Improving Cross-Validation Classifier Selection Accuracy through Meta-Learning

Jesse H. Krijthe
Student number: 1527053
MSc Computer Science
Track Media & Knowledge Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

12 July 2012

Preface

Progress in research and implementations of methods from machine learning, pattern recognition and statistics has provided a large number of methods for practitioners to choose from. In the case of classification problems, this choice is often made based on the cross-validation errors of different methods. The research presented in this thesis is concerned with whether it is possible to improve upon the accuracy of this selection strategy by reframing the classifier choice as a meta-learning problem. Unlike traditional work on meta-learning, I focus on improving selection accuracy, rather than computational efficiency. From this new perspective, I show better classifier choices can be made than the cross-validation selection strategy in some simulated cases. I study these cases and discuss a methodology to test whether these results might hold on real-world datasets as well. This thesis holds the results and insights gained from these inquiries.

The thesis paper is intended to be self-contained, no additional material is part of the main work. The chapters following the main paper contain notes from my research log on topics and experiments that did not make it into the main text. It also includes notes on my initial research topic on using data complexity measures to characterize changes in problem complexity during classifier combination. I have spent four months on this initial topic before coming up with an experiment that finally culminated in the work presented in this thesis. The appendix also contains a four-page paper [8] on the initial results of this research that was accepted to be presented at the International Conference on Pattern Recognition 2012.

Most of the research for this thesis was conducted during a six month stay in the Statistics and Learning department of Alcatel-Lucent Bell Labs in Murray Hill, New Jersey under the supervision of Dr. Tin Kam Ho (Bell Labs) and Dr. Marco Loog (Delft University of Technology). I would like to thank Tin and her team for welcoming me at the labs, making me feel part of the group and including me in their discussions. I am particularly grateful to Tin for taking the time to discuss my work even when it was not always convenient. I would also like to thank Piotr Mirowski, Harald Steck and Adriaan J. de Lind van Wijngaarden for their guidance and friendship during my visit. Last but not least, I would like to thank Marco Loog for the continued support during all phases of the project.

I would like acknowledge and thank the graduation committee consisting of the following members: Prof. dr. ir. M.J.T. Reinders, Dr. M. Loog, Prof. dr. J.N. Kok, Dr. L. van der Maaten and Mr. Yan Li.

Jesse Hendrik Krijthe, 6 July 2012

Contents

Chapter 1

Thesis paper

Improving Cross-Validation Classifier Selection Accuracy through Meta-learning ☆

Jesse H. Krijthe

Delft University of Technology
jkrijthe@gmail.com

Abstract

In order to choose from the large number of classification methods available for use, cross-validation error estimates are often employed. We present this cross-validation selection strategy in the framework of meta-learning and show that conceptually, meta-learning techniques could provide better classifier selections than traditional cross-validation selection. Using various simulation studies we illustrate and discuss this possibility. Through a collection of datasets resembling real-world data, we investigate whether these improvements could possibly exist in the real-world as well. Although the approach presented here currently requires significant investment when applied to practical applications, the concept of being able to outperform cross-validation selection opens the door to new classifier selection strategies.

Keywords: Classifier selection, Meta-learning, Error estimation

1. Introduction

Research in machine learning, pattern recognition and statistics has produced a plethora of different classification algorithms for practitioners to choose from. Given this large number of classification methodologies, as well as parameter settings and scalings or transformations of the data, the classifier selection problem remains important in machine learning practice. But how do we, preferably with a high level of accuracy, select the method that is most suitable to a given problem?

The optimal way to select a classifier would be to try all classifiers under consideration and select the method with the lowest classification error. In practical settings, we are always dealing with finite data and the classification errors therefore needs to be estimated. The prevailing method to approach this estimation in classifier selection is to use a cross-validation strategy [24]. This is an intuitive approach that provides (nearly) unbiased estimators for the errors. Unfortunately, this method is not without its drawbacks. First of all, evaluating the cross-validation errors for every possible method is computationally intensive and therefore does not scale well as the number of classifiers increases. Secondly, for small sample sizes, cross-validation error estimates have been shown to become unreliable [3, 12].

To improve upon the first drawback, meta-learning [25] was introduced to aid in the classifier selection. Meta-learning treats classifier selection as a classification problem in itself. The classes in this 'meta-problem' are the optimal classifier choices, while its features can be any statistics calculated on the dataset. In the meta-learning literature, several computationally efficient meta-features were introduced as alternatives to cross-validation errors. Their goal is to retain accuracy in selection, while improving time-efficiency. In this field, however, classifier selection through cross-validation is used as the benchmark by which meta-learning is compared in terms of accuracy. As such, meta-learning becomes an computationally efficient approximation of the presumably more accurate cross-validation selection. What has not been considered, however, is whether and how we would improve the *accuracy* of classifier selection through a meta-learning framework. Especially in the small-sample setting referred to above, a possibility and a need exist for a selection strategy that improves upon cross-validation selection.

Our work considers precisely this question: is it possible to use meta-learning techniques to improve the accuracy (rather than the computational efficiency) of classifier selection using cross-validation? The approach we take is to project cross-validation selection into the meta-learning framework. Maybe surprisingly, this constitutes a novel view to meta-learning and opens up new possibilities unexplored to this date.

Seen from this meta-learning perspective cross-validation corresponds to a static, untrained meta-classifier, using the cross-validation errors as its meta-features. The question then becomes whether using this untrained method is justifiable. We study this problem by constructing problem universes where a trained meta-learner outperforms the static rule corresponding to cross-validation selection. Thus, the main contribution of this work is to show that, at least at a conceptual level, meta-learning *can* be used to outperform cross-validation selection in terms of selection accuracy. It does not only have to serve as a time-efficient proxy.

Through a second simulation study we argue that additional

☆ Preliminary results of this research have been described in a paper with the same title in [15], which covers the first experiment presented here. We extends these results with larger experiments on pseudo real-world data and a more thorough description and discussion of our findings.

measures on the dataset, apart from the cross-validation errors, can aid in the classifier selection problem. In other words, not all information important to the classifier selection problem is captured by the cross-validation errors. This offers support for approaches like data complexity [10] which try to capture these effects. This is our second contribution.

Although we can show these effect conceptually, through simulated universes, the third contribution of this work is to study the possible improvements in accuracy of meta-learning selection on a collection of datasets resembling real-world data. Even though the effects are smaller then in the simulation studies, the important observation is that the effect might exist in the real-world settings as well. Whereas cross-validation selection has long been considered the optimal method for classifier selection based on limited training data, we show this is not necessarily the case.

