Learning with Local Models

Stefan Rüping[1]

University of Dortmund, 44221 Dortmund, Germany,
rueping@ls8.cs.uni-dortmund.de,
WWW home page: http://www-ai.cs.uni-dortmund.de/

Abstract. Next to prediction accuracy, the interpretability of models is one of the fundamental criteria for machine learning algorithms. While high accuracy learners have intensively been explored, interpretability still poses a difficult problem, largely because it can hardly be formalized in a general way. To circumvent this problem, one can often find a model in a hypothesis space that the user regards as understandable or minimize a user-defined measure of complexity, such that the obtained model describes the essential part of the data. To find interesting parts of the data, unsupervised learning has defined the task of detecting local patterns and subgroup discovery. In this paper, the problem of detecting local classification models is formalized. A multi-classifier algorithm is presented that finds a global model that essentially describes the data, can be used with almost any kind of base learner and still provides an interpretable combined model.

1 Introduction

It is commonplace knowledge that more and more data is collected everywhere and that the size of data sets available for knowledge discovery is increasing steadily. On the one hand this 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 a standard learning algorithm to all the available examples. While the accuracy of the learner typically increases with example size, other criteria are negatively affected by too much examples. This paper will deal with the criterion of interpretability of the learned model, which is an important, yet often overlooked aspect for applying machine learning algorithms to real-world tasks.

The key problem with interpretability is that humans are very limited in the level of complexity they can intuitively understand [9]. 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. What can we do about this 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. Hence, we can try to find a simple model first and then concentrate on improving only those parts of the input

space, where the model is not good enough. This will be an easier task because less examples have to be considered and hence one might use a more sophisticated learner. In other words, one constructs not one single global model for all the data, but a global model plus one or more local models to cover special cases. Also, 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 are surprising [7], i. e. if they contradict what is already known. Hence, it may also be the case that the local models actually contains the interesting cases.

To avoid problems with finding general measures of complexity for arbitrary models, this paper takes a very hands-on approach to interpretability: The user is allowed to use an arbitrary global learner, such that he can pick the learner whose results are the most understandable to him. He is also allowed to use an arbitrary clustering algorithm, which will be used to identify where to use the global algorithm and where not. So he can use the clusterer which will give him the most understandable description, where his global algorithm is right. To improve accuracy, a high-performance local learner is used to predict the rest of the examples, as long as the deviation from the global model remains below a user-defined threshold.

Summing up, the goal of this paper is to develop an hierachical, multi-classifier algorithm for learning local models which

1. learns a global model that sums up the essential properties of the data
2. can use (almost) every kind of learner to find the global and local base models
3. reduces a given criterion of model complexity for the global model.

The rest of the paper is organized as follows: The next section will discuss a broader picture of local models, complexity and interpretability. Readers that are only interested in the proposed local model algorithm may skip this section. Section 3 will review some learning algorithms which will be needed as parts of our hierachical approach. The new local model algorithm will be presented in Section 4, while Section 5 will discuss related approaches. Experimental results are given in Section 6 and Section 7 gives some conclusions.

2 Local Models

There are two aspects to local models: structure and performance. The structural aspect refers to the case where the optimal model of the data is composed of several parts that have a meaningful interpretation in terms of the application. This may be the case because the structure behind the data can only be expressed in a combination of hypothesis spaces of several standard learners (e. g. a combination of numerical and logical rules) or because there are limits in terms of interpretability. The performance aspect of local models refers to the case that a good model can theoretically be found by a single learner on the whole example set, but it is more efficient to use separate learning runs. Here, the distinction between global and local models is only meaningful in terms of

algorithmic complexity and time and space requirements, but not in any intrinsic way.

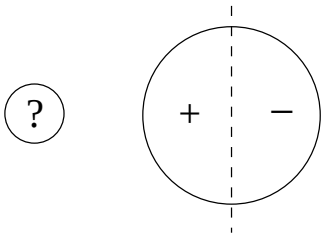

Fig. 1. Relationship between local patterns and local models. The left circle is a local pattern, but only a local model if its class is negative.

There is an obvious connection between local models and the detection of local patterns [8]. Local pattern or subgroup detection is defined as the unsupervised detection of high-density regions in the data. There are several way to formalize local patterns. In the scope of this paper, local patterns are defined as follows:

Definition: Given an input space X and some default probability measure $P_{Default}(X)$, a local pattern is a subset X' of X such that the empirical probability[1] of X' is significantly different from $P_{Default}(X')$.

The idea is that the user already has some idea of what his data looks like (for example he assumes that all items in a store are bought independently of one another) and he his interested in cases where his beliefs are wrong (for example an association rule describing which items are often bought together). In contrast, local models are meant to improve the classification, hence the interesting quantity is the class probability $P(y|x)$.

