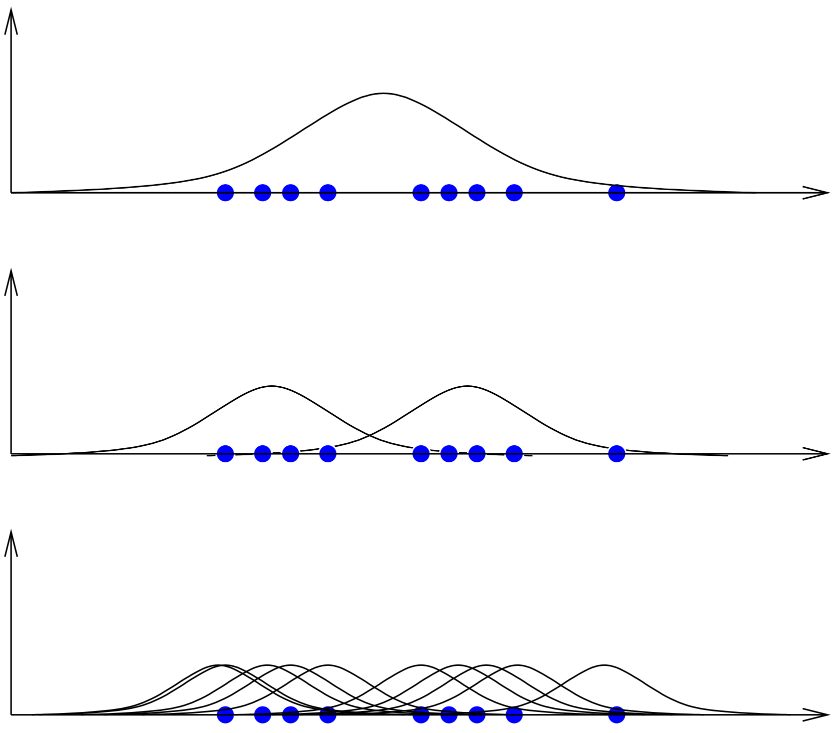


Figure 2.7: Gaussian density, mixture of Gaussians and kernel density

Density estimation will play an important role in robust statistics in Section 2.3, because the density value allows to quantify the degree of correspondence between the general structure in the data and the single observation. In particular, examples with a low density value do not fit to the rest of the data very well.

2.2.2 Clustering

The goal of clustering is to assign the observations to different groups (clusters) such that the observations in each single cluster are similar to each other and observations from different clusters are not similar to each other. Specific clustering algorithms differ in the type of similarity measure they use. Sometimes it is more useful to define clustering in terms of distance measures, where two observations are the more similar, the less distant they are.

Most clustering algorithms are either hierarchical or partitional. Hierarchical clusterers construct a hierarchy of clusters, ranging from the most specific clusters (each observation is one cluster) to the most general cluster (all observations are in one cluster), where in each level of the hierarchy clusters consist of the union of more specific clusters. In partitional clustering the number of clusters k is defined by the user and the clusterer learns a function which maps each example to its cluster number $\{1, \dots, k\}$. In the context of this thesis we are mainly interested in partitional clustering algorithms, mainly because it shares some important relations with density estimation.

A very popular clustering algorithm is k-means clustering, which can be seen as a generalization of mixture-of-Gaussians density estimation. Clusters are defined by their mean vectors μ and each observation is assigned to the cluster with the closest mean. A variant of k-means is k-medoids clustering. The medoid of a cluster C is the observation from the cluster with minimal sum of distances to each other observation in C. In particular, k-means and k-medoids have a canonical measure of similarity between a single observation and a cluster via the similarity of the observation and the mean / median. Often, these cluster membership values are equivalent to densities with respect to the construction of confidence measures of classifiers or the identification of outlying observations.

With respect to interpretability, k-medoids is preferable to k-means, because a medoid is an observation and as such related to a concrete real-world entity, while a mean vector is a mathematical construct that does not need to have a correspondence to a real-word object. For example, in patient data a mean can be 50% female and 50% male, while a medoid is a concrete patient and hence either female or male.

2.3 Robust Statistics

The field of robust statistics deals with the problem that in real data, several assumptions usually made in statistics can fail or hold only approximately. In particular, many statistical methods are based on the assumption that the observed data adheres to a previously known family of distributions (i.e.,a hypothesis language). In practice however, errors in the data are more the rule than the exception. It is not unusual that because of automatic data acquisition and the large number of data collected, even very gross errors go unnoticed. Sources of these errors contain typos (9 instead of 3 or $1,000$ instead of 1.000), errors in unit conversions (inches instead of meters), malfunctioning equipment (readings of zero because a wire got loose, spikes in measurements when a device is turned on) and recording observations that were not expected to happen (a patient moving instead of lying still when recording an ECG, an observation for which no suitable table in a database exists). And even if no such error is present, it is quite possible that a statistical method makes gross errors modeling the data because the actual distribution of the data does not fit to the family of distributions used in the particular method. In general, serious problems can occur if there are more observations lying far away from the bulk of the data than there should be (this problem is called *large tails* in statistics) or observations were assumed as independent when in fact there is a correlation.

Robust statistics deals with the question what can happen in situations where the usual assumptions are met only partially. [Hampel et al., 1986] define the main aims of robust statistics as

1. To describe the structure best fitting the bulk of the data.
2. To identify deviating data points (outliers) or deviating substructures for further treatment, if desired.
3. To identify and give a warning about highly influential data points (leverage points).
4. To deal with unsuspected serial correlations, or more generally, with deviations from the assumed correlation structures.

The immediate connection to the problem of global and local models is obvious: In this thesis, the structure best fitting the bulk of the data is called the global model. Taking the perspective of the global model, any local model will look like outliers or a deviating substructure, which leads to goal 2. Thirdly, a very influential data point in the global model may be a sign that this point is distorting the model and should better be handled by a local model. Hence, in the context of local models, goal 3 is connected with the question of discriminating between global and local structures and identifying local patterns in general. The problem formulated in goal 4 lies beyond the scope of this thesis. It is, however, an interesting task when working with time-dependent data and concept drift.

Historically, robust statistics dealt mostly with one-dimensional data and statistics such as estimators for the location and the scale of the data like mean and variance. A typical analysis in this setting is this: given a set of data points $x_1, \ldots, x_n$, the mean $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ shows some structure of the data. Then, the distance $d_i = |x_i - \bar{x}|$ can be used as a measure for the deviation of a point from the bulk of a data and one could select a threshold τ to define all points x_i with $d_i > \tau$ as outliers. Taking a closer look at $\bar{x}$, however, makes clear that even when all but one points lie very close together, the far away point can result in $\bar{x}$ lying arbitrarily far away from the rest of the data. Hence, a far away data point is very influential, as it has a pronounced effect on the discovered structure $\bar{x}$. Removing the influence of these influential points, e.g. by removing extreme points prior to calculating the mean (trimming, using the median), replacing these points with nearer points (winsorising) or calculating a weighted mean (L-estimators) leads to an estimate which lies closer to the bulk of the points. These estimators are less influenced by far away points, they are more robust.

Looking at models for outlier-contaminated data in the supervised learning setting, one can see that for a regression problem, the general definitions and aims of robust statistics can be generalized by looking at the mutual distribution $P(X \times Y)$ of the observations and their label. For a classification model, the problem becomes more complicated, as we do not have a density on $X \times \{-1, 1\}$. One can find a generalization of the usual outlier models by handling $P(X|Y = 1)$ and $P(X|Y = -1)$ separately and defining (x, y) as an outlier if x is an outlier with respect to the distribution corresponding to y. Notice that in this case (x, y) can be an outlier either because it is an outlier in the non-supervised sense, that is, x lies far away of the bulk of the data regardless of its label, or because it has the wrong label, meaning it is an outlier with respect to $P(X|Y = y)$. This demonstrates an essential problem of outliers in supervised learning: in principle, we are only interested in outliers with respect to the label, because we are only interested in predicting the label in the first place and it does not matter if x lies far away from the rest of the data as long as we can predict its y correctly. But when predicting a new observation, the decision whether this observation belongs to the main model or constitutes an outlier or not can only be made on the basis of x, because y is not known at this point.

[Barnett and Lewis, 1994] make a distinction between influential points and outliers: Consider the data in Figure 2.8. From the unsupervised perspective it seems clear that point A is an outlier while point B fits to the rest of the data. From the supervised perspective, it is reasonable to assume a linear model $Y = cX + d$ behind the data and now it turns out that point A perfectly fits the linear model, while point B disrupts the structure behind the data. This shows that inferring about outliers becomes a very complex problem with structured data, gaining even more complexity with higher dimensions, as it does not suffice to identify those points that are far away from the rest in *some* way. This observation has a deep impact on construction principles for robust methods: in the unsupervised case robustness is typically achieved by restricting the influence a single observation can have. If an observation has high influence, that is, if the model inferred from the data with and without this observation differ significantly, this is a sign that the observation does not fit to the rest of the data and should be called an outlier. In the supervised case, a high influence does not need to be a sign of an outlier. In Figure 2.8 both points A and B have a high influence on the model parameters c and d, but while the influence of B is misleading, the influence of A will not disturb the parameters estimation, but will be highly influential in reducing the standard error of the estimates. In fact, in a controlled experiment point A may be especially constructed to achieve this effect by choosing an extreme value of X and observing the corresponding Y. In conclusion, one need to distinguish carefully between (perhaps influential) outliers and (perhaps outlying) influential observations, where the important difference is the 'pattern-breaking' effect ([Barnett and Lewis, 1994], Ch. 8).

Robust statistics consists of three main parts, namely outlier models, outlier detection procedures and robust methods. Outlier models are a formalization of the assumptions how both the uncontaminated data and the outliers are generated, outlier detection methods test whether some observation x belongs to a given model or is an outlier, and robust methods estimate the main model in the presence of outliers. These three parts are closely connected. Obviously, the construction of an outlier detection method or a robust method depends on the kind of outliers one accounts for and hence on the outlier model. Further, given an outlier detection method, removing all outliers and estimating a model in a classical way gives a robust model. Vice versa, estimating a robust model, a threshold on the probability density defines an outlier detection procedure.

2.3.1 Outlier Models

The basic idea for outlier models is that there are two data sources, the “valid” data, usually assumed to belong to a known family of models and the outlier data, with a generally unknown, possibly very complex distributional form. Each observation is drawn either from the uncontaminated distribution or from

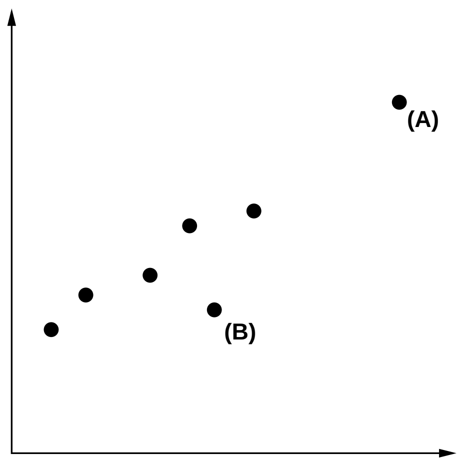


Figure 2.8: Influential point (A) and outlier (B) with of a linear model.

the outlier distribution, where the a-priori probability of an outlier is unknown, but less than 0.5. For a finite sample this translates to the property that an unknown number k of the observations in the sample are outliers.

One point that separates outlier models from models usually used in classification is that it is assumed that the observations in a sample are drawn independently of another from the underlying distribution. This assumption is necessary because the estimation of a model with several dependent observations is much more complex and often not practicable. Assuming independence basically means that a sample of size k can be used as a set of k single examples instead of one example of size k. In the case of outliers, this assumption may still be necessary and justified for the uncontaminated data, but one can usually not guarantee this for the contaminated data. For example, assume we had some technical equipment to automatically record data. As long as these sensors work properly and the observed instances are selected randomly, it is safe to assume independence of the sensor readings for each instance. However, when the sensors fail it may take some time to repair them and hence the probability of getting a faulty sensor reading is much higher when the last reading was faulty already.

Next to correlated observations, there are two kinds of outliers that can occur, namely outliers with respect to location and outliers with respect to scale. In the strict statistical sense, outlyingness with respect to scale describes the situation where the outliers have a different mean than the uncontaminated data while outlyingness with respect to scale means the outliers have a different variance. In a more general sense, one has to distinguish between the situation where the interesting properties (mean or optimal classification rule) of the bulk of the data differ from those of the outliers and the situation where the interesting properties are the same for both outliers and normal data, but the variance of the estimator of these properties is different.

According to [Gather and Becker, 1997], the most important statistical models for outlier-contaminated data are the Ferguson-type model, labeled outlier models, mixture models and the α-outlier model.

The Ferguson-type model [Ferguson, 1961, Gather, 1989] assumes that the observations are independent, that the number k of contaminants in the sample of size N are known, but the positions of the outliers in the sample are not known. The regular observations belong to an unknown distribution F from a known class of distributions (hypothesis space) $\mathcal{F}$, while each of the contaminants belongs to a possibly different distribution G_i from a class of distributions that can depend on F. For example, F could be a Gaussian distribution with unknown mean and variance and each G_i could be a Gaussian distribution with different mean but same variance.

One problem of the Ferguson-type model is that it may not coincide with the intuition about outliers. The term 'outlier' is connected with extremeness in some way, but in the Ferguson-type model it may be the case that the outlier distributions are such that the outlier lie right among the regular observations. Although this may be unlikely for the maximum likelihood Ferguson-type model [Gather and Kale, 1992], it may still be preferable to assure that only extreme observations can become outliers. This is done in labeled outlier models, where the observations are sorted in order of their 'extremeness' and assumes the first $N-k$ observations to be uncontaminated and the k most extreme observations to be the outliers. For both groups a distribution can then be found independently of another. This approach needs a formal description of the 'order of extremeness' of the observations. For one-dimensional data, the usual order of the real numbers can be used. For multivariate data, a center-outward ordering induced by the data depth can be defined [Liu et al., 1999]. A data depth measure indicates how central a given point is in comparison to the other points from the sample or with respect to a distribution. Some examples of data depths are:

Mahalanobis Depth [Mahalanobis, 1936]: this depth is defined based on the Mahalanobis distance of the point x to the mean of the distribution

$$D(x) = \frac{1}{1 + (x-\mu)\Sigma(x-\mu)}$$

where μ is the mean and Σ the covariance matrix of the underlying distribution or the sample.

Half Space Depth [Hodges, 1955, Tukey, 1975]: this depth is defined as the infimum of the probabilities of any half-space that includes the point x.

Convex Hull Depth [Barnett, 1976]: The convex hull depth is defined iteratively: all points lying on the convex hull of the data have depth 1, and all points lying on the convex hull of the data with all points of depth less than i removed are assigned depth i.

See [Liu et al., 1999] for more examples of data depths and a more detailed discussion.

Both Ferguson-type and labeled outlier models assume that the number k of outliers in the sample is known. In practice, this assumption is seldom fulfilled. Instead, one can view k itself as a random variable distributed by a Binomial distribution with parameters N and p. This is equivalent to assuming that each of the N observations is drawn from the regular distribution with probability $1 - p$ and from the outlier distribution with probability p. It is assumed that the observations are independent and that all outliers come from the same distribution. The corresponding model is called the mixture model of outliers.

The α outlier model [Davies and Gather, 1993] is a result of the α outlier identification rule. First, the α outlier region $out(\alpha, F)$ of a distribution F with density f is defined as

$$out(\alpha, F) = \{x | f(x) < \sup\{\delta | P(f(X) < \delta) \leq \alpha\}\}.$$

That is, all points with density below a threshold δ are used and we pick the largest of these thresholds which has probability of at most α. All points in $out(\alpha, F)$ are declared as outliers. Hence, the probability of mistakenly taking a point from F as an outlier is at most α. The α outlier model is a model with $N - k$ regular observations, independently drawn from a distribution F and k outliers. The only assumption about the outliers is that their support lies in the α outlier region of F.

A problem common to all outlier models is that of model selection. All outlier models assume the family of models that the regular observations are distributed by is known. However, it is not clear from the original data which model class to use, because labeling different observations as outliers may result in the rest of the points being better modeled by different model classes. For example in Figure 2.3.1 the data can either be modeled with a diagonal line, taking points A and B as outliers, or with two lines parallel to the x-axis, taking points C and D as outliers. Hence, from the contaminated data it may not be obvious which model class to use for the uncontaminated data. Moreover, the notions of "similarity" or "deviation" of observations from structures, which are a fundamental concept for the definition of outliers, depend on a certain representation of the data. For example, January, 1st and December, 31th appear to be very far away when encoded as numerical values, but are close together in the calendar. It is well known that a carefully selected representation is essential for the success of a statistical analysis or data mining application and it is important to remember that a transformation of the data can also mean a change in the assumptions about outliers.

2.3.2 Robust Methods

The problem with outlier detection and outlier models is that in practical application it is sometimes very hard to define which observation constitutes an outlier, because it is not clear what the meaningful structure behind the data looks like. Only when such a structure is known, observations that do not fit to this structure or prevent a more sensible structure to be found can be called outliers with a certain right. This brings us to the area of robust methods. Roughly speaking, a method is called robust if changes in a small part of the data only have a bounded, preferably very small, effect on the estimate.

There are several reasons to prefer a robust method. The first reason may be that one has some prior about the data at hand, such that one can specify the structure of the process one is looking for or at least can specify some observations that are definitely not part of the structure one is looking for. For example, in real-world applications one can often easily specify that a certain

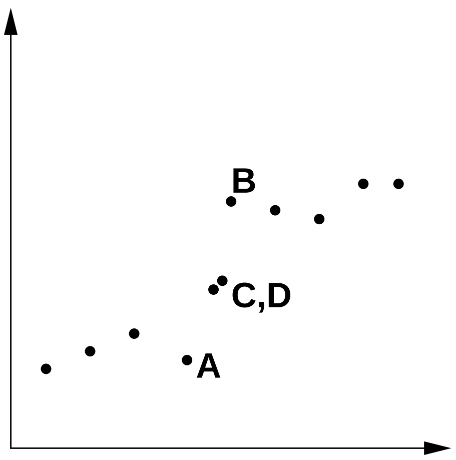


Figure 2.9: Model selection problem with outliers

measurement cannot exceed some bound, for example the heart rate of a patient cannot lie above 300. Formally speaking, one has reason to select a specific class of outlier models, and needs a procedure to analyze the regular observations only. In other situations one does not have prior knowledge about the model class, but chooses one based on measures like prediction performance. In this case, one does not have reason to priori believe that a specific observation will be an outlier or that the regular observations follow a specific structure and hence it is not justified that the influence of some observations should be limited or ignored. Still, the use of a robust method may be justified, as it is questionable whether a structure inferred from only a small fraction of points will be repeated in new data and hence lead to an increase in prediction performance. Here robustness is used as a method to avoid overfitting the data. Finally, robustness is a prerequisite for interpretability: it is meaningless to try to understand a model if a slight change in the data will give a very different model, because in this case it is safe to say that the model did not capture the basic structure of the data very good.

It is important to notice that robustness is only one property one would like to have in a learning algorithm and that indeed an increase in robustness only comes with a decrease in other performance criteria, such as prediction performance. The problem is that there is no general way to be sure that a certain observation is an outlier and not just a very unlikely regular observation. Hence, by excluding a certain kind of regular observations the learner can become biased or less efficient.

The Approach based on Influence Functions

Much of the work in robust statistics is based on the influence function. First, we consider functionals T, defined on a suitable set of probability distributions, which map a distribution to a vector $\theta \in \Theta \subseteq \mathbb{R}^d$. In particular, T should be defined on the set of empirical distributions. An empirical distribution $G_{(x_1,\ldots,x_n)}$ corresponding to a set of observations $x_1, \ldots, x_n$ places probability mass $1/n$

on each x_i and mass 0 elsewhere. In short, $G_{(x_1,\ldots,x_n)} = \frac{1}{n}\sum_{i=1}^{n} \Delta_{x_i}$, where Δ_x is the point mass in x. Hence, we can identify a set of observations with a probability distribution. Next, we only consider statistical procedures which can be written as a functional.

Definition 2.3.1 (Influence Function [Hampel, 1974])**.** The influence function of a functional T at a distribution F is defined as

$$IF(x;T,F) := \lim_{h\to 0} \frac{T((1-h)F + h\Delta_x) - T(F)}{h}$$

The influence function is a kind of differential that measures how much an infinitesimal change of the distribution F at a point x will influence the functional's T estimate of F. The influence function is by construction a local measure of robustness. There are also many other statistical concepts to formalize robustness, like Gross-Error Sensitivity, Self-Standardized Sensitivity, or Breakdown point (see [Hampel et al., 1986]).

The influence function is straightforward to convert into an easy to compute empirical measure: instead of measure the influence of an infinitesimal change of the distribution, we measure the influence of a new example x.

Definition 2.3.2 (Empirical Influence Function [Hampel et al., 1986])**.** The value of the empirical influence function of an estimator T_n at a sample $X = (x_1,\ldots,x_{n-1})$ is

$$eIF(x,T_n,X) = T_n(x_1,\ldots,x_{n-1},x).$$

A variant of the empirical influence function is the sensitivity curve:

Definition 2.3.3 (Sensitivity Curve [Tukey, 1977])**.** The sensitivity curve of an estimator T_n at a sample $X = (x_1,\ldots,x_{n-1})$ is

$$SC_n(x) = n(T_n(x_1,\ldots,x_{n-1},x) - T_{n-1}(x_1,\ldots,x_{n-1}))$$

which is a linear transformation of the empirical influence function. It measures the difference between the estimates with and without a new observation at x, scaled with the sample size. Taking the empirical distribution of the sample as an approximation of the actual distribution F and noting that the new x corresponds to a point mass contamination of F of size $1/n$, we find that the sensitivity curve can be expected to converge against the influence function for $n \to \infty$.

To get an estimate of the covariance matrix one can use the jackknife estimator.

Definition 2.3.4 (i-th Jackknifed Pseudovalue [Tukey, 1958])**.** The sensitivity curve of an estimator T_n at a sample $X = (x_1,\ldots,x_n)$ is

$$T^*_{ni} = nT_n(x_1,\ldots,x_n) - (n-1)T_{n-1}(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n)$$

Noting that $T^*_{ni} - T_n(x_1,\ldots,x_n)$ is the value of the sensitivity curve at x_i, we can approximate the variance of the influence function with respect to a distribution F by its variance with respect to the empirical distribution F_n

of the sample, and then approximate the influence function by the sensitivity curve. This yields

$$\begin{aligned}
V(T,F) &= Var_F(IF(x;T,F) \\
&\approx Var_{F_n}(IF(x;T,F) \\
&= Var(IF(x_i;T,F); i = 1\ldots,n) \\
&\approx Var(SC_n(x); i = 1\ldots,n) \\
&= Var(T^*_{ni} - T_n(x_1,\ldots,x_n); i = 1\ldots,n) \\
&= Var(T^*_{ni}; i = 1\ldots,n) \\
&= \frac{1}{n-1}\sum_{i=1}^{n}(T^*_{ni} - \mathrm{avg}(T^*_{ni}))^2
\end{aligned}$$

Another empirical measure is the breakdown point which is defined for a sample:

Definition 2.3.5 (Finite-sample Breakdown Point [Hampel et al., 1986])**.** The finite-sample breakdown point of an estimator T_n at a sample $X = (x_1,\ldots,x_n)$ is

$$\epsilon^*_n(T_n; x_1,\ldots,x_n) = \frac{1}{n}\max\{m|\max_{i_1,\ldots,i_m}\sup_{y_1,\ldots,y_m}|T(z_1,\ldots,z_n)| < \infty\}$$

where the sample $(z_1,\ldots,z_n)$ is obtained by replacing the data points $x_{i_1},\ldots,x_{i_m}$ by the arbitrary values $y_i,\ldots,y_m$.

However, it is in general unclear how to compute the supremum or to find suitable points $y_1,\ldots,y_m$, which makes this definition less practical than the other empirical measures.

In [Carbrera et al., 1997], empirical robustness measures for estimators T based on the distance of estimates and a resampling scheme are defined. It is assumed that there exists a distance measure for estimates that is meaningful for the task at hand. For a fixed values $\alpha \in]0, 0.5[$ and a set of observations $X = (x_1,\ldots,x_n)$, samples S of size $[n(1-\alpha)]$ are repeatedly drawn from X and the distance of $T(S)$ and $T(X)$ is recorded. As a robustness measure, the median of this distances divided by the maximum distance is proposed. The authors also consider the measures of the mean divided by the maximum of the interquartile range divided by the range. In general, one can define

Definition 2.3.6 (Resampling-based Robustness Measure [Carbrera et al., 1997])**.** Given a measure of distance of estimates, a sample X of size n, a constant $\alpha \in]0, 0.5[$, and a statistic S for the amount of variability in a sample of real numbers, the resampling based robustness measure based on S for an estimator T is the amount of variability in the distance of $T(X)$ to the value of T on a random subset of X of size $[n(1-\alpha)]$ over repeatedly selecting such a random subset.

The definition of the resampling-based robustness measure can be improved. The problem is that many sophisticated learners have a correction for the finite data size built in: in order to avoid over-fitting the data, these learners either automatically or by use of a tunable parameter, adapt their complexity to the number of examples available. Generally, less examples will mean a less complex model being estimated. The resampling-based robustness measure as defined above compare the model on all available data with a model on a $1-\alpha$ subset, hence the distance between the two models is only partly due to unwanted random effects related to robustness, but partly due to wanted complexity reduction. To remove the bias introduced by different levels of complexity in the estimates, it is better to use the following definition:

Definition 2.3.7 (Unbiased Resampling-based Robustness Measure)**.** Given a measure of distance of estimates, a sample X of size n, a constant $\alpha \in]0, 0.5[$, and a statistic S for the amount of variability in a sample of real numbers, the resampling based robustness measure based on S for an estimator T is the amount of variability in the distance of T on two random subsets of X of size $[n(1-\alpha)]$ over repeatedly selecting such a random subset.

Robustness in Prediction Tasks

Up to now we have only considered the estimation of a point $\theta \in \Theta \subseteq \mathbb{R}^d$. Prediction tasks, i.e. classification and regression, differ from this setting in two ways. First, the estimate is supposed to be a classification or regression function $f: X \to Y$, that is, Θ is a hypothesis space $\mathcal{H}$. Second, there is a fixed the performance criterion, that is, a definition of the optimal $\theta^* \in \Theta$. The goal is to find a function $f \in \mathcal{H}$ which minimizes the expected error with respect to a loss function L:

$$Err(f) := \int L(x, f(x), y) dP(x) \to \min .$$

Here, P is the probability distribution of the data and the loss function $L: X \times Y \times Y \to \Re_{\geq 0}$ measures the error that occurs when predicting the label of x as $f(x)$ instead of the true y. In many cases the loss function is defined to depend only on the residual $r = f(x) - y$. While specific approaches have replace the criterion of $Err(f)$ with some approximation, because the true distribution of the data P is generally not known, minimizing the prediction error is the true goal of prediction.

One measure of the distance of two prediction models, which can be used to define a resampling-based robustness measure, is the disagreement rate. In classification tasks, the disagreement rate is defined as the probability that the two models predict differently

$$Dis(f, g) = P(f(x) \neq g(x)).$$

A more general definition could be the estimated loss L of predicting f instead of g

$$Dis(f, g) = \frac{1}{2} \int \left(L(x, f(x), g(x)) + L(x, g(x), f(x)) \right) dP(x)$$

which coincides with the previous definition for the 0-1-loss usually used in classification. The empirical version of this measure can be estimated via cross-validation.

Let us first consider the case of parametric functions. In this case, the hypothesis space $\mathcal{H}$ consists of functions f_θ indexed by a parameter $\theta \in \Theta \subseteq \mathbb{R}^d$. In this case the hypothesis space can be identified with the space Θ and recovered the case of point estimation. The best known example of parametric estimation is the least sum of squares estimator which dates back to Gauss and Legendre. Given data $(x_i, y_i)_{i=1}^n$ the goal is to find a linear functions

$$f(x) = \theta * x + \theta_0 = \theta_1 x^{(1)} + \ldots + \theta_d x^{(d)} + \theta_0$$

which minimizes the L_2 error

$$L_2 = \sum_{i=1}^{n} (f(x_i) - y_i)^2.$$

The least square estimate is particularly susceptible to outliers, as can be seen from an example in [Rousseeuw and Leroy, 1987].

This is a general problem when the value of the loss function is unbounded and hence the influence of an example (x, y) on the estimated function f is unbounded. We will later see that the crucial step towards robustification of the estimates is to bound the value of the loss function of the influence one example can have.

Let us now investigate more general cases of prediction tasks. Not all hypothesis space can be cast into the framework of the estimation of a vector $\theta \in \mathbb{R}^d$. For example, Support Vector Machines estimate a function

$$f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x) + b$$

that is, they can be described by a $n + 1$-dimensional vector $(\alpha_1, \ldots, \alpha_n, b)$, which is not of a fixed dimension, but depends on the number of examples n. Logical learners like they occur in decision trees or Inductive Logic Programming cannot readily be represented as a real vector at all.

Taking a closer look at the definitions of robustness and robustness measures it turns out that the definitions can be generalized in a straightforward way to estimates in a Banach space (i.e. a complete normed vector space), where the norm is necessary to measure the size and distance of the estimates and the completeness is necessary for the limit in the definition of the influence function to exist. For example, the kernel function of a Support Vector Machine defines a Reproducing Kernel Hilbert Space $\mathcal{H}$ which contains the SVM function. As the name says, this space is a Hilbert space and hence a Banach space.

We can further generalize most of the robustness definitions and approximate the empirical measures if only a measure of the distance or similarity of estimates is defined. The general ideas and informal definitions of robustness – changes in small parts of the data should only have a small or bounded effect on the estimate – do not specify how the size of this effect should be measured at all. For the empirical measures, the empirical influence function does not need to be changed, the sensitivity curve can be adapted by using the distance instead of the difference of the estimates and in the estimate of the variance of the estimator we can use the variance of the new sensitivity curve approximation. In the definition of the finite-sample breakdown point it is easy to see that it is equivalent to replace the absolute value of the estimate by the distance of the estimate to a fixed default value. The resampling-based robustness measure again only uses a measure of distance of estimates in its definition.

Robustness and Predictive Performance

In order to investigate the connection between robustness and predictive performance, an experiment was conducted. Over 18 data sets and 10 cross-validation folds each, 4 learners were compared with respect to predictive performance and robustness. For each cross-validation fold, two sets of data were generated, each consisting of a random subset of 80% of the points from the training data. The learning algorithm was used to induce a model from each of these sets. Hence, the training sets of both models are to 80% identical. To measure the predictive performance of the learner, the error of the first model on the test set was reported. To measure the robustness, the fraction of points from the test set that were predicted differently by the two models was reported, the idea being that the more robust the model is, the less influence the 20% non-identical examples in both training sets should have. To obtain more general results, four very different learners were employed: a Support Vector Machine with dot kernel (linear numeric learner), a Support Vector Machine with radial basis kernel (nonlinear numeric learner), the J4.8 decision tree algorithm [Witten and Frank, 2000], which is a variant of the popular C4.5 algorithm [Quinlan, 1993] (propositional logic tree learner) and the JRip algorithm [Witten and Frank, 2000], which is a variant of RIPPER [Cohen, 1995] (propositional logic rule set learner).

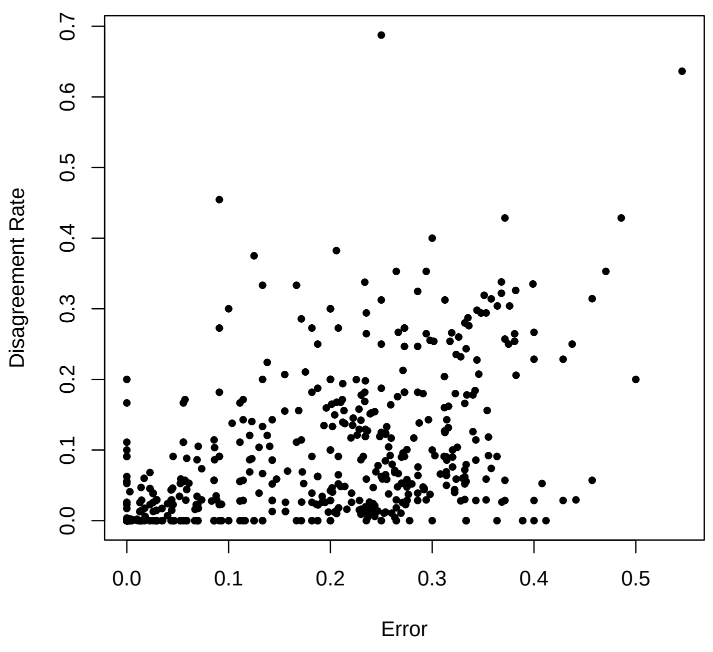


Figure 2.10: Prediction error vs. resampling-based prediction difference

Figure 2.10 compares the prediction error and the disagreement rate over four learners, 18 data sets and 10 cross-validation repetition. It shows that most of the data points (89.8%) lies below the diagonal, meaning that with high probability the disagreement rate is a lower bound of the prediction error. This result indicates that in order to achieve a low error, robustness is needed.

2.3.3 Robustification of Procedures

After realizing the importance of robustness in learning algorithms and defining robustness measures to compare different procedures, an interesting question is how to modify known algorithms to make them more robust. This is not only interesting as a method to develop novel, hopefully better algorithms, it is also important because in real-life situations there can be very different requirements and restrictions apart from accuracy and robustness that call for a particular learning algorithm or hypothesis space to be used. Hence, it may not be possible to simply use some very different algorithm that is more robust from a technical point of view. In conclusion, this leads to the question how learning algorithms can be robustified without loosing too much of their specific desirable properties.

Huber [Huber, 1964, Huber, 1973, Huber, 1981] proposed the class of M-estimators. Noticing that many statistical estimators are of the form

$$\sum_{i=1}^{n} \rho(x_i, T_n) \to \min_{T_n}$$

with $\rho = -\ln f_{T_n}$ where f is the density function in maximum likelihood estimation or $\rho = (y - f(x))^2$ in least-sum-of-squares regression, or of the form

$$\sum_{i=1}^{n} \psi(x_i, T_n) = 0$$

where ψ is the derivate of ρ, Huber proposed to use also other functions ρ or ψ.

Definition 2.3.8 (M-estimator)**.** Any estimator that can be written in the form

$$\sum_{i=1}^{n} \rho(x_i, T_n) \to \min_{T_n}$$

or in the form

$$\sum_{i=1}^{n} \psi(x_i, T_n) = 0$$

is called an M-estimator.

Obviously, the robustness properties are a matter of choosing the right function ρ or ψ. One example is to bound the derivative of ρ, e.g. by setting

$$\psi(x) = \min(c, \max(-c, \rho'(x)).$$

In particular, for prediction tasks the M-estimator translates into choosing a different loss function L. One might ask whether it is admissible to change the loss function, as the loss function is part of the definition of a prediction task and changing the loss function will result in changing the learning problem. In general, this objection is justified. When the setup of the learning problem states that a misprediction at a certain point x causes a very large error, then the model should cover this example closely, even if this means a larger error on all other examples. On the other hand, in real-world data mining tasks the loss function is seldom specified precisely, it is more common that only parts of it are specified, e.g. the desired symmetry properties, or that one simply resorts to using a standard loss function like least-sum-of-square or minimum absolute error without further consideration of the alternatives. Also, in the case of outlier-contaminated data it may be that a model fitted to the contaminated data with a modified loss function better fits the regular data with respect to the real problem-specific loss function, as the distorting influence of the outliers outweighs the errors introduced by the modified loss function.

For the case that outliers have already been identified, the M-estimator can be modified to bound the influence of these observations.

Definition 2.3.9 (Generalized M-estimators [Rousseeuw and Leroy, 1987])**.** An estimator that can be written in the form

$$\sum_{i=1}^{n} w(x_i)\rho(x_i, T_n) \rightarrow \min_{T_n}$$

or in the form

$$\sum_{i=1}^{n} w(x_i)\psi(x_i, T_n) = 0$$

is called M-estimator. □

The most simple generalized M-estimator is to set $w(x) = 0$ for all outliers, removing the outliers completely.

An approach which is somewhat orthogonal to the M-estimators is to not modify the loss function, but the way the losses at the individual points are aggregated into a single value. Instead of summing up, Rousseeuw proposed to use the median of the squared errors in regression [Rousseeuw, 1984], yielding the LMS estimator. Originally, the least median idea was only formulated for regression, as one needs to identify the worst outliers from better approximated examples, which is not possible for the 0-1-loss usually used in classification. However, for numerical classifiers of the form $sign(f(x))$ one can generalize this definition by looking at the error terms $yf(x)$.

Definition 2.3.10 (Least Median Estimator)**.** Given examples $(x_i, y_i),_{i=1}^n$, a hypothesis space $\mathcal{H}$ and a loss function L, the least median estimator is the function $f \in \mathcal{H}$ that minimizes

$$\text{median}_{i=1\ldots n} L(x_i, f(x_i), y_i).$$

A closer analysis of LMS regression exhibited that the LMS estimator has an abnormally slow convergence rate[Rousseeuw, 1984, Rousseeuw and Leroy, 1987]. This may be explained by the fact that it makes use of very little of the information in the error terms: only the size of the median is reported, from all other errors only their order, but not their size is used. Obviously, there are still many estimators possible that achieve the same median of errors, but predict the examples with errors below the median better or worse. This is improved in the least trimmed loss estimator, which is a generalization of Rousseeuw's least trimmed squares estimator [Rousseeuw, 1984].

Definition 2.3.11 (Least Trimmed Loss Estimator)**.** Given a set of examples $(x_i, y_i)_{i=1}^n$, a hypothesis space $\mathcal{H}$ and a loss function L, assume that the examples are ordered such that $L(x_1, f(x_1), y_i) \leq \ldots \leq L(x_n, f(x_n), y_n)$. For an number h, $1 \leq h \leq n$, the least trimmed loss estimator is the function $f \in \mathcal{H}$ that minimizes

$$\sum_{i=1}^{h} L(x_i, f(x_i), y_i).$$

The least trimmed loss estimator can easily be computed by use of a resampling technique. A random subset of h examples is drawn from the training set and a function is fitted using classical techniques and its error recorded. This step is repeated for a fixed number of times and the function with the lowest error is returned [Rousseeuw and Leroy, 1987]. This algorithm can further be improved by local optimization: after finding the model on the h selected examples, the h examples with the lowest loss are chosen. If this set of examples differs from the original h examples, a new model is estimated on these examples. One example of local optimization can be found in the Minimum Covariance Determinant Estimator [Rousseeuw and Van Driessen, 1999]. Finally, it is also possible to generalize the least median and least trimmed loss estimators by choosing other robust measures of the size of the loss, see [Rousseeuw and Leroy, 1987].

2.4 Probabilistic Classifiers

A particular type of numerical classifiers are estimators $\hat{P}(Y = 1|x)$ of the conditional class probability $P(Y = 1|x)$. These can be transformed into regular classifiers via

$$f(x) = \text{sign}\left(\hat{P}(Y = 1|x) - 0.5\right).$$

These types of classifiers are called probabilistic classifiers.

A probabilistic classifier should be well-calibrated, that is for each interval of probabilities $[p_1, p_2]$ the probability of drawing a positive example given the classifier predicts $\hat{P}(Y = 1|x) \in [p_1, p_2]$ should also be in $[p_1, p_2]$. However, calibration is not sufficient because it is easy to construct a perfectly calibrated classifier by assigning the default probability $P(Y = 1)$ to all examples [Niculescu-Mizil and Caruana, 2005].

A better approach is to measure the error of a probabilistic classifier by a loss function, e.g. the squared loss (using $y \in \{0, 1\}$ as the target) or the cross-entropy loss (see Section 2.1.1). For these losses it can be shown that a small error corresponds to a small distance of the distributions $P(Y = 1|x)$ and $\hat{P}(Y = 1|x)$.

2.4.1 Probabilistic Scaling Methods

Softmax Scaling

Probably the easiest scaling method is softmax scaling, which is defined by

$$\sigma_{softmax}(f(x)) = \frac{1}{1 + \exp(-2f(x))}$$

The softmax scaler monotonically maps the real function values $f(x)$ into the interval $]0,1[$ and hence adheres to the intuition that a higher value of $f(x)$ corresponds to a higher conditional class probabilities $\sigma(f(x)) = P(Y = 1|x)$. It also maps the critical value $f(x) = 0$ into $P(Y = 1|x) = 1/2$, the value of maximum uncertainty of the class. However, interpreting the softmax values as probabilities is not justified by any information from the examples; the scaler is not calibrated over the data set at all. It hence relies completely on the learner to give meaningful values $f(x) \approx -\frac{1}{2}\log(P(Y = 1|x)^{-1} - 1)$ (which, by the way, is exactly the idea of logistic regression and kernel logistic regression [Wahba, 1999]).

Gaussian Scaling

Gaussian Scaling assumes the function values $f(x)$ are distributed according to a mixture of two Gaussians, one for each class. Given a training set (x_i, y_i), the mean μ_1 and standard deviation σ_1 of the two sets $\{f(x_i)|y_i = 1\}$ and the corresponding values μ_{-1} and σ_{-1} are computed. This yields Gaussian density functions $n_{\mu_y,\sigma_y}(\cdot)$ for $y \in \{-1, 1\}$. The conditional class probability then follows from standard statistics as

$$\sigma_{Gauss}(f(x)) = \frac{n_{\mu_1,\sigma_1}(f(x))}{n_{\mu_1,\sigma_1}(f(x)) + n_{\mu_{-1},\sigma_{-1}}(f(x))}$$

As a special case, for $\sigma_1 = \sigma_{-1}$ and $\mu_1 + \mu_{-1} = \sigma_1$, the Gaussian scaling function is identical to the softmax scaler. It should also be noted that the Gaussian scaling function is only monotonic for $\sigma_1 = \sigma_{-1}$.

Isotonic Regression Scaling

Isotonic Regression [Zadrozny and Elkan, 2002] computes a monotonic, piecewise constant scaling function. The algorithm runs as follows: let the training set $(x_i, y_i)_{i\in I}$ by ordered by $f(x_i)$. Define probability values $p_i := 1$ iff $y = 1$ and $p_i := 0$ iff $y = -1$. and weights $w_i :_= 1$. For any two consecutive values i and $i+1$ that violate strict monotonicity, that is $p_i \geq p_{i+1}$, remove these two values from I and replace them by a new value i' with $f(x_{i'}) = f(x_i)$, $w_{i'} = w_i + w_{i+1}$ and $p_{i'} = (w_i p_i + w_{i+1} p_{i+1})/w_{i'}$. Iterate this step until no more violators of the monotonicity are found. Then, the final value $f(x_i), i \in I$, define the thresholds of the piecewise constant function and $p_i, i \in I$ the corresponding scaled values.

One can also post-process the piecewise constant function in order to get a continuous or smooth function, e.g. by approximation with splines.

Beta Scaling

Beta Scaling [Garczarek, 2002] is a scaling methods designed for multiple classes. Given functions $f_1(x), \cdot, f_k(x)$ that predict the membership of x in the k classes, the Beta Scaling algorithm assigns each example to the class with the highest membership value and scales each predicted class independently of the others. The membership values $f_i(x)$ are initially scaled with an ad-hoc method, e.g. softmax scaling, and then both the scaled membership values and the true classes on an independent test set are viewed as realizations of Beta distributed random variables. The Beta distributions are a flexible class of distributions that are completely characterized by their means and standard deviations. Optimally, both Beta random variables should agree and hence the parameters of the two Beta distributions are selected such that the mean of the membership values and class values agree. The standard deviation of the membership values is taken from the observations. The standard deviation of the distribution modeling the true classes is finally optimized to minimize the mean squared error.

Beta distributions are of the form

$$B(\alpha,\beta)(x) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)+\Gamma(\beta)}(1-x)^{\beta-1}x^{\alpha-1}$$

with parameters α and β dependent on the mean and standard deviation, such that the final scaling function has the form

$$\sigma_{Beta}(f(x)) = B(\alpha_T,\beta_T)^{-1}B(\alpha_M,\beta_M)(\sigma_{softmax}(f(x)))$$

where the subscript M denotes the parameters estimated from the membership function and the subscript T denotes the parameters estimated from the true class.

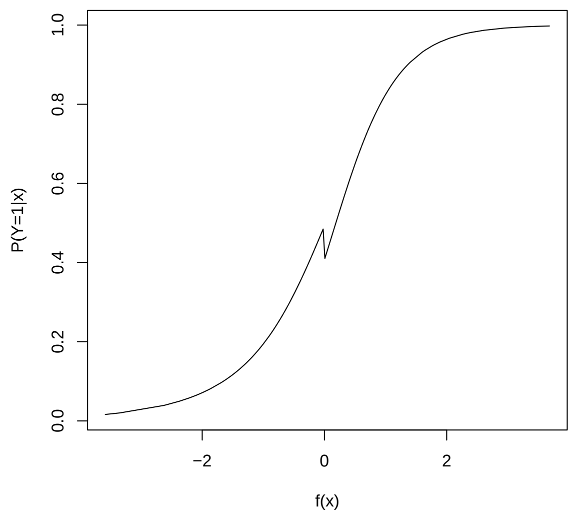


Figure 2.11: Beta Scaling Function

Beta Scaling has the problem that due to the independent scaling of the two predicted classes, the scaled function does not need to be continuous at the class borders. In Figure 2.11 we see the linear SVM decision function $f(x)$ on the Business data set plotted against the Beta Scalers estimation of $P(Y=1|x)$ on the y-axis. There is obviously a discontinuity at $f(x)=0$ which is not justified by the continuous SVM decision function. This makes the estimated values around $f(x)=0$ unreliable. In Section 2.4.2 a modification of Beta Scaling for the two class case will be presented that avoids discontinuities.

Platt Scaling

Platt Scaling [Platt, 1999] was originally introduced for scaling Support Vector Machines outputs but has been shown to be efficient for many other numerical decision functions as well [Niculescu-Mizil and Caruana, 2005]. Based on an empirical analysis of the distribution of SVM decision function values, Platt suggests to use a scaling function

$$\sigma_{Platt}(f(x)) = \frac{1}{1 + exp(Af(x) + B)}.$$

The form of the function is partly motivated by a monotonicity assumption, note that this is exactly the form of Gaussian Scaling when the Gaussian scaling function is monotonic.

The parameters A and B are optimized using gradient descend to minimize the cross-entropy error

$$-\sum_{i=1}^{n} t_i \log(\sigma_{Platt}(f(x_i))) + (1 - t_i) \log(1 - \sigma_{Platt}(f(x_i)))$$

where t_i is defined as $t_i = \frac{N_1+1}{N_1+2}$ if $y_i = 1$ and $t_i = \frac{N_{-1}+1}{N_{-1}+2}$ if $y_i = -1$ where N_y is the number of examples with label y, these values are used instead of $t_i \in \{0, 1\}$ to give a uniform uninformative Bayes prior over the true class probability.

2.4.2 Novel Probabilistic Scaling Methods

This section describes three new ideas for scaling methods, namely a fix for Beta Scaling of binary classes, a robustification of Platt Scaling, and a simple scaling algorithm for Support Vector Machines.

Binary Beta Scaling

We have seen that Beta Scaling suffers from the discontinuities at the class border which stem from the independent scaling of each predicted class. For binary problems, this problem can be circumvented by jointly modeling the positive class probability $P(Y = 1|x)$ instead of the correctness probability $P(y|x)$. This allows to model Beta distributions over both predicted classes simultaneously. The rest of the algorithm remains unchanged.