The rest of this paper is organized as follows. Section 2 discusses work related to this paper. Section 3 introduces meta-learning and cross-validation in more detail. Section 4 covers our experimental setup and results, both the constructed universes used to illustrate the possibility of outperforming cross-validation selection, as well as some experiments on pseudo real-world data. Section 5 offers a discussion of the results. Section 6 concludes.

2. Related Work

2.1. Cross-validation selection

A thorough description and evaluation of the strategy of using cross-validation errors for the classifier selection problem was first offered by [24]. This method has remained the prevalent strategy in use. The main conclusion in [24] is that the cross-validation strategy allows for a higher average accuracy and lowers the chance of poor performance, compared to sticking with a single classifier. The author also notes, however, that like any single classifier, the strategy constitutes an inductive bias. This bias may or may not correspond to the actual properties of the problem at hand. Our work can be interpreted as a way to incorporate experience gained from other datasets to adapt this bias, so it better corresponds to the types of problems the practitioner is facing.

2.2. Issues with cross-validation errors

Even though cross-validation selection is an effective approach in many cases, for small samples, cross-validation error estimates have been shown to become unreliable. [5] gives an overview of the potential problems of large variances of the cross-validation errors for feature and classifier selection in the case of small sample micro-array classification. [3] studies the effectiveness of cross-validation error estimation compared to resubstitution and bootstrapped estimates of the error on both real and simulated data. They note that, in practice, the small bias of cross-validation is often less important than its variance properties. They find that the variance of cross-validation error estimators tends to be larger than resubstition or bootstrapped error estimation. In particular, they note this variance increases as the complexity of a classifier increases. This relates to a study of the stability of classifiers of [14], that found similar results.

[12] uses only simulated data to study the problem of getting an accurate error estimate of the performance of a classifier. Noting cross-validation errors become too unreliable for this purpose in the small-sample setting, they propose to use a hold-out set to determine error estimates and confidence intervals. Whereas these papers try to get a hold on the size of the variance of the cross-validation error estimator, our work studies these small sample problems for classifier selection specifically. We try to answer the question whether meta-learning can help in situations were cross-validation errors become unreliable. Unlike the solution proposed by [12], we therefore choose not to use a hold-out set but attempt to estimate performance on the whole set of training objects available: in the small-sample case, using fewer training objects can actually change which classifier performs best. Because we assume the practitioner wants to get the most of the little data he has, we use all the data to train our classifiers.

2.3. Meta-learning

Another problem the cross-validation selection approach to classifier selection faces is that it can quickly become impractical from a computational point of view as the number of classifiers or the training set sizes increase. The main focus in meta-learning has been to develop alternative features that are less computationally intensive than cross-validation errors but can still aid the classifier choice [4]. A comprehensive overview of meta-learning is given by [25].

Since the first meta-learning study [1], various meta-features have been introduced and evaluated. Common meta-features include [13, 17]: statistical measures such as the average mean, variance, skewness and kurtosis of the variables in the dataset; information theoretic measures, such as entropy; and simple descriptive measures, such as the number of variables, classes or object in the dataset. More esoteric meta-features include measures on the structure of a decision tree trained on the data [2, 22].

The land-marking approach proposed by [23] involves using cross-validation errors of simple classifiers as meta-features to select between a set of more complex classifiers. These simple, computationally efficient classifiers form the 'landmarks' on which we anchor the performance of more elaborate classifiers. The approach taken in our research is most similar to this approach. Unlike landmarking, we are not concerned with time-efficiency, but focus on accuracy improvement. We therefore use the cross-validation errors of *all* classifiers in the set, instead of the cross-validation errors of a small set of simple classifiers.

[8] investigates whether cross-validation errors of the landmarks can predict the performance of more complex classifiers on more training data. Although this may look similar to our approach, our goal is to actually predict classifier performance on small training sets, not predict the performance of classifiers trained on large training sets using only a subsampled training set.

2.4. Data complexity measures

A recent attempt at putting together a comprehensive set of 14 meta-features and studied in [10, 19, 20] is known as the data complexity measures. Apart from combining the most effective meta-features from the literature they also introduce other measures describing the geometrical and topological properties of the dataset. Well known measures include the leave-one-out errors of a 1 nearest neighbor and a linear classifier, inspired by the landmarking approach, and the ratio of the number of objects to the number of dimensions in the dataset. The goal of the geometric properties is to establish the complexity of the classification boundary. One example of such a measure is the nonlinearity of a classifier with respect to the dataset introduced by [11]. For a given classifier, this is defined as the fraction of objects, obtained by interpolating between two objects of the same class assigned by the classifier, that share the class assignment with these two points. As [18] notes, this fraction can be used as a measure of how much two classes interleave.

These data complexity measures were then used to establish the so-called domains of competence of different classifiers [19]. The domain of competence of a classifier is the types of datasets this classifier has lower error on than other classifiers. The data complexity measures are a way to describe these types quantitatively. If the combination of these measures offers a succinct and interpretable description of datasets, a domain of competence allows us to better understand what properties of a dataset allow a classifier to succeed or fail. The measures can also be utilized as features in the meta-learning problem.

3. Methods

In this section we introduce in more detail the two approaches compared in this work, cross-validation selection and meta-learning, in the broader setting of parameterizations for classifier selection. Our intention is to illuminate the similarities between both approaches and present the cross-validation selection approach as a simple form of meta-learning.

3.1. Classifier selection

The goal in this work is to select from a set of m classifiers $C = \{c_1, c_2, \ldots, c_m\}$, the one c_i that will give the lowest classification error accuracy e_i when trained on our given dataset D. In practice, we have limited data, so e_i is unavailable. The choice of classifier therefore needs to depend on measures that we can calculate from our given data D. We will call these measures parameterizations of D. These measures offer a quantitive description (hence parameterization) of a dataset in a new space spanned by these parameters (see Figure 1). The cross-validation error estimates introduced in the next section are one possible parameterization. The meta-learning literature offers a host of others in the meta-features, for instance, statistical measures, information theoretic measures or data complexity measures [10, 19, 20].

Once the measures are chosen, a classifier selection strategy becomes a geometrical boundary in the space parameterized by these measures (see Figure 1). For instance, if we take the number of objects in the dataset n as a measure, a selection strategy could be to always use a nearest mean classifier when the number of objects in a dataset is below a certain threshold, to guard against overtraining. In this case, and in the case of cross-validation selection that we will introduce next, this geometrical boundary is fixed. Meta-learning, on the other hand, offers a dynamic boundary learned by experience on previous datasets.