Definition: Given an input space $X \times Y$ and a default (or global) class probability measure $P_{Default}(Y|X)$, a local model is a subset X' of X such that the empirical probability $P_{emp}(Y|X')$ is significantly different from $P_{Default}(Y|X')$, plus a classification rule with input space X'.

Here, the user will not be bothered with deviations from the data to his beliefs, as long as this does not have an influence on the attribute of interest y. The difference can be seen in Figure 1. In a local pattern task, the default probability could be a Gaussian approximating the large batch of data in the circle on the right and the corresponding local model would be a smaller Gaussian describing the circle on the left. In a local classification task, an obvious default model

[1] The empirical probability of X' is the frequency of X' observed in the data, in contrast to the expected frequency as defined by $P(X')$.

would be the vertical line through the large circle, classifying all points to its right as negative and the points to its left as positive. In contrast to the local pattern case, the smaller circle will only be a local model if its class is negative, i. e. different from the one predicted by the default model. Summing up, the difference between both tasks is that in local model detection the goal is not to just find regions with unsuspected density, but to identify those which can be used to improve the overall performance, e. g. the classification accuracy.

Given these definitions it is obvious that the local model task can be solved by detecting local patterns in the misclassified examples of the default classification rule. This approach learns a very simple classification model: predict the negative of the default rule. Its disadvantage is, that the simplicity of the classifier may lead to the problem of learning very complex patterns. So the complexity of the classification is not removed, but only transferred to the decision, where to apply the classifier, i. e. the local patterns. As the classification problem is usually much better investigated in machine learning than the local pattern problem, it will usually be better to define a more simple pattern together with a more complex decision function, see Figure 2. We will take this approach in this paper.

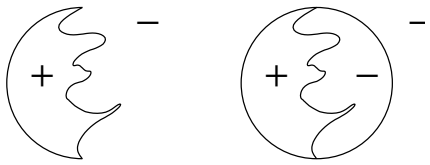

Fig. 2. Given the surrounding data is negative, the positive data can be marked in two ways: a complex cluster plus a trivial classifier (default positive, right) versus a trivial cluster (circle) plus a complex classifier (twisted line, right).

From the perspective of interpretability, the use of simple local patterns together with a more complex local classifier allows to better separate the local and global models, because it minimizes the interactions between both models. It allows to interpret both models on its own without the need to take interactions into account, which account for a large part of the complexity of the overall classifier.

Talking about interpretable models, one has to acknowledge that the concept of interpretability is very hard to formalize. Interpretability of models is often evaluated by interviewing a human expert, but reliable information is hard to come by, as this would need a survey of several independent human experts. Findings from psychology show that the size of the models plays some part, as humans are usually only able to deal with about seven cognitive units at the same time [9]. This motivates certain measures of model complexity like the number or length of rules in logical models [17], which are traditionally assumed to be more interpretable than other approaches, as logical rules can be easily cast into

sentences in natural language. However, experience shows that the experience of humans with interpreting the kind of model or its visualisation plays a crucial role. Plainly, humans tend to find the things the most interpretable, they are already acquainted with. Concludingly, this paper takes a very pragmatic approach to the question of interpretability: let the user choose whatever learning algorithm he likes best and optimize a user-given criterion of model complexity.

3 Learning Algorithms

In this section we will shortly introduce the learning algorithms that were used in the experiments in this paper. A detailed discussion of these algorithms is beyond the scope of this paper, we will restrict ourselves to those properties that are of interest for the course of this paper. The learning tasks to solve are both density estimation and probabilistic classification.

3.1 Density Estimation

Several algorithms exists for the approximation of a density $P(x)$ from examples (x_i). We limit this discussion to the approximation by a Gaussian as the prototypical parametric approach and probably easiest density estimation technique.

To approximate the data by a Gaussian distribution one has to notice that the multivariate Gaussian is completely determined by its mean and covariance matrix. Hence, it suffices to calculate the mean and covariance matrix of the data and substitute it into the Gaussian distribution. This is a very simple algorithm, but also a very unflexible, as it is limited to a very special parametric form. However, it turns out that this is sufficient for the local model algorithm of this paper, as we do not need to approximate the density excactly, but only need the densities to separate different local regions in the input space in order to split the classification model up into less complex parts. This approach is similar to the well-known k-means clustering algorithm, where one is not interested in the form and density of each cluster itself, but only in the border line between each cluster and the rest.

What is important in the course of this paper is the robustness of the density estimation. Robustness means that if a certain fraction of examples is not drawn from the distribution of interest, this should have little influence on the estimated density. Standard estimates of mean and covariance can be heavily distorted by far away outliers and hence the trivial Gaussian approximation is not robust. In this paper, we use the Minimum Covariance Determinant estimator [14], which searches for a subset of the examples of given size such that the determinant of its covariance matrix becomes minimal and hence, the examples are located very close together. The mean and covariance of this subset are then again used to define a Gaussian density.