This trick works only for the binary case, as is makes use of the fact that $P(Y \neq y|x) = 1 - P(Y = -y|x)$, thereby connecting the error probability to the correctness probability of the complementary class.

Robust Platt Scaling

Platt Scaling uses the logarithm of the predicted class in its cross-entropy target function and hence gives high weight to mispredicted points with high confidence $f(x_i)$. This becomes a problem when a small set of outliers can not be appropriately modeled by the classifier. In particular with a linear classifier, these outliers may get a very high confidence value, simply because they are very far away from the linear decision function and hence a small set of points can contribute a large part of the cross-entropy error. When the scaler lowers the probability prediction to reduce the error on these examples, due to the monotonicity assumption the prediction on the rest of the examples from the same predicted class will decrease as well, effectively increasing the error on a large part of the examples.

This motivates a robustification of Platt Scaling. Inspired by a similar problem in robust regression, the least trimmed loss and least median loss estimator [Rousseeuw, 1984, Rousseeuw and Leroy, 1987] from robust statistics are transferred to this problem.

The more simple approach is to remove a fraction τ, say $\tau = 0.05$, of the examples with highest absolute value $|f(x)|$ in order to bound the maximum influence a example can have. This approach will be denoted the Robust Platt algorithm in the experiments.

A more sophisticated approach is to optimize the fraction τ over values in $]0.5, 1]$. In order to get a reliable estimate of the overall error, the median of the errors of all training examples is used to select τ. This approach is denoted the Robust Platt Median algorithm in the experiments.

Theoretical Background and a Simple SVM Scaling

The goal of classification algorithms is to predict the class label itself, not its probability. Of course, these goals are connected, but it still raises the question if there is a bound on the error one can achieve by scaling a classifier into a probability prediction. Bartlett and Tewari [Bartlett and Tewari, 2004] show that there is a tradeoff between sparseness of a classifier and the ability to estimate conditional probabilities. Their theorem shows that if the conditional class probability can be estimated on some interval of values of f, sparseness is lost in that region.

Support Vector Machines are sparse classifiers, they only put a restriction on those examples with $yf(x) \leq 1$. Hence, an optimal class probability estimation is only possible for values $|f(x)| \leq 1$. This motivates to only scale the SVM in the region $|f(x)| \leq 1$ and use a trivial scaler for the rest of the examples.

The following trivial scaling algorithm implements this idea: the training examples are divided into three sets, those with function values $f(x) < -1$, values $f(x) \in [-1, 1]$ and values $f(x) > -1$. For the examples with values $f(x) > 1$, the scaled value is the simply the precision of the rule in this region, accordingly for the examples with values $f(x) < -1$. For the examples with values $f(x) \in [-1, 1]$, the scaled value is of the form

$$\sigma_{simple}(f(x)) = \frac{1}{1 + exp(Af(x))}$$

where the constant A is chosen such that $\sigma_{simple}(1)$ is the weighted mean of the precisions in the outer regions. This yields small discontinuities at -1 and 1, but the scaling function is still monotonic. This scaling function requires no optimization and can be computed in a single run over the training set.

2.4.3 Empirical Comparison

The following table shows the mean squared error and the cross-entropy error of the compared methods for both the linear and the radial basis function Support

Vector Machine. The given numbers are the means over the 18 standard data sets.

Method	linear		RBF	
	MSE	CRE	MSE	CRE
Platt Scaling	0.0891	0.2808	0.0809	0.2520
Beta Scaling	0.5352	8.3067	0.5312	6.7117
Isotonic Regression	0.0904	0.2847	0.0825	0.2613
Gauss	0.0877	0.2740	0.0831	0.2776
Binning	0.0982	0.3367	0.0858	0.3105
Softmax	0.0896	0.3013	0.1065	0.3491
Precision	0.0942	0.3032	0.1049	0.3344
Binary Beta	0.0887	0.2873	0.0870	0.2889
Robust Platt	0.0885	0.2793	0.0809	0.2519
Robust Platt Median	0.0869	0.2750	0.0808	0.2520
Simple Scale	0.0860	0.2716	0.1155	0.3476

As means over different data sets are hard to compare, every method was compared against every other using a paired t-test with confidence level $\alpha = 0.05$. The following table lists on how many data sets the method in the row was better than the method in the column. This table contains the results for the linear SVM and the mean squared error.

Name	Pl	Be	Is	Ga	Bi	So	Pr	BB	RP	RPM	Si
Platt Scaling	-	18	4	4	9	7	8	6	0	0	2
Beta Scaling	0	-	0	0	0	0	0	0	0	0	0
Isotonic Reg.	3	18	-	2	8	7	6	4	2	1	1
Gauss	6	18	5	-	12	7	9	6	6	4	6
Binning	2	18	1	2	-	1	5	1	1	1	1
Softmax	4	18	6	3	9	-	7	4	4	4	0
Precision	3	18	2	2	7	5	-	6	3	3	0
Binary Beta	5	18	5	5	9	5	8	-	3	1	1
Robust Platt	11	18	5	5	9	7	9	6	-	0	3
Rob. Platt Med	9	18	8	4	12	10	8	7	7	-	4
Simple Scale	5	18	6	5	11	10	11	7	4	3	-

The results for the linear SVM with the cross-entropy loss:

Name	Pl	Be	Is	Ga	Bi	So	Pr	BB	RP	RPM	Si
Platt Scaling	-	18	3	5	8	9	8	6	0	0	3
Beta Scaling	0	-	0	0	0	0	0	0	0	0	0
Isotonic Reg.	4	18	-	3	6	9	7	4	2	1	2
Gauss	9	18	7	-	8	8	10	8	9	7	7
Binning	2	18	1	2	-	5	4	2	1	0	1
Softmax	3	18	2	2	5	-	5	3	3	3	0
Precision	3	18	2	1	4	5	-	5	3	2	1
Binary Beta	5	18	3	7	5	10	8	-	4	1	2
Robust Platt	11	18	4	6	8	10	9	6	-	0	3
Rob. Platt Med	9	18	4	6	10	10	9	6	7	-	3
Simple Scale	7	18	5	5	10	9	12	7	5	4	-

The results for the radial basis SVM with mean squared error:

Name	Pl	Be	Is	Ga	Bi	So	Pr	BB	RP	RPM	Si
Platt Scaling	-	18	3	5	8	13	13	11	0	1	10
Beta Scaling	0	-	0	0	0	0	0	0	0	0	0
Isotonic Reg.	2	18	-	3	4	11	12	7	2	0	5
Gauss	0	18	3	-	4	11	13	7	0	0	8
Binning	0	18	0	0	-	9	11	5	0	0	5
Softmax	0	18	0	0	1	-	7	0	0	0	6
Precision	0	18	0	0	1	6	-	5	0	0	2
Binary Beta	0	18	0	1	4	9	9	-	0	0	6
Robust Platt	4	18	3	5	8	13	13	11	-	1	10
Rob. Platt Med	4	18	2	6	7	14	13	9	2	-	9
Simple Scale	1	18	0	2	5	6	12	4	1	1	-

The results for the radial basis SVM with the cross-entropy loss:

Name	Pl	Be	Is	Ga	Bi	So	Pr	BB	RP	RPM	Si
Platt Scaling	-	18	3	7	5	13	17	11	3	0	10
Beta Scaling	0	-	0	0	0	0	0	0	0	0	0
Isotonic Reg.	1	18	-	6	2	12	15	8	1	1	4
Gauss	1	18	2	-	3	9	13	6	1	1	7
Binning	0	18	0	4	-	10	10	6	0	0	6
Softmax	0	18	0	2	0	-	7	0	0	0	5
Precision	0	18	0	1	0	5	-	4	0	0	2
Binary Beta	0	18	0	3	0	10	10	-	0	0	6
Robust Platt	5	18	4	7	5	13	17	11	-	0	10
Rob. Platt Med	5	18	2	8	4	13	16	10	5	-	9
Simple Scale	1	18	0	3	3	9	14	7	0	0	-

The final table sums up the rows of each of the last four tables, that is, it lists the number of times the method in the row was better than some other method on some data set.

Method	linear		RBF	
	MSE	CRE	MSE	CRE
Platt Scaling	58	60	82	87
Beta Scaling	0	0	0	0
Isotonic Reg.	52	56	64	68
Gauss	79	91	64	61
Binning	33	36	48	54
Softmax	59	44	32	32
Precision	49	44	32	30
Binary Beta	60	63	47	47
Robust Platt	73	75	86	90
Rob. Platt Med	87	82	84	90
Simple Scale	80	82	50	55

Beta Scaling clearly performs worst. This is due to the fact that in Beta Scaling both classes are handled independently of another. When both classes are combined in binary Beta Scaling, the performance becomes comparable to the other methods. The best performance for the linear SVM and the MSE is the new Robust Platt Medium approach, followed by Simple Scaling. For the linear SVM and the cross-entropy, Gaussian Scaling is best with the Robust Platt Medium and Simple Scaling both second. For the radial basis SVM and both error measures, the robust Platt scaling performs best with Robust Platt Medium a close second.

In conclusion, the Robust Platt Medium performs consistently good over all data sets and error measures. A big surprise is the good performance of the Gaussian Model and the Simple Scaler, which have a good performance in particular for the linear Support Vector Machine with the lowest computational efforts.

Chapter 3

Optimizing the Global Model

This chapter deals with the optimization of a classification model in order to improve interpretability. Rather than optimizing models of a specific learner or constructing a new learning algorithm, the goal of this chapter is to investigate techniques to optimize interpretability for any learner, regardless of its functionality and with no or very limited assumptions about the inner structure of the model. The maximization of interpretability is based on the heuristics explained in Section 1.1.2.

First, Section 3.1 will deal with feature selection. Feature selection can be performed with any kind of classifier and aims at reducing the number of concepts that are necessary to describe a single observation. In particular, in Section 3.1.5 a new feature selection method will be presented that is especially targeted at the situations where feature selection matters most, namely large sets of observations, slow learning time and non-linear numerical classifiers.

A second very general approach is instance selection, which will be investigated in Section 3.2. The idea is targeted at bringing the explanation of the model close to the data by extracting a small set of examples that are representative of the data from the view of the classifier. It will be shown that a clustering algorithm can be employed to extract prototypical instances that carry information about the way the hypothesis space of the classifier and the examples labels structure the data space.

Section 3.3 deals with de-composing complex non-linear models into smaller, easier to understand linear local models. At the end of this chapter the question how the direct reduction of a technical complexity measure influences the accuracy and interpretability of a model will be investigated. Several techniques of complexity reduction exist in standard learning algorithms to fight over-fitting the data and to increase accuracy, but up to now there is no investigation of the effect of these techniques when the level of complexity of the learner is restricted to a higher extend than necessary for accuracy optimization. Section 3.4 presents an empirical investigation of this question.

3.1 Feature Selection

Feature selection [Liu and Motoda, 1998] is an important step in simplifying a classifier, because humans are very constrained in the number of mental concepts they can operate with. As usually the representation of the examples is designed driven from the application, a feature usually represents a well-defined real-world aspect of the data that is meaningful by itself. Hence, in reducing the number of features that are necessary in a classification rule, the number of concepts the user has to deal with when trying to understand the rule is minimized. For example, in an exemplary multimedia application that will be described in Section 6.3, feature selection will be used to extract from a classifier over 44 audio features and 508 text features the 10 simple keywords GROOVE, SMOOTH, CHILL, JAZZY, MOOD, FUSION, PIANO, PIECE, PAUL, and JAZZ, which are obviously sufficient to describe the musical taste the classifier has learned to predict.

An empirical investigation in [Mladenic et al., 2004] has also shown that feature selection with respect to one algorithm (in this case linear SVM weights) also works well with other learning algorithms (here Naive Bayes and the Perceptron). This seems to indicate that feature selection does not need to depend strongly on a single learning algorithm, but does capture the general importance of features.

Although the impact of feature selection on interpretability is obvious, feature selection is usually motivated from the perspective of classification performance. From the No-Free-Lunch-Theorem [Wolpert and Macready, 1997] we know that on the average over all learning problems, any two learners achieve equal performance. Hence, a carefully selected representation of the learning problem is the key to better than average performance. The difference in selecting features for interpretability and for performance is that in order to achieve a sufficient level of interpretability the number of features may have to be reduced so much that the accuracy is negatively affected.

Finally, a carefully selected set of features not only maximizes classification accuracy, a minimal set of features may also be desirable to speed up the classifier. Many classifiers can efficiently deal with only a low number of features, making learning infeasible for many real-world learning tasks. While there are other learners such as Support Vector Machines that can deal with problems of very high dimension [Joachims, 1998], there may still be a problem to efficiently apply the model to data sets containing millions of observations, e. g. in online databases for the detection of credit card fraud. For example, the application of a SVM model to new observations can be implemented as pure SQL code inside a relational database [Rüping, 2002b, Rüping, 2002a], and in this case reducing the features (and hence the database operations) can result in a dramatic speedup of the classification process.

In general, feature selection is a very complex problem, as it not only concerns the characteristics of the data set, but also the geometry of the hypothesis space, the learning strategy and the influence of the learners parameters over input spaces of different dimensions. In particular, one cannot conclude that a superset of features always performs at least as good as any of its subsets, because estimating an optimal model from a fixed size data set becomes increasingly hard in higher dimensions. This is known as the Curse of Dimensionality [Bellman, 1961]. There are several results on the complexity of feature selection. For example, finding a subset of n features, such that no two examples have identical feature values and different class value, is NP-hard [Davies and Russell, 1994]. Even in the case of the seemingly more simple linear classifiers, minimizing the number of features of a separating linear classifier is NP-hard [Amaldi and Kann, 1998].

3.1.1 General Feature Selection Methods

In general, the only way to find an optimal set of features is to test all 2^d possible feature sets. As this is practically infeasible for all but very small numbers of features, one usually has to resort to a heuristic. Trivial heuristics are known as the filter approach, where criteria such as correlation of the feature with the label or mutual information of the feature and the label are used to select promising features. However, this approach does not account for the role the hypothesis space plays and for the dependencies between the features and consequently shows only suboptimal results.

A better but computationally more expensive idea is the wrapper approach [Kohavi and John, 1997, Kohavi and John, 1998], which describes the general idea of using the actual performance of the learner on the feature sets to guide the feature selection by repeatedly calling the learner with different sets of features.

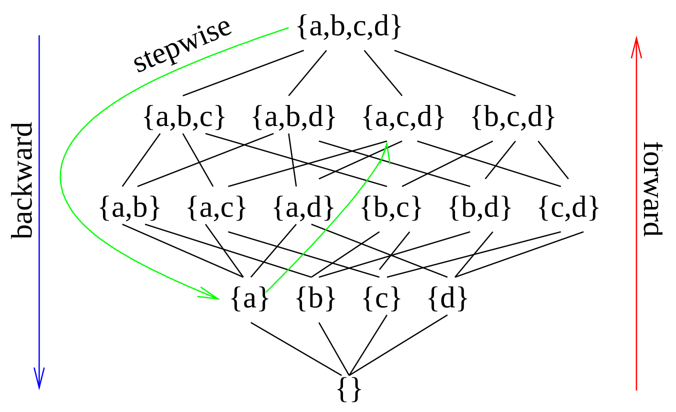


Figure 3.1: Backward selection, forward selection and stepwise selection.

One particular variant of the wrapper approach is backward selection, where one starts with the full set of features and then iteratively removes one feature as long as the error decreases. In each step, the feature to remove from the set of k available features is selected by testing the learner on all feature subsets of size $k-1$ and chosing the feature whose removal reduces classification performance the least. The order in which the features are removed gives a ranking of feature importance. The complementary approach of starting with the empty set of features and iteratively adding one feature is known as forward selection. Forward selection is computationally more simple, because it starts with smaller feature sets, but is likely to give worse results because reliable information about the importance of a feature requires a reasonably good model which is unlikely be found with too few features. A third variant is stepwise selection, which always iterates over all features and includes all features that increased the accuracy and removes all other features. Figure 3.1 shows how backward selection (blue), forward selection (red) and stepwise selection (green) navigate through the lattice of feature sets.

One can easily see that backward and forward selection require $\sum_{i=2}^{d} i = \Theta(d^2)$ runs of the learner. A shorter runtime can be achieved by a Wrapper-Filter-combination, where the error of the learner on the full feature set minus one feature is used as the weight of the feature (the higher the error when the feature is removed, the more important the feature is). That is, only the first iteration of backward selection is executed and used as performance measure for the feature filter. This results in $\Theta(d)$ runs of the learner.

A randomized approach for feature selection based on genetic algorithms has been proposed [Punch et al., 1993, Yang and Honovar, 1998]. However, genetic algorithms also need a high number of learning runs to evaluate the different feature sets. Hence, we exclude these approaches from our investigation for performance reasons.

A simple randomized feature selection can be based on resampling: this method generates a fixed number of feature sets and returns the one with best classification performance. The experiments will show that this method is surprisingly effective for a small number i of iterations, e. g. $i = \sqrt{d}$.

3.1.2 Feature Selection for Logical Classifiers

Most logical classifiers build up a rule step by step in order to exploit the structure given by generalization and specification of rule. For example, the rule $A \rightarrow C$ is more general than $A \& B \rightarrow C$ and hence one can conclude that if the first rule does not cover enough examples, neither will the second rule without further testing.

Therefore, logical classifiers implicitely contain a feature selection step in the core of the learning algorithm. For example, decision tree learning [Quinlan, 1986] can simply be described as recursively finding the best feature to split the current training set by. Similarly, the refinement step in covering rule learners is basically a feature selection step. In first-order logic, the search for the best features can additionally be guided by the user by a proper definition of the predicates and a sorted logic [Muggleton, 1992b, Morik et al., 1993].

In conclusion, feature selection is much less a problem for logical classifiers than it is for numerical classifiers. For this reason in the rest of this section we restrict the analysis to numerical classifiers.

3.1.3 Linear Feature Selection

For linear classifiers $f(x) = w * x + b$ the absolute value $|w|$ of the weight vector gives a canonical feature weight, given the features are scaled to identical variance beforehand (we will assume this for the rest of the paper). Indeed, one can see that several more complex feature selection methods for Support Vector Machines boil down to this approach [Guyon and Elisseeff, 2003]. This feature selection measure is particularly appealing as it requires to run the learner only once.

This method may not provide an optimal set of features because of the problem of feature correlation. Consider the case of d features $x_1, x_2, \ldots, x_d$

and let w be the weights of the optimal linear classifier. Now assume, we add a feature which is highly correlated with an existing feature. In the extreme case, we could add a copy of x_1 to the features. It is easy to see that for the new set of features $x_1, x_1, x_2, \ldots, x_d$, the weight vector

$$w_\lambda = (\lambda, w_1 - \lambda, w_2, \ldots, w_d)$$

describes the same linear classifier as w on the original features (with the same b) for all $\lambda \in \mathcal{R}$. We can further see that in the case of SVMs the complexity term $||w_\lambda||_2$ is minimized for $\lambda = w_1/2$. It follows that $(w_1/2, w_1/2, w_2, \ldots, w_d)$ is the optimal classifier for the larger feature set. Of course we can arbitrarily repeat the same process. In other words, we can make a feature weight arbitrarily small by simply adding highly correlated features.

Vice versa this means that the importance of a group of correlated features, of which at least one feature is needed to make a good prediction, is underestimated by this method. One can circumvent this problem by removing one feature at a time and re-training the classifier afterwards, or at least re-training the classifier after a certain number of variables are removed. One could also transform the data using principal components, forcing the features to be uncorrelated, but of course in terms of interpretability the transformed features offer no advantage over using all features at once.

The proof of the NP-hardness of minimizing the number of features of a separating linear classifier [Amaldi and Kann, 1998] shows that this is not only an effect of the 2-norm used in SVMs.

3.1.4 Feature Selection for Nonlinear Classifiers

Feature selection for general nonlinear classifiers is the hardest feature selection problem, as there is no global measure of an attribute's importance like the weight for a linear classifier or the number of times it was selected in the induced hypothesis for a logical classifier.

This section investigates feature selection for nonlinear Support Vector Machines. In spirit of the interpretability approach, the problem is defined to efficiently reduce a set of d features to a set of size $k << d$, where k is selected by the user. We approach this problem by constructing feature weights w_i, such that the feature set in question is given by the k highest weighted features (actually, we would only need a ranking of the features). This has the advantage that the user can try out different values of k without the need to start the feature selection procedure all over again.

A multitude of feature selection algorithms have already been proposed for Support Vector Machines [Guyon and Elisseeff, 2003]. Let us first survey a number of existing SVM-related feature selection algorithms.

Radius/Margin Bound Feature Selection

[Weston et al., 2000] propose a feature selection method based on the following proposition:

Theorem 3.1.1 (Radius/Margin Bound [Vapnik, 1998]). *Let $W(\alpha)$ be the target function of the SVM given by dual variables α, R be the radius of a sphere in feature space containing the training data and l be the size of the training* *set. If the training data is separated with positive margin, the expectation of the error probability over all training sets of size l is bounded by*

$$EP_{Err} \leq \frac{1}{l} R^2 W^2(\alpha).$$

This bound is called the radius/margin bound.

The idea is now to scale the features $x_1, \ldots, x_d$ by scaling factors $\sigma_1, \ldots, \sigma_d$ and minimize the radius/margin bound with respect to σ. The optimal factors σ are then used as weights of the features. Optimization is carried out by gradient descend algorithm. A SVM is trained with feature weights σ. For this SVM, the gradient of the radius/margin bound with respect to σ is computed and a gradient descend step is carried out to find new weights σ'. These steps are iterated until convergence.

The drawback of this approach is that is does not only require to learn an SVM at each gradient descend step, but also requires the solution of another quadratic programming problem to find the radius R of the data scaled with the new scaling factor.

Gradients of Performance Criteria

Rakotomamonjy [Rakotomamonjy, 2003] proposes to use the derivatives of the performance criteria for feature selection instead of the criteria themselves. The criteria he considers are the weight vector norm $||w||^2$, the radius/margin bound and the span bound [Vapnik and Chapelle, 2000]. Computing the gradients requires only one run of the SVM. In particular, the weight vector norm gradient was reported to perform consistently well in [Rakotomamonjy, 2003].

3.1.5 Large-Scale Non-Linear Feature Selection

The previously discussed feature selection methods for nonlinear classifiers all employ very computation intensive techniques. This section presents a new method for non-linear feature selection, which computes feature weights after only one run of the learning algorithm in a fashion similar to the successful approach of the weight vector for linear SVMs. This allows to efficiently employ this approach even for high-dimension datasets, which will not be possible for the slower more computation intensive methods.

In the spirit of linear feature weights different linear approximations to the SVM function

$$f(x) = \sum \alpha_i y_i K(x_i, x) + b$$

are used. That is, instead of using the gradient of some performance criterion as in Rakotomamonjy's method, we directly use the gradient $\nabla f(x)$ of the decision function. This gradient is evaluated at all training points $x \in M_f = \{x : |f(x)| \leq 1\}$, i. e. the set M_f contains the training points which fall on the border or inside of the margin. Note that M_f is a subset of the support vectors, excluding the bounded support vectors outside of the margin.

The intuition is that non-support vectors do not restrict the form of the decision function (removing them results in the same decision function) and hence, the form of the decision function and the gradient is somewhat arbitrary as long as $yf(x) > 1$. As a conclusion, these points are likely to not give reliable information about feature importance. Points exceeding the margin on the other side, i. e. points x with $yf(x) < -1$, disagree with the rest of the data so much that are likely to be classified wrong with every kind of feature selection, hence they are excluded as well.

Computing the gradient leaves us with several weights for each feature, one for each training point in M_f. To aggregate these weights into a single weight w_i for each feature, we proceed as follows: the weights for each feature i are sorted increasingly and the aggregated weight w_i is set to the weight at position $q \cdot |M_f|$ in the sorted sequence, where $q \in [0, 1]$ is a user-defined parameter. This accounts for the fact that features may have different importance in different parts of the input space. The meaning of w_i is that the i-th feature has an importance of at least w_i for a fraction of $1 - q$ of the examples in M_f. In the experiments, $q = 0.75$ was used.

For a more formal justification of this approach, note that the SVM decision function f can be written as

$$f(x) = w * \Phi(x) + b,$$

where Φ is a mapping that induces the kernel via $K(x, x') = \Phi(x) * \Phi(x')$ and w is the hyperplane normal vector in feature space. Putting differentiability considerations aside, the partial derivate of f can be written as

$$\partial_i f(x) = \partial_i (w * \Phi)(x) = w * \partial_i \Phi(x)$$

and hence

$$||\nabla f(x)|| \leq ||w|| \cdot ||\nabla \Phi(x)||.$$

As the gradient of Φ is fixed by the choice of the kernel, minimizing $||w||$ in the primal SVM problem leads to the minimization of the norm of the gradient of the SVM decision function. Hence, the SVM seeks to find the smoothest function that predicts the data. The control of the gradient shows that a high partial derivate in the direction of one feature at a point x can only be due to the fact that this feature is needed to correctly classify the points in the neighborhood of x.

In conclusion, the presented method could theoretically be applied to any classifier with derivable decision function, but promises particular good perfomance in the case of learners that find smooth function like the SVM.

Experiments

This section will validate the proposed algorithm in terms of classification performance. Following the discussion from the introduction, it would also be interesting to investigate how interpretability is affected by a specific choice of features. But as interpretability can hardly be quantified, this investigation will be restricted to directly measurable performance criteria, hoping that a high quality model with few features will somehow be interesting to the user.

A radial basis SVM was used as the nonlinear classifier. For each data set, three experiments of dimension reduction were performed. In the first experiment, the number of features was reduced from the full set of features to 75%, in the second run, 50% of the features were selected and in the final experiment, 25% of the features were selected. All reported results were obtained with 10-fold cross-validation.

The new quantile-of-gradients approach was compared against other feature selection methods that allow a fast computation, namely the backward selection filter (selecting features by only the first step of backward selection), the correlation filter (correlation between feature and class), the resampling-based feature selection (evaluating $\sqrt{d}$ random feature subsets) and Rakotomamonjy's method. More complex feature selection methods like full backward selection, iterative methods like the radius/margin bound feature selection or repeatedly removing a smaller subset of the features and re-training the classifier are likely to give better results, but are not included in the comparison, as this investigation is targeted at finding a feature selection methods for large high-dimensional data sets, where an iterative procedure is too slow.

The following table compares the complexity of several feature selection methods in term of the number of necessary calls to the learner.

Method	Complexity
Backward Selection	$O(d^2)$
1-step backward selection	d
Weston	#iterations
Resampling	$\sqrt{d}$
Rakotomamonjy	1
Quantile	1
Correlation	0

Preliminary experiments suggested that the parameter q of the gradient feature selection is best set to $q = 0.75$ over all data sets and dimensions.

The following table shows the accuracy when reducing the number of features to 3/4 of its original size. The first column contains the name of the data set, the second column the backward-selection-filter, the third column the correlation filter, the fourth column the resampling-based feature selection, the fifth column Rakotomamonjy's performance criteria gradients and the final column the new quantile of gradients approach.

Name	Backward	Correlation	Resampling	Perf. Grad.	Quantile
liver	0.710	0.608	0.710	0.686	0.684
diabetes	0.769	0.766	0.769	0.773	0.766
breast	0.969	0.967	0.966	0.970	0.969
business	0.872	0.840	0.834	0.822	0.854
wine	0.988	0.977	0.982	0.977	0.971
voting	0.965	0.951	0.960	0.956	0.965
medicine	0.803	0.811	0.811	0.811	0.813
dermatology	1.000	1.000	1.000	1.000	1.000
ionosphere	0.940	0.928	0.940	0.937	0.934
covtype	0.800	0.793	0.804	0.801	0.803
digits	0.993	0.996	0.997	0.998	0.997
physics	0.506	0.506	0.506	0.506	0.506
mushroom	0.994	1.000	0.999	0.999	0.999
insurance	0.998	0.993	0.998	0.998	0.997
promoters	0.886	0.896	0.933	0.933	0.924
garageband	0.720	0.726	0.723	0.723	0.709

The following table shows the accuracy when reducing the number of features to 1/2 of its original size. The meaning of the columns is the same as in the previous table.

Name	Backward	Correlation	Resamp.	Perf. Grad.	Quantile
liver	0.672	0.588	0.680	0.605	0.571
diabetes	0.761	0.764	0.708	0.748	0.766
breast	0.963	0.963	0.972	0.961	0.967
business	0.808	0.840	0.872	0.764	0.827
wine	0.971	0.982	0.988	0.971	0.909
voting	0.963	0.949	0.958	0.956	0.965
medicine	0.787	0.771	0.807	0.779	0.809
dermatology	1.000	1.000	1.000	1.000	1.000
ionosphere	0.937	0.920	0.937	0.931	0.923
covtype	0.791	0.784	0.801	0.794	0.807
digits	0.992	0.994	0.997	0.996	0.997
physics	0.506	0.506	0.506	0.506	0.506
mushroom	0.970	0.999	0.999	0.999	1.000
insurance	0.998	0.991	0.998	0.997	0.997
promoters	0.876	0.858	0.914	0.867	0.915
garageband	0.728	0.728	0.725	0.736	0.708

The last table shows the accuracy when reducing the number of features to 1/4 of its original size. For most data sets this is an extreme reduction with a significant reduction in accuracy.

Name	Backward	Correlation	Resamp.	Perf. Grad.	Quantile
liver	0.632	0.579	0.727	0.576	0.579
diabetes	0.755	0.759	0.744	0.742	0.746
breast	0.931	0.948	0.950	0.937	0.947
business	0.739	0.827	0.802	0.751	0.770
wine	0.938	0.932	0.966	0.887	0.898
voting	0.958	0.956	0.919	0.956	0.967
medicine	0.735	0.721	0.790	0.759	0.722
dermatology	0.972	0.994	1.000	0.994	0.994
ionosphere	0.905	0.888	0.928	0.911	0.840
covtype	0.791	0.764	0.801	0.791	0.795
digits	0.983	0.992	0.994	0.993	0.998
physics	0.506	0.506	0.506	0.506	0.506
mushroom	0.965	0.981	0.999	0.999	0.999
insurance	0.997	0.985	0.998	0.998	0.997
promoters	0.810	0.934	0.914	0.897	0.896
garageband	0.712	0.731	0.725	0.735	0.709

To better compare the performance of these algorithms, a paired t-test with significance level $\alpha = 0.05$ was conducted for each data set and each fraction of features $(3/4, 1/2, 1/4)$ to check if one algorithm performs significantly better than another. The following table contains the total number of a signifcantly higher accuracy. Each entry in the table represents the number of times that the method in the row had a significantly higher accuracy than the method in the column.

Name	Backward	Correlation	Resamp.	Perf. Grad	Quantile
Backward	-	7	0	2	4
Correlation	4	-	0	4	2
Resampling	13	14	-	6	6
Perf. Grad	7	7	0	-	4
Quantile	8	10	1	7	-

It can be seen that the resampling method performs best, with 39 wins over other methods. The quantile method comes second with 26 wins, followed by the performance criteria gradients with 18 wins and backward selection with 13 wins. Correlation based feature selection ranks last with only 10 wins, which is not surprising as correlation is essentially a measure of linear dependence and hence is not suited for nonlinear methods.

In the case of large-scale high-dimensional classifiers, where only one learning step is feasible, the new quantile outperforms all other feature selection methods by combining high accuracy with computational efficiency.

3.2 Instance Selection

Instance selection describes the problem of selecting a small set of highly informative instances from a larger set of examples [Liu and Motoda, 2001]. Informative examples can be divided into protoypes and discriminating instances. Prototypes are examples which are similar to a large number of examples and can hence be taken as a typical representative of this set of examples. Discriminating instances are examples which lie close to the class border and are hence representative of the distinction between the classes.

In the context of interpretability instances are useful because their meaning is easier for a domain expert to understand than abstract models and decision rules. The idea is to describe a model by the examples it regards as important. Hence, instance selection in this context is targeted at extracting an extremely small set of examples, possibly less than 10. With more selected instances the user runs into the risk of not seeing the wood for the trees.

A conceptual problem of instance selection in a classification setting is a missing general measure of instance importance. The problem is that the importance of an example should not reflect the information it has about other examples per se, but the information from the view of the learner. Hence, the definition of similarity of examples should reflect the geometry of the example space that the hypothesis space of the learner induces. If, for example, the learner cannot distinguish between two examples, these examples should be considered similar.

Several ad-hoc solutions to instance selection in the context of specific learners exist. For example, k-medoids clustering can be seen as the explicit search for prototypes that represent the structure of the data. But of course this structure is not optimized with respect to the class distribution. The support vectors in SVMs are discriminating instances and it can be shown that the SVM trained on the support vectors alone is identical to the SVM on the complete data set. Usually only a small fraction of the data points become support vectors, but in absolute numbers this can still be several hundreds or thousands of examples.

3.2.1 Instance Selection by Data Squashing

Data Squashing [DuMouchel et al., 1999] is a technique to reduce large datasets by constructing a set of instances with identical statistical probabilities as the original data set. Squashing consists of three steps. The data set is first partitioned into groups, within each group a number of statistical properties are computed and then pseudo-data is generated that accurately reproduces these statistical properties.

An approach to instance selection based on data squashing is presented in [Madigan et al., 2002] Assuming the optimal model for the data (x_i, y_i) is represented by a parameter $\theta \in \mathbb{R}^d$, the idea is to inspect a set of similar models $\theta_j = \theta + \delta_j$ and inspect the conditional probabilities $p_{ij} = P_{\theta_j}(y_i|x_i)$ that these models assign to the examples. The number of different models tested and the size of the deviations δ_j are given as a parameter to the method.

The vector $p_{i\cdot}$ represents the statistical properties the squashed data set should reproduce. If for two examples i and i' all models assign similar probabilities, the learner does not disinguish between these examples and they should be treated as similar. Hence, the examples (x_i, y_i) are only represented by their probability "footprint" $p_{i\cdot}$ and instance selection is performed as a k-medoids clustering of the $p_{i\cdot}$.

3.2.2 Extending Instance Selection by Data Squashing

Instance selection by data squashing is restricted to models that can be represented by a real vector θ. More sophisticated learners employ more complex structured models such as decision trees, rule sets or sets of support vectors whose size is not fixed. In this case, one can not readily construct new models θ_j and hence existing squashing methods can not be applied here.

In principle, the described squashing methods are white-box approaches. To convert instance selection by squashing to a black-box approach, and hence to a very general class of models, it is important to notice that the only general way to modify the hypothesis returned by a learner without knowledge of the learners internals and the hypothesis space is to modify the training set. Further, as we do not want an arbitrary hypothesis but one that structures the training examples and has a complexity similar to the original hypothesis used to predict the data, the training set should not be modified (the modification of examples would change the distribution $P(x)$ and the removal of examples could reduce the complexity of the induced classifier, e. g. in decision trees). This leaves the modification of the examples labels y_i as the only available way to generate additional useful hypotheses.

A new distribution of the labels y_i cannot be generated randomly, because then with high probability there will be no structure in the examples for the classifier to fit. The idea of the following algorithm is that the clustering should contain maximal information about the class label. If a number of classifiers is already known, an additional classifier will increase the information about the examples the most if it is independent of the other classifiers. In the algorithm, this independence is approximated by letting the new classifier predict where the prior classifier has the most problems fitting the data in terms of the estimated conditional class probability $P(y|x)$. This induces independent classifiers because if the previous classifier would have this information, it would use it to

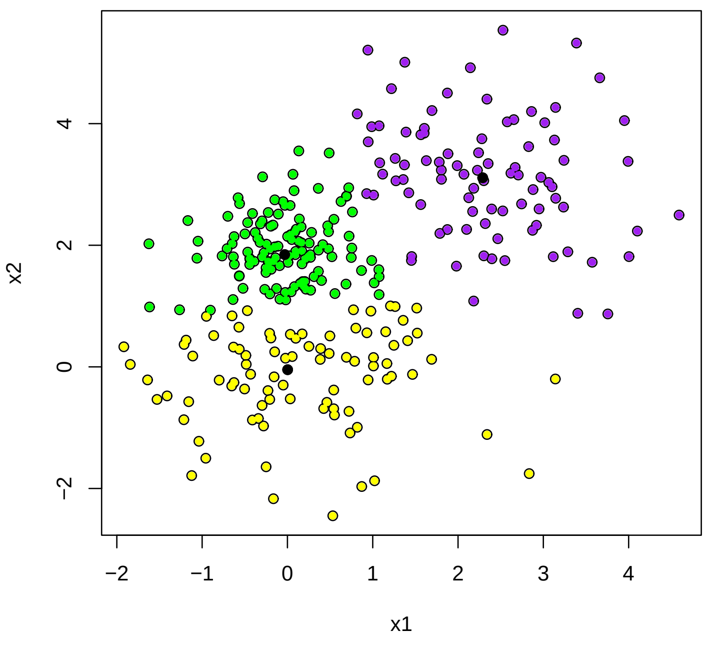


Figure 3.2: Instance selection for a linear classifier.

improve its prediction.

1. Input: Examples $(x_j, y_j)_{j \in 1 \ldots n}$, the desired number of clusters k, the number of iterations it

2. Learn a probabilistic classifier f_1 on $(x_j, y_j)_{j \in 1 \ldots n}$

3. Let p_1 be the vector of conditional class probabilites $P_1(Y = 1|x_j)$ as estimated by f_1

4. Compute a clustering C_1 with each observation represented by its conditional class probability

5. For i in $2 \ldots it$

   (a) Let $y'_j = 1$ for the 50% examples with highest $P_{i-1}(y_j|x_j)$ and $y'_j = -1$ else

   (b) Learn probabilistic classifier f_i on $(x_j, y'_j)_{j \in 1 \ldots n}$

   (c) Let p_i be the vector of conditional class probabilites $P_i(Y = 1|x_j)$ as estimated by f_i

   (d) Compute a clustering C_i with each observation represented by its i conditional class probabilities from the vectors $p_1, \ldots, p_i$

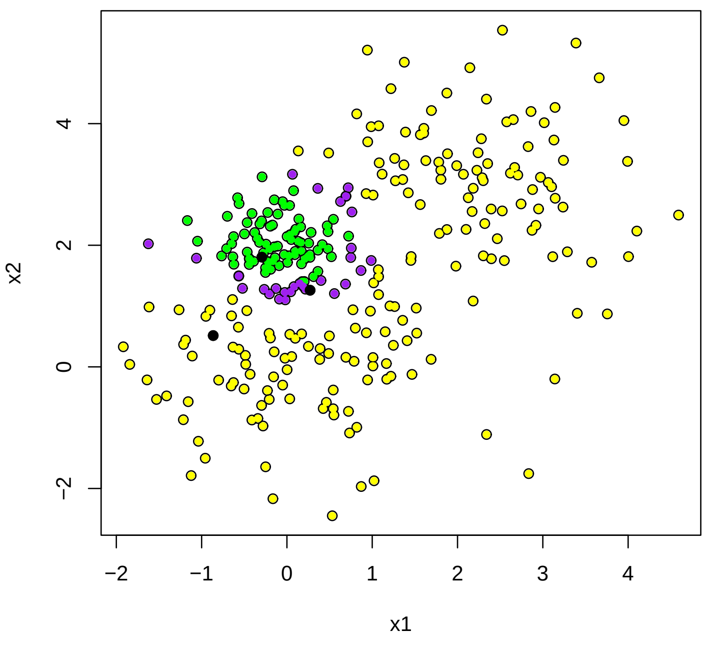


Figure 3.3: Instance selection for a nonlinear classifier.

6. Return the clustering $C_j, j \in \{1,\ldots,i\}$ with highest information about the class label.

The theoretical background of the algorithm is as follows: under the assumption that the clusterer is correct, the selected instances from the clusters will be optimal, if the constructed hypotheses f_i contain maximal information about the examples. Viewing the examples as random variables $Z = (X, Y)$ and the hypotheses as random variables $F_i = f(X_i)$, this can be formulated as

$$I(Z; (F_1, \ldots, F_i)) \rightarrow \max$$

where I is the mutual information [Cover, 1991]. If the hypotheses $\vec{F} := F_1, \ldots, F_{i-1}$ are fixed, it follows from the equation

$$I(Z; (F_1, \ldots, F_i)) = I(Z; F_i|\vec{F}) + I(Z; \vec{F})$$

that the optimal additional hypothesis F_i maximizes the information $I(Z; F_i|\vec{F})$ about the examples Z given that the values of the prior hypotheses are already known (note that this greedy selection of hypotheses F_i is not necessarily optimal, an optimal strategy would require to select all hypotheses in dependence on all other).

The mutual information can be further decomposed into

$$I(Z; F_i|\vec{F}) = H(F_i|\vec{F}) - H(F_|\vec{F}, Z)$$

where H is the entropy. Hence, a optimal new hypothesis F_i should maximize the entropy given $\vec{F}$ (i.e. F_i and $\vec{F}$ should be independent), while it should minimize the entropy of F_i given $\vec{F}$ and Z (i.e. should depend on $\vec{F}$ and Z).

Using the approximation that only the last hypothesis F_{i-1} is considered instead of all hypotheses $\vec{F}$, this maximal conditional information criterion is implemented in the training set construction of the instance selection algorithm. The independence of F_i and F_{i-1} is guaranteed by the fact that the task of F_i is to predict, whether F_{i-1} will give a high likelihood $P_{i-1}(y_j|x_j)$ or not. If there was a dependency between this information and F_{i-1}, the classifier could have used this information to select a better hypothesis (this assumes that the classifier selects the optimal hypothesis). The dependence of F_i on $\vec{F}$ and Z is obvious, as F_i is constructed using exactly this information as input.

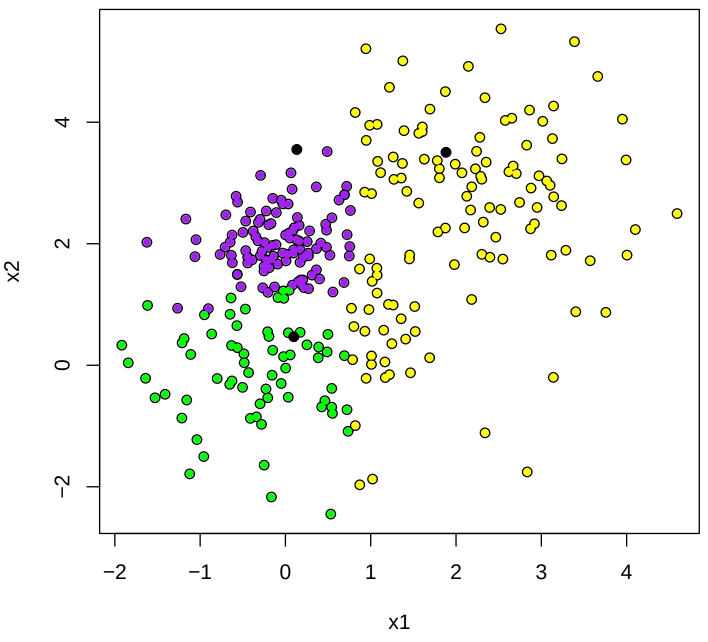


Figure 3.4: Instance selection for a logical classifier.

The figures show the instances and clusters generated by this algorithm on a 2-dimensional data set. The data set consists of three Gaussian distributions centered at $(0,0)$, $(0,2)$ and $(2,3)$ with different covariances. The label was set to $y = 1$ iff the observation came from the distribution with mean $(0,2)$. Figure 3.2 shows the result of instance selection with a linear SVM classifier. It can be seen that this induces linear bounds between each two clusters, but no cluster can be described by a single linear rule alone. The selected instances (cluster centers) are at the centers of the distributions. In Figure 3.3, a nonlinear radial basis SVM has induced nonlinear bounds on the clusters. As the nonlinear decision function effectively covers the class distribution, the three clusters cover different values of $P(Y = 1|x)$ and very little other information about the distribution of the examples. This can be seen in the selected observations, which do not reflect the distribution $P(x)$ at all. Figure 3.4 shows the situation with a decision tree as base classifier. It can be seen that this induces boundaries parallel to the coordinate axes between each two clusters. Further, the selected instances of clusters 1 and 2 lie on a line parallel to a coordinate axis, just as the center instances of clusters 2 and 3. This shows how the axis parallel decision rules of the decision tree influence the geometry on which clustering and instance selection works.

In summary, it can be seen that the method of classifier-dependent instance squashing extracts more meaningful instances and clusters than usual clustering because it reflects the information about the hypothesis space of the classifier.

3.3 Piecewise Linear Approximations

Linear function are usually much more interpretable than nonlinear functions, because they assign a unique weight to each attribute and attributes usually carry a well-defined meaning in terms of the application. Although this easy interpretation has its loopholes – correlations between attributes can make an attribute seem to appear more or less important than it actually is – this is still a clear advantage compared to nonlinear functions, where a global measure of an attribute's importance is hard to find. On the other hand, several data sets are intrinsically nonlinear, and it is clear that an easy-to-understand model is useless if it is wrong.

In some cases, there is a way to retain nonlinearity without loosing to much of the interpretability advantages of linear classifiers. Look at the data set shown in Figure 3.5. The class distribution is obviously nonlinear – the positive points form a square – but it is clear that we can easily describe the class boundary by not one but four linear functions, one for each side of the rectangle. Hence, a local linear model, consisting of several linear models plus a decision rule where to apply each model, is a simple and correct solution to this classification problem.

The idea of piecewise linearity itself is not new. It can be traced back to the definition of manifolds in mathematics, which are locally similar to the Euclidean space $\mathbb{R}^d$ for some d (with an appropriate mathematical definition of "local" and "similar", of course). This principle has been used in machine learning under the name of local linear embedding [Roweis and Saul, 2000] for dimensionality reduction. Local linear classifiers appear naturally in logical hypothesis space, where the necessary discretization of numerical values naturally introduces linear functions paralles to the coordinate axes (e.g. the decision function in Figure 3.5 could be described as IF X1 $\leq$ 1 AND X1 $\geq$ -1 AND X2 $\leq$ 1 AND X2 $\geq$ -1 THEN CLASS = BLUE). Local linear functions also naturally arise on classifiers based on the minimal distance to prototypes, such as k-nearest-neighbor classification and learning vector quantization [Kohonen et al., 1992]. These classifiers assign a test point to the prototype with minimal distance and hence induce a Voronoi tesselation on the input space, which has linear borders. However, in these approaches the linear decision functions are only defined implicitely and hence cannot easily be interpreted.

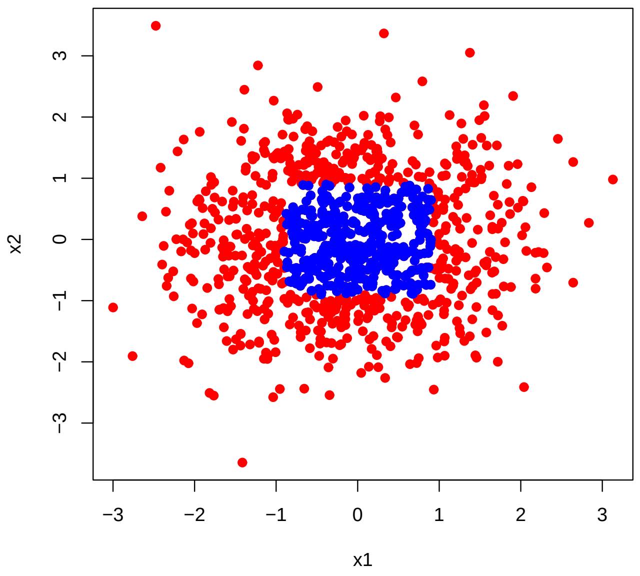


Figure 3.5: A piecewise linear classification problem.