3.2. Cross-validation

As referred to above, cross-validation error estimates are a commonly used parameterization of the dataset on which to base the classifier selection choice. Cross-validation errors are a direct estimate of the real error that each classifier would make on an independent test set. The cross-validation procedure works as follows. Our dataset, D is randomly divided into k parts, called folds, D_k, each containing $\frac{n}{k}$ objects, where n is the total number of objects in our dataset. Next, we cycle through these folds ($i = 1, 2, \ldots, k$) using the objects in $k-1$ folds, $D \setminus D_i$, for training our classifier, c, and use the objects in the remaining fold, D_i, for testing the performance of c. We then average the error made on each of these folds to obtain the cross-validation error estimate. The k-fold cross-validation error can therefore be estimated as follows. $c(\cdot, \cdot)$ is a classification function whose first input parameter is the training data and the second a feature vector for which we want to predict the corresponding class, while its output is the predicted class. $(y_j^{(i)}, x_j^{(i)})$ is the true class/features values pair for observation j in fold i. $I(\cdot, \cdot)$ is an identity function which returns 1 if its parameters have equal values and 0 otherwise.

$$\hat{e}_{cv_k} = \frac{1}{n} \sum_{i=1}^{k} \sum_{j=1}^{\frac{n}{k}} I(y_j^{(i)}, c\left(D \setminus D_i, x_j^{(i)}\right)) \tag{1}$$

The k-fold cross-validation error can be shown to be a high-biased estimator of the real error. This is because in each of the folds we train the classifier using $n - \frac{n}{k}$ samples rather than all n samples in the dataset. This bias becomes smaller as k becomes larger, whereas computational cost increases with k. The variance has slightly more complex behavior, showing high variance for both small and large values of k. It has been shown that in practice, $k = 10$ is often a good choice [9, 14] although differences are minor for reasonable values of k [3]. We have therefore chosen to use $k = 10$ in our experiments.

In repeated cross-validation we repeat the process in Equation 1 several times to reduce the variance of the estimate. This also allows us to estimate the variance of the k-fold cross-validation error. In the related work section we mentioned that the variance of the cross-validation is usually larger than that of similar methods. As noted by [3], the variance estimate using repeats only measures one source of the variance of the cross-validation errors. This can therefore be interpreted as a lower bound on the variance. Since this variance estimate might offer information on the accuracy of the cross-validation error estimate, we will add it as an extra feature in one of our experiments (see section 4.3 and 4.4).

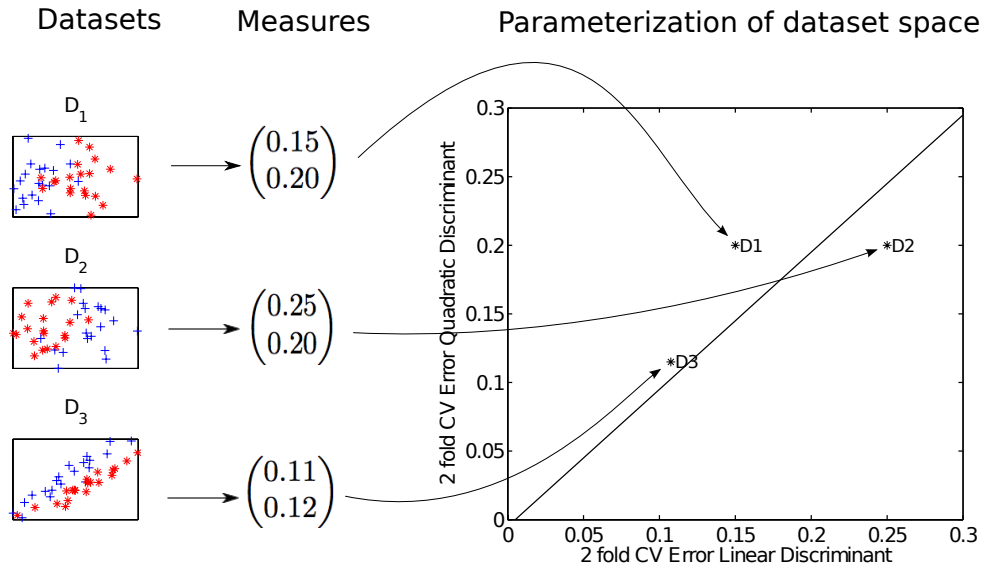


Figure 1: Each dataset (on the left) can be described by measures or statistics calculated on this dataset. Each dataset therefore form a vector in the space parameterized by these measures (on the right). The example shows the case were the measures are the cross-validation errors of 2 classifiers. The boundary in the dataset space corresponds to the cross-validation selection strategy: if the dataset characterization is above the boundary, classifier 1, the linear discriminant, is preferred (has lower error), if below the boundary classifier 2, the quadratic discriminant, is preferred.

In classifier selection, the cross-validation errors for all classifiers will be compared and the classifier with the lowest error chosen. In the space parameterized by the cross-validation errors of the different classifiers, this cross-validation selection rule will form a static boundary. Consider for instance the case where we compare two classifiers, see Figure 1. Datasets that form a point on this diagonal have cross-validation errors that are exactly equal, whereas above or below this diagonal, the cross-validation error of one of the classifiers is lower than the cross-validation error of the other. The cross-validation selection strategy is to choose the classifier with the lowest cross-validation error estimate.

A wrong selection may result if the cross-validation error differs from the real error. A high variance of the cross-validation estimator increases the likelihood of these errors being far apart and subsequently making the wrong classifier selection. In this work we introduce this high variance by considering the classifier selection choice in the small sample setting. This was partly inspired by the work of [3] on classifier error estimation in small sample micro-array classification.

3.3. Meta-learning

The field on meta-learning offers an alternative selection procedure by treating the classifier selection problem as yet another classification problem, called the meta-problem. Given a set of datasets $D = \{D^1, D^2, \ldots, D^N\}$, we have shown that each dataset can be seen as an object in the space parameterized by measures calculated on the dataset. Some examples of measures used in the meta-learning literature are the number of objects in the dataset, the error rate of a simple classifier or the size of a decision tree trained on the dataset. Because these measures act as the features of the meta-problem, they are known as meta-features. From now on, in keeping with conventions in meta-learning literature, we will therefore refer to parameterizations as meta-features. The meta-class for each meta-object (dataset) is the classifier that performs best on that dataset. Classifier selection is now done by training a classifier (called a meta-classifier) on this meta-problem and using it to predict the best classifier (meta-class) for a new dataset, again using the meta-features as the parameterization of this dataset.

Meta-learning differs from cross-validation selection in two ways. Firstly, the parameterizations (i.e. meta-features) are different, and secondly, learning occurs in the dataset space.