3.2 Probabilistic Classification

The goal of probabilistic classification is to not only find a classifier, but also an estimation of the conditional class probability $P(y|x)$. A general, but very coarse solution is to use the classifiers accuracy or its precision for the positive and negative class, respectively.

Numerical classifiers, i. e. classifiers of the form $cl(x) = sign(f(x))$, can be transformed into probabilistic classifiers by finding an appropriate scaling function σ such that $P(Y = 1|x) = \sigma(f(x))$. This scaling function can be found by minimizing the cross-entropy

$$CRE = \sum_{i:y_i=1} \log(\sigma(f(x))) + \sum_{i:y_i=-1} \log(1 - \sigma(f(x)))$$

or the mean squared error

$$MSE = \frac{1}{n} \sum_i (\frac{1 + y_i}{2} - \sigma(f(x)))^2$$

over the space of applicable scaling functions.

In this paper Support Vector Machines [21] are used as numerical classifiers. For Support Vector Machines, appropriate scaling procedures have been investigated in [11, 15]. A generic scaling function for a large variety of learners has also been proposed in [6]. Other classifiers can directly give an estimate of the conditional class probability, for example decision trees [13] where the class distribution at each leaf or its laplacian correction can be used as an estimator of the class probability of an observation classified by this leaf.

3.3 Interpretable Learners

One of the goals of this paper is the interpretability of the classifier. As was already pointed out in the introduction, the user may in principle prefer any kind of classifier. In the following, we limit the discussion to decision trees as interpretable learners, because they have some favourable properties with respect to interpretability: First, decision trees can be easily visualised, because they consist of a simple tree structure and simple tests. They can alse be transformed into a set of independent rules, which the user can investigate one after the other. Secondly, there are two simple measures of model complexity, namely the depth of the tree and its number of nodes. It is obvious that a decision tree is the more understandable, the less tests have to be made in order to classify an observation and the less tests it contains in general. This allows us to quantify the degree of complexity reduction. Thirdly, these measures of complexity can directly be optimized at the construction of the tree by cutting the tree off at the maximal allowed depth and continuing with the pruning phase of the tree induction until the maximum number of nodes is met.

3.4 Expectation Maximization

The Expectation Maximization (EM) algorithm [2] is a general technique for computing the maximum likelihood estimates of the parameters of a distribution in the presence of hidden variables. This approach assumes that in addition to the observed data X, there are hidden variables Z, such that the observations (x_i) could be modeled much better if the (z_i) were known. The EM algorithm involves two steps, the expectation step and the maximization step. The E-step computes the values of the hidden variable z_i, or a sufficient description thereof, given the current estimate of the parameters. The M-step computes the parameters as the maximum likelihood estimation given the observed data and the current estimate of the hidden variables. Both steps are iterated until convergences or a sufficient number of times. It can be shown that the EM algorithm converges to a local optimum under some very general assumptions. The well-known k-means clustering algorithm is a famous application of the expectation maximization algorithm.

4 Algorithm

Following the earlier discussion on complexity and interpretability, we assume the following problem setting: We are given data $(x_i, y_i), i = 1 \dots n$, two arbitrary learners $\mathcal{L}_{Glob}$ and $\mathcal{L}_{Loc}$, a density estimation algorithm and a real number $\tau \in]0, 1[$. We want to find a model that is a combination of a global model learned from $\mathcal{L}_{Glob}$ and several local models learned from $\mathcal{L}_{Loc}$, such that the combined model differs from the global model by at most τ. More formally, we define dummy variable $z_i, i \in 1 \dots k$ for the single base models, and let the combined model take the form

$$P(Y = 1|x) = \frac{1}{P(x)} \sum_{j=1}^{k} P(Y = 1|x, z = j)P(x|z = j)P(z = j),$$

where

$$P(x) = \sum_{j=1}^{k} P(x|z = j)P(z = j)$$

with $P(z = 1) > 1 - \tau$. We replaced the classifiers $y = f(x)$ by estimators of the conditional class probability $P(Y = 1|x)$, which can be done by one of the algorithms described in Section 3.2. Hence, by inspecting $P(Y = 1|x, z = 1)$ (the global model), the user learns how the combined model behaves on a fraction of $P(z = 1) > 1-\tau$ of the cases, while inspecting the $P(x|z = j)$ tell him where the global model is applied and where not. Note that although at each point x the combined prediction is a linear combination of the base models predictions, the combined model is a nonlinear combination, as the $P(x|z = j)$ are not linear.