We will now introduce an algorithm which approximates a nonlinear function by constructing local linear approximations. The main idea is that given a function f, the gradient $\nabla f(x_0)$ of f at a point x_0 induces a linear function $g(x) = \nabla f(x_0) * (x - x_0) + f(x_0)$. This is the Taylor approximation of f at x_0 of order 1 and by Taylor's Theorem we know that each continuously differentiable function f can locally approximated by its Taylor approximation. Now, given a training set $(x_i, y_i)_{i=1\ldots n}$ and a continuously differentiable nonlinear function f, e.g. the decision function of a radial basis SVM, we evaluate its gradient at every training point, which gives us n linear functions g_i. We could also use other points to obtain a better description of the decision function, but it is obviously meaningless to approximate the function at points that are not likely to appear. By restricting the starting points to the training set we implicitely weight the approximation quality at a point x with its probability $P_{emp}(x)$.

Given the functions g_i we cluster them into k clusters. Each cluster should represent one linear component and hence we have to control the clustering such that similar functions will fall into the same cluster. It is important to notice that we are not interested in the classification function g itself but in the class partitioning $\{x|g(x) > 0\}$ it induces, because this is what controls the classification performance. Hence, any function that induces the same partitioning, notably all αg with $\alpha > 0$, should fall into the same cluster. The same is true for functions which only differ on regions of the input space with low probability $P(x)$, because if the number of points on which these functions differ is very small, their performance will be similar. Hence, we construct a similarity measure $\sigma(g, h)$ between functions g and h defined as

$$\sigma(g, h) \quad = \quad \frac{1}{n} \# \{i | g(x_i) h(x_i) > 0\} .$$

That is, $\sigma(g, h)$ is the fraction of training points that g and h agree on. Obviously, σ formalizes the previously discussed concept of function similarity. We use a standard k-medoids clusterer with this similarity measure to obtain k clusters and hence k linear functions. In the experiments, the clusterer pam described in [Kaufman and Rousseeuw, 1990], Ch. 2 is used. To decide which of the k functions to apply to a test point x we use the training points x_i corresponding to the cluster medoids g_i as prototypes and assign the test point to the cluster with the nearest median x_i. Figure 3.6 shows the resulting clustering of the training set from Figure 3.5, where the initial function was a radial basis Support Vector Machine. Note that the data set is best described by five linear functions, one for each side of the square and another for all points far away from the positive class. The latter cluster corresponds to the default negative classifier and is induced from the training points with zero gradient; obviously this cluster could be removed as the nearest cluster from the square also classifies these points as negative. Also notice that the four inner clusters are not identical to the positive class! These four clusters cover more than the positive examples, which are indicated by the black square in the middle, but the decision boundaries from each class correspond to the sides of the square.

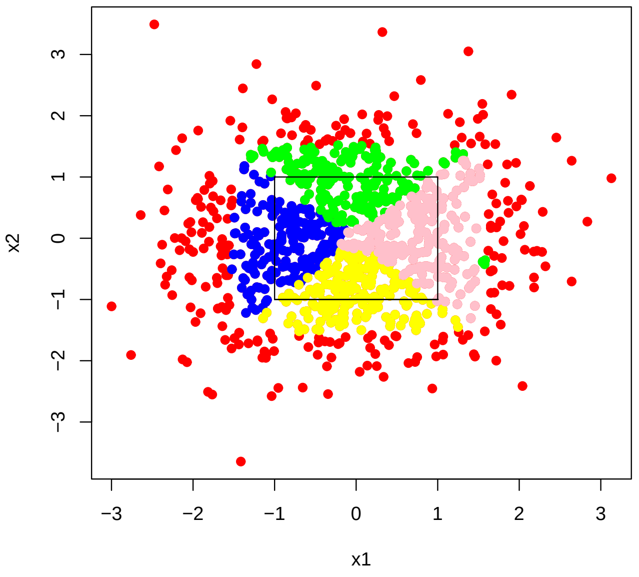


Figure 3.6: A piecewise linear classifier.

In principle, this approach can be used for any kind of numerical decision function. However, Support Vector Machines are specifically well-suited for the following reason: Note that Taylor's Theorem guarantees that the function f can be approximated locally by its gradient, but it does not give any information about the size of this local neighborhood (the approximation quality is only described in terms of limits). In general, the more complex the function f is, the smaller the neighborhood where a sensible approximation is possible. Support Vector Machines are specifically well suited because they explicitely penalize function complexity in terms of regularizer $||w||$. Note that the SVM decision function can be written in the form

$$f(x) = w * \Phi(x) + b$$

where Φ is the nonlinear mapping in the Reproducing Kernel Hilbert Space. Hence, the gradient of f is of the form

$$\nabla f(x) = w * \nabla\Phi(x)$$

and hence the norm of the gradient is bounded by

$$||\nabla f(x)|| \leq ||w||||\nabla\Phi(x)||.$$

As the gradient of Φ at the training points is constant for a given kernel and a given training set, minimizing $||w||$ is equivalent to minimizing the gradient of f. Hence, locally a good approximation by the gradient of f can be expected.

This algorithm was tested on the 18 standard data sets with the radial basis SVM as the original classifier. Values of k between 1 and 10 were tested. The following table shows the results. The first column contains the name of the data set and the second column it dimension. The next two columns show the classification error of the standard radial basis and the linear SVM, respectively. The next two columns show the error achieved by the piecewise linear approximation and the corresponding optimal number of clusters k, while the last two columns contain the disagreement rate between the radial basis SVM and its approximation, again with the corresponding k.

Name	Dim.	RBF SVM	lin. SVM	piecewise linear			
		Error	Error	Error	k	Disagree	k
iris	4	0.000	0.000	0.000	2	0.000	2
balance	4	0.014	0.052	0.028	10	0.014	10
liver	6	0.303	0.306	0.265	8	0.066	10
diabetes	8	0.262	0.252	0.246	6	0.060	10
breast	9	0.034	0.032	0.030	6	0.008	6
business	13	0.127	0.139	0.126	4	0.108	7
wine	13	0.225	0.011	0.225	2	0.011	1
voting	16	0.066	0.043	0.069	2	0.0023	2
medicine	18	0.278	0.278	0.278	1	0.000	1
dermatology	33	0.021	0.005	0.000	5	0.016	2
ionosphere	34	0.065	0.131	0.171	10	0.111	10
covtype	48	0.284	0.280	0.286	7	0.158	10
digits	64	0.008	0.004	0.004	1	0.002	4
physics	78	0.378	0.360	0.384	4	0.140	9
mushroom	126	0.074	0.002	0.450	1	0.384	1
insurance	134	0.070	0.024	0.068	1	0.002	1
promoters	228	0.123	0.055	0.460	2	0.403	1
garageband	552	0.329	0.349	0.315	1	0.137	1

At 11 of the 18 data sets, the piecewise linear approximation achieves a better or equal error rate compared to the radial basis function it approximates. Further, the error rate does not exceed the radial basis SVMs error by more than 0.01 on 14 of the 18 data sets. Given the far superior interpretability of a piecewise linear function compared to a linear combination of radial basis functions, the approximation should be preferable on these data sets as well. The error of the approximation is significantly higher than that of the approximated function on 4 data sets (balance, ionosphere, mushroom, and promoters), with an almost random prediction for the mushroom and the promoters data set. One can see that these two data sets are the high dimensional data sets where the radial basis SVM has a significantly higher error than the linear SVM, which probably means both classifiers significantly disagree and no transfer between a nonlinear and a linear function can be made.

Comparing the approximation ability of the piecewise linear function one can see that the best approximation (lowest disagreement rate) is found at a higher number of components on 6 data sets, an equal number of components at 9 data sets and a lower number of components at 3 data sets. It is not surprising that a higher number of components can increase the approximation quality, as points will move closer and closer to their nearest median. However, as can be seen this also increases the risk of overfitting and of course limits the interpretability.

3.4 Direct Complexity Reduction

To conclude the investigation of general methods for enhancing interpretability, this section presents an investigation of the relation between formal complexity measures, interpretability and accuracy. Formal complexity measure like the size of a decision tree or the number of features used by a classifier are easy to measure and to reduce and often show a direct relationship to interpretability heuristics. On the other hand, these complexity measures are primarily used to improve the accuracy of learners by avoiding over-fitting, not to optimize interpretability.

The question that is answered in the section is how the error of a classifier develops when the complexity is restricted far more than what is necessary for accuracy optimization. If there is a simple functional relationship between the complexity measure and the error, e.g. a linear dependency, the user can easily select a complexity value that optimally trades off the conflicting goals of accuracy and understandability. If this relationship is very complex, that is if it is hard to predict which complexity value will give which error, finding an optimal value will require a high computational effort (e.g. a high number of runs of the classifier), such that in practice this approach of interpretability optimization becomes infeasible. The minimal requirement is that the functional dependency is reasonably smooth, such that one can be sure that the results are not mainly a result of random effects and that the desired level of accuracy will also be met on future instance.

In short, the goal of the chapter is to investigate whether formal complexity measures enable the user to reliably trade off his requirements of accuracy against understandability.

3.4.1 Decision Tree Size

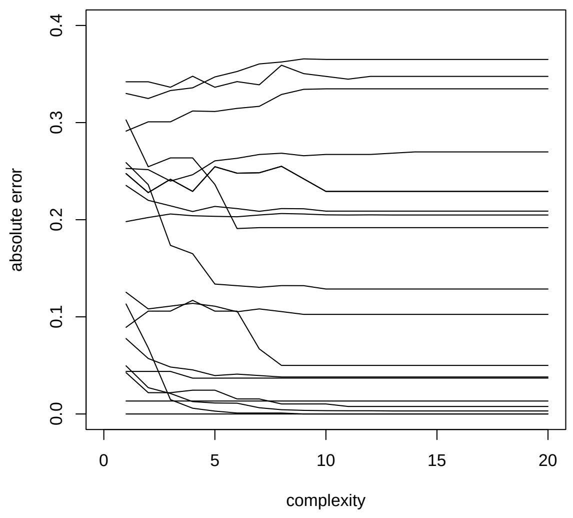


Figure 3.7: Absolute error of constrained decision tree

Decision Tree learning consists of a learning phase, which constructs the tree, and a pruning phase which removes some subtrees in order to avoid overfitting. It is straight-forward to extend the pruning phase to remove nodes from the tree until a user-defined threshold on the size of the decision tree is met.

The impact of limiting decision tree size on interpretability is obvious, but how does the prediction error behave? In order to answer this question, an experiment was conducted where the complexity is restricted to a certain level. In total 20 different complexity levels were investigated, where level i corresponds to a size restriction of $i/20$ of the nodes in the unrestricted tree. For each level, 10-fold cross-validation was performed. Figure 3.7 shows the performance curve of the 18 standard data sets (each data set one line) in dependency on the complexity. Figure 3.8 shows the error in relation to the error of the tree without size restriction on a logarithmic scale.

Several observation can be made. First, in all cases the complexity can be limited to at least half of the nodes without hurting the performance. Second, on several of the data sets the optimal decision tree is the most constrained one! Third, for other decision trees the error increases sharply, sometimes several orders of magnitude, if the complexity drops below a certain value.

The result of this investigation is that decision tree pruning is not optimal with respect to interpretability. On all data sets, considerably smaller decision trees could be obtained without hurting the classification performance. On the other hand, no simple functional form that describes the relationship between tree size and error could be obtained. Very small tree sizes can result in very different relative performance values. This makes it hard to balance accuracy

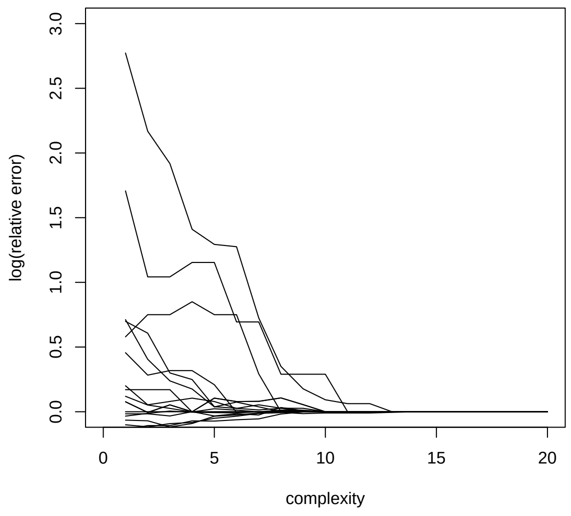


Figure 3.8: Relative error of constrained decision tree

and interpretability without testing out all different complexity values.

3.4.2 Feature Selection

Feature selection and its implications to interpretability have been discussed in Section 3.1, which discussed how to reduce the number of features to a predefined level. This section will deal with the question of how to find the optimal number of features. The discussion will be limited to backward feature selection (see Section 3.1.1), as it is currently the most popular feature selection method.

In the experiments, the number of features are fixed at different fractions of the total number of features. For each number, 10-fold cross-validation was performed. Figure 3.9 shows the performance curve with the linear SVM as base learner on the 18 standard data sets (each data set one line). The performance curve of the radial basis SVM (Figure 3.10), decision trees (Figure 3.11) and the J48 covering rule learner (Figure 3.12) are also shown.

On most data sets, the error increases linearly with decreasing number of features until a certain threshold is reached, below which it increases significantly. This is different from the results of limiting decision tree size in the previous section, which had a constant error for high complexity values. The results are similar for all learners.

The linear dependency for not too small numbers of feature allows a more fine-grained control of the trade-off between accuracy and complexity than with decision tree sizes. On the other hand, feature selection is a hard problem (see Section 3.1) and optimizing the number of features still needs several runs of the learning algorithm.

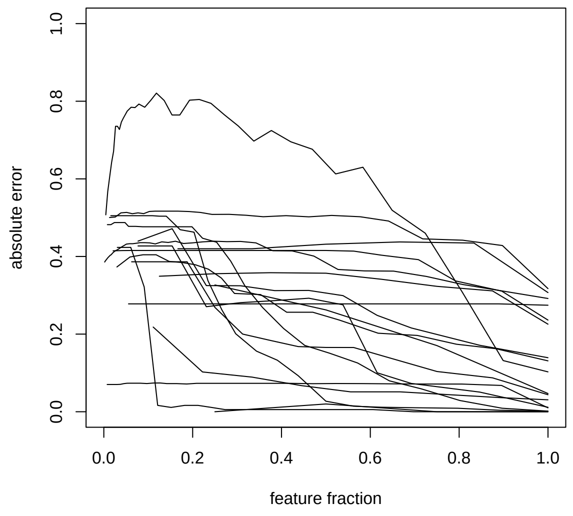


Figure 3.9: Absolute error of feature selection with linear SVM

3.4.3 VC-dimension

Support Vector Machines directly balance a complexity criterion based on the VC-dimension against the error on the data (see Section 2.1.3). Using the parameter C this trade-off can be regulated by the user. But how does this formal measure of complexity relate to user-centered measures of complexity?

To answer this question, several experiments have been conducted to show the relationship between C and the SVM error and a user-centered measure of complexity. For linear SVMs, the sparsity of the hyperplane vector, i. e. the number of its zero components, is such a user-centered measure. It describes the number of features that are not used in the classification.

Figure 3.13 shows the absolute error of a linear SVM over the range of sensible values for C for each data set (each line one data set). Figure 3.14 show the corresponding sparsity of the SVM hyperplane vector. It can be seen that only values in a certain interval give sensible predictions, outside this range the error increases significantly. A comparison of the two figures reveals that on this interval of sensible values, sparsity is very limited and almost constant in most cases. Only for very small values of C sparsity increases up to 1, which means a zero hyperplane vector (constant decision function).

Figure 3.15 shows the absolute error of a radial basis SVM over the same range of C values. As there is no explicit hyperplane vector for nonlinear SVMs, the number of support vectors is used as complexity measure. Figure 3.16 shows the corresponding number of support vectors in terms of the fraction of the number of support vectors to number of training examples. It can be seen that the number of support vectors increases with smaller C.

It can be concluded that the formal complexity measure of VC-dimension is

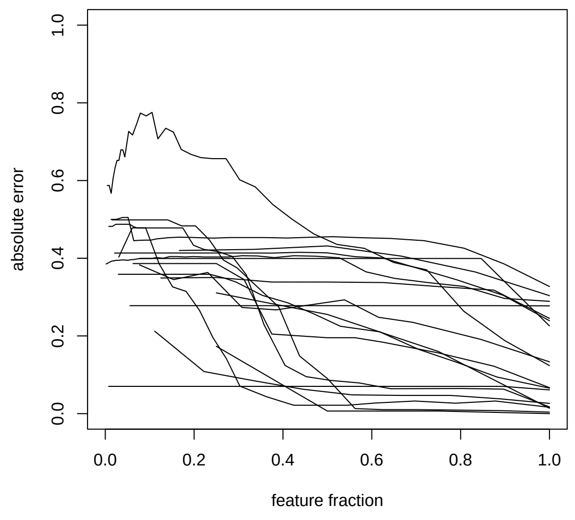


Figure 3.10: Absolute error of feature selection with Radial Basis SVM

not a valid complexity measure in terms of interpretability by the user.

3.4.4 Summary

The investigations in this section showed that formal complexity measures do not solve the problem of balancing interpretability and accuracy. Although decision tree size and number of features can be directly related to the interpretability of hypotheses, their optimization is costly, because it involves generating a large set of models and filtering out appropriate models in a subsequent step. On the other hand, the complexity measure of VC-dimension in Support Vector Machines, which can be influenced a-priori, does not lead to an optimization of the user-defined interpretability.

3.5 Conclusions

This chapter investigated the question how the interpretability of a model can be optimized in a black box scenario. Of course, some assumptions and information about the model is necessary, so the contribution of this chapter may be better described as mapping out how much interpretability is possible with how little information.

The least possible information that is necessary to train and evaluate a classifier are the structure of the observations (the classifier's input format) and the predictions it produces (the classifier's output).This information is enough to iteratively minimize the number of features evaluated by the classifier (see Section 3.1). Of course, the more information is available about the classifier the more efficient this step can be performed. This was demonstrated in Section

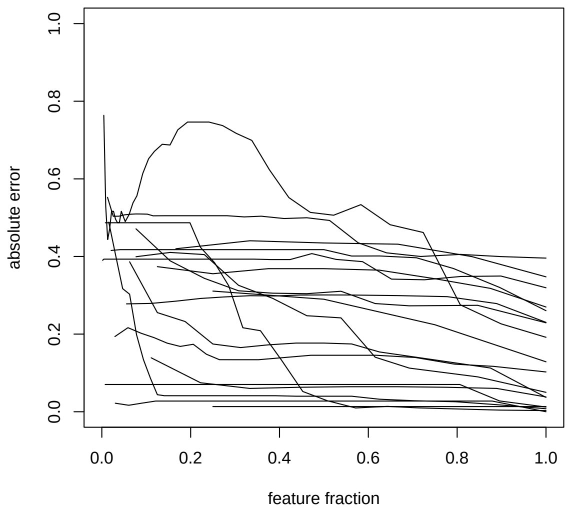


Figure 3.11: Absolute error of feature selection with decision tree

3.1.5 by making use of the classifier's gradient to select features from a nonlinear classifier in only one step, which was shown to significantly outperform all existing approaches.

Section 3.2 showed that more information about the classifiers inner structure is needed to select informative instances in order to make up for a missing measure of instance importance. It was shown that for all probabilistic classifiers the unknown structure over the example space that is induced by the classifier can be re-constructed by an algorithm based on data squashing and clustering. In contrast to existing approaches, the new squashing algorithm is a black-box approach and is hence applicable to every class of probabilistic models.

For numerical models, the gradient is a general property of the classifier that can easily be computed. Section 3.3 showed that this information can be used to construct a very general decomposition of a complex model into simpler parts by making use of simple linear approximations.

Finally, many classifiers posses a specific complexity function that is connected to their internal structure – e.g. VC dimension, depth of decision trees, or rule length. Section 3.4 showed that this information is less useful with respect to interpretability, as the complexity function is hard to optimize and its connection with a user-centered definition of interpretability is often not clear.

In conclusion, even very little information about the classifier's internals can be used to optimize understandability. The task of the next chapter will now be to investigate the complementary situation where complete information about the classifiers structure is available.

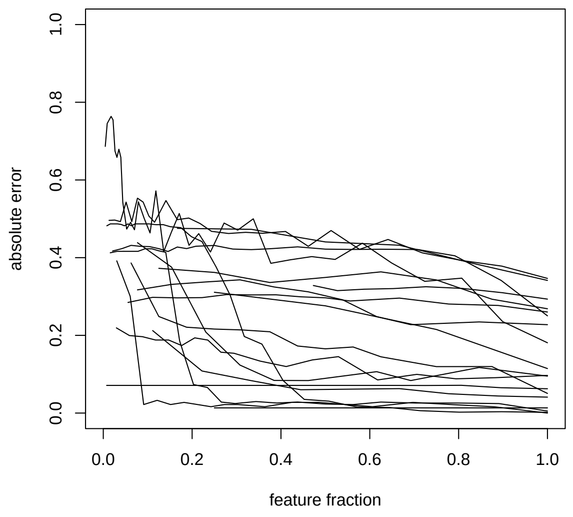


Figure 3.12: Absolute error of feature selection with covering rule learner

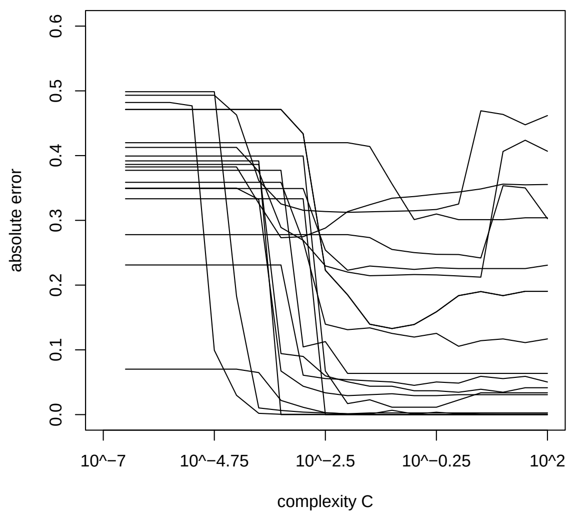


Figure 3.13: Absolute error of linear SVM

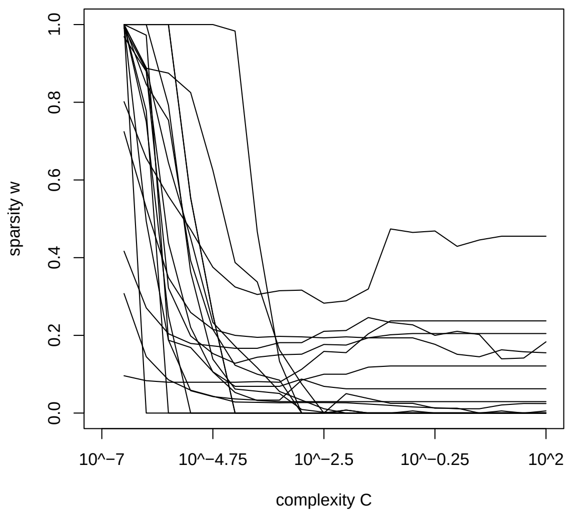


Figure 3.14: Sparsity of hyperplane vector of linear SVM

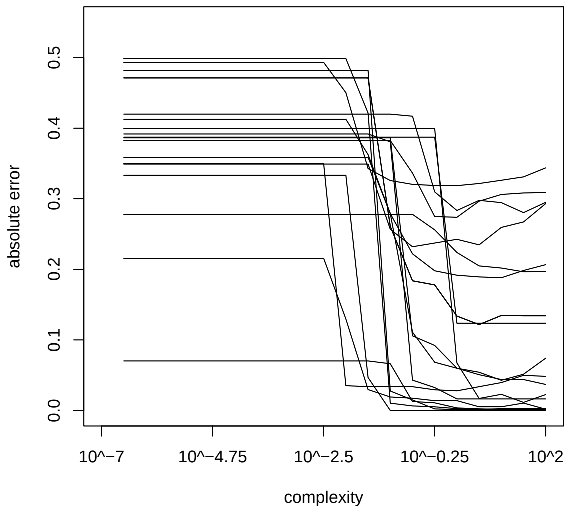


Figure 3.15: Absolute error of Radial Basis SVM

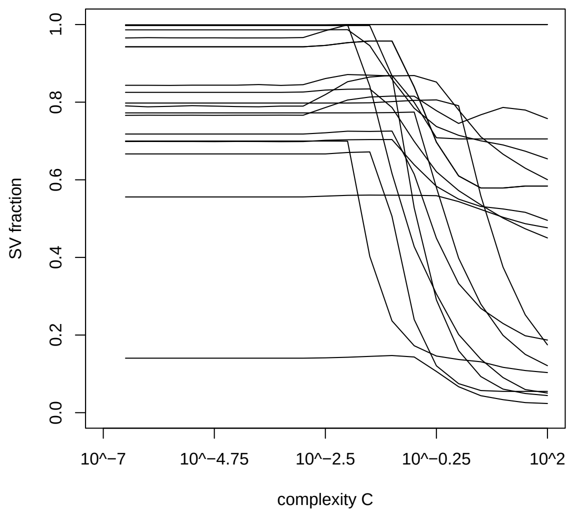


Figure 3.16: Fraction of Support Vectors of Radial Basis SVM

Chapter 4

Interpreting Support Vector Machines

This chapter deals with the problem of interpretability of Support Vector Machines. The reason for investigating SVMs in detail is that they are a very popular learning algorithm that proved to be effective over a large range of applications. Additionally this chapter exemplifies how interpretability can be optimized in a white-box scenario with extensive knowledge of the learner's internals and the structure of the hypothesis space.

Support Vector Machine classifiers are a linear combination

$$f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x) + b$$

with a high number n of basis functions $K(x_i, \cdot)$. The translation of SVMs into the language of the domain experts is usually done by data miners, for example by casting formulas into natural language sentences. For numerical learners, this translation if very complex and usually one can only approximately describe the learner in terms like *the higher X, the more likely the class is positive.* Finding the right words to talk to a domain expert is obviously a task that is very hard to automate.

This chapter presents three methods of transforming a SVM into a different form which is easier to interpret. First, in Section 4.1 a transformation of the SVM classifier in terms of different basis functions is presented. This method significantly reduces the number n of components needed to describe the classifier. Section 4.2 investigates the possibility of describing a SVM using logical formulas, which are easier to transform into natural language sentences than numerical rules. Finally, Section 4.3 presents a novel visualization method for Support Vector Machines that combines the structure of the hypothesis space with the form of the decision function.

4.1 Sparse Models

Sparse models are models with a low number of parameters. Although a SVM classifier

$$f(x) = \sum_{i=1}^{s} \alpha_i K(x_i, x) + b$$

is usually sparse in the sense that the number of support vectors s is usually much smaller than the number of training examples, it can still be very complex in the sense that s is much too large to understand the linear dependencies of the single functions $K(x_i, x)$. The obvious question is whether it is possible to express the same function $f(x)$ with less basis functions.

4.1.1 Pre-Images

The sparsity situation becomes much more simple in feature space. Let $\Phi : X \to \mathcal{X}$ be the feature map associated with the kernel K. Then, neglecting the offset b, the SVM function can be written as

$$\begin{aligned} f(x) &= \sum_{i=1}^{s} \alpha_i K(x_i, x) \\ &= \sum_{i=1}^{s} \alpha_i \Phi(x_i) * \Phi(x) \\ &= \left(\sum_{i=1}^{s} \alpha_i \Phi(x_i) \right) * \Phi(x) \\ &=: w * \Phi(x) \end{aligned}$$

For linear SVMs ($\Phi(x) = x$), the input space and the feature space fall together and w can be directly interpreted as a vector of weights for the input vectors attributes $x^{(i)}$. Another way of interpretation is to see w as prototypical for the positive class, in the sense that from the view of the SVM an example is the more likely to be positive the more similar it is to w in terms of the inner product $*$. The question whether a similar interpretation is possible for nonlinear kernels leads to the pre-image problem.

Definition 4.1.1 (Pre-Image Problem ([Mika et al., 1998])**.** Given a feature map $\Phi : X \to \mathcal{X}$ and an element of the feature space $w \in \mathcal{X}$, the pre-image problem is to find a $x \in X$ such that $\Phi(x) = w$.

The pre-image of the SVM vector $\sum_{i=1}^{s} \alpha_i \Phi(x_i)$ can then be interpreted as a prototypical example. However, pre-images do not need to exist. For example, for radial basis kernels the basis functions $\Phi(x_i)$ are Gaussians centered at x_i, but it is known that no Gaussian can be written as a linear combination of Gaussians centered at other points. To arrive at a solution, we have to relax the problem:

Definition 4.1.2 (Approximate Pre-Image Problem [Mika et al., 1998])**.** Given a feature map $\Phi : X \to \mathcal{X}$ and an element of the feature space $w \in \mathcal{X}$, the approximate pre-image problem is to find a $x \in X$ which approximates w in the 2-norm in feature space. I. e. $x = \arg\min_x ||w - \Phi(x)||^2_{\mathcal{H}}$.

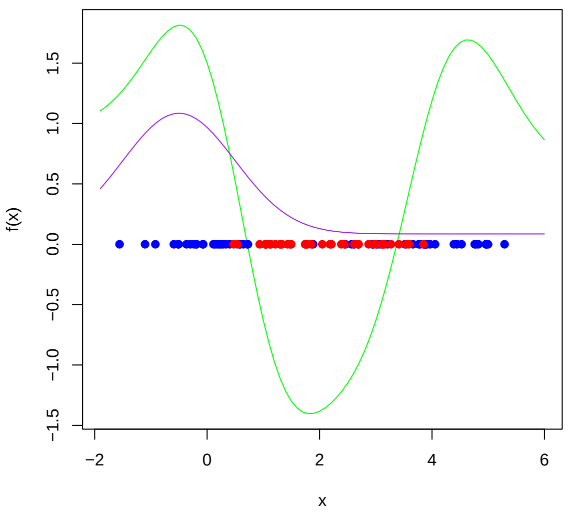


Figure 4.1: SVM decision function and pre-image

For example, in Figure 4.1 the green line shows the SVM decision function over the data set (red and blue points) and the purple line shows its approximate pre-image. The approximation is very coarse, but the pre-image at least covers the largest peak of the decision function on the left.

As the feature map Φ is generally not known, the problem has to be solved in feature space. The 2-norm is used in the definition of the problem, as it can be computed using the kernel:

$$\begin{aligned} x &= \arg\min_x ||w - \Phi(x)||_2^2 \\ &= \arg\min_x w * w - 2w * \Phi(x) + \Phi(x) * \Phi(x) \\ &= \arg\min_x -2w * \Phi(x) + \Phi(x) * \Phi(x) \\ &= \arg\min_x -2f(x) + K(x, x) \end{aligned}$$

As a special case, for kernels where $K(x, x)$ is constant for all x, we have

$$\arg\min_x -2w * \Phi(x) + \Phi(x) * \Phi(x) = \arg\max_x w * \Phi(x)$$

that is, the pre-image of w is the point x with the highest decision function value $f(x)$.

An immediate approximate solution to the pre-image problem would be to select the example in the training set with the lowest value of $K(x, x) - 2f(x)$ as the pre-image of the decision functions normal vector. For differentiable kernels K, one can use gradient search to minimize $K(x, x) - 2f(x)$ [Mika et al., 1998]. However, gradient search is iterative and hence slow and can get trapped in local minima. To avoid local minima the authors propose multiple restarts of the gradient search with random starting points.

Distance-based Pre-Image Construction

A more direct algorithm was proposed by [Kwok and Tsang, 2004], based on an idea similar to multi-dimensional scaling. The idea is that for several kernels, the distance in feature space can be related to the distance in input space. In particular, this holds for the class of isotropic kernels. A kernel is called isotropic if it only depends on the input space distance of its arguments, i. e. if $K(x, x') = \kappa(||x - x'||)$ for a real function κ. Radial basis kernels are an example of isotropic kernels. The idea is that as the feature space distances of w to the x_i in the training set can be calculated, the input space distances of the x_i to a pre-image x of w – should it exist – are also known. An approximate pre-image should have similar distances and hence one can look for a point which approximates the distance values as closely as possible. A solution to this problem can be given in closed form. For numerical stability, the paper proposes to use only the training points closest to w here. The paper does not give a suggestion on how many training points should be used. A empirical investigation on the standard data sets for this thesis showed that using only 10 training points is sufficient even for SVMs with high numbers of support vectors.

One could also think of optimizing the solution of the distance-based approach by using it as a starting point for gradient search. This combines the strength of the gradient search, which directly minimizes the target criterion, and hence might lead to a better solution, with the computational efficiency of the distance-based approach which does not fall into local optima.

Learning Pre-Images

In the case where one wants to compute several pre-images for the same feature space, [Bakir et al., 2003] suggest to learn a function $\Gamma : \mathcal{X} \to X$, to compute the pre-image, such that, approximately, $\gamma(\Phi(x)) = x$. The trick is to use a finite-dimensional basis in the feature space $\mathcal{X}$ and to work on the basis coordinates instead of the possibly infinite original space. As a finite set of points x_i always spans only a finite dimensional subspace of $\mathcal{X}$, this does not constrain the problem. The problem of estimating a pre-image function $\Gamma' : R^k \to X$ can then be reduced to a standard regression problem. The basis itself can be found be means of kernel principal component analysis with the kernel induced by Φ.

4.1.2 Reduced Set Methods

We will now show that the pre-image problem is actually too constrained. As we are only concerned with linear maps in the feature space, one can easily see that we do not need to know a pre-image of w, but actually the pre-image x_v of any vector w/β with $\beta \in \mathbb{R}\backslash\{0\}$ will do, as $w * \Phi(x) = \beta(w/\beta) * \Phi(x) = \beta\Phi(x_v) * \Phi(x)$. Hence, we can modify the approximate pre-image problem by a linear factor:

Definition 4.1.3 (Linear Approximate Pre-Image Problem)**.** Given a feature map $\Phi : X \to \mathcal{X}$ and an element of the feature space w, find a vector x in input space and a real number β, such that $(x, \beta) = \arg\min_{x,\beta} ||w - \beta\Phi(x)||^2_{\mathcal{H}}$

Figure 4.2 shows the linear pre-image of the same function as in Figure 4.1. It is obvious that due to the negative factor β the linear pre-image is a much better description of the decision function.

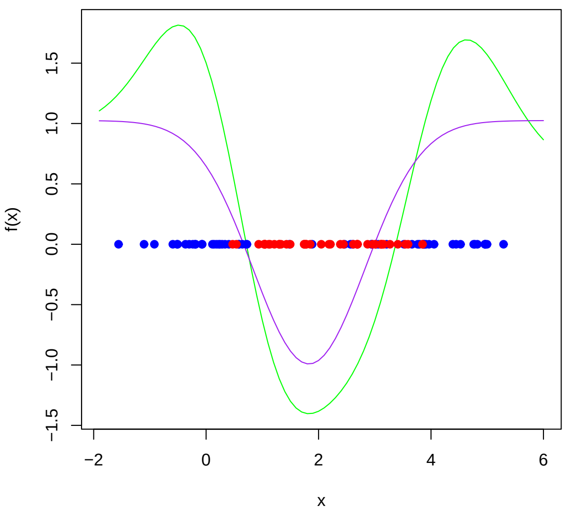


Figure 4.2: SVM decision function and linear pre-image

A little algebra shows that

$$\begin{aligned}(x, \beta) &= \arg\min_{x,\beta} ||w - \beta\Phi(x)||_2^2 \\ &= \arg\min_{x,\beta} ||w||^2 + \beta^2 K(x,x) - 2\beta w * \Phi(x) \\ &= \arg\min_{x,\beta} \beta^2 K(x,x) - 2\beta w * \Phi(x)\end{aligned}$$

Derivation with respect to β shows that the minimum is obtained at

$$\beta = \frac{w * \Phi(x)}{K(x,x)}$$

and hence we want to minimize

$$\beta^2 K(x,x) - 2\beta w * \Phi(x) = -\frac{(w * \Phi(x))^2}{K(x,x)}$$

over all x. Note that this term can be expressed in terms of the kernel and can hence be minimized by standard optimization techniques.

We can also generalize the idea of [Kwok and Tsang, 2004] to iteratively find a solution to the linear pre-image problem using a distance-based approach. Using an initial estimate β_0 of β, for example the average of the value defined in Equation 4.1 over all training points, we find a pre-image x_i of w/β using the approach for pre-images. Using x_i we can now compute the corresponding optimal β_i using Equation 4.1 and start over to find x_{i+1} and so on until convergence. We can see that this procedure monotonically decreases the distance to w and hence convergences against a local minimum by observing that both the pre-image algorithm and the optimization of β minimize the distance:

$$\begin{aligned} ||w-\beta_i\Phi(x_i)||^2 &= \beta_i^2||\frac{w}{\beta_i}-\Phi(x_i)||^2 \\ &\geq \beta_i^2||\frac{w}{\beta_i}-\Phi(x_{i+1})||^2 \\ &= ||w-\beta_i\Phi(x_{i+1})||^2 \\ &\geq ||w-\beta_{i+1}\Phi(x_{i+1})||^2 \end{aligned}$$

As this approach is based on a gradient optimization for β, it cannot be guaranteed that it will find a global minimum. Again, one can also use the distance-based approach as an initialization step to find a good starting point for the gradient approach.

The following table compares the feature-space distances $||w-\beta_i\Phi(x_i)||$ of the three linear pre-image methods. The next to last column give the statistical significance of the hypothesis that distance-based approach gives a larger error value than the combined approach. A + stands for significance at the level 0.05, a ++ stands for significance at the level 0.01, a − or −− stands for the significance of the complementary hypothesis of a lower error. The final column compares the gradient-based approach and the combined approach in the same way.

Name	Distance	Gradient	Combined	D $\geq$ C	G $\geq$ C
business	6.393	6.538	6.186	+	++
covtype	26.101	26.164	26.002	+	++
diabetes	8.141	8.304	8.137	*o*	++
digits	10.286	10.358	10.358	−−	*o*
physics	24.430	24.512	24.506	−−	+
ionosphere	6.560	6.085	5.384	++	++
liver	7.327	7.492	7.304	*o*	*o*
medicine	15.771	16.019	15.878	−−	++
mushroom	33.596	33.618	33.457	++	++
promoters	9.127	9.197	8.789	++	++
insurance	20.331	20.726	20.726	−−	++
balance	7.435	8.390	7.523	*o*	++
dermatology	6.026	6.147	5.479	++	++
iris	2.493	2.737	2.370	*o*	++
voting	9.713	9.793	9.793	−−	*o*
wine	6.577	6.665	6.665	−−	++
breast	4.090	4.414	3.568	++	++
garageband	22.664	22.713	22.713	−−	*o*

In conclusion, the combined method is far better than the gradient-based method, it has a significantly lower distance on 13 of the data sets and performs equal on another 4 data sets. In comparison to the distance-based method, the combined method perform equal. Both are significantly better than the other on 7 of the data sets. But as the combined method contains the result of the distance-based method as an intermediate step, it is easy to compare both vectors and select the one closer to the SVM hyperplane. This combines the advantages of both approaches and gives an algorithm that is never worse than the distance-based approach and better on 7 of the data sets.

Reduced Sets

Reduced set methods take the linear pre-image idea a step further. Instead of approximating w with a single pre-image $w = \beta\Phi(x)$, it is approximated by a linear combination $\sum_i \beta_i \Phi(x_i)$ of basis functions $\Phi(x)$.

Definition 4.1.4 (Reduced Set Problem [Burges, 1996])**.** Given a feature map $\Phi : X \to \mathcal{X}$ and an integer N, find vectors $z_1, \ldots, z_N$ in the input space and coefficients $\beta_1, \ldots, \beta_N$ that minimizes $||w - \sum_{i=1}^{N} \beta_i \Phi(z_i)||^2_{\mathcal{H}}$.

Note that as $w = \sum_{i=1}^{s} \alpha_i \Phi(x_i)$ is also defined in terms of a linear combination of basis functions, the solution is trivially given by the support vectors and their weights for $N \geq s$. One is usually interested in the case where $N << s$, that is, to approximate the vector w by a small number of input patterns. Note that there are no constraints on the coefficients β_i, in contrast to the SVM coefficients α_i. Hence, it is possible that one can even describe w exactly with less linear combinations than there are support vectors.

Figure 4.3 shows the reduced set approximation of the decision function of Figure 4.2 with $N = 3$. In comparison to the linear pre-image, this approximation covers most of the shape of the decision function.

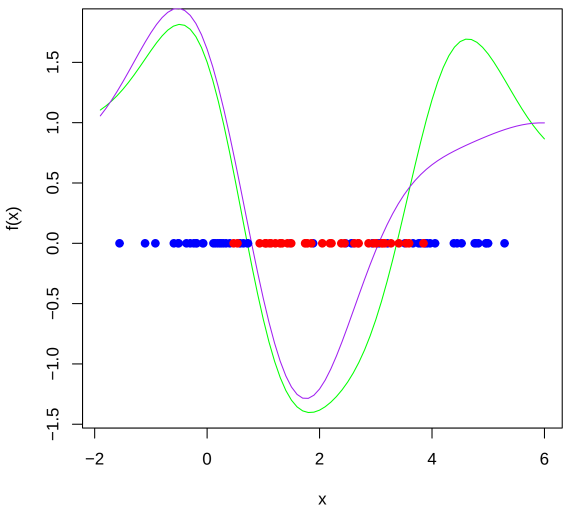


Figure 4.3: SVM decision function and reduced set approximation

In [Schölkopf et al., 1999] the problem is solved in three steps: selection of the β's, selection of a set of vectors z_i and construction of a set of vectors z_i.

First, it is shown that once the vectors z_i are known, the appropriate coefficients β can be computed directly. Deriving the distance function with respect to β_i yields that the solution must fulfill

$$K^{zx}\alpha = K^{z}\beta$$

and hence the solution is given by

$$\beta = (K^z)^{-1} K^{zx} \alpha \tag{4.1}$$

where $K^z_{ij} = \Phi(z_i)\Phi(z_j)$ and $K^{zx}_{ij} = \Phi(z_i)\Phi(x_j)$. The matrix K^z is invertible if the functions $\Phi(z_i)$ are linear independent. Otherwise, the pseudo-inverse of K^z can be used.

Given an SVM expansion $w = \sum_{i=1}^{s} \alpha_i \Phi(x_i)$ the next question is how to select a subset of N of the x_i which minimizes the approximation error. [Schölkopf et al., 1999] give two approaches. The first approach is to investigate the null space of the kernel matrix K, i. e. to find coefficients vectors $\alpha' \neq 0$ such that $K\alpha' = 0$. The existence of such a vector means that the functions $\Phi(x_i)$ are linearly dependent and one can replace one of these function by a linear combination of the others without changing the vector w. An approximative solution for small, non-zero eigenvalues of K is also given. The second approach is to directly minimize

$$||\sum_{i=1}^{s} \alpha_i \Phi(x_i) - \sum_{i=1}^{s} \beta_i \Phi(x_i)||^2 + \lambda \sum |\beta_i|$$

$\lambda > 0$ is a constant to trade off the approximation of w via the first summand against sparseness of β via the second term. The 1-norm of β is used instead of the 2-norm in the usual SVM formulation to achieve sparseness. Theoretically, one would like to directly minimize the number of nonzero components of β (the 0-norm), but this would lead to a combinatorial optimization problem which would not be efficiently solvable. Hence, the 1-norm is used as an approximation. For numerical reasons, this optimization scheme is only used to obtain the right base functions $\Phi(x_i)$ (the ones with nonzero β). The actual β's are computed by the exact formula 4.1.

The final question concerns the construction of new points z in the input space to be used in the reduced set. [Schölkopf et al., 1999] propose an iterative algorithm. Starting with the empty set of points z_k, each iteration finds a new vector z_{k+1} by solving problem 4.1 to approximate the feature space vector

$$\Delta_w = \sum_{i=1}^{s} \alpha_i \Phi(x_i) - \sum_{i=1}^{k} \beta_i \Phi(z_i)$$

Again, coefficients β are obtained by formula 4.1. The algorithm is iterated a certain number of times or until the distance of the approximate solution and the original vector w falls below a pre-defined threshold.

This algorithm can be improved. The greedy selection of new vectors z_i is not optimal. The vectors z_i can be optimized by repeatedly iterating on the set of vectors, each time removing one vector and replacing it with the linear pre-image of the SVM vector w minus the other vectors z_i. The iteration is repeated until the change of the estimate or the decrease in the distance falls below a pre-defined threshold. The optimality of the linear pre-image in approximating its feature space vector guarantees that this algorithm converges, as it guarantees that in each iteration the distance between the SVM vector and its reduced set approximation will decrease.

Figure 4.4 shows a comparison of the greedy reduced set approximation (purple line) and the iterated approach (black line) on the one-dimensional

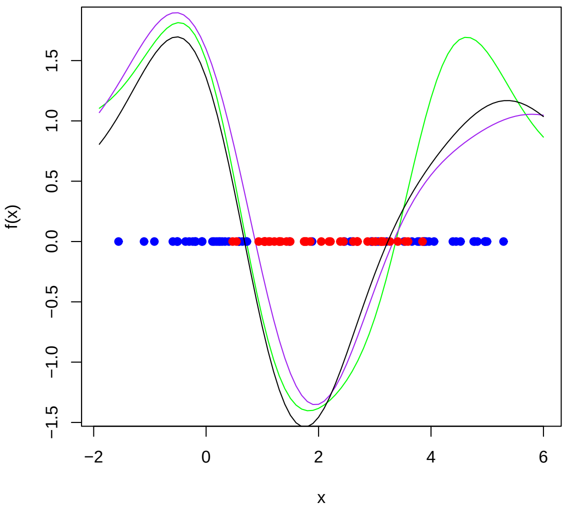


Figure 4.4: Greedy and iterated reduced set approximation

data set. The following table shows a comparison of the greedy approach of [Schölkopf et al., 1999] with the new iterated approach.