The first distinction is that whereas cross-validation selection attempts to estimate the real classification error using a computationally intensive procedure, meta-learning has traditionally provided time-efficient measures to aid in the classifier selection. However, if we are only concerned with selection accuracy and not efficiency, we propose to use cross-validation errors as meta-features since they may turn out to be very discriminative. And as we treat the selection problem as a classification problem we can include other measures calculated on the dataset as meta-features as well. It is unclear whether the cross-validation error meta-features will contain all possible information that can be used in classifier selection. For instance, is information on the sample size of the dataset sufficiently integrated into the high errors of complex classifiers or would adding these sizes separately help in the classifier selection problem?

The second distinction is that meta-learning estimates a selection rule in the dataset space by training a meta-classifier using a collection of datasets. We assume these datasets come from a probability distribution over datasets, which we will refer to as a *universe* of problems. Cross-validation selection uses an untrained rule whereby we *always* select the classifier with the lowest cross-validation error. So, contrary to meta-learning,

the cross-validation selection rule does not change given experience of previous classifier selections.

Thus, seen from a meta-learning perspective, cross-validation selection can be considered a very simple untrained meta-classifier using as its meta-features the cross-validation errors. However, this means this simple meta-classifier has to be justified in some way: is the meta-problem really so simple that an untrained rule provides optimal performance, or is a trained procedure warranted? Using several simulation studies we will demonstrate that there exist universes of problems where the space formed by the cross-validation errors does not support the superiority of this simple decision rule. Instead, using a different meta-classifier allows for an improvement in accuracy over cross-validation based selection.

4. Experiments

The following sections contain the results from several simulations and experiments conducted to test whether meta-learning can improve upon cross-validation selection. The first experiment is a simulation study meant to illustrate the concept using an small intuitive universe of problems where meta-learning clearly outperforms cross-validation selection. In the second experiment, we construct a universe in which adding additional meta-features can improve performance. Improvement in accuracy in this case, demonstrates cross-validation errors do not necessarily carry all information useful for selecting a classifier.

Whereas these first two experiments are done on simulated universes, the third experiment tests whether these effects occur on datasets that arguably conform more to what we encounter in the real world as well.

The setup of the experiments follow a common pattern: *Data* We start with the construction of a collection of datasets which we call a universe of problems. They are either obtained by artificial data generators (experiment 1 and 2) or from a collection resembling real-world datasets (experiments 3). *Setup* Next we create the meta-problem by training classifiers on the dataset and calculating the meta-features. *Results* In the last step we train and test several meta-learners on this meta-problem, including the static cross-validation selection rule, and compare their performance. All experiments were run using classification procedures from the PRTools Matlab toolbox version 4.2.1 [6]. The first experiment was also implemented in R for verification and additional testing.

4.1. Notation

Before we get to the description of the experiments, it will be worthwhile to introduce some notation. Base-problems, meaning classification problems with continuous attributes like the ones a practitioner might come across will be denoted by capital letters (i.e. G). The training and test sets that a problem consists of will be denoted by G^{train} respectively G^{test}. Meta-problems are denoted by bold capitals (i.e. $\mathbf{G}$) and the training and testing splits of this meta-problem by $\mathbf{G}^{TRAIN}$ respectively $\mathbf{G}^{TEST}$. We will call the classifier predicted to be the most suitable for a particular problem by our meta-learner the *assigned* classifier. The *best* classifier is the classifier with lowest accuracy on a large test set. In other words, the ground truth.

4.2. Simulated meta-learning example

In this section we construct a universe of problems to illustrate and visualize the concept of improving cross-validation selection.

Data

Consider an example universe of problems consisting of instances of two types of two-dimensional problems. The first problem type has strongly overlapping, two dimensional Gaussian classes (call it $\mathbf{G}$). The second problem type has a more complex boundary with 'banana-shaped' classes with Gaussian noise but, unlike the first problem, small class overlap (call it $\mathbf{B}$) (see Figure 2). We generate problem instances from $\mathbf{G}$ by selecting the separation of the class means from a uniform distribution. For instances from $\mathbf{B}$, we change the variance of the Gaussian noise. In this way we create a family of 500 instances for each problem: $\mathbf{G} = \{G_1, G_2, ..., G_{500}\}$, and $\mathbf{B} = \{B_1, B_2, ..., B_{500}\}$. From each of these problems B_i or $G_i (i = 1, ..., 500)$, we extract a sample of size N_{train} (varying uniformly randomly between 20 and 100) for training, call this G_i^{train} and B_i^{train}, and use the rest ($N_{test} = 20000 - N_{train}$) for testing, G_i^{test} and B_i^{test}. Even though for the two-dimensional setting considered the sample sizes are not particularly small, they will cause cross-validation errors to have high variance. In real world problems, with complex boundaries, we expect this effect to occur for even larger sample sizes. The large test set size is used to get an accurate estimate of the real error of the classifier for the problem instance.

Setup

The goal of our classifier selection task is to choose between two classifiers: a nearest mean classifier (NM) and a 1-nearest neighbor classifier (1-NN). Casting this as a meta-learning problem, we create the meta-features and meta-classes as follows. For each G_i, we obtain a 10-fold cross validation error for the NM classifier using G_i^{train} to obtain our first meta-feature. Doing the same for the 1-NN classifier, we obtain the value of meta-feature 2 for G_i, and similarly for each B_i.

The meta-class for each of G_i or B_i is the classifier that performs the best for the testing data G_i^{test} or B_i^{test}. This is determined by, say, using all of G_i^{train} for training the NM classifier, estimating the error rate of the resultant classifier on G_i^{test}, and doing the same for the 1-NN classifier. If the NM gives a lower error, G_i is labeled with the best classifier (or true meta-class) NM, and vice versa. Similarly we obtain the best classifier for each B_i.

For the meta-learning problem, approximately half of the 500 instances from $\mathbf{G}$ or $\mathbf{B}$ are used for meta-training and the rest are reserved for meta-testing. Recall that the cross validation rule corresponds to a static classifier, and hence requires no training. We compare this to a trainable classifier that is a linear support vector machine, trained with the meta-features for $\mathbf{G}^{TRAIN}$ and $\mathbf{B}^{TRAIN}$. The comparison is shown in Figure 2.