The local model problem as it is defined here is a most general black-box setting, as we neither assume knowledge about the internals of the learning and

density estimation algorithms, nor about the interpretation of its models. In particular, we do not assume that it is possible to assign weights to example, modify the learners parameters or deduce the influence of specific training examples to the learners model. The only possibility for the overall algorithm to interact with the given learners is to present the learners different training sets. For the outputs, we only assume that the classifiers return a real value and that the higher this value is, the more certain the classifier is that the observation belongs to the positive class (in particular, this includes the case of a simple binary classifier $f(x) \in \{-1, +1\}$). It was shown by Garczarek [6] that this is sufficient to convert the classifiers output into an estimate of the conditional class probability $P(Y = 1|x)$.

Equation 4 contains two special cases for learning interpretable models as extremes: For $P(z = 1) = 1$, we use only the global learner. This gives the user full control over what the learner does, but only in a few lucky cases the most understandable and the most accurate model will coincide. For $P(z = 1) = 0$, one may use the most accurate model. The user may still inspect a model from the global learner to understand the data, but he has no control over in which cases the interpretable and the accurate will disagree, as he is missing the explicite model $P(x|z = i)$ of global and local regions.

Of course, there are many other methods for combining several learners and one might ask, whether explicitely learning density models in the mixture approach as described in Equation 4 is not a more complex task than necessary. To justify the mixture idea, let us compare it with two simpler approaches: First, in Section 3.2 we described the idea of probabilitically scaling a classifier to obtain an estimate of the conditional class probability $P(y|x)$. Now, one might think of using the most confident classifier for each example or to combine the confidences of each classifier. But this idea is fallacious, because in general each classifier can only be trusted to give good probability estimates over the region of the input space it was trained and scaled on. For example, a linear classifier will be the more confident the further away from the decision line the observations lie, regardless of the position of its training examples. Hence, a local classifier may either give much too optimistic results on non-local examples (when scaled over the local examples only) or too pessimistic results on the local examples (when scaled over many examples it was not trained on).

A second seemingly easier approach to learning local models would be to use the local learner to predict whether the global learner is right or not. But this, too, is not a good approach, because it makes the local learning problem much more complex. The learner has not only to find structures in the data but also the structure superimposed by the global model. Besides from complicating the learning problem, this effectively prohibits to understand the overall prediction by looking at the global model, as any prediction from the global model might be negated by the second model. Hence, in this paper we use a combination of models were each learner is trained to directly predict the true class.

4.1 Learning the Combined Model

To find the combined model, we borrow ideas from two well-known learning algorithms: classification with covering algorithms and EM clustering. Covering algorithm find a logical rule which covers a part of the data, remove the examples covered by this rule and then iteratively find further rules to cover the rest of the data. Because of their iterative nature, they are well suited for the task of finding local models, when one views the first rule as the global rule and the following rules as local models Learning by covering makes explicit use of the fact that logical rules (e. g. "$X1 = a \wedge X2 \leq b \Rightarrow Y = 1$") make predictions for only a part of the input space, such that it is clear which rule can be applied. But this means that this idea cannot be directly applied to other base learners, as in general, learners may make a prediction for every observation in the input space. This is the reason why we need an additional density estimator to select which model to use.

To find both the models and the clusters, on which each model should be applied, we may proceed similar to EM clustering. EM clustering algorithms iteratively find a cluster model, then for each example estimate the probability of belonging to each of the clusters and re-estimate the clusters using these probabilities as weights (i. e. an example with low probability of belonging to some cluster will have little influence on the shape of this cluster). One might be tempted to solve the local model problem by directly using clustering as a pre-processing step, perhaps with a minimum size constraint on the first cluster, and then find a different local model for each cluster. But this strategy may fail, as clustering groups observations according to some similarity measure in X, while we are interested in grouping observations together if they can be predicted by the same model. If observations look very different in the input space, but can be correctly predicted by the same simple model, there is no reason to treat them differently as far as classification is concerned. This is the same situation as described in Figure 1.

The reason for using an EM-like approach instead of a greedy approach as in covering algorithms is that even if there is a simple structure behind parts of the data that one learner could find, the learners model may be distorted by outliers from the rest of the data. This problem can be seen in Figure 3 in Section 6, where a linear hyperplane from a Support Vector Machine is distorted by a small set of far away outliers. Once the local examples are known as such, it is very easy to improve the model by removing this examples from the training set of the classifier. Hence, it is important to re-evaluate the models once more information about clusters and outliers is present. The Expectation Maximization procedure allows to optimize both models in parallel by finding an optimal allocation from examples to learners.

As we are interested in predicting x, i. e. in the conditional class distribution $P(y|x)$, in the final application of our model only x, but not z, is known, such that we can make use of the mixture decomposition 4 of $P(Y = 1|x)$ only if z can be identified from x alone. Hence, we need the following assumption:

Assumption: The distributions $P(x|z=j)$ differ significantly for two different j_1, j_2.