Name	k	Distance		Error		Greedy $\geq$ Iterated	
		Greedy	Iterated	Greedy	Iterated	Distance	Error
business	2	5.779	5.714	0.253	0.241	+	*o*
	3	5.462	5.337	0.228	0.202	++	*o*
	10	4.251	3.793	0.158	0.145	++	*o*
covtype	2	16.027	15.880	0.388	0.371	+	*o*
	3	16.065	15.864	0.370	0.367	++	*o*
	10	15.121	14.854	0.314	0.311	++	*o*
diabetes	2	7.878	7.539	0.287	0.259	++	++
	3	7.668	7.287	0.269	0.255	++	*o*
	10	6.647	5.776	0.260	0.243	++	+
digits	2	10.358	10.240	0.498	0.501	++	*o*
	3	10.358	9.612	0.498	0.498	*o*	*o*
	10	10.358	1.951	0.498	0.003	++	++
physics	2	14.956	14.635	0.500	0.478	++	+
	3	14.750	13.155	0.492	0.356	++	++
	10	14.705	12.731	0.508	0.354	++	++

Name	k	Distance		Error		Greedy ≥ Iterated	
		Greedy	Iterated	Greedy	Iterated	Distance	Error
ionosphere	2	4.644	4.475	0.094	0.076	++	+
	3	4.394	3.994	0.088	0.074	++	*o*
	10	3.485	2.459	0.065	0.062	++	*o*
liver	2	6.744	6.354	0.457	0.441	+	*o*
	3	6.234	5.470	0.350	0.356	++	*o*
	10	5.584	3.659	0.344	0.298	++	*o*
medicine	2	5.632	5.191	0.278	0.296	*o*	*o*
	3	5.301	4.403	0.278	0.291	+	*o*
	10	5.092	3.023	0.277	0.273	++	+
mushroom	2	22.433	22.432	0.377	0.377	*o*	*o*
	3	22.428	22.428	0.365	0.366	*o*	*o*
	10	22.061	22.061	0.299	0.299	*o*	*o*
promoters	2	8.608	8.579	0.207	0.179	++	+
	3	8.550	8.514	0.188	0.160	++	+
	10	8.155	8.105	0.122	0.140	++	*o*
insurance	2	6.687	5.436	0.070	0.204	++	*o*
	3	6.969	5.567	0.069	0.316	++	−
	10	6.709	2.343	0.147	0.063	++	*o*
balance	2	7.219	6.481	0.384	0.249	++	+
	3	7.002	4.299	0.386	0.057	++	++
	10	6.459	3.298	0.308	0.057	++	++
dermatology	2	5.327	5.326	0.037	0.037	++	*o*
	3	5.219	5.216	0.037	0.037	+	*o*
	10	4.454	4.389	0.027	0.021	++	*o*
iris	2	1.804	1.427	0.033	0.033	++	*o*
	3	1.563	0.875	0.106	0.006	++	*o*
	10	0.844	0.321	0.000	0.000	++	*o*
voting	2	9.793	9.644	0.386	0.613	++	−−
	3	9.793	9.678	0.386	0.386	++	*o*
	10	9.793	0.570	0.386	0.069	++	++
wine	2	6.665	6.550	0.399	0.600	++	−−
	3	6.665	6.162	0.399	0.399	++	*o*
	10	6.665	0.548	0.399	0.220	++	++
breast	2	3.201	3.127	0.035	0.038	++	*o*
	3	2.953	2.807	0.030	0.033	++	*o*
	10	2.152	1.753	0.029	0.026	++	*o*
garageband	2	18.919	18.853	0.404	0.479	++	*o*
	3	18.975	18.568	0.382	0.382	++	*o*
	10	18.941	13.959	0.432	0.296	++	++

In conclusion, one can see that the reduced set approximates the SVM much better than a simple linear pre-image. It can also be seen that the iterated reduced set approach is significantly better than the greedy approach. However, this decrease in distance comes at the price of higher runtime, as more computations of pre-images will be necessary. In the experiments, the iterated approach was between 16% (for $k = 2$) and 85% (for $k = 10$) slower than the greedy approach.

4.1.3 Function Approximation

Viewing an SVM decision function simply as a function $f : \mathbb{R}^d \to \mathbb{R}$, one can ask whether it is possible to simplify the function term. In the case of the linear SVM it is trivial to write

$$\begin{aligned}\sum_{i=1}^{k} \alpha_i (x_i * x) &= \left(\sum_{i=1}^{k} \alpha_i x_i \right) * x \\ &=: w * x,\end{aligned}$$

thereby simplifying the term from a sum over k vectors x_i to a single inner product. As another example, in the case of polynomial kernels $K_g(x, x') = (x * x')^g$, it is easy to see that the function term

$$\sum_{i=1}^{k} \alpha_i (x_i * x)^g$$

is a linear combination of k polynomials of degree g and hence a polynomial of degree g itself. Expanding this term into a standard form is a trivial task for any computer algebra program.

These kinds of simplifications are possible because both the linear and the polynomial kernel span a finite-dimensional space of functions. The simplification of a function term does in principle correspond to the projection of the function on a given basis. For infinite dimensional function spaces such as the one spanned by radial basis functions, this is not possible.

When no exact simplified function term exists, one can ask how closely this function can be approximated by a more simple function. Mathematical approximation theory has given a large number of results in this field and, of course, one is free to use a set of functions of one's choice to approximate the SVM function if one feels that this will give a clearer representation.

In the following, we will deal with the question of approximating an SVM with other SVMs, which has the advantage of allowing a kernelized approach and not deviating too much from the original model.

Augmenting the RKHS of a Kernel

Remember that a kernel function K defines a Reproducing Kernel Hilbert Space $\mathcal{H}_K$ via the basis functions $K(x, \cdot)$ (see Chapter 2.1.3). Suppose we want to increase the expressibility of this RKHS by adding some other basis functions ψ_i. We can use the same construction as for $\mathcal{H}_K$ to define a space $\mathcal{H}_{K,\Psi}$ using linear combinations of the old basis function $K(x, \cdot)$ and the new basis functions ψ_i. However, in order to follow the construction to the end, we have to guarantee that the form

$$\begin{aligned}< f, g > &= < \sum_i \alpha_i K(a_i, \cdot) + \sum_j \beta_j \psi_j, \sum_i \alpha'_i K(a'_i, \cdot) + \sum_j \beta'_j \psi'_j > \\ &= \sum_{i,j} \alpha_i \alpha'_j < K(a_i, \cdot), K(a'_i, \cdot) > + \sum_{i,j} \beta_i \beta'_j < \psi_i, \psi'_j > \\ &\quad + \sum_{i,j} \alpha_i \beta'_j < K(a_i, \cdot), \psi'_j > + \sum_{i,j} \beta_i \alpha'_j < \psi_i, K(a'_i, \cdot) >\end{aligned}$$

is positive definite, such that it forms an inner product. We will now give such an inner product for the special case of RBF kernels.

An Alternative Representation for RBF Kernels

Radial Basis Kernels $K_\gamma(a,b) = e^{-\gamma||a-b||^2}$ allow for a different explicit representation of K as an inner product in a space of functions.

We start this section a little technically. It is well known that on the space $L_2(\Re^d)$ of square-integrable functions $\mathbb{R}^d \to \mathbb{R}$, the integral

$$f * g \;:=\; \int_{\mathbb{R}^d} f(x)g(x)dx$$

defines an inner product. In particular, for basis functions of possibly different RBF kernels K_γ, we have

$$\begin{aligned}
K_{\gamma_a}(a,\cdot) * K_{\gamma_b}(b,\cdot) &= \int_{\mathbb{R}^d} K_{\gamma_a}(a,x) * K_{\gamma_b}(b,x)dx \\
&= \int_{\mathbb{R}^d} e^{-\gamma_a||a-x||^2} e^{-\gamma_b||b-x||^2} dx \\
&= \int_{\mathbb{R}^d} e^{-\gamma_a||a-x||^2-\gamma_b||b-x||^2} dx \\
&= \int_{\mathbb{R}^d} e^{-\sum_{i=1}^d \gamma_a(a_i-x_i)^2+\gamma_b(b_i-x_i)^2} dx \\
&= \int_{\mathbb{R}^d} \prod_{i=1}^d e^{-\gamma_a(a_i-x_i)^2-\gamma_b(b_i-x_i)^2)} dx \\
&= \prod_{i=1}^d \int_{\mathbb{R}^d} e^{-\gamma_a(a_i-x_i)^2-\gamma_b(b_i-x_i)^2} dx_i
\end{aligned}$$

Further, we can write

$$\begin{aligned}
& \gamma_a(a_i-x_i)^2+\gamma_b(b_i-x_i)^2 \\
= \;& \gamma_a(a_i^2-2a_ix_i+x_i^2)+\gamma_b(b_i^2-2b_ix_i+x_i^2) \\
= \;& (\gamma_a+\gamma_b)x_i^2 - 2(\gamma_a a_i+\gamma_b b_i)x_i + \gamma_a a_i^2 + \gamma_b b_i^2 \\
= \;& (\gamma_a+\gamma_b)(x_i - \frac{\gamma_a a_i+\gamma_b b_i}{\gamma_a+\gamma_b})^2 \\
& -\frac{(\gamma_a a_i+\gamma_b b_i)^2}{\gamma_a+\gamma_b} + \gamma_a a_i^2 + \gamma_b b_i^2 \\
= \;& (\gamma_a+\gamma_b)(x_i - \frac{\gamma_a a_i+\gamma_b b_i}{\gamma_a+\gamma_b})^2 \\
& +(\gamma_a - \frac{\gamma_a^2}{\gamma_a+\gamma_b})a_i^2 - \frac{2\gamma_a a_i\gamma_b b_i}{\gamma_a+\gamma_b} + (\gamma_b - \frac{\gamma_b^2}{\gamma_a+\gamma_b})b_i^2 \\
= \;& (\gamma_a+\gamma_b)(x_i - \frac{\gamma_a a_i+\gamma_b b_i}{\gamma_a+\gamma_b})^2 \\
& +\frac{\gamma_a\gamma_b}{\gamma_a+\gamma_b}a_i^2 - \frac{2\gamma_a a_i\gamma_b b_i}{\gamma_a+\gamma_b} + \frac{\gamma_a\gamma_b}{\gamma_a+\gamma_b}b_i^2 \\
= \;& (\gamma_a+\gamma_b)(x_i - \frac{\gamma_a a_i+\gamma_b b_i}{\gamma_a+\gamma_b})^2 + \frac{\gamma_a\gamma_b}{\gamma_a+\gamma_b}(a_i-b_i)^2
\end{aligned}$$

and with the well-known identity

$$\int_{\mathbb{R}^d} (\frac{1}{2\pi})^{d/2} \det(\Sigma)^{-1/2} e^{-\frac{1}{2}(x-\mu)^t \Sigma^{-1}(x-\mu)} dx \quad = \quad 1$$

from the multivariate normal distribution we obtain

$$\begin{aligned} K_{\gamma_a}(a,\cdot) * K_{\gamma_b}(b,\cdot) &= \prod_{i=1}^{d} \int_{\mathbb{R}^d} e^{-\gamma_a(a_i-x_i)^2-\gamma_b(b_i-x_i)^2} dx_i \\ &= K_{\gamma_a\gamma_b/(\gamma_a+\gamma_b)}(a,b) \prod_{i=1}^{d} \int_{\mathbb{R}^d} e^{-(\gamma_a+\gamma_b)(x_i-\mu)^2} dx_i \\ &= K_{\gamma_a\gamma_b/(\gamma_a+\gamma_b)}(a,b) \int_{\mathbb{R}^d} e^{-\frac{1}{2}\sum_{i=1}^{d} 2(\gamma_a+\gamma_b)(x_i-\mu)^2} dx \\ &= K_{\gamma_a\gamma_b/(\gamma_a+\gamma_b)}(a,b) (\frac{\pi}{\gamma_a+\gamma_b})^{d/2} \end{aligned}$$

with $\mu = \frac{\gamma_a a_i + \gamma_b b_i}{\gamma_a+\gamma_b}$ and $\Sigma^{-1} = \mathrm{diag}(2(\gamma_a+\gamma_b))$. In particular, for $\gamma_a = \gamma_b = 2\gamma$, it follows that

$$\left((\frac{4\gamma}{\pi})^{d/4} K_{2\gamma}(a,\cdot) \right) * \left((\frac{4\gamma}{\pi})^{d/4} K_{2\gamma}(b,\cdot) \right) \quad = \quad K_\gamma(a,b)$$

Hence, the RBF kernel with parameter γ can be interpreted as the integral product of two basis functions of a RBF kernel with parameter 2γ.

The advantage of this representation is that the integral product is also well-defined for functions other than basis functions of a particular kernel. Here, we are interested in the case of basis functions for two different RBF kernels. Let γ_1 and γ_2 be their parameter and let $\gamma = 2\frac{\gamma_1\gamma_2}{\gamma_1+\gamma_2}$. We define an inner product on the space of functions given by the basis functions of both K_{γ_1} and K_{γ_2} as follows. Let

$$\begin{aligned} A(\gamma_1,\gamma_2) &:= \left(\frac{\pi}{2(\gamma_1+\gamma_2)} \right)^{d/2} \\ A(\gamma) &:= A(\gamma,\gamma) = (\frac{\pi}{4\gamma})^{d/2} \end{aligned}$$

such that $A(\gamma_1,\gamma_2)K_\gamma(a,b) = K_{\gamma_1}(a,\cdot) * K_{\gamma_2}(b,\cdot)$.

Now select a kernel parameter γ^* and set $A = A(\gamma^*,\gamma^*)$. For two basis functions $K_{\gamma_1}(a,\cdot)$ and $K_{\gamma_2}(b,\cdot)$ of kernels K_{γ_1} and K_{γ_2}, we define an inner product

$$\begin{aligned} < K_{\gamma_1}(a,\cdot), K_{\gamma_2}(b,\cdot) > &= \frac{1}{A} K_{2\gamma_1}(a,\cdot) * K_{2\gamma_2}(b,\cdot) \\ &= \frac{A(\gamma_1,\gamma_2)}{A} K_\gamma(a,b) \end{aligned}$$

where $\gamma = 2\gamma_1\gamma_2/(\gamma_1+\gamma_2)$. With this definition, we recover the Kernel function K_γ as the inner product in the case $\gamma_1 = \gamma_2 = \gamma^*$:

$$\begin{aligned} < K_{\gamma^*}(a,\cdot), K_{\gamma^*}(b,\cdot) > &= \frac{A(\gamma^*,\gamma^*)}{A} K_\gamma(a,b) \\ &= K_{\gamma^*}(a,b). \end{aligned}$$

To extend this definition to the linear space of basis functions of radial kernels $\mathcal{R} = \{K_\gamma(a, b) | \gamma \in \Re_{>0}, a \in \Re^d\}$, let f and g be linear combinations of these basis functions. That is, $f = \sum_i \alpha_i K_{\gamma_i}(a_i, \cdot)$ and $g = \sum_i \beta_i K_{\gamma_i'}(b_i, \cdot)$. Set $\Gamma(\gamma, \gamma') = 2\gamma\gamma'/(\gamma + \gamma')$ and define

$$\begin{aligned} < f, g > &= < \sum_i \alpha_i K_{\gamma_i}(a_i, \cdot), \sum_i \beta_i K_{\gamma_i'}(b_i, \cdot) > \\ &:= \sum_i \sum_j \alpha_i \beta_j < K_{\gamma_i}(a_i, \cdot), K_{\gamma_i'}(b_i, \cdot) > \\ &= \sum_i \sum_j \alpha_i \beta_j \frac{A(\gamma, \gamma')}{A} K_{\Gamma(\gamma,\gamma')}(a_i, b_j) \end{aligned}$$

To show that this definition indeed produces an inner product, we can transform it into an integral product.

$$\begin{aligned} A < f, g > &= \sum_i \sum_j \alpha_i \beta_j A(\gamma, \gamma') K_{\Gamma(\gamma,\gamma')}(a_i, b_j) \\ &= \sum_i \sum_j \alpha_i \beta_j K_{2\gamma}(a_i, \cdot) * K_{2\gamma'}(b_j, \cdot) \\ &= \sum_i \left(\alpha_i K_{2\gamma}(a_i, \cdot) * \left(\sum_j \beta_j K_{2\gamma'}(b_j, \cdot) \right) \right) \\ &= \left(\sum_i \alpha_i K_{2\gamma}(a_i, \cdot) \right) * \left(\sum_j \beta_j K_{2\gamma'}(b_j, \cdot) \right) \end{aligned}$$

As $A > 0$ and $*$ is an inner product, so is $< \cdot, \cdot >$. Again, for f and g as linear combinations of basis functions with the fixed parameter γ^*, the inner product reduces to the inner product induced by the usual kernel K_{γ^*}.

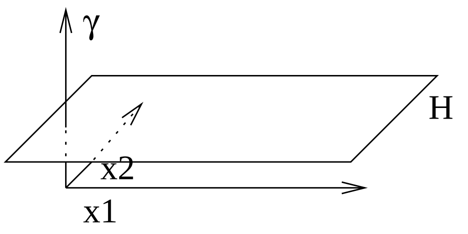


Figure 4.5: The new Hilbert space contains the RKHS H of a SVM

Hence, the space $\mathcal{R}$ with the newly defined inner product is a Hilbert Space that contains the Reproducing Kernel Hilbert Space $\mathcal{H}$ of K_γ as a subspace (see Figure 4.5). This means that, first, the $\mathcal{H}$ is a subset of $\mathcal{R}$, and second, that the inner product $< \cdot, \cdot >_{|\mathcal{H}\times\mathcal{H}}$ coincides with the inner product induced by the K_{γ^*}.

It should be noticed that $\mathcal{R}$ is not necessarily a Reproducing Kernel Hilbert Space itself, as its inner product is not induced by a single kernel function.

Back to the Pre-Images

With respect to the pre-images, the newly constructed Hilbert Space means that we have a larger set of functions, such that we can choose a better approximation. At the same time, the structure of the space, as defined by the inner product, remains the same and hence mathematical similarity in the augmented space translates to similarity with respect to the generalization ability.

The new augmented Hilbert space allows to solve the standard linear pre-image problem

$$(\beta, x) \quad = \quad \arg\min_{\beta,x} ||f - \beta\Phi(x)||^2$$

also for pre-images of basis functions of RBF kernels other than the one used to find f. In particular, we can use kernels with a smaller value of γ (higher standard deviation), which corresponds to using larger Gaussian "hats" and hence less detail to approximate the general characteristics of the function. Again, the problem can be solved purely in terms of kernels. Let $\mathcal{H}_1$ be the RKHS of a RBF kernel with parameter γ_1, f be a function in that space, Φ be a basis function of a RBF kernel with parameter γ_2, $\mathcal{H}'$ be the linear space of both kernel functions and $\gamma = 2\gamma_1\gamma_2/(\gamma_1 + \gamma_2)$. Then

$$\begin{aligned}
||f - \beta\Phi(x)||^2_{\mathcal{H}'} &= \; < f - \beta\Phi(x), f - \beta\Phi(x) >_{\mathcal{H}'} \\
&= ||f||^2_{\mathcal{H}'} + \beta^2||\Phi(x)||^2_{\mathcal{H}'} - 2\beta < f, \Phi(x) >_{\mathcal{H}'} \\
&= ||f||^2_{\mathcal{H}} + \beta^2 \frac{A(\gamma_2,\gamma_2)}{A(\gamma_1,\gamma_1)} K_{\gamma_2}(x,x) \\
&\quad -2\beta \frac{A(\gamma_1,\gamma_2)}{A(\gamma_1,\gamma_1)} \sum_i \alpha_i K_\gamma(x_i,x) \\
&= ||f||^2_{\mathcal{H}} + \beta^2 \frac{A(\gamma_2,\gamma_2)}{A(\gamma_1,\gamma_1)} \\
&\quad -2\beta \frac{A(\gamma_1,\gamma_2)}{A(\gamma_1,\gamma_1)} \sum_i \alpha_i K_\gamma(x_i,x)
\end{aligned}$$

Derivation with respect to β yields that the minimum is attained at

$$\beta \quad = \quad \frac{A(\gamma_1,\gamma_2)}{A(\gamma_2,\gamma_2)} \sum_i \alpha_i K_\gamma(x_i,x)$$

and hence it suffices to find

$$
\begin{aligned}
& \min_{\beta,x} ||f - \beta\Phi(x)||^2_{\mathcal{H}'} \\
= \quad & \beta^2 \frac{A(\gamma_2,\gamma_2)}{A(\gamma_1,\gamma_1)} - 2\beta \frac{A(\gamma_1,\gamma_2)}{A(\gamma_1,\gamma_1)} \sum_i \alpha_i K_\gamma(x_i,x) \\
= \quad & \min_{\beta,x} \beta^2 \frac{A(\gamma_2,\gamma_2)}{A(\gamma_1,\gamma_1)} - 2\beta \frac{A(\gamma_1,\gamma_2)}{A(\gamma_1,\gamma_1)} \sum_i \alpha_i K_\gamma(x_i,x) \\
= \quad & \min_x \frac{A(\gamma_1,\gamma_2)^2}{A(\gamma_1,\gamma_1)A(\gamma_2,\gamma_2)} \left(\sum_i \alpha_i K_\gamma(x_i,x) \right)^2 \\
& -2 \frac{A(\gamma_1,\gamma_2)^2}{A(\gamma_1,\gamma_1)A(\gamma_2,\gamma_2)} \left(\sum_i \alpha_i K_\gamma(x_i,x) \right)^2 \\
= \quad & \min_x - \frac{A(\gamma_1,\gamma_2)^2}{A(\gamma_1,\gamma_1)A(\gamma_2,\gamma_2)} \left(\sum_i \alpha_i K_\gamma(x_i,x) \right)^2 \\
= \quad & \max_x \left(\sum_i \alpha_i K_\gamma(x_i,x) \right)^2
\end{aligned}
$$

This problem differs from the original pre-image problem only by the kernel width γ, so it can be solved with any standard pre-image algorithm.

Figure 4.6 compares the SVM decision function (green line), the usual linear pre-image (purple line) with the linear pre-image in the augmented space (black line), where the kernel parameter γ in the augmented space was chosen to optimize the prediction error of the new pre-image. The new pre-image function exactly covers the decision region of the SVM decision function. Alternatively, one can optimize γ in order to minimize the feature space distance between the function and the pre-image, which is more appealing when one want to approximate the complete form of the decision function and not only its decision region.

In the following experiments, the value of γ for the pre-image kernel was optimized over values between $\gamma_{SVM}/10$ and γ_{SVM} by choosing the value with the lowest difference between the original SVMs predictions and the pre-image prediction (in order to describe the complete SVM function, the actual prediction error of the pre-image was not directly used). No values higher than γ_{SVM} were used because higher values correspond to smaller variance in the radial basis function and hence to a more specific kernel, while a pre-image should be more general than a single basis functions that comes from the complete SVM hypothesis. The following table compares the standard pre-images with the new multi-kernel approach, listing both the distance to the regular SVM hypothesis in feature space and the prediction error obtained. The values are the results of 10-fold cross-validation. Because of numerical problems due to the high dimension, the distance values for the garageband data set could not be computed.

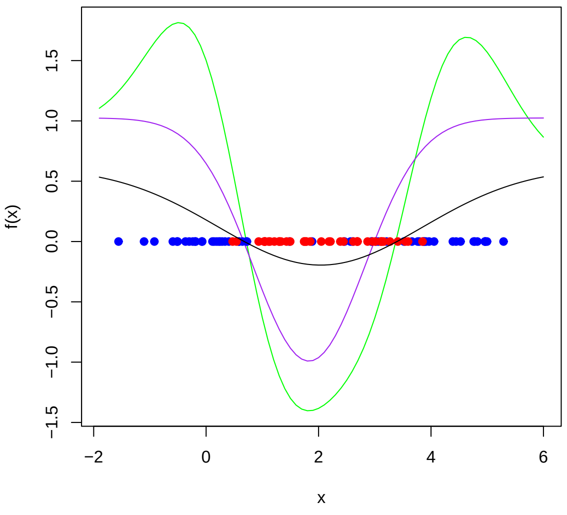


Figure 4.6: Linear pre-image and augmented kernel pre-image

Name	Standard		Multi-Kernel		Comparison	
	Dist	Err	Dist	Err	Dist	Err
business	5017.6	0.362	4873.5	0.249	+	+
covtype	7.062E11	0.407	7.015E22	0.397	++	*o*
diabetes	445.8	0.437	450.9	0.293	−	+
digits	2.047E47	0.498	2.048E47	0.498	−−	*o*
physics	1.067E38	0.491	1.067E38	0.487	−−	*o*
ionosphere	6.908E16	0.170	7.531E16	0.641	−−	−−
liver	154.6	0.457	161.3	0.385	−	+
medicine	1.029E9	0.277	1.091E9	0.277	−−	*o*
mushroom	2.877E29	0.433	2.871E29	0.414	+	*o*
promoters	9.205E108	0.159	9.205E108	0.105	*o*	*o*
insurance	3.569E97	0.070	3.696E97	0.070	−−	*o*
balance	56.674	0.500	62.829	0.213	−−	++
dermatology	1.328E8	0.037	1.329E9	0.048	−−	*o*
iris	18.61	0.286	19.92	0.126	−−	+
voting	3.726E12	0.386	3.727E12	0.386	−−	*o*
wine	1.709E10	0.399	1.713E10	0.399	−−	*o*
breast	354.7	0.093	345.7	0.039	*o*	*o*
garageband	-	0.382	-	0.382	*o*	*o*

One can see that the new pre-image algorithm performs significantly better than the standard one on 5 on the data sets and performs better or equal on another 11 of the data sets. It performs worse on only 2 of the data sets (ionosphere and dermatology), where only the performance reduction on the ionosphere data set is significant. One can also see that the distance to the original SVM varies widely and the new approach performs less good in terms of the distance. However, this has no impact on the actual error which shows that the distance in feature space is only useful as an intermediate concept, but not as a performance measure itself. This can be explained by the fact that the distance of the functions can be defined in terms of the differences of the function values, while the different error rates are only described by the difference of the signs. E.g. the functions f and $100 \cdot f$ have a large distance, but exactly the same error. An example can be seen in Figure 4.7 with the distance-based pre-image in black and the error-based pre-image in purple.

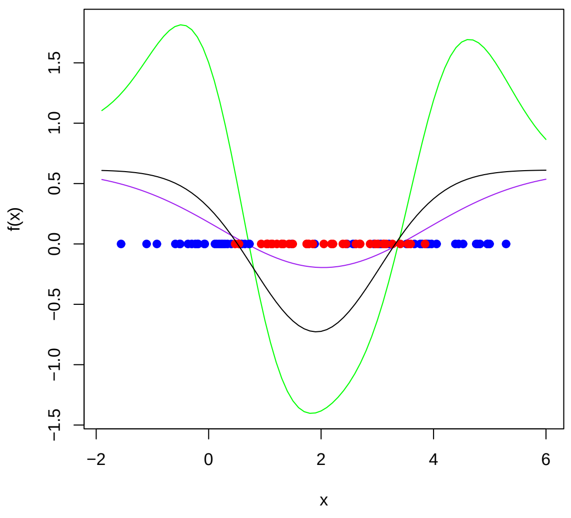


Figure 4.7: Distance-based and Error-based multi-kernel pre-image

In terms of understandability, the new augmented pre-image gives a much better approximation to the decision function SVM hypothesis, while at the same time it preserves the simple functional form of a single Radial Basis Function, i.e. the user has to inspect only a single observation x and a single linear factor β. The advantage for understandability becomes apparent in the following equation, which compares the SVM function of Figure 4.7 (first 52 lines) with its augmented pre-image (last line).

$$\begin{aligned}
f_{svm}(x) &= \cdot e^{-0.5||x-0.716||^2} + e^{-0.5||x-0.579||^2} + e^{-0.5||x-0.336||^2} \\
&+e^{-0.5||x-0.645||^2} + 0.311 \cdot e^{-0.5||x--2.319||^2} + e^{-0.5||x-0.587||^2} \\
&+0.716 \cdot e^{-0.5||x-0.288||^2} + e^{-0.5||x-0.403||^2} + e^{-0.5||x-1.481||^2} \\
&+e^{-0.5||x-0.726||^2} - 0.959 \cdot e^{-0.5||x-1.214||^2} - e^{-0.5||x-1.112||^2} \\
&-e^{-0.5||x-2.693||^2} - e^{-0.5||x-1.031||^2} - e^{-0.5||x-2.691||^2} \\
&-e^{-0.5||x-2.871||^2} - e^{-0.5||x-1.048||^2} + -0.621 \cdot e^{-0.5||x-2.685||^2} \\
&-e^{-0.5||x-0.539||^2} - e^{-0.5||x-3.157||^2} - e^{-0.5||x-3.591||^2} \\
&-e^{-0.5||x-3.270||^2} - e^{-0.5||x-3.543||^2} - e^{-0.5||x-1.135||^2} \\
&-e^{-0.5||x-0.471||^2} - e^{-0.5||x-2.961||^2} - e^{-0.5||x-3.131||^2} \\
&-e^{-0.5||x-3.006||^2} - e^{-0.5||x-1.045||^2} - e^{-0.5||x-3.409||^2} \\
&-e^{-0.5||x-0.934||^2} - e^{-0.5||x-3.113||^2} - e^{-0.5||x-3.848||^2} \\
&-e^{-0.5||x-1.041||^2} - e^{-0.5||x-3.107||^2} + e^{-0.5||x-2.568||^2} \\
&+e^{-0.5||x-3.661||^2} + e^{-0.5||x-2.949||^2} + e^{-0.5||x-3.063||^2} \\
&+e^{-0.5||x-3.208||^2} + e^{-0.5||x-3.874||^2} + e^{-0.5||x-3.763||^2} \\
&+0.553 \cdot e^{-0.5||x-3.886||^2} + e^{-0.5||x-3.862||^2} + e^{-0.5||x-2.609||^2} \\
&+e^{-0.5||x-3.022||^2} + e^{-0.5||x-2.461||^2} + e^{-0.5||x-1.885||^2} \\
&+e^{-0.5||x-3.794||^2} + e^{-0.5||x-3.519||^2} + e^{-0.5||x-2.939||^2} \\
&+0.612 \\
&\approx -1.340 \cdot e^{-0.397||x-1.916||^2}
\end{aligned}$$

Augmented Kernel Reduced Sets

Finally, not only the linear pre-image problem, but also the reduced set problem is well-defined in the new augmented Hilbert space. It is straight-forward to extend the gradient search algorithm of [Mika et al., 1998] to the case of radial basis functions with different parameters γ in order to minimize the distance in the new Hilbert space.

The following table compares the test errors of the standard reduced sets and the augmented kernel version.

Name	k	Standard	Multi-Kernel	Comparison
business	2	0.241	0.209	o
	3	0.202	0.177	o
	10	0.152	0.197	$-$
covtype	2	0.368	0.355	o
	3	0.362	0.359	o
	10	0.305	0.319	$-$
diabetes	2	0.259	0.250	o
	3	0.255	0.280	o
	10	0.242	0.243	o
digits	2	0.501	0.003	++
	3	0.498	0.003	++
	10	0.003	0.003	++

Name	k	Standard	Multi-Kernel	Comparison
physics	2	0.451	0.331	++
	3	0.350	0.455	−−
	10	0.335	0.347	−
ionosphere	2	0.076	0.076	o
	3	0.074	0.173	−−
	10	0.062	0.207	−−
liver	2	0.385	0.295	+
	3	0.341	0.336	o
	10	0.292	0.304	o
medicine	2	0.277	0.271	o
	3	0.277	0.277	o
	10	0.277	0.275	o
mushroom	2	0.383	0.35	++
	3	0.366	0.323	++
	10	0.333	0.294	++
promoters	2	0.179	0.500	−−
	3	0.160	0.178	o
	10	0.140	0.102	o
insurance	2	0.070	0.062	++
	3	0.013	0.009	o
	10	0.062	0.041	o
balance	2	0.280	0.046	++
	3	0.088	0.078	o
	10	0.081	0.032	+
dermatology	2	0.037	0.163	−
	3	0.037	0.112	−−
	10	0.021	0.032	o
iris	2	0.033	0.000	o
	3	0.006	0.000	o
	10	0.000	0.000	++
voting	2	0.613	0.069	++
	3	0.386	0.071	++
	10	0.069	0.073	o
wine	2	0.600	0.259	++
	3	0.399	0.214	++
	10	0.220	0.275	−−
breast	2	0.035	0.035	o
	3	0.033	0.032	o
	10	0.026	0.029	o
garageband	2	0.472	0.284	++
	3	0.399	0.283	++
	10	0.312	0.284	+

The comparison shows that for $k = 2$ the new method performs significantly better than the standard reduced set on 9 data sets while it is significantly worse on 2 data sets. For $k = 3$ it is better on 5 data sets and worse on 3 data sets and for $k = 10$ it is better on 10 data sets and worse on 5.

[Schölkopf et al., 1998a] also proposes an algorithm for simultaneously approximating multiple feature space vectors. In the case of radial basis functions, an interesting second feature vector to approximate is that given by the kernel functions of all n examples, each with weight $1/n$, as this will be a form of kernel density estimation, called the Parzen density estimate [Scott, 1992]. Reduced set approximations for both a SVM vector and the density estimator will give a set of points that are representative for both the decision function and the data distribution.

4.2 Logical Approximation of Numerical Functions

Numerical learners like the Support Vector Machine haven proven to be most successful on many real-world learning tasks. On the other hand, a logical representation has often found to be much more interpretable for humans. The obvious question is whether and to what extend it is possible to approximate a given numerical learner by some set of logical rules.

Given a numerical decision function $f : \mathcal{X} \to \mathbb{R}$, there are two possible approximation tasks, namely approximating the function f itself and approximating its class prediction $signf(x)$, which corresponds to identifying $\{x|f(x) > 0\}$. This section will only deal with the latter tasks, as due to the discrete logical representation the first tasks can be reduced to approximating the sets $\{x|f(x) > \delta\}$ for different values of δ (although there may of course be more sophisticated approaches).

For a probability measure P we can define $P(f \neq g)$ as a distance measure between f and its logical approximation g. Such a probability measure is necessary because a data-independent distance measure like the volume of all points predicted differently by f and g will usually be not finite. This formulation allows the following theorem:

Theorem 4.2.1 (Existence of a Logical Approximation)**.** *Let P be a probability measure with a density ρ, i. e. $P(A) = \int_A \rho(x)dx$. Let f be a continuous classifier. Then for all $\eta > 0$ there exists a logical classifier g with*

$$P(signf(x) \neq g(x)) < \eta$$

and

$$Err(g) < Err(f) + \eta.$$

Proof. It is sufficient to show that there exists a g with $P(signf(x) \neq g(x)) < \eta$, because $g(x) \neq y$ implies $g(x) \neq signf(x) \vee signf(x) \neq y$ and hence

$$P(g(x) \neq y) \leq P(f(x) \neq y) + P(signf(x) \neq g(x))$$

which means

$$Err(g) \leq Err(f) + P(signf(x) \neq g(x)).$$

As P is a finite Borel measure, P is regular. Hence, there exists a compact set $K \subset R^n$ with $P(K) = P(R^n) - \frac{\eta}{2} = 1 - \frac{\eta}{2}$. As P has a measure with respect to the Lebesgue measure λ, P is continuous with respect to λ by the Theorem of Radon-Nikodyn. Hence there exists a $\delta > 0$ such that for every Borel set A we have $\lambda(A) < \delta \Rightarrow P(A) < \frac{\eta}{2}$.

Let $K' = K \cap \{x|signf(x) = 1\} = K \cap f^{-1}([0,\infty[)$. As f is continuous, $f^{-1}([0,\infty[)$ is closed and K' is compact. Therefore, there exists a finite set of boxes Q_i such that $K' \subseteq Q := \bigcup_i Q_i$ and $vol(K') > vol(Q) - \delta$. Let $g = 2 \cdot 1_Q - 1$ (which defines a logical classifier based on the discretization induced by the Q_i), then $g(x) \neq signf(x)$ implies $x \notin K$ or $x \in Q \backslash K'$. It follows that

$$\begin{aligned} P(g(x) \neq signf(x)) &\leq P(x \notin K) + P(Q \backslash K') \\ &< \frac{\eta}{2} + \frac{\eta}{2} = \eta \end{aligned}$$

□

Of course, this theorem is only of theoretical interest because it does not allow to estimate the number of discrete values needed to achieve a certain error. For the goal of understandability, we need to find logical classifiers with a small number of discrete values.

Several algorithms for logical approximations of numerical functions, a task which is also called rule extraction, have been proposed in the literature, see [Andrews and Diederich, 1996]. The main evaluation criteria for rule extraction algorithms according to [Craven and Shavlik, 1999] are the comprehensibility of the extracted rules, their fidelity (rate of agreement between rules and numerical classifier), their accuracy, and the scalability and generality of the extraction algorithm. According to the authors, the last two points are the main weaknesses of existing algorithms.

The most well-known rule extraction algorithm is the Trepan algorithm [Craven and Shavlik, 1996]. Trepan is a most general approach based on decision tree construction [Quinlan, 1986]. Trepan effectively makes use of the property that the numerical learner can label any given observation, such that theoretically infinitely many training examples are available. In the Trepan decision tree, each leaf represents a region in the input space and the corresponding example distribution $P(x)$. The algorithm starts with a decision tree consisting of only one leaf, which represents the whole example set. In each iteration it selects the leaf v where the approximation is worst according to the measure

$$Err(v) = number(v) \cdot (1 - fidelity(v))$$

where $number(v)$ is the estimated number of examples that fall into this leaf when classified by the tree and $fidelity(v)$ is the estimated approximation accuracy of the leaf. For the selected leaf, a number of additional examples is generated by estimating the probability $P(x)$ in this leaf based on the training examples and drawing additional observations for the numerical model to classify. This additional information is used to find the optimal split for this leaf.

4.2.1 Empirical Limits for Rule Extraction

A problem of rule extraction from black-box classifiers like the Trepan approach is that although it is theoretically possible to generate infinitely many example labels from the classifier, the problem of which unlabeled x to choose is more tricky than it seems. Generally, it is impossible to approximate a classifier with zero error and hence to minimize the error one has to concentrate on the most likely parts of the input space. For example, in figure 4.8 the logical classifier (red) approximates the nonlinear decision line (blue) on a certain part of the input space (green). Without the space restriction, the region where both classifiers disagree could have an infinite volume.

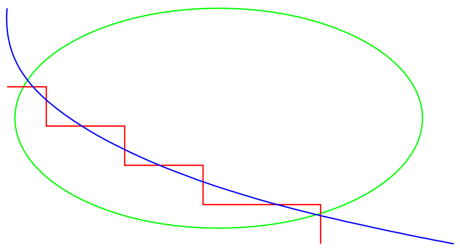


Figure 4.8: A Logical Classifier Approximates a Numerical Classifier

Using an estimated distribution like in the Trepan algorithm allows to approximate the classifier more closely with respect to this distribution, but it is not clear whether this holds for the true distribution. In the worst case, the extracted rules build up much useless complexity in regions with low probability or are biased towards artificial examples, sacrificing accuracy on the true training examples. This becomes a problem when the size of the extracted rule set is bounded to improve interpretability. In short, a heuristic for machine learning says to never solve a more complex problem (density estimation) in order to solve a more simple one (rule extraction).

Another conceptual problem of rule extraction is the following: as the numerical classifier approximates the data and the extracted rule approximates the numerical classifier, the extracted rule also approximates the data. So, how far can one distinguish between logical rules that predict the data and rules that describe the classifier? In particular, when two such classifiers predict the data with an error of ε, both cannot disagree more than 2ε and on the average one would suspect an error of only ε. This has to be taken into account when the quality of an approximation algorithm is evaluated.

Accordingly, the goal of this section is to investigate two problems:

- How does a logical classifier that predicts the data differ from a logical classifier that predicts the output of a numerical classifier trained on the data?
- How much better than the trivial method of training a logical classifier on the numerical classifiers outputs can a rule extraction method perform?

Numerous approaches for extracting rules from numerical models exist, for example [Craven and Shavlik, 1996, Nunez et al., 2002, Chen, 2004]. However, none of these approaches considered the above questions.

Predicting the True Class vs. Predicting the Classifiers Output

In the following experiments, linear and radial basis Support Vector Machines are used as numerical classifiers, while the logical approximation is generated by a decision tree classifier. For each test, two decision trees are generated, one predicting the data and one approximating the SVM model. The goal is to investigate the relationship between both decision trees (see Figure 4.9). In other words, is it empirically possible to distinguish both learning tasks?

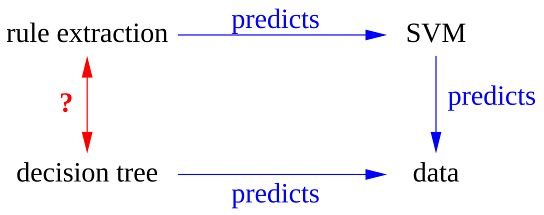


Figure 4.9: First rule extraction problem

The following table shows the results of using the J48 decision tree algorithm to predict the true class y_i (columns labeled "class") and predicting the output of the SVM $f(x_i)$ (columns labeled "SVM"). In both cases, the columns give the disagreement rate, that is the fraction of examples that are predicted differently by the SVM and the decision tree. The final column in each case shows whether the disagreement rate of the tree predicting the SVM output is significantly lower than the disagreement rate of the regular decision tree ("++" denotes significance at the 0.99 confidence level and "+" denotes significance at the 0.95 confidence level).

Name	linear SVM			radial basis SVM		
	class	SVM	Sig	class	SVM	Sig
Business	0.165	0.133	o	0.147	0.145	o
Covtype	0.182	0.052	++	0.159	0.093	++
Diabetes	0.132	0.066	++	0.142	0.080	++
Digits	0.010	0.009	o	0.008	0.015	o
Physics	0.301	0.095	++	0.293	0.146	++
Ionosphere	0.119	0.105	o	0.071	0.073	o
Liver	0.318	0.220	+	0.269	0.240	o
Medicine	0.230	0.040	++	0.191	0.063	++
Mushroom	0.000	0.000	o	0.001	0.001	o
Promoters	0.256	0.238	o	0.180	0.180	o
Insurance	0.002	0.002	o	0.008	0.005	++
Balance	0.165	0.149	o	0.142	0.156	o
Dermatology	0.000	0.000	o	0.016	0.016	o
Iris	0.013	0.013	o	0.013	0.013	o
Voting	0.020	0.000	++	0.038	0.025	o
Wine	0.061	0.066	o	0.220	0.135	++
Breast	0.030	0.019	o	0.029	0.026	o
Garageband	0.281	0.225	++	0.267	0.210	++

We can see that for each learner only on 7 out of 18 data sets the task of predicting the SVM output instead of the true class produced significantly better results at approximating the SVM. This shows that in most of the cases, the task of predicting the SVM output is empirically very hard to distinguish from the task of predicting the true labels. It follows that a regular decision tree should always be considered as a benchmark approach.

Practical Limits for Approximating a Classifier

Every probability estimation in an intermediate step of rule extraction invariably introduces the risk of constructing a sub-optimal approximator with respect to the true distribution $P(x)$. Hence, the most straight-forward way is to extract rules only based on the given training examples of the numerical classifier. But in this case, finding an optimal rule extractor gives rise to the well-known problem of comparing two classifiers based on a finite number of examples, upon which only a finite number of classifiers can be distinguished.

The idea of this section is that one can infer the bound on the proportion of misclassified examples that an approximator has to exceed in order to be distinguishable from an assumed optimal approximator. Assume there is an approximator with zero empirical approximation error. As the error is computed on n examples, the Bayes-optimal estimator of the true error rate is $\hat{\pi} = \frac{1}{n+2}$ [Polasek, 1997]. This allows to directly compute the confidence region of a $B(\hat{\pi}, n)$ binomial distribution.

The following experiment employs the direct rule extractor that learns a decision tree on the given observation with the numerical learners predictions as labels. A linear and a radial basis SVM are used as numerical classifiers. The following table checks whether the disagreement rate of the decision tree predicting the SVM is significantly higher than an assumed optimal approximator of the SVM with a disagreement rate of $\frac{1}{n+2}$, where n is the number of test examples. The numbers in the columns give the number of cross-validation runs where the approximating tree had a higher disagreement rate than the default with a confidence of 0.99 or 0.95, respectively. There were 10 cross-validation runs in total.

Name	linear SVM		RBF SVM	
Confidence level	0.99	0.95	0.99	0.95
business	0	1	1	1
covtype	10	10	10	10
diabetes	6	7	8	10
digits	0	0	0	0
physics	10	10	10	10
ionosphere	3	5	0	4
liver	9	10	10	10
medicine	10	10	10	10
mushroom	0	0	0	0
promoters	2	4	1	2
insurance	0	4	7	9
balance	9	10	10	10
dermatology	0	0	0	0
iris	0	0	0	0
voting	0	0	0	0
wine	0	1	0	3
breast	0	0	0	1
garageband	10	10	10	10

We can see that on 6 data sets for the linear SVM and on 5 data sets for the radial basis SVM, the disagreement rate was never significantly higher than the theoretical optimum. On 11 data sets for the linear SVM and on 10 data sets for the radial basis SVM, the disagreement rate was significantly higher on at most half of the runs.

This shows that in most cases, the second simple approach, which predicts the SVM decision function on the training examples, cannot be significantly improved. This result holds for all rule extractors, as no assumptions about the algorithm have been made. Note that this result is even an optimistic bound on the ability of a SVM approximator to produce better results than the standard decision tree, in general one must expect an approximation algorithm to already have a higher disagreement rate than $\frac{1}{n+2}$ on a test sample. In conclusion, for most learning problems a very simple rule extraction approach is already optimal.

4.3 Visualization of Support Vector Machines

The trick of a good visualization method is that it displays the relevant structure of the object of interest. For the visualization of Support Vector Machines, many well-known visualization methods [Keim, 2002] can of course be used. This section will deal with visualization a a dimension reduction problem with the goal of retaining most of the relevant structure with respect to the SVM.

The structure of the possible SVM hypotheses is given by the Reproducing Hilbert Kernel Space of the SVM kernel (see Section 2.1.3). To represent the structure of the data in the RKHS and for dimension reduction with respect to the features extracted by the kernel, kernel principal component analysis (kPCA) has been proposed [Schölkopf et al., 1999].

kPCA is an extension of the regular (linear) principal component analysis (PCA). The idea of PCA is shown in Figure 4.10: Given a set of data, the vector along which the data shows the most variance is the first principal component. Given the first i principal components, the $i+1$-st principal component is the vector orthogonal to the first i principal components along which the data shows the most variance. It follows that the best reconstruction of the data in an i-dimensional subspace is given by the first i principal components. Computationally PCA reduces to a eigenvector decomposition of the covariance matrix.

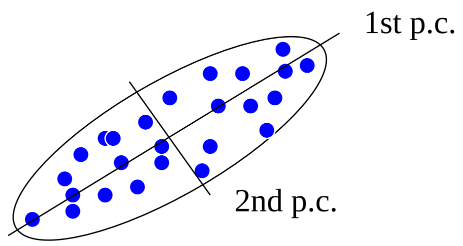


Figure 4.10: Linear Principal Component Analysis

A useful property of PCA is that it solely depends on the inner product and hence one can use the kernel trick to develop a nonlinear variant, the kPCA. Let the kernel K be given by the nonlinear mapping Φ, i. e. $K(x, x') = \Phi(x) * \Phi(x')$.