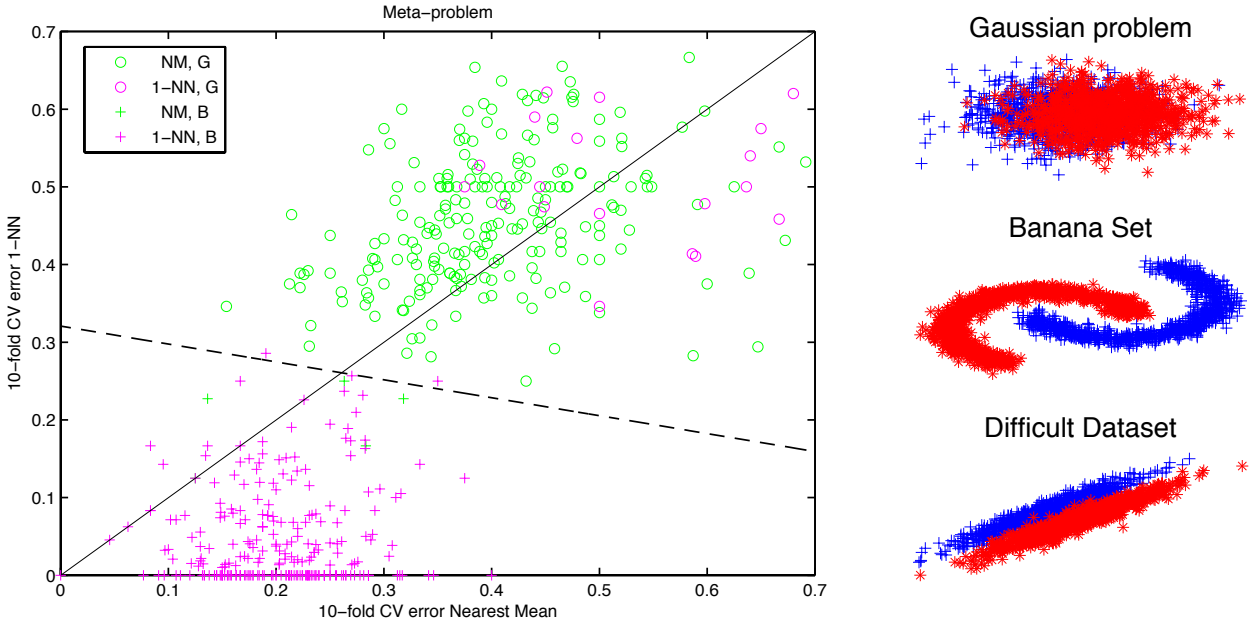


Figure 2: On the left: an example of a universe of problems where static cross-validation selection does not perform well. On the right: the problems 'Gaussian', 'Banana Set' as discussed in Section 4.2, and the 'Difficult Dataset', a 2D version of the problems discussed in Section 4.3.

Results

In Figure 2, the points plotted represent problem instances from $\mathbf{G}^{TEST}$ and $\mathbf{B}^{TEST}$. Their coordinates are given by the two meta features, and they are marked with symbols denoting their source family ($\circ$ for the Gaussian problems, or $+$ for the Banana problems), and colors denoting their true meta-class (green for NM, and pink for 1-NN).

We can observe that 1-NN almost always gives better performance for the cluster of Banana problems (almost all points in the cluster of $+$ are marked pink). The other cluster formed by the Gaussian problems primarily favors the NM classifier (most $\circ$'s are marked green). Note that in this particular universe, the cross-validation rule does not give the best separation: the decision boundary crosses through the cluster of Gaussian problems ($\circ$'s), forcing one to choose the 1-NN classifier for the points below the diagonal, while performance in this cluster is in fact almost always better with the NM classifier. The selection error, defined as the proportion of datasets in $\mathbf{B}^{TEST} \cup \mathbf{G}^{TEST}$ assigned by the meta-rule to a classifier different from their true meta-class, is 0.16 for cross-validation. The support vector machine reduces the selection error to 0.06, offering a more accurate alternative.

4.3. Additional meta-features

The example in the previous section has shown that learning the selection rule can improve performance in some cases where the cross-validation errors have high variance. In more elaborate universes of problems, the meta-classes may overlap heavily, making it difficult to select a classifier based solely on the cross-validation errors. In this experiment we study whether other measures characterizing the dataset may reduce this overlap by introducing extra meta-features that can aid the selection task. A different universe of problems is used than in the previous section. This is because the universe in the previous experiment was constructed specifically to demonstrate the change in the selection boundary, while leaving little need for improvement from additional features.

Data

The universe we consider in this experiment consists of 100 dimensional two-class problems where most of the class variance is along the class boundary (the 'Difficult Dataset' from the PRTools library), see Figure 2 (right). Instead of introducing more types of problems we introduce variability to the universe by randomly selecting the class-priors using a uniform distribution between 0 and 1. Extra randomness is introduced by varying the training set size (N_{train} is now between 20 and 100). Again $N_{test} = 20000 - N_{train}$, in order to get accurate estimates of the true error. We generate 1000 datasets $\mathbf{D} = \{D_1, \ldots, D_{1000}\}$ this way to be able to train and evaluate different meta-learners.

Setup

The classifiers we consider in this setting are the nearest mean classifier and the Fisher classifier. For each dataset D_i we calculate the 10 fold cross-validation estimate of both classifiers on the training set D_i^{train}. We repeat this five times. The averages of these five runs are used as the cross-validation error meta-features. The variances of the five runs are also used as meta-features. We also add the number of objects in D_i^{train} as an extra meta-feature. The meta-classes are again determined by training both classifiers on D_i^{train} and evaluating their performance on D_i^{test}. As before, the classifier with the lowest

error on this test set is the meta-class. We evaluate different meta-learners and meta-features by estimating the selection error using 10 times repeated, 10 fold cross-validation. Given the size of the meta-problem $\mathbf{D}$, which consists of 1000 datasets, we assume the cross-validation error is sufficiently accurate.

Results

Cross-validation selection gives a selection error of 0.237. Using a k nearest neighbor classifier as a meta-learner, we get an error of 0.238. Compare this to an error of 0.271 when always selecting the most common classifier in the training set as the assigned classifier. Adding to the meta-features an extremely simple characterization of the sampling density of the dataset, n the number of objects in the dataset, causes a reduction in error to 0.151. Adding the cross-validation variance estimates gives an error of 0.221, whereas adding both these meta-features gives an error of 0.127. For a linear discriminant classifier the error rates are respectively 0.241, 0.159, 0.239 and 0.110, for adding no-extra meta-features, only the number of samples, only cross-validation variance and adding both meta-features.

The average difference over the datasets between the classifier with the lowest true classification error and the true error of the selected classifier, gives an indication of the size of the improvement that is attained by making a good classifier selection. Cross-validation selection gives an average error that is 0.016 higher than the classification error obtained when always choosing the best classifier. For the linear discriminant with both meta-features this mean difference is 0.004 while the variance is also smaller (0.0013 vs. 0.0002).

4.4. Pseudo real-world datasets

We have shown the concept of improving classifier selection through meta-learning works in some simulated universes. It is unclear, however, whether these findings also hold in real-world data. In this section we use a collection of datasets that shares properties with real-world data to see whether meta-learning still offers improvements.