We will not define formally, what a significant difference of two probability distributions is, as this is not crucial for the algorithm presented here. What is important is the intuition that the decomposition in k distributions will only be of benefit if the distributions live in different part of the input space X. Accordingly, in this paper the terms *j-th probability distribution* $P_j(x)$, *j-th cluster* and *j-th batch of data* are used interchangeably. It is easy to check whether this assumption holds by computing

$$P(Z=j|x) = \frac{1}{P(X)} \sum_i P(x|Z=j)P(Z=j)$$

for each x in the trainig set. If the assumption holds, the distribution of $P(Z=j|x)$ should follow an U-form, i. e. most examples should either clearly belong to the j-th cluster ($P(Z=j|x) \approx 1$) or clearly belong to a different cluster ($P(Z=j|x) \approx 0$).

In a similar way to $P(z|x)$, we can also compute $P(z|x,y)$ as

$$P(z|x,y) = \frac{P(x,y,z)}{P(x,y)} = \frac{P(x,y,z)}{\sum_z P(x,y,z)}$$

with

$$P(x,y,z) = P(y|x,z)P(x|z)P(z).$$

Note that here we need to transform the conditional class estimate $P(Y=1|x,z)$ given by the learner into the probability $P(y|x,z)$ that the learner predicts the correct class. Following the update of $P(z|x,y)$, the default probability $P(z)$ can be re-estimated as $P(z) = avg_i P(z_i|x_i,y_i)$.

Theoretically, we could now follow the algorithm for EM clustering algorithm by using $P(z|x,y)$ to assign examples to clusters and then use the learner and density estimator on each cluster to learn an update of $P(y|x,z)$ and $P(x|z)$. But it turns out that we need another intermediate step. The problem is the allocation of examples to batches in the presence of noise. Imagine the case when a large part of the error is due to random noise in y, independent of x, e. g. by independently flipping any label with a fixed, small probability. When we have an approximately correct global model, every non-flipped example will have a high $P(z|x,y)$, while every flipped example will have a very small probability belonging the global model, as the model can be quite certain that the example belongs to its cluster (high $P(x)$) and can be quite certain that its prediction is correct on the average (high $P(y|x)$), but the prediction indeed is wrong (flipped label). As a result, the examples with the lowest probability $P(z|x,y)$ will most likely be the errors of the global model and hence, the best local classifier to learn from these examples is the negative of the original classifier. Now one would need to know if the first classifier is wrong beforehand, because by the independence of $P(noise)$ and $P(x)$, the observation x is completely uninformative to the correctness of the first classifier. This is of course totally useless.

To remedy this problem we double the EM-step of the algorithm: in the first step, we allocate examples to batches with respect to $P(z|x, y)$ (E-step 1) and then learn a cluster model $P_j(x) = P(x|z)$ only (M-step 1). In the second step, we allocate examples to batches with respect to the learned $P(z|x)$ (E-step 2) and conclude with learning the classifier $P_j(x) = P(y|x, z)$ from the new batches (M-step 2). The trick is that now in the first M-step the density estimator only sees the examples that this model can predict better than any other, while in the second M-step the classifier does see all the examples it will be asked to predict later. That is, the first step is to find a local pattern (deviation from the combination of the rest of the models), and the second step is to learn a model for exactly this pattern.

In each E-step, we take care to partition the examples into disjunct batches for each learner by first choosing the examples for the batches with higher $p(z)$ and then choosing the examples for batches with lower $p(z)$ from the rest. This makes sure that different batches do not learn redundant models.

Alternatively, one could also give each learner the complete example set plus weights based on $P(z|x)$ or $P(z|x, y)$, as in standard k-means. We do not use this approach here, because we want to be able to 'plug-in' as many different learners as possible and there exist many types of learners that cannot deal with example weights. A possible alternative to the approach defined here is to not partition the example set but allow the batches to overlap themselves to a certain degree, in order to account for undecisive batch probabilities.

4.2 Finally, the Algorithm

The final algorithm looks like this:

```
Local Model Algorithm:
1  input: data (xi,yi), #clusters k, #iterations i,
          threshold tau
2  independently learn k models Pj(X,Y)
3  repeat i times
4      for all j estimate P(zi=j|xi,yi) from Pj
5      adjust P(zi=j|xi,yi) such that P(z=1) >= 1-tau
6      assign observations (xi) to batch j = argmax(P(zi=j|xi,yi))
7      for all j estimate P(zi=j|xi) from Pj
8      adjust P(zi=j|xi) such that P(z=1) >= 1-tau
9      assign examples (xi,yi) to batch j = argmax(P(zi=j|xi))
10     for all j learn model Pj(x,y) from batch j
```

If $P(z = 1) < \tau$ in step 5 or 8, we set $P(z = 1) = \tau$ and adjust all other $P(z = j)$ linearly.