To perform PCA in feature space, the covariance matrix C

$$C = \frac{1}{n}\sum_{i=1}^{n}\Phi(x_i)\Phi(x_i)^t$$

is investigated. Here we assume that the data is centered in feature space, that is $\sum_{i=1}^{n}\Phi(x_i) = 0$. The task is to find eigenvalues λ and eigenvectors V that satisfy

$$\begin{aligned} & \lambda V = CV \\ \Leftrightarrow\quad & \lambda V = \frac{1}{n}\sum_{i=1}^{n}\Phi(x_i)\Phi(x_i)^t V \\ \Leftrightarrow\quad & \forall_{k=1}^{n} : \lambda\Phi(x_k)^t V = \Phi(x_k)^t C V \end{aligned}$$

where the last equation follows because all solutions V lie in the span of the $\Phi(x_k)$, i. e. $V = \sum_{i=1}^{k}\alpha_i\Phi(x_i)$. Note that there are n principal components V_k, each with a weight vector $\alpha^{(k)} = (\alpha_1^{(k)}, \ldots, \alpha_n^{(k)})$. This yields the equation

$$n\lambda K\alpha = K^2\alpha$$

which is equivalent to finding the solutions of

$$n\lambda\alpha = K\alpha$$

because all solutions $\alpha^{(k)}$ of the latter equation satisfy the former and it can be shown that any additional solutions of the latter equation do not make a difference in the expansion of V_k. Finally, the eigenvectors are normalized in feature space which translates to setting $|||\alpha^{(k)}||^2 = 1/\lambda_k$. Principal component extraction for an observation x is computed by projecting x on the eigenvectors V_k, i. e. computing

$$\begin{aligned} V_k\Phi(x) &= \sum_{i=1}^{n}\alpha_i^{(k)}\Phi(x_i)\Phi(x) \\ &= \sum_{i=1}^{n}\alpha_i^{(k)}K(x_i, x). \end{aligned}$$

For non-centered data, i. e. $\sum_{i=1}^{n}\Phi(x_i) \neq 0$ the vectors $\Phi(x)$ are replaced by their centered counterparts

$$\tilde{\Phi}(x) = \Phi(x) - \frac{1}{n}\sum_{i=1}^{n}\Phi(x_i).$$

The kPCA solution can then be computed using the kernel matrix $\tilde{K}$ corresponding to $\tilde{\Phi}$:.

$$n\lambda\vec{\alpha} = \tilde{K}\vec{\alpha}$$

It can be shown that $\tilde{K}$ can be written as

$$\tilde{K} = K - 1_nK - K1_n + 1_nK1_n$$

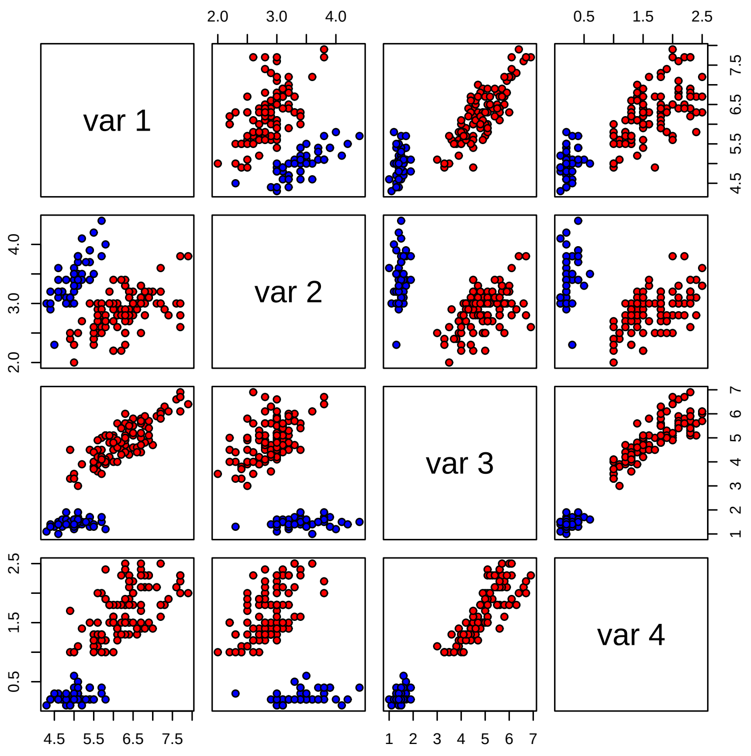


Figure 4.11: Iris Data Set

where 1_n is the $n \times n$ matrix with entries $1/n$ (see [Schölkopf et al., 1998b]).

Kernel PCA can be used to extract features from the data in order to show up the structure defined by the kernel. For example, let us take a look at the well-known Iris data set [Fisher, 1936]. Figure 4.11 shows the scatterplot matrix of this 4-dimensional data set. The data set consists of three species of flowers, where one of the species (Setosa) is marked blue in the plots. Figure 4.12 shows 9 plots of the same data set, projected on the first and third feature (i. e. every plot of Figure 4.12 corresponds to the plot in the third row of the first column of Figure 4.11). This time, the colors encode the values of points projected on the first 9 principal components. A red point marks a negative value and a blue point marks a positive value. One can see that the first principal component separates the lower cluster from the upper cluster, the second principal component divides the upper cluster in half and the following principal components highlight other, more complex structures in the data (these structures may be better observable in plots of other attributes).

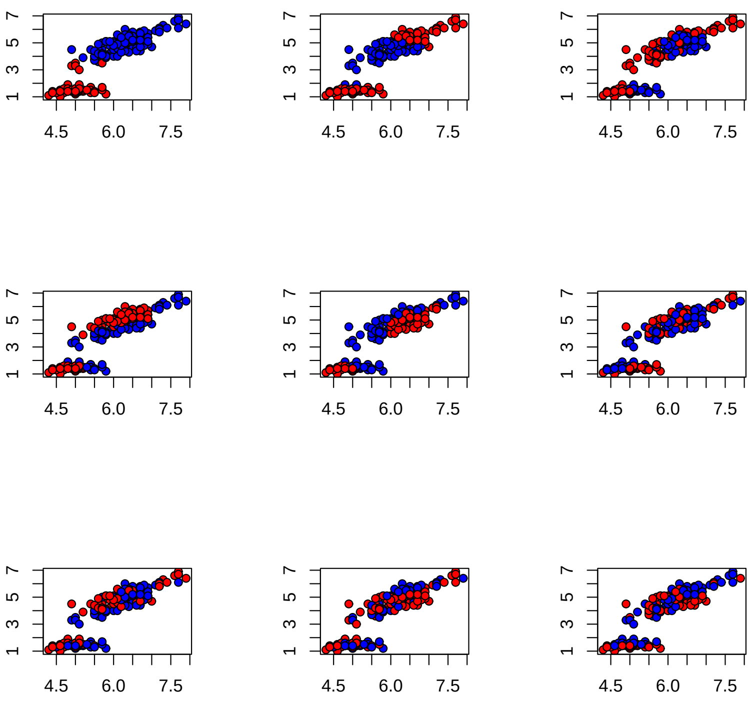


Figure 4.12: Kernel Principal Component Analysis

4.3.1 Projected Kernel Principal Component Analysis

Kernel PCA is a useful tool for extracting the structure of the data and the kernel itself, but it does not include information about the classification task and SVM solutions. The problem is that the class structure may not correspond to the variance structure extracted by the PCA. It may very well be the case the the SVM hyperplane is oriented along the smallest principal component and that plot along the largest components do not exhibit any useful structure with respect to classification. It may also be the case that the SVM prediction is more or less correlated with the projections on all principle components, such that there is no low-dimensional projection that includes sufficient information to analyze the SVM hyperplane.

To circumvent this problem, information about the SVM hyperplane can be incorporated into Kernel PCA. The idea is to use the SVM hyperplane vector w as the first principle component. In this way, all information about the SVM prediction is bundled into the first component while all other principle components are uncorrelated to the SVM and hence include a maximum of additional information. This novel variant of kPCA will be called Projected Kernel Principal Component Analysis (pkPCA).

To define the pkPCA algorithm, let $w = \sum_{i=1}^{n} \beta_i \Phi(x_i)$ be a SVM solution. Assume that w is normalized to $||w|| = 1$. To force kPCA to take w as the first principal component, the data is projected onto the orthogonal complement of w and a $kPCA$ is carried out with the transformed data. This forces w in the span of the eigenvectors with zero eigenvalue, which can be ignored and replaced by w in the later feature extraction step.

Basic linear algebra states that the projection $P\Phi(x)$ of $\Phi(x)$ onto the orthogonal complement of w is given by

$$P(x) \quad := \quad \Phi(x) - ww^t\Phi(x).$$

and it is straight-forward to show that P defines a new kernel function with kernel matrix

$$\tilde{K} \quad = \quad K - KBK$$

where K is the kernel matrix of Φ and $B = \beta\beta^t$. Similar to kPCA, a centering step is carried out by replacing $P(x)$ with

$$\bar{P}(x) = P(x) - \frac{1}{n}\sum_{i=1}^{n} P(x_i).$$

with corresponding kernel matrix

$$\bar{K} \quad = \quad \tilde{K} - 1_n\tilde{K} - \tilde{K}1_n + 1_n\tilde{K}1_n.$$

Projected Kernel PCA is then equivalent to solving the kPCA eigenvector problem with the new kernel matrix $\bar{K}$.

Figure 4.13 shows the results of pkPCA on the Iris data set. The SVM used was generated by defining a classification tasks to separate the Setosa species (lower cluster) from the rest. Notice that none of the principal components separated the lower cluster from the rest, as this is already done by the SVM decision function (the projection on w is not plotted here). Instead, independent structure is extracted by the PCA.

To validate the independence of the extracted structures, the following table show the correlation between the SVM parameter vector β and the 10 largest principal components $\alpha^{(k)}$ for both the usual kPCA and the new pkPCA.

#	kPCA	pkPCA
1	-0.123	-3.525e-17
2	-0.108	-4.084e-17
3	0.153	1.034e-16
4	-0.122	-6.669e-17
5	-0.206	1.969e-17
6	-0.071	-2.241e-16
7	-0.091	-6.055e-17
8	0.027	-4.816e-16
9	0.072	-7.158e-16
10	0.081	-4.127e-16

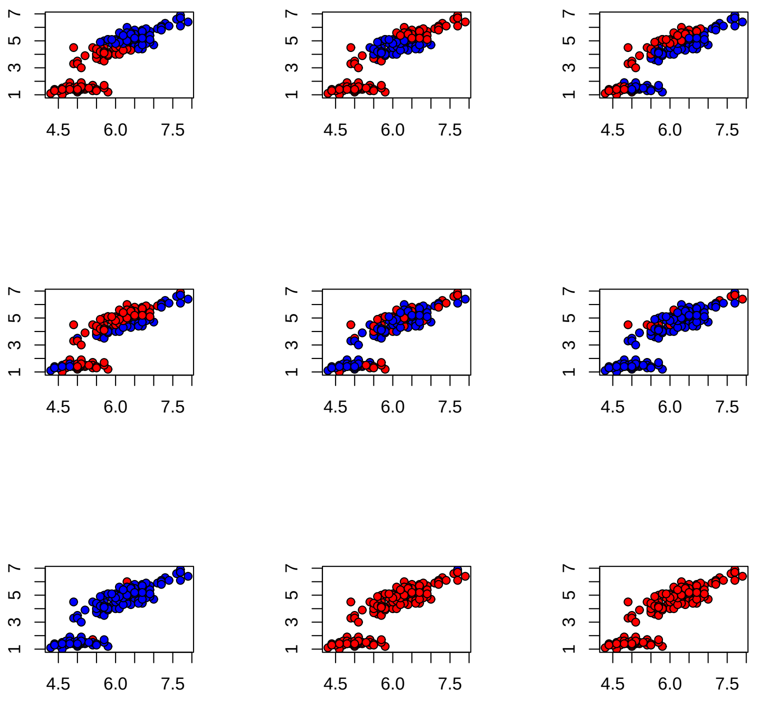


Figure 4.13: Projected Kernel Principal Component Analysis

It can be seen that there is a considerate amount of correlation in the $kPCA$ (although the first kPCA component already provides a good separation of both clusters, see Figure 4.12), while the pkPCA is uncorrelated. This shows that pkPCA is much better suited for describing the SVM-relevancy of the structure in the data than usual kPCA.

The plots generated by this technique optimally separate the class structure that the SVM extracts (i.e. the decision function) and the structure that the SVM could theoretically extract, but does ignore (uncorrelated components of the same functional form). Still, the analysis of these plots in order to find out whether the SVM is missing some information in the data that could be extracted, e.g. by a feature transformation, lies in the responsibility of the user.

4.4 Conclusions

Detailed information about a classifiers internals allows many approaches to construct a provably optimal transformation of the classifier into a more understandable form. This chapter has presented three such transformations of Support Vector Machines: into formulas, rules, and plots.

Section 4.1 embedded the Reproducing Kernel Hilbert Space of SVMs into a larger space, which is even more complex, but which contains simple functions with high approximation quality. From these functions, an approximation can be found that is both mathematically well-defined and desirable from understandability aspects.

A negative result was obtained in Section 4.2: even with complete knowledge about the classifier's hypothesis space it may be not possible to transform the classifier into a very different representation – in this case rules – that is more sensible than a very trivial approach when the variance from the finite number of examples is taken into account.

Finally, Section 4.3 introduced a new method for the visualization of Support Vector Machines that is based on a dimension reduction of the hypothesis space that is maximally informative about the decision function.

The most important result of this chapter is that the optimal transformation of an accurate classifier into an understandable form depends largely on the type of features and the structure the learner internally extracts from the data – e.g. kernelized principal components and Reproducing Kernel Hilbert Spaces. When this structure is known, more meaningful results can be obtained than in the case where this structure has to be estimated from the data, as it was the case in the last chapter. When this structure does not fit to the language the model is to be described in, as in Section 4.2, only trivial transformations can be performed.

Chapter 5

Detecting Local Patterns

The previous two chapters dealt with the task of describing a model itself; the question of interest was "how does the model predict the data?" In this chapter, the goal is slightly different. Instead of describing the model, the focus lies on the performance of the model, that is the question becomes "how good does the model classify the data"[1]. To answer this question, this chapter will look for regions of remarkably high or low classification accuracy. This leads us to the problem of local pattern detection.

Local pattern detection [Hand et al., 2002, Morik et al., 2005] is defined as the unsupervised search for local accumulations of data points with unexpected high density with respect to some background model [Hand, 2002]. For example, in a retail store the background model may state that customers decide to buy an item independently of other items. Deviations from this model are items which are often bought together (e.g. pretzels and beer) or items which are seldomly bought together (e.g. different brands of dish washing detergent), and these are very interesting to retail stores to optimize their assortment of goods. The search for frequent combinations of item has become very popular under the name of "association rule mining" [Agrawal and Srikant, 1994].

Local pattern detection has similarities to the field of robust statistics (see Section 2.3) . They share the idea that examples do not only come from one distribution, but instead a small fraction of examples comes from a different distribution. But in contrast to local models, robust statistics usually is not interested in describing these local exceptions, but in removing their influence. In this sense, local pattern detection and robust statistics are complementary problems.

This chapter deals only with local patterns from the view of a classifier, that is we are only interested in subsets of the data that deviate from the rest of the data with respect to the ability of predicting their label. See for example Figure 5.1. In terms of usual unsupervised local pattern detection, the large circle on the right is a likely background model, stating that all data lies in this circle, while the small circle on the right then becomes a local pattern. In the case of supervised local pattern detection, the vertical line dividing the large circle on the right is a probable background model, declaring all points on the left as positive and all points on the right as negative. In contrast to the unsupervised

[1]Note that this is a different question than "how sure is the classifier of its prediction?", which is the focus of probabilistic classifiers (Section 2.4)

case, the smaller circle on the left only becomes a local pattern if its examples are negative, because the focus lies on predicting the class, not describing the location of the data.

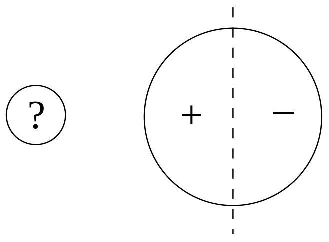


Figure 5.1: Background model and a possible local pattern.

The problem of finding local patterns for classification can be formalized as follows:

Definition 5.0.1 (Predictive Local Pattern Problem)**.** Given examples $(x_i, y_i)_{i=1}^n$ i.i.d. drawn from a probability distribution P, a learner L, and a number $\tau \in]0,1[$, the predictive local pattern problem is to find a subset C of the input space X such that

1. $P(C) < \tau$ (locality constraint)
2. the predictive performance of L on $X \backslash C$ is maximized.

□

The definition states the goal of minimizing the errors on the global part, which is equivalent to concentrating the errors in the local part. More intuitively, the Predictive Local Pattern Problem can be stated as: Given Hand's equation [Hand, 2002]

$$Data = Global\,Model + LocalModels + Noise,$$

give me the regions C where

$$Data = Global\,Model + Noise$$

still holds, taking the same level of noise in both cases.

Three fundamental questions arise from the definition of the Predictive Local Pattern Problem.

Verification: Given a set C, is it a predictive local pattern? To answer this question, one has to empirically verify that $P(C) < \tau$ and that the performance of L on $X \backslash C$ is significantly higher than its performance on X. These are readily solvable statistical standard problems.

Existence: Given a set of examples, does a predictive local pattern exist? In general, this question cannot be answered, because if a local pattern exists, the performance gain of L could be arbitrarily small, while there is a fixed minimal performance value that can be distinguished from zero using a finite set of examples. If the problem is restricted to find only local patterns on the training data, it can be solved by enumerating all possible $\sum_{k=1}^{\tau n} \binom{n}{k}$ possible local patterns.

Optimality: Given a local pattern C, is it optimal, i.e. does no other local pattern exists, on which L has a higher performance gain? Again, this problem cannot be solved from a finite set of training data, as only finitely many patterns can be distinguished from this data set, but there are infinitely many possible patterns and any two patterns that coincide on the training data may have a significantly different true performance difference.

In conclusion, the problem of identifying predictive local patterns suffers from the same theoretical problems as supervised learning in general: it is possible to verify the existence of a given structure in the data, but it is not possible to verify if the data contains any structure at all (see e.g. the definition of Kolmogorov complexity in Section 2.1.4). Hence, the set of admissible patterns C has to be restricted to form a practically solvable problem. A specific instance of the Predictive Local Pattern Problem is defined by the hypothesis language $L_C \subset \mathcal{P}(X)$ of admissible patterns C.

How much performance gain can we expect from the local pattern approach? Given a global classifier G with error ε, the best possible local pattern C contains as much errors of G as possible. This leaves a classifier with $\max\{\varepsilon - \tau, 0\}$ errors on a set of size $1 - \tau$. The worst case is that the local pattern covers none of the errors, which implies an increased error rate due to the smaller region of size $1 - \tau$. Hence, assuming $\varepsilon \geq \tau$, for the error rate ε_G on the global part the following bounds hold:

$$\frac{\varepsilon}{1-\tau} \geq \varepsilon_G \geq \frac{\varepsilon - \tau}{1-\tau} \tag{5.1}$$

It follows that for fixed τ the relative error $\frac{\varepsilon_G}{\varepsilon}$ is an appropriate measure to quantify the performance of a predictive local pattern procedure.

For the criterion of interpretability, the performance of this approach largely depends on the choice of the hypothesis language L_C and hence on the local pattern learner. A plethora of learners could be applied for this task and it is practically impossible to validate the interactions of global learner and pattern learner for all such combinations. To focus the investigation, this chapter will concentrate on two approaches motivated by the interpretability heuristics from Section 1.1.2. The most important heuristic says that whenever possible the user should select the learner to use. Given that the user has already selected the global learner as being understandable, it is a good idea to use the same learner to describe the local patterns. This will be the main approach pursued in this chapter. The other approach is based on the heuristic to stay close to the data and avoid complex formal models, which proposes the use of a prototype-based clustering algorithm for finding local patterns.

The rest of this chapter presents three approaches to local pattern detection. Section 5.1 investigates how local patterns can be used to describe a classifier by identifying the regions where the classifier fails to perform well. In Section 5.2, the information of the local patterns will be used to improve the classifier on the rest of the data. Section 5.3 finally will investigate unsupervised approaches to local patterns.

5.1 Describing a Classifier by Local Patterns

The crucial part of local pattern detection is finding the right hypothesis space for the local patterns. This becomes even more important when interpretability constraints are involved: as the user is supposed to not only understand the global model, but also the description of the patterns, the question arised how many different representations one can expect the user to become aquainted with. As the used is expected to understand the global model anyway, from an interpretability point of view the obvious solution is to encode the global model and the patterns in the same hypothesis language.

This suggests to implement a predictive local pattern finder by first using the learner to find a global model and then using the same learner again to predict for each example whether the global models prediction will be correct. To ensure that the locality constraint is met, we assume that the learner returns a numerical or probabilistic classifier f (a classifier can be cast into a probabilistic classifier via the methods presented in Chapter 2.4). We can then find a threshold t such that the classifier f exceeds t only on a fraction of τ of the training examples and define the local pattern as $C = \{x|f(x) > t\}$, which is the region where the pattern detection classifier is most sure the global classifier will mispredict.

Classifiers with a strongly restricted hypothesis space like linear classifiers may profit from this combination as the hypothesis space will be significantly larger. For example, it may be possible to improve accuracy be inverting the global classifier on the local pattern, should the error there exceed 50%. More flexible classifiers like radial basis SVMs or logical classifiers, which can adapt to the data very much, will usually not be improved by this approach, because the hypothesis space will not gain much expressibility from using two classifiers instead of one. If it were possible to improve the classifier by using some more rules or radial basis functions, the global learner would likely have found these rules in the first place. However, it is important to remember that the goal was not to improve the predictions of the global classifier, but to identify regions with higher error rates, independently of whether it is possible to improve the classifier on this regions. For example, if the overall error rate of the classifier is 5%, identifying a region with 20% error will be highly informative, even if one cannot readily see how the classifier could be improved.

The following table gives the results of this approach on the 18 standard data sets. The most important information is given in the second column, it contains the error of the classifier on the global region in relation to the error on the complete data set, i.e. $Err_{rel} = \frac{Err_{global}}{Err}$. A low relative error means that the local pattern captures many of the incorrectly predicted examples, as the points from the local pattern are present in the complete data set but missing in the global part. The relative error is reported instead of the absolute difference in errors because it is obviously easier to remove errors if many of them are present. The following columns of the result table contain the error rates on the global part and on the local pattern and the fraction of test examples classified as local examples. The final column indicates whether the error rate on the global part alone is significantly lower than the error rate on all examples (computed with a paired t-test. ++ indicates a 99% level of confidence for a lower error and + a 95% level of confidence. $--$ and $-$ indicate the same confidences for a significantly larger error. Finally, o indicates confidences below 95%). The classifiers used were the linear SVM, the radial basis SVM, J4.8 and JRip. The local fraction was set to $\tau = 0.1$ and 10-fold cross-validation was used.

Here are the results using the linear SVM classifier.

Name	Rel. Err	Err Global	Err Local	Local	C $\geq$ G
business	0.925	0.127	0.233	0.116	*o*
covtype	0.996	0.235	0.242	0.105	*o*
diabetes	0.974	0.216	0.280	0.096	*o*
digits	1.000	0.001	0.000	0.083	*o*
physics	0.980	0.310	0.365	0.111	++
ionosphere	0.557	0.084	0.317	0.170	++
liver	0.974	0.301	0.349	0.095	*o*
medicine	0.872	0.239	0.566	0.105	++
mushroom	0.822	0.000	0.006	0.017	*o*
promoters	0.970	0.092	0.066	0.047	*o*
insurance	0.528	0.005	0.068	0.076	++
balance	0.885	0.040	0.112	0.088	+
dermatology	1.000	0.000	0.000	0.000	*n.a.*
iris	1.000	0.000	0.000	0.000	*n.a.*
voting	0.695	0.031	0.205	0.094	+
wine	1.018	0.012	0.000	0.066	*o*
breast	1.047	0.032	0.014	0.098	*o*
garageband	0.975	0.285	0.321	0.179	+

The result using the radial basis SVM classifier:

Name	Rel. Err	Err Global	Err Local	Local	C $\geq$ G
business	0.742	0.094	0.241	0.228	+
covtype	0.916	0.219	0.368	0.133	++
diabetes	0.933	0.225	0.322	0.123	+
digits	1.011	0.004	0.000	0.077	*o*
physics	0.969	0.317	0.393	0.128	++
ionosphere	0.502	0.038	0.181	0.208	++
liver	0.913	0.283	0.414	0.182	*o*
medicine	0.848	0.235	0.653	0.100	++
mushroom	1.000	0.017	0.000	0.000	*o*
promoters	1.000	0.123	0.000	0.000	*o*
insurance	0.036	0.002	0.568	0.103	++
balance	0.717	0.007	0.064	0.086	*o*
dermatology	1.000	0.016	0.000	0.000	*o*
iris	1.000	0.000	0.000	0.000	*n.a.*
voting	0.667	0.041	0.348	0.096	++
wine	0.954	0.221	0.200	0.090	*o*
breast	0.325	0.006	0.264	0.087	++
garageband	0.978	0.283	0.294	0.471	*o*

The result using the JRip classifier:

Name	Rel. Err	Err Global	Err Local	Local	C $\geq$ G
business	1.000	0.227	0.000	0.000	*o*
covtype	0.981	0.245	0.219	0.019	+
diabetes	0.973	0.260	0.174	0.024	*o*
digits	1.000	0.005	0.000	0.000	*o*
physics	0.988	0.342	0.411	0.069	+
ionosphere	0.892	0.084	0.300	0.022	+
liver	0.971	0.332	0.180	0.020	*o*
medicine	0.972	0.252	0.250	0.028	+
mushroom	1.000	0.001	0.000	0.000	*o*
promoters	1.000	0.180	0.000	0.000	*o*
insurance	0.989	0.061	0.366	0.001	+
balance	0.971	0.110	0.100	0.010	*o*
dermatology	1.000	0.000	0.000	0.000	*n.a.*
iris	1.000	0.013	0.000	0.000	*o*
voting	0.878	0.046	0.083	0.011	*o*
wine	1.000	0.095	0.000	0.000	*o*
breast	0.902	0.037	0.116	0.014	+
garageband	0.992	0.290	0.141	0.015	*o*

The result using the J48 classifier:

Name	Rel. Err	Err Global	Err Local	Local	C $\geq$ G
business	1.000	0.229	0.000	0.000	*o*
covtype	0.999	0.260	0.055	0.003	*o*
diabetes	1.002	0.270	0.158	0.020	*o*
digits	1.000	0.007	0.000	0.000	*o*
physics	1.000	0.395	0.000	0.000	*o*
ionosphere	1.000	0.102	0.000	0.000	*o*
liver	1.001	0.348	0.033	0.014	*o*
medicine	0.994	0.228	0.095	0.018	*o*
mushroom	1.000	0.002	0.000	0.000	*o*
promoters	1.020	0.194	0.000	0.018	*o*
insurance	1.000	0.012	0.000	0.000	*o*
balance	1.000	0.128	0.000	0.000	*o*
dermatology	1.000	0.000	0.000	0.000	*n.a.*
iris	1.000	0.013	0.000	0.000	*o*
voting	1.000	0.036	0.000	0.000	*o*
wine	1.000	0.050	0.000	0.000	*o*
breast	1.000	0.038	0.000	0.000	*o*
garageband	0.990	0.316	0.262	0.007	*o*

This approach fails to perform as expected, which can be clearly seen from the following summary table. It reports for each classifier the average relative error (first column) the number of data sets with a significant reduction of error (second column), the number of times no local pattern could be found (third column) and the number of times the fraction of test examples assigned to the local pattern exceeded the local threshold τ by more than 0.01 (fourth column).

Classifier	rel. err	sig. err	no local	local $> \tau$
linear SVM	0.901	7	2	4
RBF SVM	0.806	9	4	7
JRip	0.973	6	7	0
J48	1.000	0	12	0

Although the error reduction is good in some cases, it can be seen that in many cases the classifiers either assigned too many examples to the local pattern or found no local pattern at all. In total, in 36 of the 4*18 tests (50%), the approach did not perform as expected. Additionally, in some cases the good performance in error reduction may be due to the fact that the local pattern contains much more examples, and hence much more errors, than allowed; see for example the ionosphere data set for the radial basis SVM, which has a relative error of 0.5 but contains twice as much local examples than allowed. The next section will explain this behaviour of the local patterns.

Adapting the Conditional Class Probability Threshold

The reason for the suboptimal behaviour of the direct approach can be found by taking a closer look at the models learned for predicting the local pattern. Figure 5.1 shows the values of the linear SVM decision function for predicting the errors on the business data set. Obviously, the SVM has learned the default negative hypothesis, with very low variation around the value of $f(x) = -1$. The maximum decison function value is -0.997. Similarly, the decision tree learner has learned the default negative model consisting of only one node. The difference is that while the decision tree prediction is constant for all examples, which prohibits the identification of local patterns on the basis of the predicted values, the SVM function is still slightly influenced by the mispredicted examples and this slight difference in the predicted values may be sufficient to identify local patterns in the experiments.

This explains the both cases of errors found in the naive approach of finding local patterns: when the global classifiers makes few errors, a local pattern present in the errors is largely covered by the majority of the correctly predicted examples. Hence, the best hypothesis in many cases is to assign all examples to the negative class (no local example). In the cases of piecewise constant classifiers as J48 and JRip, no local patterns can be defined by this model. In the case of numerical classifiers as the two SVMs, slight variations in the numerical function around -1 can be used to define a local pattern by putting a threshold on the decision function. However, given that the differences are very small and random influences are inevitable, it is very likely that many more observations of the test set than allowed are assigned to the local pattern.

A solution to this problem can be found by analysing the optimal classifier for a probabilistic problem, the Bayes classifier.

Theorem 5.1.1 (Bayes-Classifier [Hastie et al., 2001])**.** *Given a probability distribution $P(x, y)$ with $Y = \{-1, 1\}$, the classifier with the minimal expected prediction error is the Bayes classifier given by*

$$\begin{aligned} f_{Bayes}(x) &= \arg\max_y P(y|x) \\ &= \begin{cases} 1 & \textit{if } P(Y=1|x) > \frac{1}{2} \\ -1 & \textit{if } P(Y=1|x) \leq \frac{1}{2} \end{cases} \end{aligned}$$

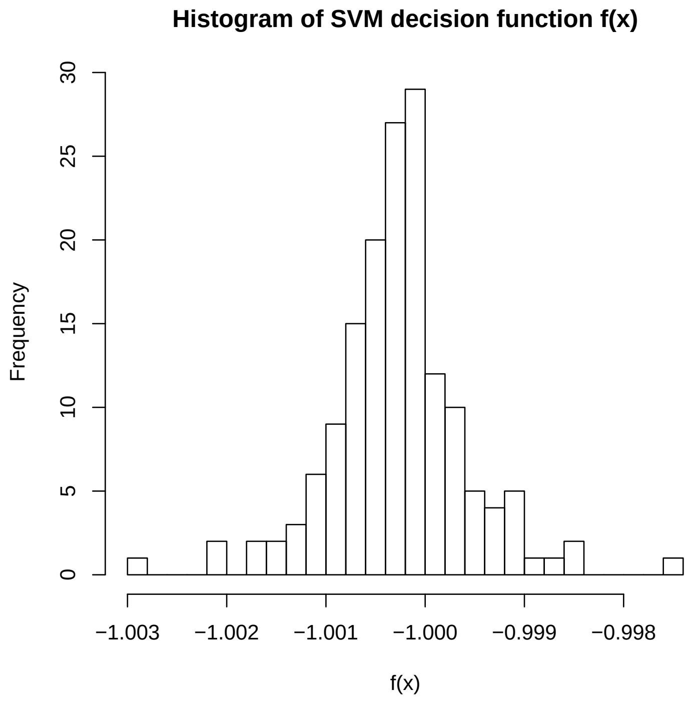


Figure 5.2: Histogram of the SVM decision function for local patterns

Proof. The expected prediction error is equal to the expected risk with respect to the 0-1-loss. We can decompose $P(x, y)$ into $P(y|x)P(x)$. For any classifier $f : X \to Y$ we can write

$$\begin{aligned} R(f) &= \int L_{01}(f(x), y)dP(x, y) \\ &= \int\int L_{01}(f(x), y)dP(y|x)dP(x) \end{aligned}$$

The inner integral is

$$\begin{aligned} \int L_{01}(f(x), y)dP(y|x) &= \int \mathbf{1}_{f(x)\neq y}dP(y|x) \\ &= \begin{cases} P(Y = -1|x) & \text{iff } f(x) = 1 \\ P(Y = 1|x) & \text{iff } f(x) = -1 \end{cases} \\ &\geq \min_y P(y|x) \end{aligned}$$

It follows that

$$\int L_{01}(f_{\text{Bayes}}(x), y)dP(y|x) = \min_y P(y|x).$$

This means that for each x the Bayes classifier attains the minimal error, and from the monotonicity of the integral operator follows that

$$\begin{aligned} R(f) &\geq \int \min_y P(y|x)dP(x) \\ &= R(f_{\text{Bayes}}) \end{aligned}$$

for all classifiers f. □

This theorem tells us that in order to achieve a low classification error, a classifier has to get a good estimate of the decision boundary $\{x|P(y|x) = 1/2\}$. Estimating other values of the conditional class probability $P(y|x)$ is not important for good classification performance. For detecting local patterns we define a new random variable $Y' \in \{-1, 1\}$ where $y' = 1$ iff $y \neq f(x)$ and try to estimate the function $f' : x \mapsto y'$. However, we are actually not interested in finding points x with $P(Y' = 1|x) > 1/2$, that is, points that are more likely to be mispredicted by f than not, but we generalize the problem to finding points with $P(Y' = 1|x) > \alpha$ for some value $\alpha \in [0, 1]$. In particular, we are interested in $\alpha = P(y \neq f_{global}(x))$, meaning we want to find points that are more likely to be mispredicted than what is described by the general misclassification error of the the global classifier. For a reasonable classifier f, the error rate $P(y \neq f(x))$ should be significantly less than 1/2 and hence using a standard learning algorithm may not optimally solve this problem, as it will usually concentrate on getting a good estimate of $\{x|P(y'|x) = 1/2\}$.

There are two approaches to finding an estimator of $P(Y = 1|x) = \alpha$ for a binary random variable Y.

Theorem 5.1.2 (Asymmetric Loss Functions)**.** *Let $L(f(x), y)$ be a loss function such that $L(y, y) = 0$ and*

$$\frac{L(1, -1)}{L(-1, 1)} = \beta.$$

For any distribution $P(x, y)$, the classifier with minimal expected error with respect to L is the classifier

$$f(x) = \begin{cases} 1 & \textit{if } P(Y = 1|x) \geq \frac{\beta}{1+\beta} \\ -1 & \textit{else} \end{cases}$$

Proof. As in the proof of Theorem 5.1.1, it is sufficient to minimize the expected loss with respect to $P(y|x)$ for all x.

$$\int L(f(x), y)dP(y|x) = \begin{cases} L(1, -1)P(Y = -1|x) & \text{iff } f(x) = 1 \\ L(-1, 1)P(Y = 1|x) & \text{iff } f(x) = -1 \end{cases}$$

Obviously, it is optimal to predict 1 iff $L(1, -1)P(Y = -1|x) \leq L(-1, 1)P(Y = 1|x)$. This is equivalent to

$$\begin{aligned} & L(1, -1)P(Y = -1|x) \leq L(-1, 1)P(Y = 1|x) \\ \Leftrightarrow\ & \frac{L(1, -1)}{L(-1, 1)} \leq \frac{P(Y = 1|x)}{P(Y = -1|x)} \\ \Leftrightarrow\ & \beta \leq \frac{P(Y = 1|x)}{1 - P(Y = 1|x)} \\ \Leftrightarrow\ & P(Y = 1|x) \geq \frac{\beta}{1 + \beta} \end{aligned}$$

□

Hence, if we have a learner that can operate with asymmetric loss functions, e.g. Support Vector Machines (see [Rüping, 1999] and Section 2.1.3), we can optimize it to estimate decision lines of other conditional probabilities by setting an asymmetric loss function according to Theorem 5.1.2. This is of course subject to the assumption that the learned classifier approximates the optimal decision function reasonably well. If the classifier performs good at $P(Y = 1|x) = 1/2$ (i.e. it is a good global classifier), this assumption may be valid for other fixed values of $P(Y = 1|x)$ as well (i.e. the classifier is a good for detecting local patterns).

The approach with asymmetric loss functions was tested with Support Vector Machines on the 18 standard data sets. The experimental setup was identical to that of the direct approach reported earlier in this section, except that the loss was adapted according to Theorem 5.1.2 to predict $P(f_{\text{global}}(x) \neq y|x) > P(f_{\text{global}}(x) \neq y)$. That is, for each x an error probability higher than the average error should be predicted by the pattern classifier.

Here are the results using the linear SVM classifier.

Name	Rel. Err	Err Global	Err Local	Local	C $\geq$ G
business	0.936	0.124	0.275	0.114	*o*
covtype	0.959	0.226	0.327	0.094	++
diabetes	0.981	0.221	0.272	0.108	*o*
digits	1.000	0.001	0.000	0.083	*o*
physics	0.953	0.302	0.440	0.102	++
ionosphere	0.771	0.101	0.400	0.091	++
liver	0.949	0.289	0.386	0.095	+
medicine	0.874	0.239	0.562	0.107	++
mushroom	0.822	0.001	0.006	0.017	*o*
promoters	0.970	0.092	0.066	0.047	*o*
insurance	0.143	0.001	0.079	0.107	++
balance	0.742	0.036	0.184	0.100	+
dermatology	1.000	0.000	0.000	0.000	*n.a.*
iris	1.000	0.000	0.000	0.000	*n.a.*
voting	0.535	0.020	0.295	0.094	++
wine	1.018	0.012	0.000	0.100	*o*
breast	1.091	0.033	0.000	0.089	−−
garageband	0.984	0.286	0.415	0.038	*o*

The local error is significantly higher than the global error on 8 of the data sets, as opposed to 7 times with the original approach. On the other hand, on one data set (breast) the local error is significantly lower than the global error, which did not happen with the original algorithm. Now, there is only one data set with a very high fraction of local examples (business with 0.114) instead of 4 with the original approach. This may indicate that the balanced cost approach is not as stable as the original approach. The average relative error is 0.874 instead of 0.901 and the average difference of the error on the local to the error on the global part is 0.095, as opposed to 0.064 for the original algorithm. This is a clear improvement.

Here are the results using the radial basis SVM classifier.

Name	Rel. Err	Err Global	Err Local	Local	C $\geq$ G
business	0.742	0.096	0.483	0.095	++
covtype	0.960	0.229	0.408	0.050	++
diabetes	0.934	0.228	0.422	0.085	+
digits	1.021	0.004	0.000	0.081	*o*
physics	0.954	0.312	0.460	0.103	++
ionosphere	0.612	0.049	0.286	0.068	++
liver	0.980	0.296	0.356	0.081	*o*
medicine	0.830	0.230	0.636	0.115	++
mushroom	1.000	0.017	0.000	0.000	*o*
promoters	1.000	0.123	0.000	0.000	*o*
insurance	0.035	0.002	0.598	0.098	++
balance	0.835	0.009	0.054	0.093	*o*
dermatology	1.000	0.016	0.000	0.000	*o*
iris	1.000	0.000	0.000	0.000	*n.a.*
voting	0.933	0.062	0.114	0.096	*o*
wine	1.053	0.233	0.133	0.123	*o*
breast	0.300	0.006	0.207	0.105	++
garageband	0.987	0.285	0.426	0.021	++

The local error is significantly higher than the global error on 9 of the data sets, which was the same with the original approach, but this time only 2 of the data sets have a local probability greater than 0.11 instead of the original 7 data sets. Hence, the results show that this approach is much better at producing consistent results than the original approach.

In case we cannot influence the loss function, the next theorem gives another method, which influences the conditional class probability at the decision boundary.

Theorem 5.1.3 (Biased Sampling [Domingos, 1999])**.** *Let $P(X,Y)$ be any probability distribution with $0 < P(Y=1) < 1$ and $\alpha \in]0,1[$. Let $\tilde{P}(X,Y)$ be another probability distribution such that*

$$\begin{aligned}\tilde{P}(X,1) &= P(X|Y=1)\alpha \\ \tilde{P}(X,-1) &= P(X|Y=-1)(1-\alpha).\end{aligned}$$

Then the classifier with minimal prediction error for $\tilde{P}$ is given by

$$f(x) = \begin{cases} 1 & \textit{iff } P(Y=1|x) \geq \beta \\ -1 & \textit{else} \end{cases}$$

with

$$\beta = \frac{P(Y=1)(1-\alpha)}{\alpha - 2\alpha P(Y=1) + P(Y=1)}.$$

In particular, for $\alpha = 1/2$ we have $\beta = P(Y=1)$.

Proof. First, notice that $\tilde{P}(X,Y)$ is a convex combination of two probability distributions and hence a valid probability distribution itself. By definition we have $\tilde{P}(Y=1) = \alpha$. A sample of examples distributed by $\tilde{P}$ can be obtained by taking a sample of P and choosing a subsample that consists of a fraction of α positive examples and $1-\alpha$ negative examples. As $0 < P(Y=1) < 1$, such a subsample will exists with positive probability.

By Theorem 5.1.1 the optimal classifier for $\tilde{P}$ is of course its Bayes classifier

$$f_{\text{Bayes}}(x) = \begin{cases} 1 & \text{iff } \tilde{P}(Y=1|x) \geq 1/2 \\ -1 & \text{else} \end{cases}$$

and hence it is sufficient to relate $\tilde{P}(Y=1|x)$ to $P(Y=1|x)$.

We have

$$\begin{aligned} \tilde{P}(Y=1|x) &= \frac{\tilde{P}(Y=1,x)}{\tilde{P}(x,Y=1)+\tilde{P}(x,Y=-1)} \\ &= \frac{P(x|Y=1)\alpha}{P(x|Y=1)\alpha + P(x|Y=-1)(1-\alpha)} \end{aligned}$$

and hence

$$\begin{aligned} & \tilde{P}(Y=1|x) \geq 1/2 \\ \Leftrightarrow\ & 2P(x|Y=1)\alpha \geq P(x|Y=1)\alpha + P(x|Y=-1)(1-\alpha) \\ \Leftrightarrow\ & P(x|Y=1)\alpha \geq P(x|Y=-1)(1-\alpha) \\ \Leftrightarrow\ & \frac{P(Y=1|x)P(x)}{P(Y=1)}\alpha \geq \frac{P(Y=-1|x)P(x)}{P(Y=-1)}(1-\alpha) \\ \Leftrightarrow\ & \frac{P(Y=1|x)}{P(Y=1)}\alpha \geq \frac{1-P(Y=1|x)}{1-P(Y=1)}(1-\alpha) \\ \Leftrightarrow\ & P(Y=1|x) \geq \frac{P(Y=1)(1-\alpha)}{\alpha - 2\alpha P(Y=1) + P(Y=1)} \end{aligned}$$

□

This theorem tells us that in order to estimate $\{x|P(Y=1|x) \geq \beta\}$ we can draw biased samples with a fraction of α positive examples and use a standard learner, of course again with the assumption that the learned classifier approximates the optimal decision function reasonably well.

This approach was tested in an experimental setup identical to that of the direct approach at the beginning of this section, except that for the local pattern classifier a sample with $\alpha = 1/2$ was drawn in order to predict $P(f_{\text{global}}(x) \neq y|x) \geq P(f_{\text{global}}(x) \neq y)$. That is, as in the balanced cost approach, for each x an error probability higher than the average error should be predicted by the pattern classifier. The sample was generated by repeatedly including examples mispredicted by the global classifier until there was an equal number of mispredicted and correctly predicted examples. This approach was chosen instead of removing correctly predicted examples in order to not lose the information present in the x values of the correctly predicted examples.

These are the results of the linear SVM:

Name	Rel. Err	Err Global	Err Local	Local	C $\geq$ G
business	0.899	0.122	0.300	0.108	+
covtype	0.962	0.227	0.309	0.102	++
diabetes	0.997	0.224	0.235	0.109	*o*
digits	1.000	0.001	0.000	0.089	*o*
physics	0.956	0.303	0.427	0.105	++
ionosphere	0.861	0.110	0.360	0.082	+
liver	0.960	0.294	0.371	0.092	+
medicine	0.869	0.238	0.569	0.107	++
mushroom	0.821	0.000	0.007	0.015	*o*
promoters	0.951	0.088	0.100	0.028	*o*
insurance	0.180	0.001	0.079	0.104	++
balance	0.993	0.045	0.058	0.078	*o*
dermatology	1.000	0.000	0.000	0.000	*n.a.*
iris	1.000	0.000	0.000	0.000	*n.a.*
voting	0.575	0.023	0.279	0.101	++
wine	1.025	0.012	0.000	0.123	*o*
breast	1.096	0.033	0.000	0.093	$--$
garageband	0.985	0.287	0.373	0.045	*o*

These are the results of the radial basis SVM:

Name	Rel. Err	Err Global	Err Local	Local	C $\geq$ G
business	0.831	0.108	0.350	0.121	*o*
covtype	0.959	0.229	0.423	0.049	++
diabetes	0.909	0.223	0.436	0.102	++
digits	1.016	0.004	0.000	0.085	*o*
physics	0.957	0.313	0.454	0.099	++
ionosphere	0.620	0.049	0.311	0.074	++
liver	0.969	0.295	0.321	0.098	*o*
medicine	0.817	0.226	0.661	0.116	++
mushroom	1.000	0.017	0.000	0.000	*o*
promoters	1.000	0.123	0.000	0.000	*o*
insurance	0.039	0.002	0.626	0.093	++
balance	0.839	0.009	0.047	0.097	*o*
dermatology	1.000	0.016	0.000	0.000	*o*
iris	1.000	0.000	0.000	0.000	*n.a.*
voting	0.919	0.061	0.155	0.089	*o*
wine	1.060	0.234	0.133	0.129	*o*
breast	0.292	0.004	0.193	0.118	++
garageband	0.988	0.285	0.471	0.024	++

These are the results of the JRip classifier:

Name	Rel. Err	Err Global	Err Local	Local	C $\geq$ G
business	0.919	0.203	0.333	0.056	+
covtype	1.000	0.250	0.000	0.000	*o*
diabetes	1.000	0.268	0.000	0.000	*o*
digits	1.018	0.005	0.000	0.045	*o*
physics	1.000	0.346	0.000	0.000	*o*
ionosphere	0.785	0.076	0.196	0.102	+
liver	1.000	0.341	0.000	0.000	*o*
medicine	1.000	0.260	0.000	0.000	*o*
mushroom	0.700	0.000	0.078	0.006	*o*
promoters	1.041	0.184	0.000	0.037	−
insurance	1.000	0.062	0.000	0.000	*o*
balance	0.968	0.110	0.065	0.015	*o*
dermatology	1.000	0.000	0.000	0.000	*n.a.*
iris	1.000	0.013	0.000	0.000	*o*
voting	0.691	0.031	0.287	0.092	++
wine	0.995	0.092	0.100	0.028	*o*
breast	0.651	0.025	0.179	0.086	++
garageband	1.000	0.293	0.000	0.000	*o*

These are the results of the J4.8 decision tree:

Name	Rel. Err	Err Global	Err Local	Local	C $\geq$ G
business	0.973	0.227	0.200	0.044	*o*
covtype	0.986	0.256	0.287	0.099	*o*
diabetes	0.968	0.263	0.217	0.054	*o*
digits	1.005	0.007	0.000	0.006	*o*
physics	0.991	0.392	0.422	0.099	+
ionosphere	1.002	0.102	0.020	0.028	*o*
liver	1.003	0.349	0.253	0.101	*o*
medicine	0.937	0.215	0.354	0.101	++
mushroom	0.910	0.002	0.116	0.000	*o*
promoters	0.987	0.188	0.183	0.104	*o*
insurance	0.678	0.008	0.684	0.006	++
balance	0.924	0.115	0.213	0.125	*o*
dermatology	1.000	0.000	0.000	0.000	*n.a.*
iris	1.000	0.013	0.000	0.000	*o*
voting	0.956	0.032	0.140	0.061	*o*
wine	1.005	0.050	0.000	0.016	*o*
breast	0.932	0.033	0.166	0.029	*o*
garageband	0.996	0.318	0.332	0.077	*o*

The following summary table compares all three approaches. It reports for each classifier the average relative error (second column), the number of data sets with a significant reduction of error (third column), the number of times no local pattern could be found (fourth column) and the number of times the fraction of test examples assigned to the local pattern exceeded the local threshold τ by more than 0.01 (fifth column).

Sampling

Classifier	rel. err	sig. err	no loc	loc $> \tau$
linear SVM	0.896	8	2	1
RBF SVM	0.845	8	4	4
JRip	0.931	4	9	0
J48	0.958	3	2	1

Balanced Cost

Classifier	rel. err	sig. err	no local	local $> \tau$
linear SVM	0.874	8	2	1
RBF SVM	0.843	9	4	2

Original

Classifier	rel. err	sig. err	no local	local $> \tau$
linear SVM	0.901	7	2	4
RBF SVM	0.806	9	4	7
JRip	0.973	6	7	0
J48	1.000	0	12	0

Interpreting this numbers one has also to keep in mind that on some of the data sets (`balance` and `iris`) a global classifier with zero test error can be found. It is not an error when no local model is found on these data sets.