Data

We utilize a collection of 300 datasets from a study by [19] where it was used to study domains of competence of different classifiers. The collection is constructed to span the space formed by data complexity measures, that attempt to characterize different sources of difficulty in classification problems. As such, a major advantage of using this collection is that it covers a large space of different problems we could come across in the real-world. Because we assume a collection of real-world datasets would also exhibit a wide range of different difficulties, we consider the collection as a whole to have some resemblance to real-world data. The flip side of this collection is that this coverage was achieved by perturbing the objects in five original seed datasets, making the perturbed datasets less like problems we would actually come across in the real-world. The resulting 300 datasets have divergent properties: the number of dimensions varies between 8 and 20, while the number of samples in the original datasets is between 230 and 950. The smallest number of samples per dimension is 15 while the largest is 85.

Using experimental, rather than simulated data poses two major problems. (1) We can no longer have a large number of test samples for each base-problem to establish the ground truth with arbitrarily high accuracy. (2) The number of datasets in our collection may be too small to train our meta-learner well.

The solution we have chosen is to estimate a density function on the datasets in our collection and use this density to generate more data. In effect, we are assuming that these estimated densities correspond to real-world data generating processes. Problem (1) is solved by generating enough data from this distribution to accurately estimate the real classification errors. Problem (2) is solved generating many different small datasets from these densities to form our meta-problem.

For, problem (1), generating data to establish the ground truth, it is not clear whether this approach removes us too far from properties of real-world data: do the densities, estimated with limited data from a limited set of problems, still correspond to problems we might come across in the real world? We will come back to this in the discussion. As a solution for problem (2) it is easier to check whether this approach is reasonable: if we take into account the dependencies between the datasets sampled from the same original problem during evaluation, increased accuracy of the meta-learner means we can be confident the assumption has some merit. That is, if the assumption the densities resemble real-world problem hold. For now we will assume the meta-problem derived here has at least some semblance to real-world data.

Setup

For each problem original problem $\mathbf{D}_i$ (i between 1 and 300) we estimate the densities of the classes using a Parzen density estimator, where the width parameter is set using a maximum likelihood procedure. Note the $\mathbf{D}_i$'s now correspond to different problem types, from which we generate more data, similar to $\mathbf{G}$ and $\mathbf{B}$ before. We assume that the density derived using this approach forms a data-generating process that we could come across in the real world. This way we can generate as much data as is needed to accurately estimate the true error of our classifiers. Similar to our earlier experiments, we now generate 20000 objects from this distribution. $N_{train} = 50$ of these are randomly selected for training and the rest for testing.

To increase the number of datasets, we repeat this procedure 5 times for each $\mathbf{D}_i$. Using the same procedure as before, we use cross-validation to estimate the error on the training set and use the test set to estimate the true error. We are left with a meta-problem with 1500 base-problems (5 from each of our original 300 problems). We evaluate the performance of the meta-learners using a restricted form of leave-one-out cross-validation: in each iteration of the cross-validation algorithm, we do not just leave out one dataset from the meta-problem but all 5 datasets that originated from a single dataset $\mathbf{D}_i$ in our original collection. This corresponds to a total of 300 folds. This is done to correct for the dependence between the datasets with the same underlying problem.

Classifier	Best on
Nearest Mean	236
k-Nearest Neighbor	118
Fisher	243
Quadratic Discriminant	32
Parzen Density	286
Decision Stump (Purity Criterion)	221
Linear Support Vector Machine	164
Radial Basis Support Vector Machine	200

Table 1: List of classifiers the meta-learners have to select between and the corresponding number of datasets they perform best on (class priors). There are a total of 1500 datasets in the meta-problem.

To simulate a real-world setting, we include a larger set of classifiers in the selection. The list and corresponding meta-class prior on the 1500 problems can be found in Table 1. Although set of classifiers will be comprehensive, we have attempted to include different types of classifiers in the mix, while keeping the number of classifiers in check. The table shows the number of problems in the meta-problem on which each classifier has lowest error on the test set. Except for the quadratic discriminant, most classifiers seem to have a similar number of problems where they perform best.

Results

Figure 3 shows the meta-problem of choosing between the Fisher classifier and the Parzen classifier. Compare this to the meta-learning space from our first experiment in Figure 2. Improvements in the classification boundary are more difficult in this real universe due to the large class overlap. In fact, the difference in selection error between the cross-validation selection rule and the linear discriminant meta-learner in the figure is only 0.015.

Selecting from the full-set of 8 classifiers, cross-validation selection has a selection error of 0.695. Using the restricted leave-one-set out cross-validation strategy, a linear discriminant meta-learner gives a selection error of 0.618, a difference of 0.077. A k-nearest neighbor meta-learner does even better with an error rate of 0.605. The learning is not just caused by knowing the most common best classifier, as this strategy gives a selection error as high as 0.809.

By including additional meta-features we can get even more performance improvement. Adding the cross-validation variances (similar to the previous experiment in section 4.3, using 5 repeats), the linear discriminant error becomes 0.599 whereas the k-nearest neighbor error becomes 0.587, a total improvement of 0.108.

Somewhat surprisingly, using unrestricted cross-validation to evaluate the meta-classifiers yields very similar, although somewhat more optimistic error rates for the meta-learners. This might indicate the dependency between datasets generated from the same original problem may not be as troublesome as we feared. Estimating the difference in error rates between the best classifier and the chosen classifier using this unrestricted cross-validation gives an average difference of 0.018 for cross-validation selection. The k-nearest neighbor meta-learner additional meta-features attains an average difference of 0.014. The variances are approximately equal.

5. Discussion

5.1. Simulations

In our first example, in section 4.2, the reason meta-learning in this universe is possible seems clear. By construction, all problems with high Bayes error belong to one type of problem and have one optimal classifier whereas low Bayes error problems have a different type and optimal classifier. So, the meta-learner predicts problems with high errors (of either classifier) are Gaussian in nature and that low error corresponds to Banana type problems. The reason for choosing this extreme example is didactic: the extreme case visually shows the possible difference between the optimal decision boundary and the static cross-validation selection boundary. But, even if we were to reduce the large differences between the error on the Gaussian datasets and the Banana datasets meta-learning still shows improvements over the static rule, although the effects are smaller and visually less distinct.

Whereas the reason learning is possible in this extreme scenario seems clear, it is less clear why learning is possible if we make the problem types more similar. One explanation is that, simply by allowing a more flexible meta-learner instead of a static rule, the meta-learner can adjust for slight differences in the probabilities of different classifiers in different parts of the meta-space.