As we do not use weights for the examples, but a hard threshold to decide whether or not to include an example in a training set, we cannot give a convergence result as in standard k-means or other EM algorithms (in k-means, the models are continuous in the weights of the examples, but of course there is

no continuity in including or excluding an example). Hence, the final model is chosen as the model with minimal training error.

5 Related Work

There exists several approaches for combining multiple classifiers, for example Voting, combination by order statistics [20], Meta-Level Learning [1], Stacking [22], Cascade Generalization [5] and Boosting [3]. In Boosting, the combined classifier is a linear combination of the base classifiers. The single classifiers and their weights are learned iteratively. In each step, explicit information about the error of the combined classifier so far is used and the classifier is added, that reduces the error of the combined classifier the most in terms of a certain loss function [4]. This idea is implemented by assigning a weight to each example. After each step, the weights of the correctly classified examples are reduced an the weights of the misclassified examples are increased. The weights can be seen as a probability distribution over misclassifications.

Boosting is a most successful approach in terms of accuracy, but the interpretability of its model is very limited. Following the terminology from this paper, one might be tempted to call the first base model the global model and the following models, that are learned with respect to the error distribution of the combined model so far, local models. The problem is, that the following models are not meaningful by themselves, but only with respect to all the models and their weights learned so far. 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 combined classifier so far has for some reason estimated too much influence of these attributes to the negative class. Also, Boosting is a greedy combined learner, i. e. previous models are not corrected once they have been learned, even if it turns out that they are wrong in several parts.

With respect to interpretability, most other combined learners suffer from the same problem, namely that to understand the model one has to understand every single base model plus the way these models are combined. Even if the base models are trivial, their combination can be quite complex. Boosting with decision tree stumps is an example of that.

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 [18] and [19]. 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 [16] decision trees and kernel density estimators have been combined to smoothen the posterior class probabilities.

However, in most cases existing approaches are usually either not easily interpretable or limited to a specific class of base learners. The goal of this paper was to find an algorithm that keeps up interpretation and works with arbitrary base classifiers.

6 Experiments

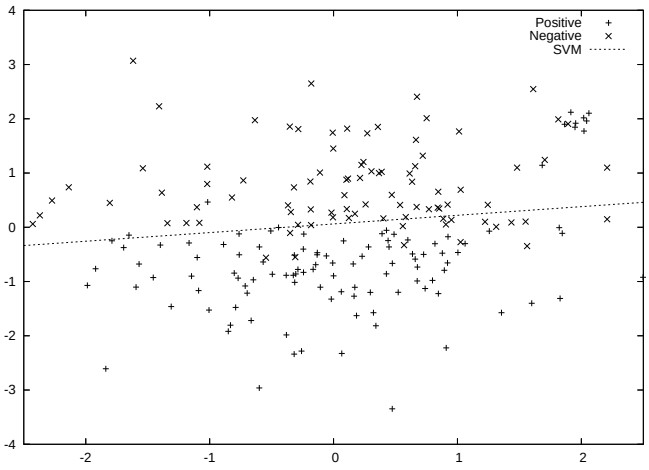

Fig. 3. Global model learned by a linear Support Vector Machine on a simple data set.

Let us first investigate the proposed local model algorithm on an artificial data set. Figure 3 shows a 2-dimensional data set of 200 observations consisting of two Gaussians, centered at $(0, 0)$ and $(2, 2)$, respectively. The first batch contains 95% of the observations with a standard deviation of $\sigma = 1$, while the second batch is smaller both in term of number of observations (5%) and in standard deviation ($\sigma = 0.1$). The positive examples are the examples from the first batch with negative second coordinate plus the examples from the second batch. An additional error of 5% in the labels was randomly added.

The straight line in Figure 3 shows a linear SVM classifier learned over all examples. One can see how the linear hyperplane is pulled into an ascending slope by the positive examples from the second batch.