In comparison, both approaches with an adapted probability threshold produce far more consistent results than the direct approach. These approaches are also superior at finding a local model in the case of the J48 classifier, but not for the JRip classifier. The comparison of the error reduction between the original and the new approaches is problematic as inconsistent local patterns can greatly reduce the global error simply by removing much more examples.

In comparison between the sampling approach and the approach with a balanced cost function, the balanced cost approach is superior for the radial basis SVM. The explanation for this observation is that it is better to directly optimize the estimation of the misclassification probability border $P(y|x) \geq \alpha$ than to indirectly encode this task in the examples distribution.

5.2 Optimizing Learning with Local Patterns

In order to further improve local pattern detection, some theoretical background is needed. An analysis from the view of statistical learning theory can be found in [Vapnik, 1998], Ch. 6.6. Statistical learning theory in general deals with the goal of minimizing

$$R(f) = \int L(f(x), y) dP(x, y),$$

the expected risk (see Section 2.1.1). The goal is to find a function f that predicts y well globally, that is with respect to $P(x)$. However, depending on the function class it may be easier to split up the input space into several regions and for each region search a function which approximates the data only locally. An analysis of the minimization of the local risk is presented in [Vapnik, 1998], Ch. 6.6. The concept of locality is modeled by a "vicinity function" $v(x, x_0, \beta) \in$

$[0, 1]$ that depends on a center x_0 and a parameter $\beta > 0$. In particular, the hard threshold vicinity function

$$v(x, x_0, \beta) = \begin{cases} 1 & \text{if } ||x - x_0|| < \beta \\ 0 & \text{else} \end{cases}$$

and the soft threshold vicinity function

$$v(x, x_0, \beta) = \exp(-\frac{(x - x_0)^2}{\beta^2})$$

are considered. The hard threshold vicinity function defines exact borders of a region around x_0 while the soft threshold vicinity function defines a density function around x_0 (up to a multiplicative constant). Using some vicinity function, the following problem is defined:

Definition 5.2.1 (Local Risk Minimization Problem [Vapnik, 1998])**.** Let $P(x, y)$ be a probability distribution, $x_0 \in X$ be an observation and $v(x, x_0, \beta)$ be a vicinity function. Define $v(x_0, \beta) = \int v(x, x_0, \beta)dP(x)$. Given a set of functions $f_\alpha, \alpha \in \Lambda$, the local risk minimization problem is to find a function f_α and a parameter $\beta > 0$ that minimizes the local risk

$$R(\alpha, \beta, x_0) = \int L(y, f_\alpha(x))\frac{v(x, x_0, \beta)}{v(x_0, \beta)}dP(x, y).$$

□

The center point x_0 is held fixed, only β is modified to define the locality. Using the 0-1-loss, the following theorem can be proven.

Theorem 5.2.1 ([Vapnik, 1998], Theorem 6.14)**.** *Let the set of loss functions $L_{01}(y, f_\alpha(x)), \alpha \in \Lambda$ have VC dimension h_1 and let the set of functions $v(x, x_0, \beta)$, $\beta > 0$, have VC dimension h_2. Then, on a set of l training points, with probability $1 - 2\eta$ simultaneously for all $\alpha \in \Lambda$ and all $\beta > 0$ the following inequality holds:*

$$R(\alpha, \beta, x_0) \leq \frac{2R_{emp}(\alpha, \beta, x_0) + \mathcal{E}_{h_1}(l) + \mathcal{E}_{h_2}(l)}{2\left(v_{emp}(x_0, \beta) - \sqrt{\mathcal{E}_{h_2}(l)}\right)_+}\left(1 + \sqrt{1 + \frac{4R_{emp}(\alpha, \beta, x_0)}{\mathcal{E}_{h_1}(l) + \mathcal{E}_{h_2}(l)}}\right)$$

where

$$R_{emp}(\alpha, \beta, x_0) = \frac{1}{l}\sum_{i=1}^{l} L_{01}(y_i, f(x_i))v(x_i, x_0, \beta)$$

$$\mathcal{E}_{h_i}(l) = 4\frac{h_i\left(\ln\frac{2l}{h_i} + 1\right) - \ln\frac{\eta}{4}}{l}$$

$$v_{emp}(x_0, \beta) = \frac{1}{l}\sum_{i=1}^{l} v(x_i, x_0, \beta)$$

In short, this theorem tells us that in order to get a good generalization, we have to simultaneously minimize the empirical risk R_{emp} and maximize the local "likelihood" term v_{emp}, both with functions of low complexity h_i. In the approach used in this chapter, the vicinity functions are given by the set of local pattern classifiers $g(x)$, scaled to give values in $[0, 1]$. For example,

$$v(x, x_0, \beta) \quad = \quad \begin{cases} 1 & \text{if } g(x) <= \beta \\ 0 & \text{else} \end{cases}$$

can be used. It does not matter that the definition of the function is independent of x_0, as long as all the term in Definition 5.2.1 are well defined. Note that the examples that are local in terms of the vicinity function (the ones where the risk shall be minimized) are the global examples in terms of the definitions used in this thesis.

A consequence of this theorem is the particular importance of the capacity (VC dimension) of the set of local pattern functions. Even with a consistent global learner, a too complex learner for the local patterns could remove all mispredicted examples of the global learner from the training set and lead to the inconsistent error estimate of 0.

Also, the bounds of Theorem 5.2.1 suggest to use the quantity

$$\frac{R_{\text{emp}}(\alpha, \beta, x_0)}{v_{\text{emp}}(x_0, \beta)}$$

as an error measure for the predictive local pattern problem. Note that by definition

$$\begin{aligned} \frac{R_{\text{emp}}(\alpha, \beta, x_0)}{v_{\text{emp}}(x_0, \beta)} &= \frac{\sum_{i=1}^{l} L(y_i, f(x_i, \alpha)) v(x_i, x_0, \beta)}{\sum_{i=1}^{l} v(x_i, x_0, \beta)} \\ &= \sum_{i=1}^{l} L(y_i, f(x_i, \alpha)) \hat{v}(x_i, x_0, \beta) \end{aligned}$$

where we define

$$\hat{v}(x_i, x_0, \beta) = \frac{v(x_i, x_0, \beta)}{\sum_{i=1}^{l} v(x_i, x_0, \beta)}.$$

This $\hat{v}$ can be viewed as an empirical estimator of the probability distribution $P(x|Glob)$ of the global examples x, which has the attractive property of viewing local risk minimization as usual risk minimization with a modified distribution $P(x)$:

Definition 5.2.2 (Locally Weighted Risk Minimization Problem)**.** Given a set of examples $(x_i, y_i)_{i=1...n}$ and two classes of function $f_\alpha : X \to Y, \alpha \in \Lambda$ and $v_\beta : X \to [0, 1], \beta \in \Lambda'$, the locally weighted risk minimization problem is to find functions f_α and v_β that minimize

$$R_L(\alpha, \beta) = \sum_{i=1}^{n} L(y_i, f_\alpha(x_i)) v_\beta(x_i)$$

such that $\sum_{i=1}^{n} v_\beta(x_i) = 1$. □

The role of the function v_β is to estimate $\hat{v}$ and hence the locally weighted risk minimization problem is to minimize the quantity

$$\frac{R_{\text{emp}}(\alpha, \beta, x_0)}{v_{\text{emp}}(x_0, \beta)}$$

as an approximation for minimizing the empirical part in the bound of Theorem 5.2.1. As an approximative algorithm for minimizing the complete bound of Theorem 5.2.1, we use function classes of restricted complexity, and solve the locally weighted risk minimization problem in these function classes. Several classes of different complexities can be tried and the solution with the minimal estimated error will be used.

This approach is similar to the Support Vector Machine idea (see Section 2.1.3): the Structural Risk Minimization principle requires an evaluation of the error bounds, which is troublesome because the VC dimension of the function classes is usually hard to derive. Hence, Structural Risk Minimization is replaced by an approximation to the actual bound (the regularized risk for SVMs and the locally weighted risk here) to guide the search for a consistent hypothesis, the best complexity class being chosen by an estimation of the risk, e.g. by cross-validation.

We can now show that the approach of detecting predictive local patterns by means of two classifiers as in Section 5.1 solves the locally weighted risk minimization problem. The only difference in the approaches is that in locally weighted risk minimization the second function v describes the global examples in terms of a density $P(x|glob)$, while in the approach of Section 5.1 the function g is based on an estimate of the local pattern probability $P(loc|x)$. Both quantities are connected by

$$\begin{aligned} P(x|glob) &= \frac{P(glob|x)P(x)}{P(glob)} \\ &= \frac{(1-P(loc|x))P(x)}{P(glob)}. \end{aligned}$$

As an empirical estimate, we can set $P(x_i) = 1/n$ for each example in the training set and hence $P(x|glob)$ is a strictly monotone decreasing function of $P(loc|x)$. The value $P(loc|x)$ itself can either be obtained by probabilistic scaling of g or one can choose $P(loc|x) \in \{0,1\}$, depending on the sign of g. The latter alternative corresponds to the selection of examples in the approach of Section 5.1. In order to use a classifier to define the local patterns, we replace the density estimating functions v in the previous definition by a pattern classification function g and modify the condition on g such that it sums up to a prior estimate $\tau = P(loc)$

Definition 5.2.3 (Locally Weighted 2-Classifier Risk Minimization Problem)**.** Given a constant $\tau \in]0,1[$, a set of examples $(x_i, y_i)_{i=1\ldots n}$ and two classes of function $f_\alpha, \alpha \in \Lambda$ and $g_\beta : X \to [0,1], \beta \in \Lambda'$, the locally weighted 2-classifier risk minimization problem is to find functions f_α and g_β that minimize

$$R_L(\alpha,\beta) = \sum_{i=1}^{n} L(y_i, f_\alpha(x_i))(1 - g_\beta(x_i))$$

such that $\sum_{i=1}^{n} g_\beta(x_i) = \tau$. □

5.2.1 Expectation Maximization for Local Patterns

The following iterative algorithm solves the locally weighted 2-classifier risk minimization problem:

1. Input: examples $(x_i, y_i)_{i=1\ldots n}$, constant $\tau \in]0,1[$, learners for the global model and the local pattern

2. Set $G = \{1, \ldots, n\}$

3. Learn function f on $(x_i, y_i)_{i \in G}$

4. Define $z_i = sign(-y_i f(x_i))$ and learn function g on $(x_i, z_i)_{i=1\ldots n}$

5. Define G as the indices of the $(1-\tau)n$ examples with highest confidence values $g(x_i)$

6. Return to 3.

In particular, the weights in step 5 can be defined as $w_i = 1$ for the τn examples with highest confidence $g(x_i)$. In the following, this algorithm will be called the EM Local Risk Algorithm.

This algorithm is a variant of the Expectation Maximization algorithm [Dempster et al., 1977]. It postulates and estimates hidden variable z_i in order to better model the known variables (x_i, y_i). As in the original EM algorithm, the algorithm can be shown to converge to a local minimum of the local risk under certain conditions.

Theorem 5.2.2 (Convergence of the EM Local Risk Algorithm)**.** *Assume the learner for the global classifier $f : X \to \{-1, 1\}$ minimizes the empirical 0-1-error and assume the learner for the local pattern classifier $g : X \to \{0, 1\}$ minimizes the error on its training set under the condition $\#\{i | g(x_i) = 1\} = \lfloor \tau n \rfloor$. Then, the sequence of local risks*

$$R_L(\alpha, \beta) = \sum_{i=1}^{n} L_{01}(y_i, f_\alpha(x_i))(1 - g_\beta(x_i))$$

in each iteration step of the EM Local Risk Algorithm converges to a lower bound R_L^*.

Note that in the theorem we define g as a binary classifier in the set $\{0, 1\}$ instead of a probabilistic classifier.

Proof. As R_L is bounded below by 0, it is sufficient to show that it is non-increasing in each iteration to guarantee the convergence of R_L to some lower bound R_L^*.

Obviously, for a fixed g the target function R_L only depends on the x_i with $g(x_i) \neq 1$ and hence only on the set G. Therefore, for a fixed g minimizing the error of f on G as in step 3 of the EM Local Risk algorithm is equivalent to minimizing R_L.

For a fixed f, R_L only depends on the examples with positive loss and hence on the examples with $z_i = 1$. Minimizing R_L is then equivalent to maximizing

$g(x_i)$ on the set $E = \{i|z_i = 1\}$ Further,

$$\begin{aligned} nErr(g) &= \#\{i|z_i = 1 \wedge g(x_i) = 0\} + \#\{i|z_i = 0 \wedge g(x_i) = 1\} \\ &= \sum z_i - \#\{i|z_i = 1 \wedge g(x_i) = 1\} + \#\{i|z_i = 0 \wedge g(x_i) = 1\} \\ &= \sum z_i - \#\{i|z_i = 1 \wedge g(x_i) = 1\} \\ &\quad + \sum g_i - \#\{i|z_i = 1 \wedge g(x_i) = 1\} \\ &= \sum z_i + \sum g_i - 2\#\{i|z_i = 1 \wedge g(x_i) = 1\} \end{aligned}$$

As the sum of z_i is constant for fixed f and $\sum g_i = \#\{i|g(x_i) = 1\} = \lfloor \tau n \rfloor$ is constant, minimizing the error of g is equivalent to maximizing $g(x_i)$ for positive z_i and hence equivalent to minimizing R_L. □

Two remarks: Note that the theorem does not guarantee the convergence of the functions f and g, it only guarantees that the functions found are in the limit equivalent in terms of R_L (the solution does not need to be unique). Also note that if we do not fix $\sum g_i = \#\{i|g(x_i) = 1\} = \lfloor \tau n \rfloor$ but only require $\sum g_i \leq \#\{i|g(x_i) = 1\} = \lfloor \tau n \rfloor$, we still optimize an upper bound on $\#\{i|z_i = 1 \wedge g(x_i)\}$ by minimizing the error of g.

Theorem 5.2.3 (Expected Convergence of the EM Local Risk Algorithm)**.** *Assume the examples are distributed by some $P(X,Y)$. Let the learner for the global classifier $f : X \rightarrow \{-1,1\}$ minimize the expected 0-1-error and assume the learner for the local pattern classifier $g : X \rightarrow \{0,1\}$ minimizes the expected error under the condition $E\{i|g(x_i) = 1\} = \tau n$. Then, the sequences of local risks*

$$R_{L,exp}(\alpha,\beta) = E\left(L_{01}(y_i, f_\alpha(x_i))(1 - g_\beta(x_i))\right)$$

in each iteration step of the EM Local Risk Algorithm converges to a lower bound $R^*_{L,Exp}$.

Proof. Replace the sums and counts by their expected values in the proof of Theorem 5.2.2. □

Of course, in practice the distribution P is not known and the learner can not be assumed to minimize the expected risk. Hence, some modifications to this approach are necessary. The problem is that we want to minimize the expected risk, but do not know the probability distribution behind the examples. We could minimize the empirical risk, but this would lead to overfitting (this is the basic problem of Machine Learning described in Chapter 2). In order to generalize well, learners usually minimize some regularized risk (see Section 2.1.12) or some MDL criterion (see Section 2.1.4) in order to approximate the expected error. The problem is that an equivalent theorem for learners which minimize the regularized risk or a MDL criterion does not exist, because in this approach we do not combine the complexities of both learners with each other, but optimize them separately. In practice, we do not optimize the bound of Theorem 5.2.1 but an approximation. To counter overfitting, it is necessary to check in each step that the local risk R_L does not increase.

The algorithm has empirically been tested with the same experimental setup as in Section 5.1. The iteration has been run until the empirical error increased in one step. For the sake of brevity, only the summary table is reported here.

Balanced Cost

Classifier	rel. err	sig. err	no local	local > τ
linear SVM	0.862	8	2	0
RBF SVM	0.839	9	4	1

Sampling

Classifier	rel. err	sig. err	no loc	loc > τ
linear SVM	0.884	8	2	3
RBF SVM	0.837	9	5	4
JRip	0.931	4	9	2
J48	0.958	2	2	1

As a result, one can see that the relative errors are slightly smaller than the ones reported in Section 5.1, but essentially the results are similar (one significant error reduction more for the linear SVM with sampling, but one less for J48 and sampling).

One can show that this result is an effect of overfitting the local pattern. A first hint is that if one reduces the chance of overfitting by using a low-complexity learner (the linear SVM) plus a larger local pattern (25% of the examples instead of 10%), the relative error reduces as expected (from 0.810 to 0.801). A more thorough investigation of the overfitting effect will be presented in the next section.

5.2.2 1.5-Step Iteration

In order to avoid overfitting one can fix the local pattern classifier after the first iteration and only optimize the global model. That is, we only perform one full step of the optimization and the first half of the second step. This approach has been tested in the same experimental setup as before. Here are the results of the linear SVM with the balanced cost approach.

Name	Rel. Err	Err Global	Err Local	Local	C $\geq$ G
business	0.940	0.124	0.275	0.114	*o*
covtype	0.949	0.234	0.364	0.094	++
diabetes	0.949	0.207	0.315	0.108	*o*
digits	1.006	0.002	0.000	0.085	*o*
physics	0.955	0.306	0.438	0.102	++
ionosphere	0.700	0.095	0.491	0.091	++
liver	0.917	0.290	0.448	0.095	+
medicine	0.862	0.239	0.597	0.107	++
mushroom	0.810	0.001	0.014	0.017	*o*
promoters	0.970	0.092	0.066	0.047	*o*
insurance	0.377	0.009	0.157	0.107	++
balance	0.764	0.038	0.164	0.100	+
dermatology	1.000	0.000	0.000	0.000	*n.a.*
iris	1.000	0.000	0.000	0.000	*n.a.*
voting	0.535	0.020	0.295	0.094	++
wine	0.938	0.025	0.100	0.100	*o*
breast	1.091	0.035	0.000	0.089	−−
garageband	0.978	0.277	0.466	0.038	+

The radial basis SVM with the balanced cost approach performs as follows:

Name	Rel. Err	Err Global	Err Local	Local	C $\geq$ G
business	0.814	0.123	0.483	0.095	$+$
covtype	0.954	0.236	0.449	0.050	$++$
diabetes	0.917	0.225	0.422	0.085	$+$
digits	1.021	0.004	0.000	0.082	o
physics	0.960	0.315	0.442	0.103	$++$
ionosphere	0.624	0.055	0.316	0.071	$++$
liver	0.959	0.290	0.430	0.084	o
medicine	0.828	0.230	0.642	0.115	$++$
mushroom	1.000	0.017	0.000	0.000	o
promoters	1.000	0.123	0.000	0.000	o
insurance	0.078	0.005	0.657	0.098	$++$
balance	0.839	0.009	0.054	0.097	o
dermatology	1.000	0.016	0.000	0.000	o
iris	1.000	0.000	0.000	0.000	$n.a.$
voting	0.929	0.054	0.114	0.096	o
wine	0.772	0.307	0.900	0.129	$++$
breast	0.300	0.006	0.221	0.105	$++$
garageband	0.994	0.281	0.343	0.022	$+$

The linear SVM with the biased sampling can be found in the following table:

Name	Rel. Err	Err Global	Err Local	Local	C $\geq$ G
business	0.880	0.118	0.300	0.120	$+$
covtype	0.954	0.240	0.360	0.098	$++$
diabetes	0.963	0.215	0.285	0.110	o
digits	1.020	0.004	0.000	0.089	o
physics	0.959	0.304	0.410	0.109	$++$
ionosphere	0.812	0.102	0.388	0.091	$+$
liver	0.942	0.324	0.442	0.098	$+$
medicine	0.861	0.239	0.600	0.107	$++$
mushroom	0.807	0.000	0.022	0.015	o
promoters	0.951	0.077	0.100	0.028	o
insurance	0.397	0.010	0.168	0.106	$++$
balance	0.902	0.038	0.110	0.088	o
dermatology	1.000	0.000	0.000	0.000	$n.a.$
iris	1.000	0.000	0.000	0.000	$n.a.$
voting	0.575	0.023	0.275	0.103	$++$
wine	0.885	0.026	0.133	0.129	o
breast	1.094	0.035	0.000	0.092	$--$
garageband	0.976	0.277	0.405	0.045	$+$

Here are the results of the radial basis SVM and biased sampling.

Name	Rel. Err	Err Global	Err Local	Local	C $\geq$ G
business	0.821	0.130	0.483	0.127	+
covtype	0.951	0.230	0.456	0.050	++
diabetes	0.902	0.219	0.418	0.098	+
digits	1.016	0.004	0.000	0.085	*o*
physics	0.966	0.317	0.432	0.097	++
ionosphere	0.627	0.056	0.311	0.076	++
liver	0.990	0.281	0.291	0.092	*o*
medicine	0.813	0.226	0.662	0.118	++
mushroom	1.000	0.017	0.000	0.000	*o*
promoters	1.000	0.123	0.000	0.000	*o*
insurance	0.083	0.005	0.677	0.095	++
balance	0.847	0.011	0.047	0.099	*o*
dermatology	1.000	0.016	0.000	0.000	*o*
iris	1.000	0.000	0.000	0.000	*n.a.*
voting	0.880	0.051	0.172	0.092	*o*
wine	0.770	0.307	0.900	0.129	++
breast	0.259	0.006	0.225	0.118	++
garageband	0.999	0.281	0.285	0.029	*o*

Here are the results of JRip and biased sampling.

Name	Rel. Err	Err Global	Err Local	Local	C $\geq$ G
business	0.969	0.218	0.265	0.094	*o*
covtype	1.000	0.250	0.000	0.000	*o*
diabetes	1.000	0.268	0.000	0.000	*o*
digits	1.018	0.006	0.000	0.045	*o*
physics	1.000	0.346	0.000	0.000	*o*
ionosphere	0.901	0.091	0.200	0.102	+
liver	1.000	0.341	0.000	0.000	*o*
medicine	1.000	0.260	0.000	0.000	*o*
mushroom	0.797	0.001	0.112	0.006	*o*
promoters	1.050	0.165	0.100	0.084	*o*
insurance	1.000	0.062	0.000	0.000	*o*
balance	0.916	0.100	0.131	0.043	*o*
dermatology	1.000	0.000	0.000	0.000	*n.a.*
iris	1.000	0.013	0.000	0.000	*o*
voting	0.657	0.030	0.260	0.087	++
wine	0.872	0.070	0.200	0.028	*o*
breast	0.522	0.025	0.311	0.087	++
garageband	1.000	0.293	0.000	0.000	*o*

Finally, the J48 decision tree and biased sampling

Name	Rel. Err	Err Global	Err Local	Local	C $\geq$ G
business	0.968	0.227	0.200	0.044	*o*
covtype	0.985	0.249	0.282	0.103	+
diabetes	0.986	0.265	0.239	0.066	*o*
digits	1.002	0.006	0.000	0.006	*o*
physics	0.984	0.383	0.442	0.095	++
ionosphere	1.002	0.108	0.020	0.028	*o*
liver	0.972	0.328	0.350	0.104	*o*
medicine	0.933	0.211	0.358	0.102	++
mushroom	0.893	0.002	0.166	0.000	*o*
promoters	0.895	0.160	0.266	0.104	+
insurance	0.741	0.011	0.684	0.006	++
balance	0.921	0.122	0.229	0.126	*o*
dermatology	1.000	0.000	0.000	0.000	*n.a.*
iris	1.000	0.013	0.000	0.000	*o*
voting	0.905	0.036	0.240	0.061	*o*
wine	1.011	0.051	0.000	0.016	*o*
breast	0.898	0.028	0.200	0.027	*o*
garageband	0.985	0.298	0.356	0.071	*o*

Balanced Cost

Classifier	rel. err	sig. err	no local	local $> \tau$
linear SVM	0.874	9	2	1
RBF SVM	0.833	10	4	2

Sampling

Classifier	rel. err	sig. err	no loc	loc $> \tau$
linear SVM	0.888	9	2	3
RBF SVM	0.829	9	4	4
JRip	0.928	3	9	0
J48	0.949	5	2	1

One can see that this approach is indeed considerably better than the non-iterative approach. The average relative error is smaller for all methods except for the linear SVM with balanced costs, where it remains constant. The number of significant error reductions is higher for every method except for JRip, where it drops from 4 to 3. This shows that with a controlled complexity for the local patterns, an iterative EM-like method can considerably improve the performance of learning with local patterns.

A more detailed discussion of the performance on the individual data sets follows in the next section.

5.2.3 Effects of the Local Pattern on the Global Model

In order to get a deeper understanding of the effects the information of the detected local pattern has on the optimality of the global model, and in particular in order to understand how the global model can be further optimized outside the local pattern, it is instructive to formalize the ideas behind the iterative approach.

The following theorem formalizes the idea, that a classifier is optimized by training only on the examples which it is supposed to predict.

Theorem 5.2.4 (Local Optimization of Hypotheses)**.** *Let R be a risk measure with respect to an arbitrary loss function L and probability measure P, i.e.*

$$R(f) \quad = \quad \int L(x, y, f(x))dP(x).$$

Let $\mathcal{F}$ be a hypothesis space and $f^* \in \mathcal{F}$ be the hypothesis with minimal risk and let A be a subset of the input space X, such that the risk of f^* on A

$$\begin{aligned} R_{|A}(f^*) &= \int L(x, y, f^*(x))dP(x|A) \\ &= \frac{1}{P(A)} \int_A L(x, y, f^*(x))dP(x) \end{aligned}$$

is higher than the risk of f^* on $X \backslash A$. Then the optimal hypothesis in $\mathcal{F}$ on $X \backslash A$ will have lower risk than f^*.

Proof. We can decompose $R(f)$ into

$$\begin{aligned} R(f) &= \int L(x, y, f(x))dP(x) \\ &= \int_A L(x, y, f(x))dP(x) + \int_{X \backslash A} L(x, y, f(x))dP(x) \\ &= P(A)R_{|A}(f) + (1 - P(A))R_{|X\backslash A}(f) \\ &\geq P(A)R_{|X\backslash A}(f) + (1 - P(A))R_{|X\backslash A}(f) \\ &= R_{|X\backslash A}(f) \\ &\geq min_{f\in\mathcal{F}} R_{|X\backslash A}(f) \end{aligned}$$

□

Notice that by choosing P as the empirical probability measure defined by the training data, the theorem in particular holds for the empirical risk. Intuitively, the theorem is clear: if we remove parts of the input space that are hard to predict, then the performance on the rest will be better.

Unfortunately, the experimental results do not show the expected results. For example, the linear SVM with the balanced costs on the data set COVTYPE achieves a global error of 0.226 after the first iteration (see the table in Section 5.1, the approach taken there is exactly the first iteration of the EM algorithm). After the next optimization step for the global model, the global error increases to 0.234 (see the 1.5 iteration in Section 5.2.2). The reason for this apparent contradiction is that the actual situation differs from the assumptions of Theorem 5.2.4 in two important points.

First, learners usually neither optimize the empirical risk (because this is not wanted) nor the true expected risk (because this is infeasible). Instead, in order to avoid overfitting, most learners can be described as optimizing a regularized risk

$$R_{reg}(f) = R_{emp}(f) + comp(f)$$

(see Definition 2.1.12). One may argue that a good learning algorithm should approximate the expected risk reasonably well, such that the theorem still holds approximately. If one assumes that not much complexity is spent for approximating the data in a high-error local pattern – there is no gain in increasing complexity if the error does not decrease – both the complexity and the error on this local pattern will only have a small influence on the choice of the hypothesis and the learner will return similar hypotheses on the complete data set and on the data set with the local pattern removed. However, it is still possible that under certain circumstances large deviations may happen.

Second, Theorem 5.2.4 assumes that the hypothesis space $\mathcal{F}$ remains unchanged, such that the old global model upper-bounds the possible error. In practice, however, the hypothesis space over which the risk is minimized usually depends on the training set. For example, most decision trees only induce splits over attribute values that are contained in the training set, and Support Vector Machines construct decision functions with basis functions centered at the training examples. As a consequence, it cannot be guaranteed that the hypothesis which was induced over the complete data set is still contained in the hypothesis space over a subset of examples, and even if an appropriate hypothesis exists, the complexity assigned to it may be arbitrarily large.

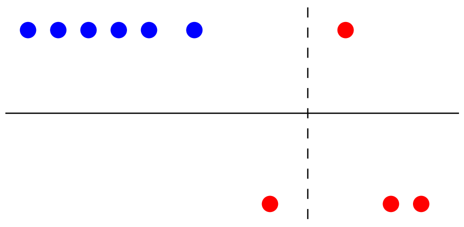


Figure 5.3: Increased global error by local patterns

As a result, there may be a situation as in Figure 5.3: let the class of the 10 examples be denoted by its y-value and suppose we compare decision trees with zero and one nodes. A decision tree with zero nodes is constant and can achieve a minimal error of 3/10 on the depicted data set. A decision tree with one node could place a threshold between the sixth and seventh example from the left and predict the examples as labeled by the point's colors, this tree will have an error of only 1/10. Now suppose the local pattern is described by the dashed line, separating the 7 examples on the left with prediction error 0/7 from the examples on the right with prediction error 1/3. When a global classifier is learned on the 7 global examples, the constant decision tree has an error of 1/7 and the decision tree with one node an error of 0/7. The difference between 1/7 and 0 is much smaller than the previous difference between 1/10 and 3/10, and hence in a setting minimizing the regularized risk, it may be possible that this smaller improvement in accuracy is no longer enough to justify a more complex function. In this case, the simpler constant function will be chosen and the error increases due to the local pattern.

In short, the problem is now that local examples may have information about the class of near global examples that should not be ignored. To find a way out of this dilemma one has to bear in mind that the idea of local models consists of two tasks, on the one side to describe the classifier, but on the other side also to improve the classifier by use of a local model. The goal of this chapter is to describe the classifier, which for example motivated the choice of a different conditional class probability threshold in Section 5.1, but in order to improve the global classifier we have to focus on those examples which clearly degrade the performance of the global classifier.

As a solution, one can make use of two local pattern classifiers. The first classifier as before describes the local patterns, i.e. regions with a higher than default error probability, while the second one describes only the clearly identified errors of the global classifier. This second classifier is trained with an unmodified conditional class probability threshold and is only used for identifying the examples to be removed from the global learners training set in the next iteration.

The experimental evaluation of this approach is summarized in the following table:

Balanced Cost

Classifier	rel. err	sig. err	no local	local > τ
linear SVM	0.868	11	2	1
RBF SVM	0.843	11	4	2

Sampling

Classifier	rel. err	sig. err	no loc	loc > τ
linear SVM	0.889	8	2	3
RBF SVM	0.828	9	4	4
JRip	0.921	4	9	1
J48	0.946	5	2	1

In comparison with the 1.5-step iteration in Section 5.2.2, this algorithm is significantly better for the balanced cost approach (both 11 significant error reduction instead of 9 and 10, respectively). For the biased sampling approach, it performs comparable (1 significant error reduction less for the linear SVM, but 1 more for JRip).

5.3 Local Patterns by Unsupervised Learning

In the previous section, local patterns were defined by a classifier that separated the global from the local examples. This is a discriminative approach, where in order to perform well, the classifier must describe the differences between the two kinds of examples. Accordingly, by understanding this classifier the user understands how the two kinds of examples differ. In this section, an alternative approach will be presented. This approach employs unsupervised learning to describe the local pattern itself, instead of discriminating it from the global pattern. By understanding this description, the user gets more information about the complete local pattern. The difference can be seen in Figure 5.4: the green line discriminates the global examples (blue) from the local examples (red), using only the information about the horizontal position. The magenta circle describes the local pattern, taking information from both the horizontal and vertical position into account. Accordingly, the local pattern can be far better reconstructed from the circle than from the line. Further, the most informative example in the classification case is the filled blue point (it lies closest to the decision boundary), while in the descriptive case the red point is most informative (it is most similar to all the other red points).

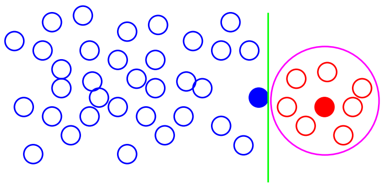


Figure 5.4: Discriminative versus descriptive local patterns.

A main advantage of the descriptive approach is in the reliability of the local pattern. Assume that in Figure 5.4 some new test points are introduced that lie on the far right side of the figure, yielding the situation of Figure 5.5. On these new observations the linear classifier will be most certain of its decision that these points belong to the local pattern, because the observations are far away from the decision boundary. But actually it should be clear that no reliable decision about these points can be made, because these observations do not look similar to any point that the classifier has seen in training. Consequently, the descriptive model where the new observations are not part of the local pattern is much more reliable.

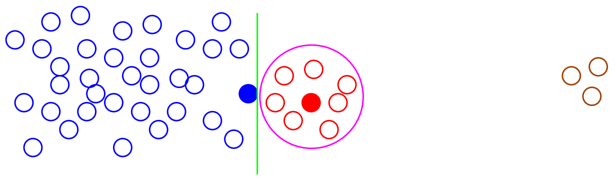


Figure 5.5: Reliability of discriminative and descriptive local patterns.

5.3.1 Local Patterns by Clustering and Density Estimation

The approach of detecting predictive local patterns with unsupervised learning has been tested with k-medoids clustering and Gaussian density estimation (see Section 2.2). Similar to the supervised approach, the global model is learned in the first step. Then, k-medoids clustering or Gaussian density estimation is applied to the examples that were mispredicted by the global model to obtain a local pattern membership function. Finally, a threshold on the membership values is chosen such that only a given fraction of τ of the training examples lie in the local pattern. For the k-medoids approach, the parameter k has been chosen in the range of 1 to 10 to optimize the training error.

There is a technical problem with Gaussian density estimation in this algorithm: Gaussian density estimation needs a non-singular covariance matrix in order to define a proper density function. In the case of nominal attributes in the data, this cannot be guaranteed as it can easily be the case that only examples with a fixed attribute value are mispredicted by the global model. In this case, the data has to be projected into the linear subspace that is defined by the nonzero eigenvalues of the covariance matrix and density estimation has to be carried out in the projected space. However, in this approach it is numerically difficult to distinguish between small and zero eigenvalues and to decide whether a new observation lies in the hyperplane or just very close to it. This problem becomes even worse for the mixture of Gaussians density estimation, because, every single Gaussian component could lie inside a different linear subspace at some iteration of the algorithm. An approach to solve this problem using a randomized algorithm with multiple restarts has been proposed in [Rüping, 2005]. However, this algorithm still turned out to be too unstable in the case of small data sets with a high number of nominal attributes like in the task of local pattern extraction. For this reason, mixture of Gaussians density estimation is not included in the comparison here.

Here are the results of local patterns with Gaussian density estimation and the linear SVM as global learner:

Name	Rel. Err	Err Global	Err Local	Local	C $\geq$ G
business	1.000	0.139	0.000	0.000	*o*
covtype	1.018	0.240	0.198	0.101	$-$
diabetes	0.948	0.213	0.347	0.098	+
digits	1.000	0.001	0.000	0.000	*o*
physics	1.000	0.317	0.000	0.000	*o*
ionosphere	1.039	0.139	0.033	0.099	$-$
liver	0.944	0.288	0.421	0.081	+
medicine	0.902	0.247	0.502	0.104	++
mushroom	0.900	0.001	0.002	0.005	*o*
promoters	1.000	0.102	0.000	0.000	*o*
insurance	0.705	0.007	0.038	0.092	++
balance	0.526	0.019	0.339	0.090	++
dermatology	1.000	0.000	0.000	0.000	*n.a.*
iris	1.000	0.000	0.000	0.000	*n.a.*
voting	0.851	0.035	0.126	0.087	*o*
wine	1.000	0.011	0.000	0.094	*o*
breast	0.742	0.025	0.084	0.088	+
garageband	1.000	0.291	0.000	0.000	*o*

Here are the results of local patterns with Gaussian density estimation and the radial basis SVM as global learner:

Name	Rel. Err	Err Global	Err Local	Local	C $\geq$ G
business	0.813	0.111	0.400	0.090	*o*
covtype	1.020	0.243	0.195	0.100	$--$
diabetes	0.978	0.238	0.317	0.098	*o*
digits	1.000	0.003	0.000	0.034	*o*
physics	1.000	0.327	0.000	0.000	*o*
ionosphere	1.077	0.070	0.016	0.094	*o*
liver	1.013	0.307	0.230	0.104	*o*
medicine	0.905	0.250	0.512	0.100	++
mushroom	1.000	0.017	0.000	0.000	*o*
promoters	1.000	0.123	0.000	0.000	*o*
insurance	0.595	0.036	0.455	0.058	++
balance	1.027	0.014	0.000	0.024	*o*
dermatology	1.000	0.016	0.000	0.000	*o*
iris	1.000	0.000	0.000	0.000	*n.a.*
voting	0.800	0.054	0.267	0.087	+
wine	0.894	0.201	0.366	0.055	+
breast	0.996	0.025	0.026	0.086	*o*
garageband	1.000	0.289	0.000	0.000	*o*

Here are the results of local patterns with Gaussian density estimation and JRip as global learner:

Name	Rel. Err	Err Global	Err Local	Local	C $\geq$ G
business	0.896	0.216	0.175	0.109	*o*
covtype	1.019	0.254	0.205	0.099	$-$
diabetes	0.938	0.252	0.405	0.098	++
digits	1.021	0.005	0.000	0.071	*o*
physics	1.000	0.346	0.000	0.000	*o*
ionosphere	1.090	0.104	0.020	0.097	$-$
liver	1.013	0.345	0.237	0.113	*o*
medicine	0.925	0.240	0.433	0.103	++
mushroom	0.800	0.000	0.018	0.009	*o*
promoters	0.906	0.169	0.200	0.048	*o*
insurance	0.499	0.031	0.498	0.067	++
balance	0.882	0.100	0.230	0.088	+
dermatology	1.000	0.000	0.000	0.000	*n.a.*
iris	1.000	0.013	0.000	0.000	*o*
voting	0.885	0.038	0.136	0.103	+
wine	0.996	0.092	0.050	0.028	*o*
breast	0.882	0.036	0.097	0.080	*o*
garageband	1.000	0.293	0.000	0.000	*o*

Here are the results of local patterns with Gaussian density estimation and J48 as global learner:

Name	Rel. Err	Err Global	Err Local	Local	C $\geq$ G
business	0.888	0.206	0.333	0.120	++
covtype	1.019	0.265	0.220	0.103	−
diabetes	0.941	0.252	0.411	0.102	+
digits	0.926	0.006	0.012	0.063	*o*
physics	1.000	0.395	0.000	0.000	*o*
ionosphere	1.021	0.103	0.077	0.094	*o*
liver	0.993	0.344	0.315	0.104	*o*
medicine	0.937	0.215	0.348	0.108	++
mushroom	1.000	0.002	0.000	0.000	*o*
promoters	1.024	0.195	0.050	0.057	*o*
insurance	0.921	0.010	0.022	0.099	+
balance	0.803	0.103	0.356	0.097	++
dermatology	1.000	0.000	0.000	0.000	*n.a.*
iris	1.000	0.013	0.000	0.000	*o*
voting	0.912	0.031	0.095	0.085	+
wine	1.000	0.050	0.000	0.000	*o*
breast	0.687	0.029	0.119	0.079	+
garageband	1.000	0.319	0.000	0.000	*o*

Before we some up the results, let us also take a look at the results of local patterns with k-medoids clustering. Here are the results with the linear SVM as base learner:

Name	Rel. Err	Err Global	Err Local	Local	C $\geq$ G
business	0.919	0.123	0.300	0.052	*o*
covtype	0.975	0.230	0.302	0.082	++
diabetes	0.946	0.213	0.364	0.080	+
digits	1.000	0.001	0.000	0.000	*o*
physics	0.973	0.308	0.413	0.082	++
ionosphere	1.031	0.131	0.098	0.082	*o*
liver	0.978	0.300	0.391	0.066	*o*
medicine	0.907	0.249	0.510	0.096	++
mushroom	0.900	0.001	0.002	0.004	*o*
promoters	1.000	0.102	0.000	0.000	*o*
insurance	0.693	0.006	0.043	0.088	++
balance	0.863	0.038	0.202	0.072	*o*
dermatology	1.000	0.000	0.000	0.000	*n.a.*
iris	1.000	0.000	0.000	0.000	*n.a.*
voting	0.860	0.037	0.175	0.055	*o*
wine	1.000	0.011	0.000	0.061	*o*
breast	0.712	0.022	0.103	0.085	+
garageband	1.009	0.293	0.270	0.095	*o*

Here are the results with the radial basis SVM as base learner:

Name	Rel. Err	Err Global	Err Local	Local	C $\geq$ G
business	0.964	0.117	0.400	0.083	*o*
covtype	0.980	0.234	0.288	0.082	++
diabetes	0.956	0.235	0.347	0.088	++
digits	1.000	0.003	0.000	0.038	*o*
physics	0.984	0.322	0.377	0.096	++
ionosphere	0.991	0.062	0.086	0.071	*o*
liver	1.037	0.312	0.108	0.064	*o*
medicine	0.887	0.245	0.567	0.099	++
mushroom	1.000	0.017	0.000	0.000	*o*
promoters	1.000	0.123	0.000	0.000	*o*
insurance	0.413	0.025	0.409	0.092	++
balance	1.030	0.014	0.000	0.057	−
dermatology	1.000	0.016	0.000	0.000	*o*
iris	1.000	0.000	0.000	0.000	*n.a.*
voting	0.686	0.046	0.416	0.068	++
wine	1.024	0.228	0.100	0.039	*o*
breast	0.683	0.017	0.101	0.076	+
garageband	1.008	0.291	0.254	0.099	*o*

Here are the results with JRip as base learner:

Name	Rel. Err	Err Global	Err Local	Local	C $\geq$ G
business	0.850	0.210	0.170	0.103	*o*
covtype	0.968	0.242	0.333	0.091	++
diabetes	0.935	0.252	0.431	0.087	++
digits	1.018	0.005	0.000	0.079	*o*
physics	0.983	0.340	0.403	0.094	+
ionosphere	0.851	0.088	0.141	0.085	+
liver	1.008	0.343	0.341	0.066	*o*
medicine	0.915	0.237	0.484	0.093	++
mushroom	0.874	0.001	0.010	0.012	*o*
promoters	0.901	0.174	0.200	0.058	*o*
insurance	0.510	0.031	0.394	0.085	++
balance	0.802	0.091	0.358	0.085	++
dermatology	1.000	0.000	0.000	0.000	*n.a.*
iris	1.000	0.013	0.000	0.000	*o*
voting	0.789	0.039	0.181	0.052	+
wine	0.868	0.085	0.100	0.033	*o*
breast	0.892	0.036	0.075	0.074	*o*
garageband	1.005	0.294	0.264	0.092	*o*

Here are the results with J48 as base learner:

Name	Rel. Err	Err Global	Err Local	Local	C $\geq$ G
business	0.942	0.214	0.266	0.082	*o*
covtype	0.983	0.256	0.312	0.088	+
diabetes	0.921	0.250	0.397	0.100	+
digits	0.928	0.006	0.011	0.068	*o*
physics	0.982	0.389	0.458	0.086	+
ionosphere	0.967	0.098	0.091	0.088	*o*
liver	1.001	0.348	0.248	0.069	*o*
medicine	0.953	0.219	0.332	0.096	++
mushroom	1.000	0.002	0.000	0.000	*o*
promoters	0.931	0.190	0.100	0.046	*o*
insurance	0.539	0.006	0.074	0.084	++
balance	0.725	0.091	0.426	0.105	++
dermatology	1.000	0.000	0.000	0.000	*n.a.*
iris	1.000	0.013	0.000	0.000	*o*
voting	0.964	0.033	0.075	0.036	*o*
wine	1.000	0.050	0.000	0.000	*o*
breast	0.965	0.035	0.058	0.087	*o*
garageband	1.008	0.321	0.286	0.098	*o*

The following table sums up the results for both unsupervised local patterns. It also contains the best result of the supervised local patterns from Section 5.1, namely the balanced cost approach for the SVMs and the sampling approach for JRip and J48.

Gaussian Density Estimation

Classifier	rel. err	sig. err	no loc	local > τ
linear SVM	0.921	6	7	0
RBF SVM	0.951	4	6	0
JRip	0.931	5	4	1
J48	0.948	7	6	1

k-Medoids

Classifier	rel. err	sig. err	no local	local > τ
linear SVM	0.931	6	4	0
RBF SVM	0.925	7	4	0
JRip	0.898	8	2	0
J48	0.934	6	4	0

Supervised

Classifier	rel. err	sig. err	no local	local > τ
linear SVM	0.874	8	2	1
RBF SVM	0.843	9	4	2
JRip	0.931	4	9	0
J48	0.958	3	2	1

The first result is that k-Medoids clustering is clearly superior to Gaussian density estimation with a total of 27 significant error reductions instead of 22 for Gaussian density estimation. Only for the J48 classifier Gaussian density estimation performs slightly better.

What is more surprising is that the unsupervised approach is clearly superior to the supervised approach for the JRip and J48 classifiers. This is surprising because in clustering the information about the correctly predicted examples is completely ignored. This result can be explained by the fact that both JRip and J48 are propositional logic learners and hence can only make use of decision boundaries that are locally parallel to the coordinate axes. Very smooth but nonlinear decision boundaries are hard to approximate in this hypothesis space. The partitions introduced by k-Medoids on the other hand are very smooth and flexible and the combination of these very different types of functions leads to the significant increase in the local pattern performance.

5.4 Conclusions

This chapter showed that describing a classifier in terms of patterns in its errors cannot be thoroughly solved by applying a standard learner to predict the other classifiers errors. The unbalanced distribution between many correctly classified instances and few errors is a problem which can quite well be solved by asymmetrical cost functions or biased sampling (Section 5.1). What is more challenging is the increased risk of overfitting by the local pattern because of the dependencies between both learners, in particular when both models shall be optimized (Section 5.2). In the experiments only very carefully selected optimization strategies (Sections 5.2.2 and 5.2.3) could improve the results, which is in accordance with the theoretical analysis in Section 5.2.3. Finally, with an independent learner that increases the diversity of the available models, local patterns can be optimized. This was shown in Section 5.3 for detecting local patterns with clustering.

Chapter 6

Learning Local Models

In this chapter, the results of the previous chapters will be put together: given an accurate learner, an understandable learner and a description of the error patterns of the understandable model, the task is to create a model that combines accuracy with interpretability. Section 6.1 will present an approach that can be viewed as an extension of the local patterns in Chapter 5. To circumvent some problems that occur when combining models of independent learners, Section 6.2 will present a SVM formulation of the local model task. Finally, Section 6.3 will present an exemplary application of local models in order to demonstrate the superior understandability of local models.

The role of local models themselves in the local model framework is to achieve a high accuracy classification in the regions defined by the local patterns. These regions will usually cover the hard to predict cases, such that a considerably more complex classifier than the one that found the global model and the local patterns may be needed. On the other hand, no interpretability restrictions are placed on the local models, such that it is solely its accuracy that is to be optimized. As a result, the problem may be solved as a standard classification problem, where only the examples from the local patterns are used to train the classifier.

What distinguishes local model learning from standard classification is the task of optimally combining the global and the local models. In Chapter 5, the task of the local pattern was to describe where the global classifier is wrong, but there is no guarantee that the local classifier can actually be better than the global. Hence, regarding accuracy it is better to let a local pattern describe the regions where the local classifier is better than the global one. This leads to the task of iteratively optimizing the global classifier, the local patterns, and the local models.