Another possible explanation may come from the sources of uncertainty that make the cross-validation errors unreliable. The difference between the true error e_{true} and the cross-validation estimate $\hat{e}_{cv}$ determine how reliable the cross-validation estimate is. The variance of this difference has various sources [3, 12]. As [3] also notes, perhaps the most important of these sources we should consider is the use of surrogate classifiers. In order to predict the performance of a classifier trained on n samples, we use k classifiers, 1 for each fold, estimated on only $n - \frac{n}{k}$ samples. For complex classifiers and small n, these k surrogate classifiers can become very different from the actual classifier we are trying to evaluate. Suppose a simple classifier works well on a subset of datasets within our meta-problem. Given the small variance of the errors in the meta-problem, this subset will form a tight cluster in the meta-space whereas for other subsets, were the datasets exhibit high variability (in other words: the surrogates do not form good proxies for the real classifier), the meta-learner becomes less confident. If a new dataset is evaluated that belongs to the tight subset of problems, it will still likely be assigned to this simple classifier, even if small changes to the error estimate occur.

In other words, the method guards against the high variability of the error estimates of complex classifiers. This may also explain the effectiveness of including the variance estimate of the cross-validation errors as a meta-feature in our second experiment. This variance is an estimate of the internal variance referred to above. As this is one part of the full variance [9], it could form a low-biased estimate of the full variance of a cross-validation error.

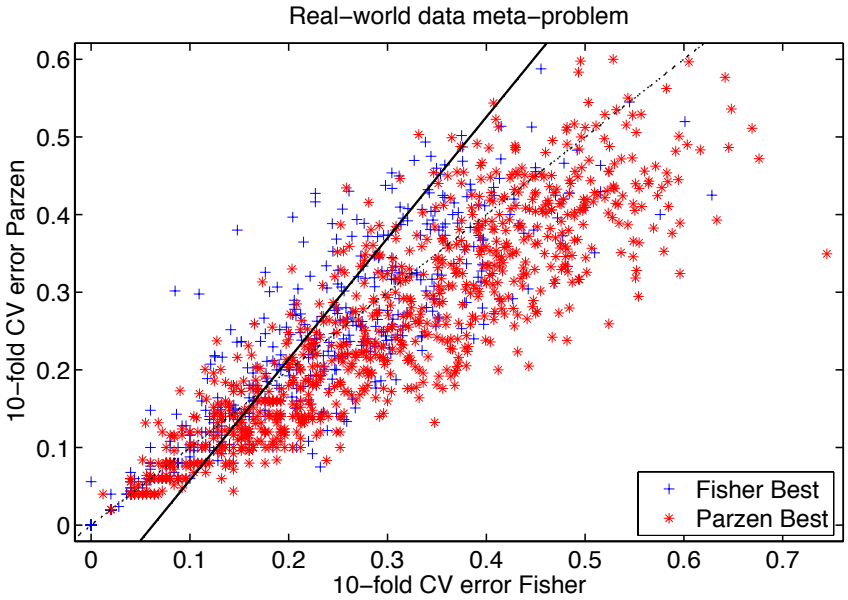


Figure 3: The meta-problem for the pseudo real-world datasets, comparing two classifiers, Fisher and a Parzen density classifier. Shown are the linear discriminant meta-classifier (solid line) and the cross-validation rule (dotted line). Class overlap is much larger in this pseudo real-world problem than our constructed universe in Figure 2.

5.2. Additional meta-features

The finding that additional meta-features, aside from cross-validation errors can increase selection accuracy is interesting in its own right. It goes to show that the error estimates do not fully contain all information present in the dataset. Even simple measures such as the number of objects in the dataset can increase performance. This might be caused by some measures having a lower variance, or even being deterministic. Estimation for these measures is less of an issue and they might still contain some important information for classifier selection. Since we treat classifier selection as be classification problem, it would be expected that adding new features would yield additional increases our ability to discriminate between classes. Even when these features are mostly uninformative, given enough training data the accuracy of a classifier should improve. The size of the effect that we observe of adding the two meta-features is, however, quite surprising. Notably, in the experiment in section 4.3, the cross-validation variance estimate by itself was not particularly successful, but in combination with the number of objects, there is significant improvement in selection accuracy. The combination of these variables is effective because large variances are more likely to occur for small sample sizes. Therefore, knowing the sample size allows the meta-learner to estimate whether a variance is unexpectedly large or small.

5.3. Pseudo real world experiments

The experiments on pseudo real-world data in section 4.4 shows the effects observed in the contrived universes also occur in this collection of datasets. However, the results of these experiments do not directly transfer to real-world datasets. There are several reasons why this may not be the case.

First off, one might wonder how close our collection of problems still is to real-world data. As indicated before, the collection was constructed to exhibit a wide range of classification difficulties one might come across. Furthermore, the datasets contain these difficulties in varying degrees, almost forming a continuum over each complexity dimension. However, apart from the changes made to the seed datasets to obtain to datasets with different levels of complexity, there were only 5 seed problems of which only one was an actual real-world problem. The other four are synthetic datasets. So although the collection as a whole shares properties with real-world data, the datasets themselves are not.

The reason we do treat these problems as resembling real-world problems is because they all have various divergent properties, which is what we expect to see in a collection of real-world datasets as well. Obtaining a similarly sized collection of datasets with divergent properties of real-world data would require a significant amount of work, although repositories such as UCI's machine learning repository [7] offer a good starting point. We therefore settle for a collection which shares these properties, but for a large part lacks the basis of actually coming from machine learning practice.

Secondly, in order to generate enough data to accurately establish the best classifier, we make the assumption that the density estimate made using the dataset correspond to a data generating process that would occur in the real-world. Especially when there are few observations to estimate this density with, this assumption clearly does not hold. The alternative is to use the data not in the training sample (D_i^{train}) selected for learning to act as a test set. However, many of the datasets do not have enough observations to predict ,with extremely high cer-

tainty, which classifier is best. The density sampling approach was therefore chosen to circumvent these issues. Unfortunately, it makes the dataset less like real-world datasets.

Even though this collection does not directly correspond to real-world data, we can still learn a lot about how our approach works on a universe of problems that was not specifically constructed to favor meta-learning and were the problem characteristics are diverse. First of all, in these more complex universes, selecting between just 2 classifiers gives a smaller performance improvement as before. As can be seen in Figure 3, class overlap is quite large in this meta-problem. However, because we are usually interested in selecting between a larger number of classifiers, it is encouraging that we find a substantial improvement when choosing from the set of 8 classifiers we considered. Because the collection of datasets was not constructed specifically for our purposes and the effect is quite large (the selection error is reduced from 0.695 to 0.587) we believe these effects are likely to transfer over to collections of real-world data as well. However, real-world experiments will still be needed to confirm or disprove this.