Figure 4 shows the result of the local model algorithm after one iteration. In addition to the global linear classifier, a local linear classifier was added to classify the examples from the second batch as positive. Notice that the global

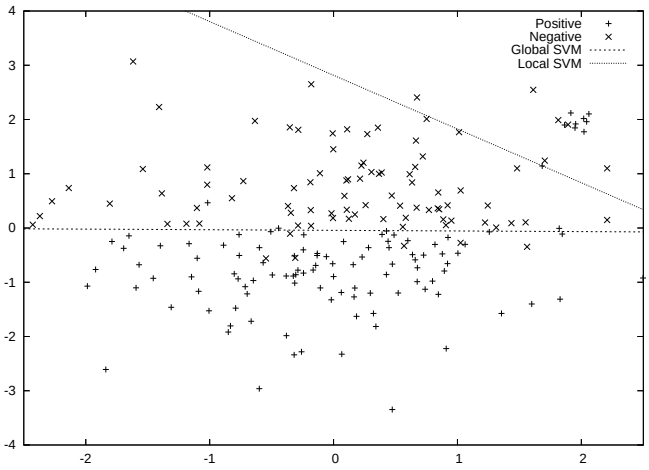

Fig. 4. Global model plus local model learned by the proposed algorithm with linear SVMs as the base learner.

classifier has gone back to the axis-parallel alignment that is optimal for the first batch.

6.1 Complexity Measure Reduction

The effect of interpretability improvement is of course hard to measure. To obtain quantifiable results, a C4.5 [12] decision tree learner is used in the following experiments, such that *interpretability* can be reduced to meaning *having a small number of nodes or levels*. To ensure an interpretable tree, the learning algorithm was modified such that the maximum depth of the tree was cut off at 75% the depth of the tree from the vanilla algorithm.

The experiments were conducted on 7 data sets, including 5 data sets from the UCI Repository [10] (breast, covtype, diabetes, ionosphere, liver) and 2 other real-world data sets: a business cycle analysis problem (business) and intensive care patient monitoring data (medicine). Prior to learning, nominal attributes were binarised and the attributes were scaled to expectancy 0 and variance 1. Multi-class-problems were converted to two-class problems by arbitrarily selecting two of the classes (covtype) or combining smaller classes into a single class (business, medicine). For the covtype data set, a 1% sample was drawn. The following table sums up the description of the data sets:

Name	Size	Dimension
breast	683	10
covtype	4951	48
diabetes	768	8
ionosphere	351	34
liver	345	6
business	157	13
medicine	6610	18

In addition to depth and number of nodes of the decision tree, the error of the combined classifier *C-error*, the error of the global decision tree classifier *G-error* and the disagreement between global and combined classifier *disagree*, i. e. the fraction of examples predicted differently by both classifiers have been recorded. All numbers reported are results of a 5-fold cross-validation.

The Support Vector Machine was used as local classifier. The type of kernel function (linear or radial basis) and the kernel parameters were selected beforehand by optimization over all examples. The C parameter was set at a default value. For density estimation, k-medoids, a robust version of k-means, was used.

In all experiments, one local model was learned and a fraction of $\tau = 0.3$ examples were allowed for this local model. 3 EM iterations were performed. The following table sums up the performance of the local model algorithm:

Data	Iteration	Depth	Nodes	C-Error	G-Error	Disagree
breast	1	5.0	17.4	0.042	0.042	0.0
	2	2.8	10.6	0.236	0.074	0.226
	3	1.2	3.4	0.029	0.067	0.055
covtype	1	22.4	528.2	0.228	0.229	0.001
	2	15.6	332.6	0.227	0.227	0.054
	3	15.6	366.6	0.227	0.239	0.058
diabetes	1	6.8	24.6	0.266	0.276	0.016
	2	4.0	14.6	0.242	0.250	0.251
	3	4.0	15.8	0.247	0.251	0.030
ionosphere	1	7.0	21.4	0.096	0.096	0.0
	2	4.0	9.4	0.116	0.119	0.054
	3	4.0	10.2	0.099	0.122	0.034
liver	1	9.8	49.0	0.313	0.318	0.005
	2	5.2	20.2	0.368	0.339	0.220
	3	6.0	24.6	0.350	0.347	0.205
business	1	6.4	20.2	0.223	0.223	0.0
	2	3.6	10.2	0.210	0.255	0.094
	3	3.6	12.2	0.216	0.222	0.057
medicine	1	19.2	389.4	0.202	0.204	0.017
	2	13.6	245.4	0.206	0.215	0.073
	3	13.6	239.8	0.211	0.219	0.104

It can be seen that the complexity of the tree classifier is reduced dramatically compared to the tree from the usual C4.5 algorithm (the first step is done

without cutting off the tree, hence the size of the vanilla C4.5 tree can be seen in iteration 1). However, this complexity reduction does not decrease the classification performance, with the exception of the liver data set. On the average, in the third iteration the error is reduced by 4% while the size of the decision tree is reduced by 46%. Global and combined classifier differ in 7% of the cases. This shows, that the local model algorithm can effectively find a much less complex approximation to the optimal model.