This approach is related to delegating classifiers [Ferri et al., 2004], or classifiers with reject option [Chow, 1970]. Delegating classifiers consist of two classifiers and use a measure of confidence for the first classifier in order to decide whether this classifier should classify the observation or delegate it to the other classifier. The confidence measure in delegating classifiers is derived from a numerical decision function, which may not be a reliable measure of confidence, as was discussed in Section 2.4. Other related approaches are arbitrating [Ortega et al., 2001] and grading [Seewald and Fürnkranz, 2001]. An orthogonal approach to local models was taken in [Scholz, 2005], where a sampling approach is used to find rules that extract the most class information from the data given that some rules are already known. These new rules still make prediction over the complete input space, but modify the conditional class probability predicted by the learner.

6.1 Local Models by Combining Local Patterns

In Section 5.1 a local pattern algorithm was presented that consists of two classifiers, one for the global model and one for the local pattern. It is straightforward to convert this algorithm into a local model learner. The role of the local model is exactly the same as the role of the global model: to predict the examples on its respective side of the local pattern boundary. Hence, the local model problem can be viewed as two local pattern detection problems that share the same local pattern. This gives rise to the following algorithm:

1. Input: examples $(x_i, y_i)_{i=1\dots n}$, threshold τ
2. Learn an initial global classifier f
3. Learn an initial local classifier g (this can be done either by applying the local learner to all examples, or by executing a local pattern step and applying the local learner to the examples in the pattern)
4. Define $z_i \in \{-1, 1\}$ such that $z_i = 1$ iff $f(x_i)$ is false but $g(x_i)$ is correct. Learn the local pattern classifier h on $\{(x_i, z_i) | i = 1 \dots n\}$, with the restriction that $P(h(x) = 1) < \tau$.
5. Adapt f to optimize the performance on $\{x | h(x) \neq 1\}$
6. Adapt g to optimize the performance on $\{x | h(x) = 1\}$
7. Return to 4.

Steps 5-7 are optional. The goal of the function h is to predict the examples x where the global classifier can be improved by the local classifier. Note that we require $P(h(x) = 1) < \tau$ for all x, not just the x_i in h's training set, to ensure the property of being local. This algorithm will be called the EM Local Model Algorithm.

Theorem 6.1.1 (Correctness of the EM Local Model Algorithm)**.** *Assume both the learners for the global classifier $f : X \to \{-1, 1\}$ and the local classifier $g : X \to \{-1, 1\}$ minimize the empirical 0-1-error and assume the learner for the local pattern classifier $h : X \to \{0, 1\}$ minimizes the error on its training set under the condition $\#\{i | h(x_i) = 1\} = \lfloor \tau n \rfloor$. Then, the EM Local Risk Algorithm minimizes an upper bound of the local risk*

$$R_L(\alpha, \alpha', \beta) = \sum_{i=1}^{n} L_{01}(y_i, f_\alpha(x_i))(1 - h_\beta(x_i)) + L_{01}(y_i, g_{\alpha'}(x_i))h_\beta(x_i).$$

Proof. As R_L is bounded below by 0, it is sufficient to show that it is non-increasing in each iteration to guarantee the convergence of R_L to some lower bound R_L^*.

Obviously, for a fixed h the problem consists of two independent learning tasks, namely to minimize $L_{01}(y_i, f_\alpha(x_i))$ over $\{i|h_\beta(x_i) = 0\}$ and to minimize $L_{01}(y_i, g_{\alpha'}(x_i))$ over $\{i|h_\beta(x_i) = 1\}$. By assumption, this is done in steps 5. and 6. of the algorithm.

Since for fixed f and g the L_{01}-terms are constants, minimizing R_L in this case is equivalent to minimizing

$$\sum_{i=1}^{n}(L_{01}(y_i, g_{\alpha'}(x_i)) - L_{01}(y_i, f_\alpha(x_i))h_\beta(x_i)$$

This is obviously minimized if

$$h_\beta(x_i) \quad = \quad \begin{cases} 1 \text{ iff } L_{01}(y_i, g_{\alpha'}(x_i)) = 0 \wedge L_{01}(y_i, f_\alpha(x_i)) = 1 \\ 0 \text{ iff } L_{01}(y_i, g_{\alpha'}(x_i)) = 1 \wedge L_{01}(y_i, f_\alpha(x_i)) = 0 \\ \text{arbitrary else} \end{cases}$$

In the *else*-case, $L_{01}(y_i, g_{\alpha'}(x_i)) - L_{01}(y_i, f_\alpha(x_i)) = 0$ holds and h may any value. One can see that the values z_i defined in step 4 are such a set of values, such that if $h_\beta(x_i) = z_i$ for all i, the error is minimized.

The problem lies in the case where no hypothesis h_β with zero error exists. In this case, the learner may optimize its prediction of z_i by accepting more errors on the examples with different errors $L_{01}(y_i, g_{\alpha'}(x_i)$ and $L_{01}(y_i, f_\alpha(x_i)$ if this reduces the error in the examples x_i with $L_{01}(y_i, g_{\alpha'}(x_i) = L_{01}(y_i, f_\alpha(x_i)$. But as every example with a nonzero term $(L_{01}(y_i, g_{\alpha'}(x_i) - L_{01}(y_i, f_\alpha(x_i))h_\beta(x_i)$ is also an error with respect to predicting z_i, the number of errors of predicting z_i is an upper bound of the number of local risk errors. Hence, by minimizing the empirical error of h an upper bound of the local risk is minimized (both under the τ-constraint of $\#\{i|h(x_i) = 1\} = \lfloor \tau n \rfloor$). □

This proof shows that the catch in the algorithm – the reason why it is not possible to show a convergence to a minimal error – is that the learner does not distinguish between the errors in the local model algorithm and the errors in predicting the z_i which do not lead to errors in the local model prediction. This raises the question why points with identical prediction of the global and the local learner cannot simply be ignored in the intermediate task of learning h. The reason is the τ-constraint. When the pattern learner does not know about the additional examples, it cannot guarantee that it will satisfy the τ-constraint over all examples. The prediction of the classifier h on the additional examples may be identical to its prediction of the examples with $z_i = 1$, such that both kinds of examples can even not be distinguished by some post-processing method. In short, the simultaneous optimization of two criteria – minimal error and τ-constraint – is beyond the capabilities of a usual classifier. While this section is concerned with the use of standard classifiers as base learners and hence may have to settle for a suboptimal solution, Section 6.2 will present a novel learning method that can effectively learn under the τ-constraint.

6.1.1 Empirical Evaluation

To evaluate the local model algorithm an appropriate experimental setup has to be defined. The local model idea assumes that the global learner is good at most of the data, but there is a small region where a better learner exists.

This excludes for example the radial basis SVM from being used as both global and local learner, because the radial basis SVM is inherently a local learner. Hence, defining a linear combination of radial basis functions on one part of the input space and defining a second linear combination on the other part of the input space will be not much different from using one such combination on the complete input space in the first place. It can also often be observed that even very different learners perform similar on most data sets and hence it is likely that on a region where one learner has problems fitting the data other learners are not much better.

When local models are used for interpretability reasons, a second assumption is that the global learner and the pattern learner are restricted by some interpretability constraints. Hence, in the following experiments an experimental setup similar to the one from [Morik, 2002] is used: both the global and the pattern learner are restricted to use only a small, pre-defined number of features, which was selected such that the error is significantly higher than with all features. The local learner is allowed to use all features. This ensures that both the local learner can achieve lower error than the global one and that the global model and the local pattern are more interpretable than the local model. In the experiments, linear Support Vector Machines have been used.

The following table shows the results of the local model algorithm when only one iteration is used.

Name	Error			Significance			
	Glob	Local	Comb	$G \geq C$	$L \geq C$	$G \geq L$	$L < \tau$
business	0.260	0.139	0.228	+	−	++	*o*
covtype	0.249	0.235	0.249	*o*	−	+	*o*
diabetes	0.307	0.225	0.281	++	−−	++	*o*
digits	0.072	0.001	0.054	+	−−	++	*o*
physics	0.424	0.317	0.414	+	−−	++	*o*
ionosphere	0.227	0.131	0.176	++	*o*	+	*o*
liver	0.361	0.304	0.358	*o*	−−	++	*o*
medicine	0.277	0.274	0.277	*o*	*o*	*o*	*o*
mushroom	0.030	0.001	0.001	+	*o*	+	*o*
promoters	0.185	0.102	0.159	*o*	−	+	−
insurance	0.070	0.010	0.070	*o*	−−	++	*o*
balance	0.161	0.053	0.144	++	−−	++	*o*
dermatology	0.010	0.000	0.010	*o*	*o*	*o*	*o*
iris	0.173	0.000	0.173	*o*	−−	++	*o*
voting	0.046	0.043	0.046	*o*	*o*	*o*	*o*
wine	0.061	0.011	0.056	*o*	−−	++	*o*
breast	0.111	0.030	0.106	*o*	−−	++	*o*
garageband	0.347	0.291	0.347	*o*	−−	++	*o*

In total, on 7 of the data sets the combined classifier is better than the global classifier alone. Additionally, on 3 of the data sets, the local classifier does not outperform the global classifier when applied to all examples, which means that the combination cannot become better than the global classifier alone. On 13 of the data sets, the local classifier alone is better than the global one, which means that the more the local pattern size τ is increased, the better the combination will become.

An interesting result has been obtained with respect to the optimization by iterating the local model algorithm. When the global learner is trained again on the examples from outside the detected local pattern, the combined classifier still outperforms the global classifier on the 7 data sets from before. The only noticeable effect is that one a single data set the local model is no longer significantly better than the global one, which indicates that some outliers have been removed. When the local learner is trained again on the local examples alone, the performance deteriorates: the combined classifier outperforms the global classifier on only 6 data sets, additionally on 7 of the data sets the local classifier is no longer significantly better than the global classifier. In conclusion, iterating the local model algorithm does not improve performance, but is likely to decrease performance. This can be explained by the increased threat of over-fitting that comes with assigning examples to different training sets and the increased variance in the classifiers due to the smaller training sets (in particular, the training set of the local classifier is severely reduced by restricting it to the local examples).

Similar to the method in Section 5.1, the results can be improved by adapting the probability threshold β in the local pattern learner. In the following approach, an iteration over several values of β has been used in the training of the pattern classifier and the classifier with lowest training error has been chosen.

Name	Error			Significance			
	Glob	Local	Comb	$G \geq C$	$L \geq C$	$G \geq L$	$L < \tau$
business	0.260	0.139	0.222	$+$	$-$	$++$	o
covtype	0.249	0.235	0.249	o	$-$	$+$	o
diabetes	0.307	0.225	0.277	$++$	$--$	$++$	o
digits	0.072	0.001	0.065	$+$	$--$	$++$	o
physics	0.424	0.317	0.411	$++$	$--$	$++$	o
ionosphere	0.227	0.131	0.213	o	$-$	$+$	o
liver	0.361	0.304	0.341	$+$	$-$	$++$	o
medicine	0.277	0.274	0.278	o	$-$	o	o
mushroom	0.030	0.001	0.001	$+$	o	$+$	o
promoters	0.185	0.102	0.177	o	$-$	$+$	o
insurance	0.070	0.010	0.069	o	$--$	$++$	o
balance	0.161	0.053	0.144	$++$	$--$	$++$	o
dermatology	0.010	0.000	0.010	o	o	o	o
iris	0.173	0.000	0.166	o	$--$	$++$	o
voting	0.046	0.043	0.046	o	o	o	o
wine	0.061	0.011	0.056	o	$--$	$++$	o
breast	0.111	0.030	0.102	$+$	$--$	$++$	o
garageband	0.347	0.291	0.326	$+$	$--$	$++$	o

This approach clearly improves the results. Now the combined approach is better than the global one on 9 of the data sets.

In analogy to the approach from Section 5.3.1, the local model algorithm can also be applied with a clusterer for describing the local pattern. The following table show the results of the local model algorithm without iteration and with k-medoids clustering as local pattern learner.

Name	Error			Significance			
	Glob	Local	Comb	$G \geq C$	$L \geq C$	$G \geq L$	$L < \tau$
business	0.260	0.139	0.228	++	−	++	*o*
covtype	0.249	0.235	0.241	++	*o*	+	*o*
diabetes	0.307	0.225	0.276	++	−−	++	*o*
digits	0.072	0.001	0.050	+	−	++	*o*
physics	0.424	0.317	0.414	+	−−	++	*o*
ionosphere	0.227	0.131	0.210	*o*	−−	+	*o*
liver	0.361	0.304	0.350	+	−−	++	*o*
medicine	0.277	0.274	0.274	+	*o*	*o*	*o*
mushroom	0.030	0.001	0.002	+	−	+	*o*
promoters	0.185	0.102	0.185	*o*	−	+	*o*
insurance	0.070	0.010	0.033	++	−−	++	*o*
balance	0.161	0.053	0.137	++	−−	++	*o*
dermatology	0.010	0.000	0.010	*o*	*o*	*o*	*o*
iris	0.173	0.000	0.120	++	−−	++	*o*
voting	0.046	0.043	0.046	*o*	*o*	*o*	*o*
wine	0.061	0.011	0.061	*o*	−−	++	*o*
breast	0.111	0.030	0.102	*o*	−−	++	*o*
garageband	0.347	0.291	0.338	+	−−	++	*o*

Similar to the results for local pattern detection, this approach is clearly superior to pattern detection by a classifier. Now the combined algorithm outperforms the global learner on 12 of the data sets instead of 9.

6.2 Basis Pursuit

The analysis in Section 6.1 has shown that the critical part in learning with global and local models is the combination of both models. In particular it has been shown that the problem is to integrate the τ-constraint with the goal of minimum accuracy, which a standard learner is not designed to do. This section will tackle this problem by defining a new SVM-type classifier that integrates the information of a global model, such that it combines the excellent performance of Support Vector Machines with a minimal disagreement from the global model.

In order to keep this section concise, it will deal only with logical learners, in particular JRip, as global models and radial basis Support Vector Machines for the local part. We will however see that the approach does not build upon the specific properties of these learners and can in fact be used with any global model and any SVM kernel.

Several approaches have been discussed in the literature in order to combine the excellent understandability of a logical representation with a better generalization performance of numerical models. For example, [Botta and Piola, 2000] proposes to optimize first order rules on numerical domains by using a gradient descend method to optimize the discretization that is necessary to transform numerical data into first-order representation. First-order rules are used as features for a numerical learner in [Kijsirikil and Sinthupinyo, 1999] by constructing a bit vector of the outputs of each rule. This approach does not increase understandability, as the combination of the features is completely unintelligible, but is well suited to tackle noise in the input data. Similar approaches can also be found in [Basilio et al., 2001] and [Popescul et al., 2002].

Basis Pursuit [Chen et al., 1998] is a method for the sparse approximation of a real function f in terms of a set of basis functions g_i. There is no restriction on the form of the basis functions. The idea is to generate a set of test points x_i and find a linear combination $\sum_i \alpha_i g_i$ of the basis functions that approximates the target function on this test points as closely as possible. To achieve sparseness, the 1-norm $||\alpha||_1 = \sum_i |\alpha_i|$ of the parameter vector is added as a complexity term similar as in Support Vector Machines. This norm is chosen because it will give much sparser result than the 2-norm, but is computationally more tractable than the otherwise preferable 0-norm $||\alpha||_0 = \#\{i|\alpha_i \neq 0\}$. The criterion of sparseness will guarantee that only the most informative basis functions will be chosen in the final function approximation.

The new idea is now to apply the same idea to a classification framework with Support Vector Machines. Standard Support Vector Machines already achieve good generalization performance by combining basis function $g_i(x) = K(x_i, x)$ given by the kernel centered at the training points x_i. When the prediction $r(x)$ of a rule learner is added as another basis function, it should already give a good prediction of the labels and hence be a very informative feature, such that most of the other features can be ignored.

Remember that the standard SVM optimization task (see Section 2.1.3) is given by

$$\begin{array}{rcl} ||w||^2 + C\sum_{i=1}^{n} \xi_i & \to & \min \\ & & \text{w.r.t.} \\ & \forall_{i=1}^{n} & y_i f(x_i) \geq 1 - \xi_i \\ & \forall_{i=1}^{n} & \xi_i \geq 0 \end{array}$$

where the decision function f has the form

$$f(x) = w * \Phi(x) + b = \sum_i \alpha_i K(x_i, x) + b.$$

To integrate the rule learners predictions $p(x)$ into the SVM and to achieve focus on the local errors, the form of the decision function is now changed to

$$\begin{array}{rcl} f(x) & = & p(x) + \sum_i \alpha_i K(x_i, x) \\ & =: & p(x) + f_{num}(x) \end{array}$$

where it is assumed that $p(x) \in \{-1, 1\}$. This SVM formulation will be called the Local Model SVM. The change in the decision function has two effects. First, for any example (x_i, y_i) that is correctly classified by p it holds that

$$\begin{array}{rl} & y_i f(x_i) \geq 1 - \xi_i \\ \Leftrightarrow & y_i(p(x_i) + f_{num}(x_i)) \geq 1 - \xi_i \\ \Leftrightarrow & y_i p(x_i) + y_i f_{num}(x_i) \geq 1 - \xi_i \\ \Leftrightarrow & y_i f_{num}(x_i) \geq -\xi_i \end{array}$$

which is automatically fulfilled when $f_{num}(x_i) = 0$ and hence for the examples correctly predicted by p the optimal solution is $w = 0$. Additionally, the constant b has been removed from the decision function. This has a special effect for locally concentrated basis functions such as the radial basis kernel function

$$f_{\gamma,x_i}(x) = e^{-\gamma||x_i - x||^2}.$$

Any nonzero term b would require that in order to meet $f_{num}(x_i) \geq 0$ there need to be basis functions with nonzero α if there are examples such that $by_i < 0$. Hence, setting $b = 0$ leads to a sparse decision function on the region correctly predicted by p. It follows further that only errors of p lead to a value of $f_{num}(x)$ that is significantly different from 0 and hence the absolute value of f_{num} may give an indication of the error probability of p.

Using the standard technique of Lagrangian multipliers it can easily be seen that the dual formulation of the new SVM problem is given by

$$W(\alpha) = \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i * x_j - \sum_{i=i}^{n} (1 - y_i p(x_i))\alpha_i \quad \rightarrow \quad \min$$
$$\text{subject to: } \forall_{i=1}^{n} 0 \leq \alpha_i \leq C$$

In particular, the removal of b leads to the removal of the linear constraint $\sum y_i \alpha_i = 0$ and hence to a simpler optimization problem.

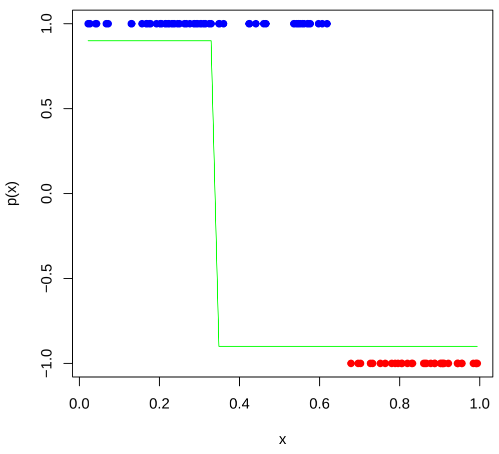


Figure 6.1: Data for Local Model SVM

Figure 6.1 shows an example data set for the local model SVM. The one-dimensional data set is given by the blue and red points while the prediction of the logical model p – in this case a simple threshold rule – is plotted in green. Figure 6.2 shows the prediction of the local model SVM, more precisely the function $f_{num}(x)$ in purple. The training points where the prediction of the logical rule $p(x)$ is false are marked as crosses. Notice that f_{num} drops to zero outside the error region of p and that for all points $y_i(p(x_i) + f_{num}(x_i)) \geq 1$ holds, which implies that the local model SVM fulfills the margin property of the standard SVM, which is imperative for good generalization performance. However, it must be noticed that the generalization property of the local model

SVM, in particular the susceptibility to overfitting, is directly dependent on the quality of the global model p. A completely overfit model p will lead to a zero local portion and hence to a overfit completely combined model. This risk can for example be reduced by training p and the local model SVM on different data sets.

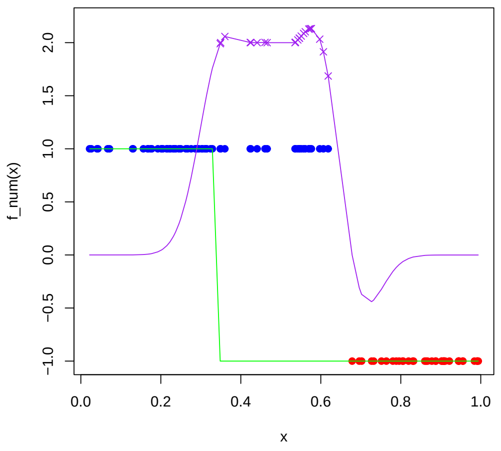


Figure 6.2: Prediction of Local Model SVM

Finally, in order to meet the τ-constraint the local decision function

$$f(x) = p(x) + f_{num}(x)$$

can easily be replaced by

$$f_\lambda(x) = p(x) + \lambda f_{num}(x)$$

where $0 \leq \lambda \leq 1$ and λ is reduced until the disagreement rate between f_λ and p drop below τ. As f_{num} reaches is maximum for points mispredicted by p, this new function will still be maximally correct.

6.2.1 Empirical Evaluation

The following experiments compare the local model SVM against two baseline techniques. As the local model SVM principally a linear combination of the global classifier p and a SVM-type classifier, the most direct solution would be to learn a linear model on the outputs of p and a standard SVM. This approach is an instance of Stacking [Wolpert, 1992]. It is straight-forward to modify the stacked classifier

$$s(x) = Ap(x) + Bf_{svm}(x)$$

into a classifier

$$s_\lambda(x) = Ap(x) + B\lambda f_{svm}(x)$$

that adheres to the τ-constraint.

On the other hand, the stacked classifier does not incorporate any local information present in the classification errors of p. This can be changed by removing all examples that are correctly classified by p from the SVMs training set. With this modified SVM classifier $f_{svm,rm}$ one defines the stacked classifier

$$r_\lambda(x) = Ap(x) + B\lambda f_{svm,rm}(x)$$

accordingly to s_λ.

The following table compare the cross-validation errors of the local model SVM, stacking and reduced stacking. The JRip classifier was used to construct the global model p. In order to simulate incomplete information in the global model, the most specific rules in each rule set were removed, such that on the average only half of the rules found by JRip were used (as the more general rules cover more examples, still most of the examples were classified by their original JRip rules). The local threshold was set to $\tau = 0.1$. The significance values for the error differences at levels $\alpha = 0.05$ and $\alpha = 0.01$ are also given.

Name	LMSVM	Stacking	$L \leq S$	Red. Stacking	$L \leq R$
business	0.216	0.255	o	0.293	++
covtype	0.241	0.264	+	0.274	++
diabetes	0.245	0.253	o	0.27	++
digits	0.027	0.003	−−	0.027	o
physics	0.409	0.417	+	0.417	+
ionosphere	0.137	0.105	−	0.174	++
liver	0.327	0.292	−	0.33	o
medicine	0.248	0.247	o	0.254	o
mushroom	0.482	0.482	o	0.482	o
promoters	0.372	0.391	o	0.391	o
insurance	0.03	0.068	++	0.07	++
balance	0.247	0.219	o	0.257	o
dermatology	0.317	0.273	o	0.317	o
iris	0.013	0.02	o	0.013	o
voting	0.233	0.207	−	0.233	o
wine	0.14	0.175	o	0.14	o
breast	0.078	0.086	o	0.099	o
garageband	0.298	0.316	+	0.337	++

Comparing the local model SVM with stacking we can see that both perform significantly better than the other on 4 data set, so it is safe to say that both perform equally well. In comparison to reduced stacking, the local model SVM performs better on 7 data sets and is never worse.

To investigate whether the local classifier f_{num} holds information about the errors of p, the following experiment uses isotonic regression (see Section 2.4.1) to transform $|f_{num}|$ into an estimator of the error probability of p. Isotonic regression was used because there is no information about the distribution of the errors except the assumption that higher values of $|f_{num}|$ indicate a higher error probability. In the following table, the mean squared error between the predicted probability and the actual occurrence of an error is given.

Name	LMSVM	Stacking	$L \leq S$	Red. Stacking	$L \leq R$
business	0.187	0.228	$+$	0.236	$++$
covtype	0.196	0.202	o	0.226	$++$
diabetes	0.193	0.196	o	0.202	o
digits	0.027	0.027	o	0.027	o
physics	0.227	0.263	$++$	0.257	$++$
ionosphere	0.123	0.143	$+$	0.155	$++$
liver	0.226	0.223	o	0.233	o
medicine	0.187	0.186	o	0.187	o
mushroom	0.086	0.336	$++$	0.383	$++$
promoters	0.34	0.275	$--$	0.303	o
insurance	0.014	0.034	$++$	0.033	$++$
balance	0.186	0.214	$+$	0.218	o
dermatology	0.021	0.194	$++$	0.195	$++$
iris	0.016	0.014	$-$	0.014	$-$
voting	0.049	0.162	$+$	0.159	$++$
wine	0.129	0.124	o	0.137	o
breast	0.078	0.111	$++$	0.08	o
garageband	0.236	0.241	o	0.284	$++$

We can see that the local model SVM has a significantly lower mean squared error than the other two methods on 9 of the data sets and is significantly worse on only 2 data sets for stacking and only 1 for reduced stacking. This shows that for the local model SVM $|f_{num}|$ carries much information about the error probability of p. This means in particular that the local model SVM will perform well for any value of τ, because selecting a value of λ is equivalent to cutting of the influence of the points with lowest values of $|f_{num}|$, which are least likely to be mispredicted by p.

To validate whether the local model SVM also leads to a sparser solution, the following table compares the number of Support Vectors for the standard SVM (as used in stacking) and the local model SVM.

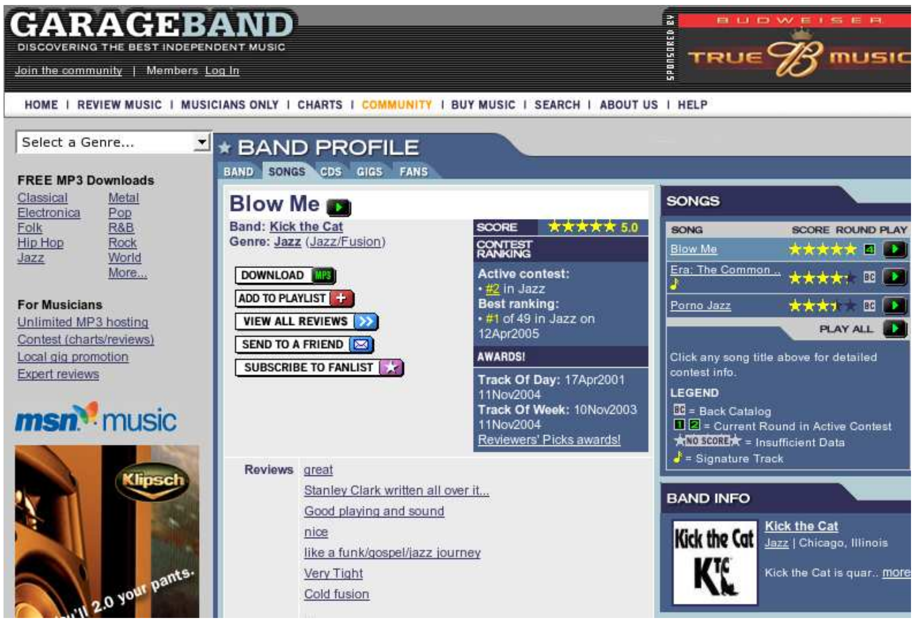
Name	SV Local	SV All	$L \leq A$
business	65.5	92.2	$++$
covtype	618.8	746.5	$++$
diabetes	349.9	388.1	$++$
digits	42.9	146.9	$++$
physics	717.4	801.6	$++$
ionosphere	110.8	119.3	o
liver	194.3	252.6	$++$
medicine	592.6	577.8	o
mushroom	999.9	999.4	$-$
promoters	78.7	95.4	$+$
insurance	174.8	178.3	o
balance	117.5	125.9	$+$
dermatology	100.1	116.9	o
iris	83.5	12.1	$-$
voting	134.4	181.9	$+$
wine	35.7	128.9	$++$
breast	68.9	86.8	$++$
garageband	734.8	807.2	$++$

We can see that on 12 of the data sets, the local model SVM returns significantly less Support Vectors than the standard SVM. It should also be noted that the kernel parameter γ of both SVMs was selected to optimize the classification performance of the standard SVM (the same values that were used throughout this thesis). An optimization of γ for the local model SVM is likely to achieve even better results.

In conclusion, the local model SVM gives far superior information about the local models errors compared to the competing solutions and is hence very well suited for extracting sparse local models and local patterns at the same time without any loss of accuracy.

6.3 A Multimedia Application of Local Models

To end this chapter on global models, an instructive application of global and local models to the field of multimedia data will be presented. In mining multimedia data, interpretability is a very important aspect because of the apparent discrepancy between the intuitive form of the objects being observed (a picture, a song) and their complex technical representations (bitmaps, Fourier transforms, phase spaces). Also the very complex dependencies between the different parts of the data (a movie and its soundtrack, a song and its lyrics) can make the interpretation of a model very hard.



Blow Me

Band: Kick the Cat
Genre: Jazz (Jazz/Fusion)

SCORE ★★★★★ 5.0

CONTEST RANKING

Active contest:
• #2 in Jazz
Best ranking:
• #1 of 49 in Jazz on 12Apr2005

AWARDS!

Track Of Day: 17Apr2001 11Nov2004
Track Of Week: 10Nov2003 11Nov2004
Reviewers' Picks awards!

Reviews
great
Stanley Clark written all over it...
Good playing and sound
nice
like a funk/gospel/jazz journey
Very Tight
Cold fusion

SONGS

SONG	SCORE	ROUND	PLAY
Blow Me	★★★★★	4	▶
Era: The Common... ♪	★★★★	BC	▶
Porno Jazz	★★★	BC	▶

Click any song title above for detailed contest info.

LEGEND
BC = Back Catalog
1 2 = Current Round in Active Contest
NO SCORE = Insufficient Data
♪ = Signature Track

BAND INFO

Kick the Cat
Jazz | Chicago, Illinois
Kick the Cat is quar... more

Figure 6.3: The Garageband.com website

The multimedia data set investigated here is the GARAGEBAND data set [Homburg et al., 2005] that was used throughout this thesis. Let us first take a closer look at this data set. *Garageband.com* is a website where independent bands can upload their music and present themselves, while users can review and rate the songs. A typical page can be seen in Figure 6.3. It contains the song ready for download, information about the band, lyrics of the song, the reviews and ratings of the users, and a classification a the song into different genres (pop, rock, jazz etc.).

The data set consists of 1885 songs from 8 genres. Each song is represented by 44 features describing the audio information, which where generated with the method described in [Mierswa and Morik, 2005], plus 508 text features which encode the reviews of the song. As the learning task defined in [Homburg et al., 2005] consists only of subsets of the songs, a new classification task was defined by letting a user listen to all 1885 songs and classify them with respect to his personal taste (like / don't like).

This data set is a good example of the problems one encounters when investigating interpretability: The user is only interested in the songs themselves and can very easily make up his mind about a song when he listens to the MP3 file. Turning the musical information into a good set features to classify the song with, however, turned out to be a very complex problem. Although given enough time good sets of features can be found using the method of [Mierswa and Morik, 2005], these features consists of complex transformations of the data like peaks of Fourier transforms and hence are are completely unintuitive. The review texts contain much less information for the user, because people have very different tastes and pay attention to very different aspects of the music when judging a song. Further, the reviews available here differ very much in quality and size, some are very detailed, some only say "nice" and for some songs there does not even exist a review. Hence, reviews usually only serve as a first indicator to filter out the most or least promising songs. On the other hands, text classification is a well-investigated problem which can be solved with high accuracy [Joachims, 2002]. The standard approach is to use the so-called bag-of-words representation, that is to represent a text by the counts of words appearing in the text. The most important words of a classification rule provide a set of keywords which usually gives the user a good intuition about the classification rule.

The idea of this section is to model this relationship between audio and text features by global and local models. The global classifier will work on the text features alone and will be restricted to a small set of features, thereby extracting a set of keywords to filter out the most and least relevant examples. The local pattern will by described by a set of clusters, again restricted to a small set of text features to extract meaningful keywords. The clusters will be described by medoids; it is obvious that a medoid corresponds to a single song and hence contains much more information than a formal cluster model on the selected text features alone. The local classifier will be obtained by training on all available features and examples.

This approach is an instance of the problem of learning with multiple views [Rüping and Scheffer, 2005], which means learning from data that is represented by independent sets of features.

As text and audio classification are well-investigated machine learning tasks, the cluster model for the local patterns will be the focus of this investigation. In particular, it will be shown that the local pattern cluster can be well improved by using additional information in the clustering, in particular information about the global model and the general structure of the set of examples. This will be implemented by using Conditional Information Bottleneck clustering, which will by explained in the next section.

Information Bottleneck

The information bottleneck method [Tishby et al., 1999] extracts structure from the data by viewing structure extraction as data compression while conserving relevant information. With the data modeled by a random variable U^1, relevant information is explicitely modeled by a second random variable V, such that there is no need to implicitly model the relevant structure in terms of appropriately choosing distance or similarity measures as in standard clustering algorithm. The idea is to construct a probabilistic clustering, given by a random variable C, such that the mutual information $I(U, C)$ between the data and the clusters is minimized, i. e. C compresses the data as much as possible, while at the same time the mutual information $I(V, C)$ of the relevant variable V and the clusters is maximized, i. e. the relevant structure is conserved. Hence, the random variable C acts as a bottleneck for the information U has about V. Both goals are balanced against each other by a real parameter $\beta > 0$, such that the goal becomes to find a clustering $P(c|u)$ which minimizes

$$F = I(U, C) - \beta I(V, C).$$

It can be shown that this problem can be solved by iterating between the following three equations

$$\begin{aligned} P(c) &= \sum_u P(u)P(c|u) \\ P(v|c) &= \sum_u P(v|u)P(u|c) \\ P(c|u) &\propto P(c)e^{\beta P(v|u)\log P(v|c)}. \end{aligned}$$

The first two equations ensure the consistency of the estimated probabilities, while the third equation gives a functional form of the clustering, depending on the Kullback-Leibler-distance between $P(v|u)$ and $P(v|c)$ (removing factors independent of c). The input consists of the probability distributions $P(u)$ and $P(v|u)$.

Condition Information Bottleneck

In [Gondek and Hofmann, 2003] and [Gondek and Hofmann, 2004] Gondek and Hoffmann extend the information bottleneck approach by considering not only information about relevant, but also about irrelevant structure. It is often easier to express what is already known and hence is uninteresting, than to specify what is interesting and relevant. Examples of such irrelevant structures are general categorization schemes and well-known properties of the data, when instead one is interested in how the data differs from what one thinks it looks like.

Conditional information bottleneck (CIB) is formulated by introducing a random variable W to describe the irrelevant information. The learning problem

[1] We use the letters U, V, W instead of the usual X, Y, Z in order to avoid confusion with the classification features X and labels Y used later

corresponds to that of standard information bottleneck with the new target function

$$F = I(U,C) - \beta I(V,C|W)$$

That is, one wants to maximize the information that C has of V, given that W is already known. In a way, the goal is to extract information orthogonal to what one can already infer via W.

Again, the problem can be solved by iterating between three estimation equations

$$\begin{aligned} P(c) &= \sum_u P(u)P(c|u) \\ P(v|w,c) &= \sum_u P(v|u,w)P(u|w,c) \\ P(c|u) &\propto P(c)e^{\beta \sum_w P(w|u) \sum_y P(v|u,w) \log P(v|w,c)} \end{aligned}$$

The probabilities $P(v|u,w)$, $P(w|u)$ and $P(u)$ have to be given to the learner as input.

Informed Clustering for Local Patterns

Informed clustering describes the setting where the desired structure to be extracted by a clustering is not only defined implicitly using the distance or similarity function, but also explicit information about relevant and irrelevant structure is given. The conditional information bottleneck algorithm is one such approach, where one can explicitely define irrelevant structure which the clustering algorithm should ignore.

There are two kinds of irrelevant information one can exploit. First, one can use a probabilistic clustering $p(c|x)$ of the complete training observations $(x_i)_{i=1\ldots n}$. This clustering shows, what the data generally looks like and can be used as a background model to discriminate the local examples against. Note that we do not require an information bottleneck clustering at this stage, we could also use any other probabilistic clusterer. It would also be possible to use an existing description of the data set at this stage (e. g. an ontology given by the user).

The other method is to define the prediction of the global model or, more precisely the conditional class probability $P_{orig}(Y=1|x)$ as irrelevant. The idea here is that the global model is used anyway and that it is better to look for independent sources of information.

In either case, one arrives at a well-defined probability $P_{cib}(w|u)$. Now one can set up the conditional information bottleneck problem to find the local patterns as follows: Identify U with the index i and the relevant features V with the classification features X:

- $P_{cib}(u) = P(x_i) = P_{orig}(x_i|Y \neq f(x_i))$
- $\forall w : P_{cib}(v|u,w) = P_{ib}(v|u)$

In other words, the problem is to compute a probabilistic clustering $P(c|u) = P(c|x_i)$ of the observations x_i which are misclassified by the classifier f (controlled by $P_{cib}(u) = P_{orig}(x_i|Y \neq f(x_i))$), such that the clustering describes how the local examples differ from the complete data set (via defining the cluster information $P_{ib}(v|u)$ of the complete data set as irrelevant) or how the local examples differ from structure from the global model (via $P_{orig}(Y = 1|x)$).

To arrive at local models, we rank the CIB features y by their relevance to the clusters c according to the mutual information $I(y, c) = P(y, c) \log \frac{P(y,c)}{P(y)P(c)}$. The larger $I(y, c)$ is, the higher the probability that y and c appear together. In particular, $I(y, c) > 0$ iff the joint probability $P(y, c)$ is higher than the probability $P(y)P(c)$ that could be predicted from assuming independence. An average of $I(y, c)$ over all c gives a relevance measure for the features over all clusters. Selecting the few most relevant features introduces a low-dimensional view on the examples, which may give the user an intuition about the clusters.

As the information bottleneck method does not return a cluster model, but only the cluster membership probabilities $p(c|x)$, a model that induces these memberships has to be found in order to apply the clustering to new observations. Following the goal of interpretability, it is advantageous to use a k-medoids clustering model, as it is often easier to interpret single examples than models. For each cluster, we choose that example as medoid, which – in the space of projections on the most relevant features – minimizes the expected distance of the medoid to the examples in the cluster, where expectation is taken with respect to the probability $P_{cib}(x, c)$ for the cluster c. The local pattern indexed by c consists of all points x which lie as least as close the medoid of the cluster c as one of the $\tau n P_{cib}(c)$ closest training examples, where τ is the user-defined local pattern threshold. This definition takes into account the sizes $P_{cib}(c)$ of the cluster from the CIB method and ensures that only a fraction of τ of all examples lie in a local pattern.

6.3.1 Experimental Investigation

In these experiments, a linear Support Vector Machine was used as both global and local classifier. Feature selection for the global classifier was performed by repeatedly removing the features with lowest absolute weight in the decision function.

To encode the text information as a probability distribution $P(v|u)$ for the information bottleneck clustering, the encoding of feature counts as probabilities from [Gondek and Hofmann, 2003] was used here. This necessary encoding step is the reason why the approach here is not readily transferable to other data sets with continuous features.

The initial information bottleneck clustering was parameterized to return 8 cluster, in correspondence with the 8 music genres, and its parameter β was set to maximize the correspondence to the genres. However, the most informative words regarding the clustering were HELLO, POWER, BLEND, SOUNDS, BABY, FAT, QUIET, BIT, NIGHT, and GIVE, which do not seem to reveal any obvious genre structure.

Feature selection for classification returned GROOVE, SMOOTH, CHILL, JAZZY, MOOD, FUSION, PIANO, PIECE, PAUL, and JAZZ as the most important features. It is obvious that a certain music taste can be associated with this set of keywords[2].

[2]The reader might ask who Paul (second to last keyword) is. An inspection of the texts showed that in fact there are several people with first name Paul in the data, who all make good music according to the users taste. This was sufficient to make the keyword PAUL very

The CIB clustering returned TALENT, BABY, SOUNDS, CHECK, NEAT, PASS, TRUE, NICE, SEXY, and CHORUS as the most important features. Interestingly, extracting two medoids from this clustering showed that the first medoid consists only of the word CHORUS with no occurrence of the other keywords and the second medoid consists of the words SOUNDS and NICE with no occurrence of the other keywords. This is a result of the sparse structure of the text data, as a medoid as any other example will have only a few nonzero features. For sparse data it may be instructive to try out a different procedure to describe the CIB clusters. However, the second medoid with the keywords "sounds nice" seems to indicate that there are two aspects to musical taste in this data set, the genre – which the initial clustering was optimized against – (the classifier indicates that the user seems to like jazz) – and the quality of the music independent of the style (whether the song sounds nice or not).

5-fold cross-validation showed an accuracy of the global model of 0.624 ($\sigma = 0.0284$), while the local model achieved an accuracy of 0.670 ($\sigma = 0.0164$) measured over all examples. The combined model achieves an accuracy of 0.649 ($\sigma = 0.0230$). This lies between the accuracies of the global and the local model, which was expected, as the amount of examples that the global and the combined differ on is bounded by the parameter τ (in this experiment, $\tau = 0.05$), which stops the local model from correcting more errors of the global model.

To validate that the increase in performance is indeed a result of the conditional information bottleneck approach, the experiment was repeated with a standard information bottleneck clustering of the global models errors instead of the CIB step (all other parameters left constant). With the same accuracies for the global and local classifiers, the accuracy of the combined classifier dropped to 0.627 ($\sigma = 0.0329$). This proves that the conditional information bottleneck clustering finds novel structure in the errors of the global classifier.

To validate the effect of the parameter τ and the number of features for the CIB clustering, more experiments were conducted. The result can be seen in the following table.

Parameters		Accuracy			Disagree
τ	#features	global	local	combined	
0.05	10	0.624	0.670	0.648	0.147
0.05	20	0.624	0.670	0.653	0.201
0.025	10	0.646	0.670	0.642	0.070
0.025	20	0.646	0.670	0.646	0.019

The table shows the accuracies of the global, local and combined models and the disagreement rate (fraction of examples classified differently) between the global and the combined model. We can see that the combined model performs better when more features for the CIB clustering are present. We also see that the actual disagreement rate is higher than the given threshold τ. This is again a result of the sparse nature of the data, as in the space projected on the most important keywords, several different examples fall together, which prevents a more fine grained control of the number of local examples. An obvious tradeoff between interpretability in terms of number of features and accuracy can be observed here.

informative

In conclusion, this specific example showed how local patterns and local models can serve to model the approach of filtering songs with a keyword-based search, as most users will do when they look for interesting songs by hand. Further, local patterns could be interpreted themselves and showed up an interesting aspect about the data that was not modeled by the global classifier, namely the general judgment of the other users about the quality of the songs. Indeed, the rating that is available on the website was not included in the features and with these results of the local patterns one could think of including this information if one wants to further improve the results.

6.4 Conclusions

This chapter presented two approaches for learning local models. The approach in Section 6.1 is more general, because it can be applied with any base learner. The approach in Section 6.2 is restricted to SVMs as local model learners, but the experimental investigation showed that it can be performed without an a-priori selected value of τ.

Section 6.3 presented an exemplary application of local models that showed how local models can generate a very clear and concise description of the underlying patterns in the data, in this case consisting of 10 keywords out of more than 500 words.

A main result of this chapter is that although local models are easily understandable, finding these models is a very complex task. Hence, it is imperative to have a clear concept on the semantics of local models, e.g. as given by the τ-constraint and apply rigorous tests of model correctness in the learning phase.

Chapter 7

Conclusions

Learning interpretable models is a challenging task, whose complexity comes from the problems that interpretability is a fuzzy, subjective concept and human mental capabilities are in some ways astonishingly limited. Interpretability is a critical problem, because it is crucial for problems that cannot be solved purely automatically.

In order to find an interpretable model, it is not sufficient to generate a model that is easily understandable to the user. At the same time, the model must be accurate enough such that it is related to the true structures behind the data in a sensible way. Moreover, as interpretability is only one of several goals in a knowledge discovery process, optimizing interpretability should be efficient enough such that it does not become a performance bottleneck. The work presented in this thesis is structured along these three dimensions of understandability, accuracy and efficiency.

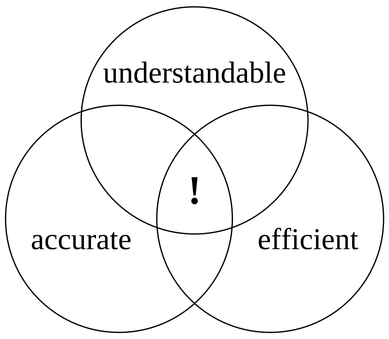


Figure 7.1: Three goals of interpretability

No single method exists that can successfully solve all requirements for interpretable models. Accordingly, the approach of this thesis was to construct a set of approaches to cover all single aspects of the interpretability problem, taking into account the relations to the two other aspects, respectively.

7.1 Contributions of this Thesis

This thesis contains contributions on the levels of the optimization of a learner with and without knowledge of its internals (white box and black box approach), the description of a models errors by local patterns and the improvement of global models with local models. This necessarily includes a contribution to the basic method of probabilistic scaling.

Probabilistic Scaling: Probabilistic information is a basic instrument to analyze a learner's output beyond the level of simple averages. Correctness probabilities are the formal and widely understood measure of the confidence of the learner in its prediction. They allow the user to form an opinion about the quality of the predicted label itself. This gives the user important information about the level of trust he can put in the prediction in critical situations. Further, confidence probabilities serve as an intermediate tool in the detection of local patterns to quantify how much this example fits to structure encoded in the overall hypothesis.

Section 2.4.2 introduced an improved probabilistic scaling method by integrating the concept of robustness, which is particularly important for scaling models with undetected local patterns. It was shown that this improved the quality of probabilistic scalers significantly.

7.1.1 Black Box Optimization

Black box optimization considers the methods to control a learned model that are accessible from outside the learner. These are the input data, in particular its size and dimension, the hypothesis space of the learner, which means choosing between one learner and another, and the parameters the learner might have. The feasibility of optimizing the understandability of the final model by influencing each of these controls were investigated.

Feature Selection: A well-known method for the optimization of classification models is to reduce the number of features that the model uses. Although the connection between Feature Selection and interpretability is obvious, it is usually used with the goal of optimizing classification performance. With respect to interpretability, it is important that the feature selection strategy can deal with very high amounts of features and many attributes, where interpretability matters most. Since interpretability is often pursued as a secondary goal next to performance (e.g. in an additional visualization step), it is important that the feature selection method is fast.

Section 3.1.5 presented a feature selection method for large-scale non-linear classifiers such as the SVM. This methods requires almost no additional effort after building the model and it was shown that it provides a better selection than most other methods and surpasses all other methods with respect to the relation of accuracy to efficiency.

Instance Selection: Selecting the most relevant observations in a data set is the complementary problem to feature selection. But unlike feature selection, the quality of an instance selection method is hard to evaluate directly, because learning with an extremely reduced number of instances is usually infeasible. Hence, a direct evaluation of the approximation quality of the selected instances fails.

Section 3.2 showed that by extending the idea of data squashing it is possible to select a number of examples and a clustering of the input space that reproduces the structure that the hypothesis space of the learner imposes on the input space. It was shown that these prototypes can be derived from an argument of maximizing the information the clustering has about the data. In combination, this shows that the selected instances are representative for the data set from the view of the learner.