7 Conclusion and Future Work

Local models are the extension of local patterns to the supervised learning case. They provide a good way to improve the interpretability of a classifier by restricting the classifier to the essential parts of the model and leaving out patterns that it hardly can approximate. In this paper, a local model algorithm was presented that learns a global classifier plus local models to reduce complexity of the global model, ensure the prediction quality of the combined model and provide guarantees that combined and global model will differ only up to a user-specified degree. This allows the user to restrict his attention to the global model and still get valid information about the high-quality combined model.

Another important aspect of this approach is the interpretability of the local models. In applications global rules often express only trivial knowledge about the data, that the user is already aware of, while the significant exceptions of these rules are highly informative. This aspect of local models will be dealt with in future work.

Another open problem concerns the runtime of the algorithm. Each iteration requires to learn a density and a classification model for both the global learner and the local learners. While the local models hopefully do not pose a problem, as only a small part of the data is concerned, the global models require to deal with possibly very large data sets. Theoretically, learning with local models could alleviate this problem, as it allows room for errors of the global model, which can be dealt with by local models. This would allow to use sampling or a faster, less accurate learner for the bulk of the data. It remains to be investigated how well this works in practice.

Acknowledgments

The financial support of the Deutsche Forschungsgemeinschaft (Collaborative Research Center SFB 475, "Reduction of Complexity for Multivariate Data Structures") is gratefully acknowledged.

References

1. Philip K. Chan and Salvatore Stolfo. Experiments in multistrategy learning by meta-learning. In *Proceedings of the second international conference on information and knowledge management*, pages 314–323, Washington, DC, 1993.

2. A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *J. Royal Statist. Soc. Ser. B*, 39:1–38, 1977.
3. Y Freund and R.E Schapire. Game theory, on-line prediction and boosting. In *Proceedings of the 9th Annual Conference on Computational Learning Theory*, pages 325 – 332, 1996.
4. Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: A statistical view of boosting. Technical report, Departement of Statistics, Stanford University, Stanford, California 94305, July, 23 1998.
5. João Gama and Pavel Brazdil. Cascade generalization. *Machine Learning*, 41(3):315–343, 2000.
6. Ursula Garczarek. *Classification Rules in Standardized Partition Spaces*. PhD thesis, Universität Dortmund, 2002.
7. Isabelle Guyon, Nada Matic, and Vladimir Vapnik. Discovering informative patterns and data cleaning. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 2, pages 181–204. AAAI Press/The MIT Press, Menlo Park, California, 1996.
8. David Hand. Pattern detection and discovery. In David Hand, Niall Adams, and Richard Bolton, editors, *Pattern Detection and Discovery*. Springer, 2002.
9. G. Miller. The magical number seven, plus or minus two: Some limits to our capacity for processing information. *Psychol Rev*, 63:81 – 97, 1956.
10. P. M. Murphy and D. W. Aha. UCI repository of machine learning databases, 1994.
11. John Platt. *Advances in Large Margin Classifiers*, chapter Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. MIT Press, 1999.
12. John Ross Quinlan. *C4.5: Programs for Machine Learning*. Machine Learning. Morgan Kaufmann, San Mateo, CA, 1993.
13. R.J. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.
14. P.J. Rousseeuw. Least median of squares regression. *Journal of the American Statistical Association*, 79:871–880, 1984.
15. Stefan Rüping. A simple method for estimating conditional probabilities in svms. In A. Abecker, S. Bickel, U. Brefeld, I. Drost, N. Henze, O. Herden, M. Minor, T. Scheffer, L. Stojanovic, and S. Weibelzahl, editors, *LWA 2004 - Lernen - Wissensentdeckung - Adaptivität*. Humboldt-Universität Berlin, 2004.
16. Padhraic Smyth, Alexander Gray, and Usama M. Fayyad. Retrofitting decision tree classifiers using kernel density estimation. In *International Conference on Machine Learning*, pages 506–514, 1995.
17. Edgar Sommer. *Theory Restructering: A Perspective on Design & Maintenance of Knowledge Based Systems*. PhD thesis, University of Dortmund, 1996.
18. L Todorovski and S. Dzeroski. Combining multiple models with meta decision trees. In *Proceedings of the Fourth European Conference on Principles of Data Mining and Knowledge Discovery*, pages 54–64. Springer, 2000.
19. Ljupco Todorovski and Saso Dzeroski. Experiments in meta-level learning with ILP. In J. M. Zytkow and J. Rauch, editors, *Proceedings of third European Conference on Principles of data mining and knowledge discovery (PKDD-99)*, volume 1704, pages 98–106. Springer, 1999.
20. Kagan Tumer and Joydeep Ghosh. Order statistics combiners for neural classifiers. In *Proceedings of the World Congress on Neural Networks*, 1995.
21. V. Vapnik. *Statistical Learning Theory*. Wiley, Chichester, GB, 1998.
22. D. Wolpert. Stacked generalizations. *Neural Networks*, 5:241–259, 1992.