Piecewise Linear Approximation: Choosing a simpler representation of a model can have a big influence on understandability. For example, in the class of numerical representations, linear models are a very easy to understand and frequently used form. On the other hand, simple representations in many cases fail to cover the structure of the entire data. An approach that combines simple representations with a higher expressive power by constructing several sub-models was introduced in this thesis.

In Section 3.3 it was shown that although a linear model itself is often not adequate for complex data, the combination of a small number of linear models can effectively approximate more complex nonlinear models. In particular, a novel algorithm was introduced that combines linear models with clustering in order to find an effective piecewise linear approximation of a nonlinear model.

Direct Complexity Reduction: To avoid over-fitting the data, learning algorithms usually possess one or more parameters that control the complexity of the model. This raises the question about the dependency between this control of formal complexity and understandability. Can understandability be controlled by this parameters?

An empirical investigation of this question that was presented in Section 3.4 showed both positive and negative results. On the positive side, it was shown that in many cases a high reduction of complexity was possible with little impact on performance, that is, for a standard learning algorithm which is designed to optimize accuracy, it is not necessary to construct a most compact and non-redundant model. On the negative side, no dependency of a parametric form could be obtained, such that it is not possible to construct an analytical guideline for choosing the optimal parameter. This means that the combined optimization of both the performance and the complexity criterion requires the high computational effort of testing out a large set of parameter values.

7.1.2 White Box Optimization

The optimization of the interpretability of a model is much easier if one can analyze the structure of a model directly. But it follows on the other hand that any white box optimization is limited to a specific class of models. In this thesis, the popular class of Support Vector Machines was investigated, which is widely recognized as a very effective, but not easily interpretable algorithm. In order to understand a complex numerical model such as SVM models, one can either try to simplify the model in the hypothesis space it is given in or represent it in another, more easily accessible space. Both approaches were investigated, where logical models and visualizations where considered as alternative representations.

Reduced Set Approximations of Support Vector Machines: Pre-images and reduced set approximations of Support Vector Machines are an approach to simplify a SVM model in the functional space that is defined by the kernel function. While this task is relatively easy for finite-dimensional hypothesis spaces, infinite dimensional cases like Radial Basis Functions pose a hard problem of simplification and approximation.

Next to an algorithmic optimization of reduced set methods, Section 4.1 presented a new functional space for Radial Basis SVMs, that allows to approximate SVMs in an extended kernel space. The augmented space allows to find better approximations, while at the same time its inner product and hence its geometric properties remain identical to the original inner product defined by the kernel. This means that the new distance measure of functions is based on the same smoothness properties that form the basis of the good SVM inductive properties.

Logical Approximations of Numerical Functions: Translating a model into another formal language can very much increase understandability. This was exemplified by the description of numerical models by logical formulas in Section 4.2.

It was empirically shown that when there does not exist a way to directly translate one model into the other, the only dependency given via the training data, a trivial approximation algorithm is already statistically optimal. In other words, existing approaches that promise to extract additional structure, e.g. by using the given model to generate new labeled examples, are misleading because they construct finer approximations on some regions of the input space without any empirical evidence that this region is important.

Visualization of Support Vector Machines: The key of a good visualization of high dimensional data is to present the user the structure that is important and filter out the irrelevant information. To visualize which the structure that a classification models extracts from the data, one can either plot the model along the dimensions of the data and let the user reason about the structure of the model, or plot along the structure of the model and let the user discover meaningful real-world structures.

While a multitude of techniques exist for the first approach, the complementary approach is harder to solve in the general case, as it requires a meaningful feature extraction from the model. For the case of SVM models, Section 4.3 presented an approach that solves this problem. It combines for the first time feature extraction and kernel principal component analysis to extract features which at the same time describe the model itself and the additional structure that the hypothesis space induces on the data.

7.1.3 Local Patterns

Models cannot be arbitrarily simplified without losing some amount of information. When this happens, it is important to give the user detailed information on the impact of this simplification on accuracy. In order to give understandable assertions about the approximation quality of the less complex model, and with the goal of distinguishing different levels of complexity in the data, local patterns in classification models where introduced.

Describing a Model by Local Patterns: Section 5.1 introduced the idea that when a learner has been found to give interpretable models, it can not only be used to give an approximation of the data, but also to describe the quality of the approximating model. This gives qualitative guarantees about the model correctness, which give the user far more information than traditional numerical estimates such as error rates.

Optimizing a Model by Local Patterns: Examples from Robust Statistics (see Section 2.3) shows that for several learners it is possible that a model may be arbitrarily disturbed by a small set of mischievously placed observations. This raises questions about the quality of the restricted interpretable model.

Sections 5.2 and 5.3 investigated this problem. It was shown that an optimization of the interpretable model is possible with an EM-style algorithm. However, a theoretical analysis showed that this is a very complex task with much risk of over-fitting, such that the optimization algorithm has to be very much restricted. In particular, this shows that traditional approach of residual analysis, i.e. the analysis of errors made by a model in order to improve this model, cannot directly be applied with complex learners.

7.1.4 Local Models

When too much accuracy is lost by optimizing for interpretability, it is necessary to use two different models for prediction and for analysis by the user. Extending the idea of Chapter 5, Chapter 6 introduced the idea modeling the local error patterns of the interpretable global model by additional, more complex local models. While the idea of combining several models into one complete model itself is not new, the idea of integrating an understandable approximation (global model and local pattern) and a hierarchical model into the same approach has not yet been considered.

The novel approach of achieving understandability by enforcing interpretability constraints at the levels of the local patterns and the global model and by the τ-constraint requires much different algorithms than usual hierarchical models. Two such approaches have been introduced in this thesis.

Learning Local Models: Section 6.1 presented an EM algorithm for local models on the basis of the EM local pattern algorithm from Section 5.2. This algorithm is applicable to all base learners and has been shown to optimize the global and local models, however at the cost of a high chance of over-fitting and high computational effort.

Basis Pursuit for Local Models: An alternative approach based on Support Vector Machine optimization was introduced in Section 6.2. This approach is more efficient and less prone to over-fitting than the EM local model algorithm, but at the trade-off of being restricted to an SVM model for the local models.

In summary, these two approaches show that interpretability can be optimized in a classifier-independent way, but that the most potential lies in constructing learning algorithms that incorporate interpretability considerations from the beginning.

7.2 Summary

What can we learn from this thesis about interpretable models? First of all, the more knowledge that is put into the learner, the better the model will be, in terms of accuracy, performance, and interpretability. This can be seen from the good performance of the white box approach (Chapter 4) and the local model SVM (Section 6.2). Additionally, the selection of the learning algorithm and the set of features that is right for the user is crucial, see the multimedia example in Section 6.3.

However, as knowledge discovery nowadays changes more and more from a pure research topic into large-scale, highly automated applications, it is necessary to limit the amount of human interaction that is needed. Ideally, the end-user should be capable of carrying out standard knowledge discovery tasks by his own, without involvement of highly experienced data mining experts, and still be capable of understanding what models he has found. This real-world scenario – the optimization of interpretability with standard tools and algorithms – has been the focus of this thesis.

It has been found that a high amount of understandability can already be gained by automatizable optimization techniques such as feature or parameter selection. This is mainly a question of efficiency of the applied procedures, see for example the one-step feature selection in Section 3.1.4 versus complete feature selection with $\Omega(2^d)$ steps. For widely used standard learning algorithms, such as the Support Vector Machine, it is possible to prepare a set of interpretability optimizing techniques, such as the ones presented in Chapter 4. The availability of such techniques should be taken into account in the choice of the learning algorithm. Finally, when a joint optimization of interpretability and accuracy in one model fails, local patterns and local models offer a structured way to split the model into parts of different complexity.

The main insight of this thesis is the importance of considering the requirements of understandability from the very beginning of the data mining process and in the design of learning algorithms themselves. Interpretability is hard to achieve as a post-processing step of an existing model, instead the relation of requirements for accuracy and understandability in the context of the available time-frame for the analysis have to be considered jointly. A main point in this context is to give guarantees about the relationship of accuracy and understandability, because nothing is more misleading than correctly understanding a false model.

In summary, the contribution of this thesis is a structured investigation of interpretability in classification models. Starting from an analysis of the requirements for and measures of interpretability in the context of knowledge discovery, the diverse possible approaches of generating understandable models have been investigated, with a particular focus on interpretable SVMs and local effects in the data. This allowed to analyze problems of existing techniques and ad-hoc approaches to understandability optimization and develop new, improved algorithms.

Chapter 8

Data Sets

This chapter gives a short description of the data sets used in this thesis. Data sets were selected both from the well-known UCI machine learning repository [Murphy and Aha, 1994] and from several real-world applications and chosen as to cover a wide range of number of attributes, dimensionality and complexity, ranging from the trivial iris data set to the very complex garageband data.

For all data sets, continuous attributes were scaled to expectancy 0 and variance 1. Multi-class data sets have been converted to binary tasks by selecting two of the classes or by joining several classes to one.

8.1 Description of the Data Sets

What follows is a short description of each data set. If nothing else is stated, the data sets can be found in the UCI machine learning repository.

Balance

This data models results from psychological experiments. The original data set contains 4 numeric attributes (values in $\{1,\ldots,5\}$) and 625 examples in 3 classes. In the experiments here, only the examples from the two largest classes (L and R) have been selected, resulting in a binary problem with 576 examples, 288 from each class.

Breast Cancer

This data set consists of patients records for breast cancer diagnosis. It originally consists of 9 attributes and 699 examples in two classes. For the experiments here, 16 examples with missing values were removed.

Covtype

The task here is to predict forest cover types from cartographic variables. The original data set consists of 581012 instances in 7 classes with 54 attributes. Here, only a 1% sample of the two largest classes (Spruce-Fir and Lodgepole Pine) was used and attributes that were constant on the sample were removed. This results in a data set of size 4951 with 48 attributes.

Dermatology

This data contains examples of dermatologic diseases. It originally consists of 34 attributes and 366 examples in 6 classes. Here, the two largest classes (psoriasis and lichen planus) were used, resulting in 184 examples.

Diabetes

The Pima Indians Diabetes Database consists of 768 examples with 8 attributes.

Digits

This data set is concerned with the optical recognition of handwritten digits from 32x32 bitmaps, converted to 8x8 integer matrices. Here, only the digits 1 and 7 are used to construct a data set with 776 examples.

Ionosphere

This data set of radar measurements of the ionosphere consists of 351 examples with 34 attributes.

Iris

This simple data set describes iris plants. It originally consists of 150 examples in 3 classes with 4 attributes. In the experiments here, the task was changed in discriminating one class (setosa) against the rest.

Liver

This liver-disorders database consists of 345 instances with 7 attributes.

Mushroom

The task in this data set is to decide whether a mushroom is poisonous based on its physical characteristics. It originally consists of 8124 examples with 22 nominal attributes. Here, the data was converted to 126 binary attributes.

Promoters

This molecular biology databases originally consists of 106 examples with 57 attributes with 4 distinct values each. Converting the data into binary form resulted in 228 attributes.

Voting

This database contains voting records from the United States Congress. The task is to classify the voter as Republicans or Democrats based on their votes on different issues. It consists of 435 examples and 16 attributes. The binary data with missing values was converted into numerical attributes in $\{-1, 0, 1\}$ with 0 representing missing values.

Wine

The data set contains chemical analyses of wine and the task is to determine the origin of the wine. It originally contains 13 attributes and 178 instances in 3 classes. Here, the task was changed to discriminate the largest class (class 2) against the rest.

Business

The task in this data set is to classify German business cycles based on certain macroeconomic variables. It originally consists of 157 observations in 4 classes with 13 attributes. In the experiments here, the problem was changed into binary classification by combining the classes upswing and upper-turning-point into one class and downswing and lower-turning-point into the other. More information about this data set can e.g. be found in [Heilemann and Münch, 2001] and [Morik and Rüping, 2002].

Insurance

This data set consists of 10000 record of insurance customers with 135 attributes identifying the changes in the customers contract. The task is to predict whether a customer will cancel his contract. See [Morik and Köpcke, 2004] for more information.

Physics

This data set is the physics data set from the 2004 KDD Cup. A 10% sample was drawn resulting in 5000 examples and 78 attributes.

Medicine

The task in this data set from intensive care medicine is to decide whether the dose of a certain drug (dobutrex) should decreased or not. It consists of 6610 examples and 18 classes and is described in [Morik et al., 2002].

Garageband

This music data set defines the task of classifying songs according to a users taste. The data consists of 44 audio features describing the songs themselves plus 508 text features in TF/IDF representation encoding reviews of the song from other users. In total, there are 552 attributes and 1885 examples. The observations from the data set are described in [Homburg et al., 2005]. The labels are not part of the original data set and were generated independently by the author of this thesis by listening to the 1885 songs.

8.2 Data Set Statistics

The following table sums up the format of the data sets:

Name	Size	Dimension
balance	576	4
breast-cancer	683	9
covtype	4951	48
dermatology	184	33
diabetes	768	8
digits	776	64
ionosphere	351	34
iris	150	4
liver	345	6
mushroom	8124	126
promoters	106	228
voting	435	16
wine	178	13
business	157	13
insurance	10000	135
physics	5000	78
medicine	6610	18
garageband	1885	552

Finally, in the following table the accuracy values of the four learners used in this thesis are given. The reported values are the results of 10-fold cross-validation tests.

Name	lin. SVM	RBF SVM	J48	JRip
balance	0.951	0.986	0.871	0.885
breast	0.969	0.973	0.961	0.958
covtype	0.785	0.807	0.791	0.775
dermatology	1.000	1.000	1.000	1.000
diabetes	0.774	0.770	0.730	0.731
digits	0.997	0.997	0.992	0.994
ionosphere	0.888	0.940	0.897	0.903
iris	1.000	1.000	0.986	0.986
liver	0.698	0.696	0.652	0.658
mushroom	1.000	0.999	1.000	1.000
promoters	0.936	0.904	0.808	0.819
voting	0.967	0.965	0.963	0.949
wine	0.988	0.988	0.950	0.904
business	0.854	0.866	0.770	0.772
insurance	0.998	0.993	0.996	0.962
physics	0.686	0.685	0.635	0.669
medicine	0.752	0.804	0.795	0.755
garageband	0.725	0.728	0.665	0.696

Bibliography

[Agrawal et al., 1996] Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., and Verkamo, A. I. (1996). Fast discovery of association rules. In Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., editors, *Advances in Knowledge Discovery and Data Mining*, chapter 12, pages 307–328. AAAI Press/The MIT Press, Cambridge Massachusetts, London England.

[Agrawal and Srikant, 1994] Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules in large data bases. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB '94)*, pages 478–499, Santiago, Chile.

[Akaike, 1973] Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In *Second International Symposium on Information Theory*, pages 267–281.

[Amaldi and Kann, 1998] Amaldi, E. and Kann, V. (1998). On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209:237–260.

[Andrews and Diederich, 1996] Andrews, R. and Diederich, J., editors (1996). *Rules and Networks – Proceedings of the Rule Extraction from Trained Artificial Neural Networks Workshop.* Queensland University of Technology, Neurocomputing Research Center.

[Aronszajin, 1950] Aronszajin, N. (1950). Theory of reproducing kernels. *Transactions of the Mathematical Society*, 68:337–404.

[Bakir et al., 2003] Bakir, G. H., Weston, J., and Schölkopf, B. (2003). Learning to find pre-images. In *Advances in Neural Information Processing Systems*, volume 16, pages 449–456.

[Barnett, 1976] Barnett, V. (1976). The ordering of multivariate data. *J. Roy. Statist. Soc. Ser. A*, 139:319–354.

[Barnett and Lewis, 1994] Barnett, V. and Lewis, T. (1994). *Outliers in Statistical Data.* Wiley Series in Probability and Statistics. Wiley, 3rd edition edition.

[Bartlett and Tewari, 2004] Bartlett, P. L. and Tewari, A. (2004). Sparseness versus estimating conditional probabilities: Some asymptotic results. In *Learning Theory: 17th Annual Conference on Learning Theory, COLT 2004*, pages 564–578.

[Basilio et al., 2001] Basilio, R., Zaverucha, G., and Barbosa, V. C. (2001). Learning logic programs with neural networks. In Rouveirol, C. and Sebag, M., editors, *ILP 2001*, volume 2157 of *LNAI*, pages 15–26. Springer.

[Baxter, 1989] Baxter, J. (1989). Children's understanding of familiar astronomical events. *International Journal of Science Education*, 11:502–513.

[Bellman, 1961] Bellman, R. (1961). *Adaptive control processes: A guided tour.* Princeton University Press.

[Bishop, 1995] Bishop, C. (1995). Neural networks for pattern recognition. *Oxford University Press, Oxford, England.*

[Botta and Piola, 2000] Botta, M. and Piola, R. (2000). Refining numerical constants in first order logic theories. *Machine Learning Journal*, 38:109–131.

[Bratko, 1996] Bratko, I. (1996). Machine learning: Between accuracy and interpretability. In Della Riccia, G., Lenz, H.-J., and Kruse, R., editors, *Learning, Networks and Statistics*, volume 382 of *CISM Courses and Lectures*, pages 163–177. Springer.

[Breuer, 1970] Breuer, M. A. (1970). Simplification of the covering problem with application to boolean expressions. *Journal of the ACM*, 17(1):166–181.

[Burges, 1996] Burges, C. (1996). Simplified support vector decision rules. In *Proceedings of the International Conference on Machine Learning*, pages 71–77.

[Carbonell et al., 1983] Carbonell, J., Michalski, R., and Mitchel, T. (1983). An overview of machine learning. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning — An Artificial Intelligence Approach*, volume 1, chapter I, pages 3–25. Morgan Kaufmann, Palo Alto, CA.

[Carbrera et al., 1997] Carbrera, J., Maguluri, G., and Singh, K. (1997). Indices of empirical robustness. *Statistics & Probability Letters*, 33:49–62.

[Chan and Stolfo, 1993] Chan, P. K. and Stolfo, S. (1993). Experiments in multistrategy learning by meta-learning. In *Proceedings of the second international conference on information and knowledge management*, pages 314–323, Washington, DC.

[Chapman et al., 1999] Chapman, P., Clinton, J., Khabaza, T., Reinartz, T., and Wirth, R. (1999). The crisp–dm process model. Technical report, The CRIP–DM Consortium NCR Systems Engineering Copenhagen, Daimler-Chrysler AG, Integral Solutions Ltd., and OHRA Verzekeringen en Bank Groep B.V.

[Chatfield, 1984] Chatfield, C. (1984). *The Analysis of Time Series: An Introduction.* Chapman and Hall, 3rd edition.

[Chen, 2004] Chen, F. (2004). Learning accurate and understandable rules from svm classifiers. Master's thesis, Simon Fraser University.

[Chen et al., 1998] Chen, S. S., Donoho, D. L., and Saunders, M. L. (1998). Atomic decomposition by basis pursuit. *SIAM Journal of Scientific Computing*, 20(1):33–61.

[Chow, 1970] Chow, C. K. (1970). On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, IT-16(1).

[Cohen, 1995] Cohen, W. (1995). Fast effective rule induction. In *Proccedings of the Twelfth International Conference on Machine Le arning*, pages 115–123, Tahoe City, CA. Morgan Kaufmann.

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. N. (1995). Support–vector networks. *Machine Learning Journal*, 20:273–297.

[Cover, 1991] Cover, T. (1991). *Elements of Information Theory.* Wiley.

[Craven and Shavlik, 1999] Craven, M. and Shavlik, J. (1999). Rule extraction: Where do we go from here? Technical Report 99-1, University of Wisconsin, Machine Learning Research Group.

[Craven and Shavlik, 1996] Craven, M. W. and Shavlik, J. W. (1996). Extracting tree-structured representations of trained networks. In *Advances in Neural Processing Systems*, volume 8.

[Davies and Gather, 1993] Davies, P. L. and Gather, U. (1993). The identification of multiple outliers. *J. Amer. Statist. Assoc.*, 88:782–792.

[Davies and Russell, 1994] Davies, S. and Russell, S. (1994). Np-completeness of searches for smallest possible feature sets. In *Proceedings of the 1994 AAAI Fall Symposium on Relevance*, pages 37–39. AAAI Press.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Ser. B*, 39:1–38.

[Domingos, 1999] Domingos, P. (1999). Metacost: A general method for making classifiers cost-sensitive. In *Knowledge Discovery and Data Mining*, pages 155–164.

[DuMouchel et al., 1999] DuMouchel, W., Volinsky, C., Johnson, T., Cortes, C., and Pregibon, D. (1999). Squashing flat files flatter. In *Proceedings of the 5th ACM Conference on Knowledge Discovery and Data Mining.*

[Fender, 2003] Fender, T. (2003). *Empirische Risikominimierung für dynamische Datenstrukturen.* PhD thesis, Department of Statistics, Univ. Dortmund.

[Ferguson, 1961] Ferguson, T. (1961). On the rejection of outliers. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 253–287.

[Ferri et al., 2004] Ferri, C., Flach, P., and Hernandez-Orallo, J. (2004). Delegating classifiers. In *Proceedings of the 21st International Conference on Machine Learning.*

[Fisher, 1936] Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188.

[Foster and Kesselman, 2004] Foster, I. and Kesselman, C., editors (2004). *The Grid: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann, 2nd edition.

[Freund and Schapire, 1996] Freund, Y. and Schapire, R. (1996). Game theory, on-line prediction and boosting. In *Proceedings of the 9th Annual Conference on Computational Learnin g Theory*, pages 325–332.

[Fürnkranz, 1999] Fürnkranz, J. (1999). Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54.

[Fürnkranz and Flach, 2005] Fürnkranz, J. and Flach, P. (2005). ROC 'n' Rule Learning – Towards a Better Understanding of Covering Algorithms. *Machine Learning*, 58(1):39–77.

[Gama and Brazdil, 2000] Gama, J. and Brazdil, P. (2000). Cascade generalization. *Machine Learning*, 41(3):315–343.

[Garczarek, 2002] Garczarek, U. (2002). *Classification Rules in Standardized Partition Spaces.* PhD thesis, Universität Dortmund.

[Gather, 1989] Gather, U. (1989). Testing for multisource contamination in location/scale families. *Comm. Statist. A*, 18:1–35.

[Gather and Becker, 1997] Gather, U. and Becker, C. (1997). Outlier identification and robust methods. In Maddala, G. and Rao, C., editors, *Handbook of Statistics*, volume 15, pages 123–143. Elsevier Science.

[Gather and Kale, 1992] Gather, U. and Kale, B. (1992). Outlier generating models – a review. In Venugopal, N., editor, *Contributions to Stochastics*, pages 57–85. Wiley.

[Giraud-Carrier, 1998] Giraud-Carrier, C. (1998). Beyond predictive accuracy: What? In *ECML'98 Workshop Notes - Upgrading Learning to the Meta-Level: Model Selection and Data Transformation*, pages 78–85. Technical University of Chemnitz.

[Goethals and Zaki, 2003] Goethals, B. and Zaki, M. J., editors (2003). *Frequent Itemset Mining Implementations.* CEUR-WS.

[Gondek and Hofmann, 2003] Gondek, D. and Hofmann, T. (2003). Conditional information bottleneck clustering. In *3rd IEEE International Conference on Data Mining, Workshop on Clustering Large Data Sets.*

[Gondek and Hofmann, 2004] Gondek, D. and Hofmann, T. (2004). Non-redundant data clustering. In *Proceedings ot the 4th IEEE International Conference on Data Mini ng.*

[Grünwald, 2005] Grünwald, P. D. (2005). Minimum description length tutorial. In Grünwald, P. D., Myung, J., and Pitt, M. A., editors, *Advances in Minimum Description Length: Theory and Applications*, chapter 2, pages 23–80. MIT Press.

[Guyon and Elisseeff, 2003] Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *JMLR Special Issue on Variable and Feature Selection*, 3:1157 – 1182.

[Guyon et al., 1996] Guyon, I., Matic, N., and Vapnik, V. (1996). Discovering informative patterns and data cleaning. In Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., editors, *Advances in Knowledge Discovery and Data Mining*, chapter 2, pages 181–204. AAAI Press/The MIT Press, Menlo Park, California.

[Hampel, 1974] Hampel, F. R. (1974). The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, pages 383–393.

[Hampel et al., 1986] Hampel, F. R., Ronchetti, E. M., Rousseeuw, P. J., and Stahel, W. A. (1986). *Robust Statistics.* Wiley.

[Hand, 2002] Hand, D. (2002). Pattern detection and discovery. In Hand, D., Adams, N., and Bolton, R., editors, *Pattern Detection and Discovery.* Springer.

[Hand et al., 2002] Hand, D., Adams, N., and Bolton, R., editors (2002). *Pattern Detection and Discovery.* Springer.

[Hand et al., 2001] Hand, D., Mannila, H., and Smyth, P. (2001). *Principles of Data Mining.* Massachusetts Institute of Technology Press, Cambridge, Massachusetts.

[Hastie et al., 2001] Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer series in statistics. Springer.

[Heilemann and Münch, 2001] Heilemann, U. and Münch, H. (2001). Classification of german business cycles using monthly data. Technical Report 8/2001, Universität Dortmund, SFB 475, Dortmund, Germany.

[Hodges, 1955] Hodges, J. (1955). A bivariate sign test. *Ann. Math. Stat.*, 26:523–527.

[Homburg et al., 2005] Homburg, H., Mierswa, I., Möller, B., Morik, K., and Wurst, M. (2005). A benchmark dataset for audio classification and clustering. In *Proc. of the International Symposium on Music Information Retrieval 2005.*

[Huber, 1964] Huber, P. J. (1964). Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35:73–101.

[Huber, 1973] Huber, P. J. (1973). Robust regression: Asymptotics, conjectures and monte carlo. *Annals of Statistics*, 1:799–821.

[Huber, 1981] Huber, P. J. (1981). *Robust Statistics.* John Wiley & Sons.

[Jennings et al., 1982] Jennings, D., Amabile, T., and Ros, L. (1982). Informal covariation assessments: Data-based versus theory-based judgements. In Kahnemann, D., Slovic, P., and Tversky, A., editors, *Judgement under uncertainty: Heuristics and biases*, pages 211 – 230. Cambridge University Press, Cambridge.

[Joachims, 1998] Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In Nédellec, C. and Rouveirol, C., editors, *Proceedings of the European Conference on Machine Learning*, pages 137 – 142, Berlin. Springer.

[Joachims, 1999] Joachims, T. (1999). Making large-scale SVM learning practical. In Schölkopf, B., Burges, C., and Smola, A., editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11. MIT Press, Cambridge, MA.

[Joachims, 2002] Joachims, T. (2002). *Learning to Classify Text using Support Vector Machines*, volume 668 of *Kluwer International Series in Engineering and Computer Science.* Kluwer.

[Kaufman and Rousseeuw, 1990] Kaufman, L. and Rousseeuw, P. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley and Sons, Inc.

[Keerthi et al., 2002] Keerthi, S. S., Duan, K., Shevade, S. K., and Poo, A. (2002). A fast dual algorithm for kernel logistic regression. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 299–306.

[Keim, 2002] Keim, D. A. (2002). Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 7(1):100–107.

[Kijsirikil and Sinthupinyo, 1999] Kijsirikil, B. and Sinthupinyo, S. (1999). Approximate ilp rules by backpropagation neural network: A result on thai character recognition. In Dzeroski, S. and Flach, P., editors, *ILP-99*, volume 1634 of *LNAI*, pages 162–173. Springer.

[Klinkenberg and Rüping, 2003] Klinkenberg, R. and Rüping, S. (2003). Concept drift and the importance of examples. In Franke, J., Nakhaeizadeh, G., and Renz, I., editors, *Text Mining – Theoretical Aspects and Applications*, pages 55–77. Physica-Verlag, Berlin, Germany.

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

[Kohavi and John, 1998] Kohavi, R. and John, G. H. (1998). The wrapper approach. In Liu, H. and Motoda, H., editors, *Feature Extraction, Construction, and Selection: A Data Mining Perspective*, pages 33–50. Kluwer.

[Kohonen et al., 1992] Kohonen, T., Kangas, J., Laaksonen, J., and Torkkola, K. (1992). LVQ PAK: A program package for the correct application of learning vector quantization algorithms. In *Proceedings of the International Joint Conference on Neural Networks*, pages 725–730.

[Kolmogorov, 1965] Kolmogorov, A. (1965). Three approaches to the quantitative definition of information. *Problems Inform. Transmission*, 1(1):1–7.

[Kwok and Tsang, 2004] Kwok, J. T. and Tsang, I. W. (2004). The pre-image problem in kernel methods. *IEEE Transactions on Neural Networks*, 15(6):1517–1525.

[Lavrac, 1998] Lavrac, N. (1998). Data mining in medicine: Selected techniques and applications. In *IDAMAP-98, Third Workshop on Intelligent Data Analysis in Medicine and Pharmacology*, pages 6–16, Brighton. an ECAI'98 workshop.

[Li and Vitanyi, 1997] Li, M. and Vitanyi, P. (1997). *An Introduction to Kolmogorov Complexity and its Applications*. Springer, 2nd edition edition.

[Liu and Motoda, 1998] Liu, H. and Motoda, H. (1998). *Feature Extraction, Construction, and Selection: A Data Mining Perspe ctive*. Kluwer.

[Liu and Motoda, 2001] Liu, H. and Motoda, H. (2001). *Instance Selection and Construction for Data Mining*. Kluwer Publishers.

[Liu et al., 1999] Liu, R. Y., Parelius, J. M., and Singh, K. (1999). Multivariate analysis by data depth: Descriptive statistics, graphics and inference. *The Annals of Statistics*, 27(3):783–858.

[Madigan et al., 2002] Madigan, D., Raghavan, N., DuMouchel, W., Nason, M., Posse, C., and Ridgeway, G. (2002). Likelihood-based data squashing. *Data Mining and Knowledge Discovery*, 6(2):173–190.

[Mahalanobis, 1936] Mahalanobis, P. (1936). On the generalized distance in statistics. *Proc. Nat. Acad. Sci. India*, 12:49–55.

[Mangasarian, 2000] Mangasarian, O. L. (2000). Generalized support vector machines. In Smola, A. J., Bartlett, P., Schökopf, B., and Schuurmans, D., editors, *Advances in Large Margin Classifiers*, pages 135–146. MIT Press.

[Mercer, 1909] Mercer, J. (1909). Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London, A*, 209:415–446.

[Merriam Webster, 2004] Merriam Webster (2004). Merriam-Webster Online Dictionary. http://www.merriam-webster.com.

[Michalski, 1969] Michalski, R. S. (1969). On the quasi-minimal solution of the general covering problem. In *Proceedings of the V International Symposium on Information Processing (FCIP 69)*, volume A3, pages 125–128, Bled, Yugoslavia.

[Mierswa and Morik, 2005] Mierswa, I. and Morik, K. (2005). Automatic feature extraction for classifying audio data. *Machine Learning Journal*, 58:127–149.

[Mika et al., 1998] Mika, S., Schölkopf, B., Smola, A. J., Müller, K.-R., Scholz, M., and Rätsch, G. (1998). Kernel pca and de-noising in feature spaces. In *Advances in Neural Information Processing Systems*, volume 11, pages 536–542.

[Miller, 1956] Miller, G. (1956). The magical number seven, plus or minus two: Some limits to our capacity for processing information. *Psychol Rev*, 63:81 – 97.

[Mitchell, 1997] Mitchell, T. M. (1997). *Machine Learning.* McGraw Hill, New York.

[Mladenic et al., 2004] Mladenic, D., Brank, J., Grobelnik, M., and Milic-Frayling, N. (2004). Feature selection using linear classifier weights: interaction with classification models. In *SIGIR 2004*, pages 234–.241.

[Morik, 2000] Morik, K. (2000). The Representation Race - Preprocessing for Handling Time Phenomena. In de Mántaras, R. L. and Plaza, E., editors, *Proceedings of the European Conference on Machine Learning 2000 (ECML 2000)*, volume 1810 of *Lecture Notes in Artificial Intelligence*, Berlin, Heidelberg, New York. Springer Verlag Berlin.

[Morik, 2002] Morik, K. (2002). Detecting interesting instances. In Hand, D. J., Adams, N. M., and Bolton, R. J., editors, *Proceedings of the ESF Exploratory Workshop on Pattern Detection and Discovery*, volume 2447 of *LNAI*, pages 13–23, Berlin. Springer Verlag.

[Morik et al., 2005] Morik, K., Boulicaut, J.-F., and Siebes, A., editors (2005). *Local Pattern Discovery.* Lecture Notes in Computer Science. Springer.

[Morik et al., 1999] Morik, K., Brockhausen, P., and Joachims, T. (1999). Combining statistical learning with a knowledge-based approach – A case study in intensive care monitoring. In *Proc. 16th Int'l Conf. on Machine Learning (ICML-99)*, Bled, Slowenien.

[Morik et al., 2002] Morik, K., Joachims, T., Imhoff, M., Brockhausen, P., and Rüping, S. (2002). Integrating kernel methods into a knowledge-based approach to evidence-based medicine. In Schmitt, M., Teodorescu, H.-N., Jain, A., Jain, A., Jain, S., and Jain, L. C., editors, *Computational Intelligence Processing in Medical Diagnosis*, volume 96 of *Studies in Fuzziness and Soft Computing*, pages 71–99. Physica-Verlag.

[Morik and Köpcke, 2004] Morik, K. and Köpcke, H. (2004). Analysing Customer Churn in Insurance Data - A Case S tudy. In Boulicaut, J.-F., Esposito, F., and an d Dino Pedreschi, F. G., editors, *Knowledge Discovery in Databases: PKDD 2004*, volume 3202 of *Lecture Notes in Computer Science*, pages 325–336. Springer.

[Morik and Muehlenbrock, 1999] Morik, K. and Muehlenbrock, M. (1999). Conceptual Change in the Explanations of Phenomena in Astronomy. In Kayser, D. and Vosniadou, S., editors, *Modelling Changes in Understanding*, Advances in learning and instruction series, chapter 5, pages 138–167. Pergamon.

[Morik and Rüping, 2002] Morik, K. and Rüping, S. (2002). A multistrategy approach to the classification of phases in business cycles. In Elomaa, T., Mannila, H., and Toivonen, H., editors, *Machine Learning: ECML 2002*, volume 2430 of *Lecture Notes in Artificial Intelligence*, pages 307–318, Berlin. Springer.

[Morik et al., 1993] Morik, K., Wrobel, S., Kietz, J.-U., and Emde, W. (1993). *Knowledge Acquisition and Machine Learning - Theory, Methods, and Applications.* Academic Press, London.

[Muggleton, 1992a] Muggleton, S. (1992a). Inductive logic programming. In Muggleton, S., editor, *Inductive Logic Programming.*, number 38 in The A.P.I.C. Series, chapter 1, pages 13–28. Academic Press, London [u.a.].

[Muggleton, 1992b] Muggleton, S. (1992b). *Inductive Logic Programming.* Number 38 in APIC series. Academic Press, London.

[Murphy and Aha, 1994] Murphy, P. M. and Aha, D. W. (1994). UCI repository of machine learning databases.

[Neiling and Lenz, 2004] Neiling, M. and Lenz, H. J. (2004). The german administrative record census - an object identification problem. *Journal of the German Statistical Society*, 88:259–277.

[Niculescu-Mizil and Caruana, 2005] Niculescu-Mizil, A. and Caruana, R. (2005). Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 625–632.

[Nunez et al., 2002] Nunez, H., Angulo, C., and Catala, A. (2002). Rule extraction from support vector machines. In *Proceedings of the European Symposium on Artificial Neural Networks*, pages 107–112, Bruges, Belgium.

[Nussbaum, 1989] Nussbaum, J. (1989). Classroom conceptual change: Philosophical perspectives. *International Journal of Science Education*, 11:530–540.

[Ortega et al., 2001] Ortega, J., Koppel, M., and Argamon, S. (2001). Arbitrating among competing classifiers using learned referees. *Knowledge and Information Systems*, 3:470–490.

[Platt, 1999] Platt, J. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In Smola, A., Bartlett, P., Schölkopf, B., and Schuurmans, D., editors, *Advances in Large Margin Classifiers.* MIT Press.

[Polasek, 1997] Polasek, W. (1997). *Schliessende Statistik.* Springer.

[Popescul et al., 2002] Popescul, A., Ungar, L. H., Lawrence, S., and Pennock, D. M. (2002). Towards structural logistic regression: Combining relational and statistical learning. In *Proceedings of the Workshop on Multi-Relational Data Mining (MRDM-2002) at KDD-2002*, pages 130–141, Edmonton, Canada.

[Punch et al., 1993] Punch, W., Goodman, E., Hovland, P., and Enbody, R. (1993). Further research on feature selection and classification using genetic algorithms. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 557–564, Palo Alto, CA, USA. Morgan Kaufman.

[Pyle, 1999] Pyle, D. (1999). *Data Preparation for Data Mining.* Morgan Kaufmann Publishers.

[Quinlan, 1993] Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning.* Machine Learning. Morgan Kaufmann, San Mateo, CA.

[Quinlan, 1986] Quinlan, R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

[Rakotomamonjy, 2003] Rakotomamonjy, A. (2003). Variable selection using svm-based criteria. *JMLR Special Issue on Variable and Feature Selection*, 3:1357–1370.

[Rissanen, 1978] Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14:465 – 471.

[Rissanen, 1983] Rissanen, J. (1983). A universal prior for integers and estimation by minimum description length. *Annals of Statistics*, 11:416–431.

[Rissanen, 1984] Rissanen, J. (1984). Universal coding, information, prediction and estimation. *IEEE Transactions on Information Theory*, 30:629–636.

[Rissanen, 1996] Rissanen, J. (1996). Fisher information and stochastic complexity. *IEEE Transactions on Information Theory*, 42(1):40–47.

[Rousseeuw and Van Driessen, 1999] Rousseeuw, P. and Van Driessen, K. (1999). A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41:212–223.

[Rousseeuw, 1984] Rousseeuw, P. J. (1984). Least median of squares regression. *J. Am. Stat. Assoc.*, 79:871–880.

[Rousseeuw and Leroy, 1987] Rousseeuw, P. J. and Leroy, A. M. (1987). *Robust Regression and Outlier Detection*. Wiley.

[Roweis and Saul, 2000] Roweis, S. and Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

[Rüping, 1999] Rüping, S. (1999). Zeitreihenprognose für Warenwirtschaftssysteme unter Berücksichtigung asymmetrischer Kostenfunktionen. Master's thesis, Universität Dortmund. In German.

[Rüping, 2001a] Rüping, S. (2001a). Incremental learning with support vector machines. In Cercone, N., Lin, T., and Wu, X., editors, *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 641–642. IEEE.

[Rüping, 2001b] Rüping, S. (2001b). Svm kernels for time series analysis. In Klinkenberg, R., Rüping, S., Fick, A., Henze, N., Herzog, C., Molitor, R., and Schröder, O., editors, *LLWA 01 - Tagungsband der GI-Workshop-Woche Lernen - Lehren - Wissen - Adaptivität*, Forschungsberichte des Fachbereichs Informatik der Universität Dortmund, pages 43–50, Dortmund, Germany.

[Rüping, 2002a] Rüping, S. (2002a). Efficient kernel calculation for multirelational data. In Kokai, G. and Zeidler, J., editors, *Proceedings der FGML 2002*, pages 121–126, Learning Lab Lower Saxony, Hannover, Germany.

[Rüping, 2002b] Rüping, S. (2002b). Support vector machines in relational databases. In Lee, S.-W. and Verri, A., editors, *Pattern Recognition with Support Vector Machines — First International Workshop, SVM 2002*, LNCS 2388, pages 310–320. Springer.

[Rüping, 2005] Rüping, S. (2005). Robust clustering for nominal and continuous data. Talk at 29th Annual Conference of the German Classification Society.

[Rüping and Morik, 2003] Rüping, S. and Morik, K. (2003). Support vector machines and learning about time. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'03).*

[Rüping and Scheffer, 2005] Rüping, S. and Scheffer, T., editors (2005). *Proceedings of the ICML Workshop on Learning With Multiple Views.*

[Schlittgen and Streitberg, 2001] Schlittgen, R. and Streitberg, B. H. J. (2001). *Zeitreihenanalyse.* Oldenburg, 9. edition.

[Schölkopf et al., 1999] Schölkopf, B., Mika, S., Burges, C. J. C., Knirsch, P., Müller, K.-R., Rätsch, G., and Smola, A. J. (1999). Input space versus feature space in kernel-based methods. *IEEE Transactions On Neural Networks*, 10(5):1000–1017.

[Schölkopf et al., 1999] Schölkopf, B., Smola, A., and Müller, K.-R. (1999). Kernel principal component analysis. In Schölkopf, B., Burges, C., and Smola, A., editors, *Advances in Kernel Methods - Support Vector Learning*, pages 69–88. MIT Press.

[Schölkopf and Smola, 2002] Schölkopf, B. and Smola, A. J. (2002). *Learning with Kernels – Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press.

[Schölkopf et al., 1998a] Schölkopf, B., Smola, A. J., Knirsch, P., and Burges, C. (1998a). Fast approximation of support vector kernel expansions, and an interpretation of clustering as approximation in feature spaces. In *DAGM-Symposium*, pages 125–132.

[Schölkopf et al., 1998b] Schölkopf, B., Smola, A. J., and Müller, K.-R. (1998b). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319.

[Scholz, 2002] Scholz, M. (2002). Using real world data for modeling a protocol for ICU monitoring. In Lucas, P., Asker, L., and Miksch, S., editors, *Working notes of the IDAMAP 2002 workshop*, pages 85–90.

[Scholz, 2005] Scholz, M. (2005). Knowledge-Based Sampling for Subgroup Discovery. In Morik, K., Boulicaut, J.-F., and Siebes, A., editors, *Local Pattern Detection*, volume LNAI 3539 of *Lecture Notes in Artificial Intelligence*, pages 171–189. Springer.

[Schwartz, 1979] Schwartz, G. (1979). Estimating the dimension of a model. *Annals of Statistics*, 6(461–464).

[Scott, 1992] Scott, D. (1992). *Multivariate Density Estimation: Theory, Practice, and Visualization.* Wiley.

[Seewald and Fürnkranz, 2001] Seewald, A. and Fürnkranz, J. (2001). An evaluation of grading classifiers. In *Advances in Intelligent Data Analysis (IDA-01)*, pages 115–124.

[Shtarkov, 1987] Shtarkov, Y. M. (1987). Universal sequential coding of single messages. *Problems of Information Transmission*, 23(3):3–17.

[Sieling, 2003] Sieling, D. (2003). Minimization of decision trees is hard to approximate. In *Proceedings ot the 18th Annual IEEE Conference on Computational Complexity*, page 84.

[Smyth et al., 1995] Smyth, P., Gray, A., and Fayyad, U. M. (1995). Retrofitting decision tree classifiers using kernel density estimation. In *International Conference on Machine Learning*, pages 506–514.

[Sommer, 1996] Sommer, E. (1996). *Theory Restructering: A Perspective on Design & Maintenance of Kno wledge Based Systems.* PhD thesis, University of Dortmund.

[Tishby et al., 1999] Tishby, N., Pereira, F., and Bialek, W. (1999). The information bottleneck method. In *Proceedings of the 37-th Annual Allerton Conference on Communicat ion, Control and Computing*, pages 368–377.

[Todorovski and Dzeroski, 1999] Todorovski, L. and Dzeroski, S. (1999). Experiments in meta-level learning with ILP. In Zytkow, J. M. and Rauch, J., editors, *Proceedings of third European Conference on Principles of data mi ning and knowledge discovery (PKDD-99)*, volume 1704, pages 98–106. Springer.

[Todorovski and Dzeroski, 2000] Todorovski, L. and Dzeroski, S. (2000). Combining multiple models with meta decision trees. In *Proceedings of the Fourth European Conference on Principles of Da ta Mining and Knowledge Discovery*, pages 54–64. Springer.

[Tukey, 1975] Tukey, J. (1975). Mathematics and picturing data. In *Proceedings of the 1975 International Congress of Mathematics*, volume 2, pages 523–531.

[Tukey, 1958] Tukey, J. W. (1958). Bias and confidence in not-quite large samples (abstract). *Ann. Math. Statist.*, 29:614.

[Tukey, 1977] Tukey, J. W. (1977). *Exploratory Data Analysis.* Addison-Wesley.

[Tumer and Ghosh, 1995] Tumer, K. and Ghosh, J. (1995). Order statistics combiners for neural classifiers. In *Proceedings of the World Congress on Neural Networks.*

[Vapnik, 1998] Vapnik, V. (1998). *Statistical Learning Theory.* Wiley, Chichester, GB.

[Vapnik and Chapelle, 2000] Vapnik, V. and Chapelle, O. (2000). Bounds on error expectation for support vector machines. *Neural Computation*, 12.

[Vosniadou and Brewer, 1992] Vosniadou, S. and Brewer, W. (1992). Mental models of the earth: A study of conceptual change in childhood. *Cognitive Psychology*, 24:535–585.

[Wahba, 1999] Wahba, G. (1999). Support vector machines, reproducing kernel hilbert spaces and the randomized GACV. In Schölkopf, B., Burges, C., and Smola, A., editors, *Advances in Kernel Methods - Support Vector Learning*, pages 69–88. MIT Press.

[Wapnik and Tscherwonenkis, 1979] Wapnik, W. and Tscherwonenkis, A. (1979). *Theorie der Zeichenerkennung.* Akademie Verlag, Berlin.

[Weston et al., 2000] Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T., and Vapnik, V. (2000). Feature selection for SVMs. In *Advances in Neural Information Processing Systems*, volume 13.

[Winter Corporation, 2005] Winter Corporation (2005). 2005 winter top ten program. http://www.wintercorp.com/VLDB/2005_TopTen_Survey/TopTenWinners_2005.asp.

[Witten and Frank, 2000] Witten, I. and Frank, E. (2000). *Data Mining – Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann.

[Wolpert, 1992] Wolpert, D. (1992). Stacked generalizations. *Neural Networks*, 5:241–259.

[Wolpert and Macready, 1995] Wolpert, D. and Macready, W. (1995). No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fé Institute, Santa Fé, CA.

[Wolpert and Macready, 1997] Wolpert, D. and Macready, W. (1997). No free lunch theorems for optimisation. *IEEE Trans. on Evolutionary Computation*, 1:67 – 82.

[Yang and Honovar, 1998] Yang, J. and Honovar, V. (1998). Feature subset selection using a genetic algorithm. In Liu, H. and Motoda, H., editors, *Feature Extraction, Construction, and Selection – A Data Mining Perpective.* Kluwer.

[Zadrozny and Elkan, 2002] Zadrozny, B. and Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 694–699.

[Zelic et al., 1997] Zelic, I., Kononenko, I., Lavrac, N., and Vuga, V. (1997). Induction of decision trees and bayesian classification applied to diagnosis of sport injuries. *Journal of Medical Systems*, 21(6):429–444.

Index

Eigenanteil an in Kooperation erzielten Ergebnissen

Die folgenden in Kooperation entstandenen Veröffentlichungen sind in diese Dissertation eingeflossen:

- [Morik et al., 2002]: Der vom Autor dieser Dissertation entwickelte Anteil besteht aus der Evaluierung des Entscheidungsfunktion der SVM als Konfidenzmaß für die Klassifikation.
- [Morik and Rüping, 2002]: Der vom Autor dieser Dissertation entwickelte Anteil besteht aus der Diskretisierung numerischer Werte für das benutzte logische Lernverfahren.
- [Klinkenberg and Rüping, 2003]: Der vom Autor dieser Dissertation entwickelte Anteil besteht aus der Integration von Beispielgewichten in einen SVM-Klassifikator.
- [Rüping and Morik, 2003]: Dieser Artikel ist ein reiner Übersichtsartikel ohne neue Ergebnisse.