5.4. Practical application

The goal in this work was to select the single best classifier. One might wonder whether this makes sense in practice. In all experiments, we found reasonable improvements in selection accuracy. However, ultimately, we are interested in getting the lowest error on a given dataset. But, as [24] notes "It is only when two models are almost equally good $[\ldots]$ and distinguishing between them is relatively unimportant that cross-validation begins to falter." However, even though the decrease in classification error may seem small and we need to collect a large number of datasets to train a meta-learner, in some settings this could be worth the added accuracy caused by selecting the best classifier.

A selection strategy that increases the chance of inadvertently selecting a classifier with significantly higher error rate than cross-validation selection may not be useful in practice. In our experiments both the variance and the maximum difference between the lowest classifier possible and the assigned classifier did not differ significantly between cross-validation or meta-learning. However, in general we suggest to use a hybrid strategy where one uses cross-validation selection unless the meta-learner is confident that a different classifier yields better performance. This could be implemented using a regularized version of the meta-learner: if the parameters of the meta-learning are penalized for differing much from those implied by the cross-validation selection rule, the meta-learner will only change the selection if there is high confidence this change is indeed better.

The computational complexity of the proposed approach may be an issue in practice. As indicated before, this work is merely concerned with showing the conceptual point of improving cross-validation selection and not computation time. In practice, the same investment/accuracy improvement tradeoff referred to in the previous paragraph should be made. Since, the meta-learner, once learned, can be used to predict the best class for many new datasets, at least the computational investment of constructing the meta-problem can be spread over a larger number of classifier selections. The possibility of outperforming cross-validation selection presented in this work also opens the door to developing new meta-features that may be more time-consuming than traditional meta-features but less expensive than cross-validation errors.

In order to get the proposed approach to work in the real world several steps will be needed. First of all, a perpetual problem in meta-learning is the lack of availability of large collections of real world datasets. Some projects [7, 16, 21] supporting the publication of datasets might help alleviate this problem, but currently offer still too few real-world datasets. For specific restricted application domains, the possibility of finding a reasonably large set of datasets is more plausible. An example of such a domain, were the small-sample error estimation problem seems to be most studied, is micro-array classification. A second problem is to establish the ground truth for these datasets. This makes the proposed approach more suitable to applications were we have to initially select a classifier based on few samples, while we can later evaluate this choice when more samples become available.

Although we have studied the small sample case here, the results are fairly robust for the size of the dataset. For extremely small n meta-learning becomes difficult due to the fact that most classifiers behave the same anyway. Consider the case of $n = 2$, for instance, then every classifier will have very similar behavior. When n becomes larger enough the cross-validation estimates can be accurately estimated and meta-learning offers no improvement. In our experiment on pseudo real-world data happened from $n = 200$ and higher. For n between 20 and 200 however, improvements due to meta-learning seem possible. We believe these are not unreasonably small sample sizes for practical problems.

The precarious problem is that we have very limited data not just to train the classifier on, but also to estimate the errors with. One might claim the proposed solution is impractical given the need to collect many datasets. The alternative is to (1) come up with a lower variance estimator of the error or (2) obtain extra data to estimate the error. Work on (1) has, as of yet, not yielded methods that significantly improve upon the variance cross-validation errors. Adding more data (2) for error estimation is not an option: we would want to use the new data to improve our classifiers. Also, with more data, the performances of the classifiers change. The solution proposed here then, is to use this larger data that we *do* have in earlier datasets, to aid in the decision on the dataset at hand.

5.5. Methodological interpretation

We have considered classifier selection as a classification problem in its own right. The reason for doing classifier selection in the first place is an assumption that different classification problems may need different classifiers. If we really believe this to be true, this assumption would also hold for the meta-problem, which we also treat as a classification problem. Therefore we should allow different meta-classifiers. Yet cross-validation based selection is a single static, untrained meta-

learner. In other words, assuming classifier selection needs to be done and only using the cross-validation selection rule for this selection leads to a contradiction.

Another interesting point related to this, is that as the meta-problem is a classification problem in itself, it could be parameterized as a vector in the meta-space as well. So the meta-problem can contain itself! In a similar vein, a meta-learner can give a prediction of which meta-learner to use. Although interesting, we have not considered this possibility because our universes are constructed and not simply the set of all problems that can occur in practice, of which the meta-problem would be one.

5.6. Future work

This work has presented a conceptual point which we think helps pave the way to more developments on the topic of classifier selection and meta-learning. In particular, it shows meta-features can be constructed that may not be more computationally efficient than cross-validation errors, but increase the accuracy of a selection procedure. Even adding simple features, like the number of objects in the dataset, considered here, may offer these improvements.

The most important work remains to be done on both showing these results truly hold on real-world data and using them in a practical setting. Given the problem of accurately establishing the ground truth, some practical applications may also lie outside the field of classification. A promising application could be early model selection for time-series. Here the ground truth can easily be established as more data comes in.

6. Conclusion

We have demonstrated that, at least on a conceptual level, cross-validation classifier selection is not necessarily the best classifier selection strategy, especially in the small sample setting. By treating cross-validation selection as an untrained meta-learner, there is promise of improving classifier selection by using trained meta-classifiers. Adding additional meta-features to the meta-problem, apart from cross-validation errors, may provide additional improvements in selection accuracy.

An experiment on a collection of datasets that resembles some properties of real-world data, shows these results may hold on real-world data as well. More experiments on actual real-world data are needed to verify these results. There are some practical concerns in applying this approach practice. The most important are the collection of large numbers of real-world datasets, the problem of establishing the correct meta-classes and the computational time. In applications were a classification method has to be selected early in the process, before much data is available, this method could be useful.

Overall, the results presented here motivate further development of meta-learning, not just as a computationally-efficient way of doing classifier selection, but potentially as more accurate as well.

References

[1] Aha, D.W., 1992. Generalizing from case studies: A case study, in: In Proceedings of the Ninth International Conference on Machine Learning, Morgan Kaufmann. pp. 1–10.

[2] Bensusan, H., Giraud-Carrier, C., Kennedy, C., 2000. A higher-order approach to meta-learning. Technical Report. Bristol, UK, UK.

[3] Braga-Neto, U., Dougherty, E.R., 2004. Is cross-validation valid for small-sample microarray classification? Bioinformatics 20, 374–380.

[4] Brazdil, P., Gama, J., Henery, B., 1994. Characterizing the applicability of classification algorithms using meta-level learning, in: Bergadano, F., De Raedt, L. (Eds.), Machine Learning: ECML-94. Springer Berlin / Heidelberg. volume 784 of *Lecture Notes in Computer Science*, pp. 83–102.

[5] Dougherty, E., 2001. Small sample issues for microarray-based classification. Comparative and functional genomics 2.

[6] Duin, R.P.W., 2000. Prtools - version 3.0 - a matlab toolbox for pattern recognition, in: Proc. of SPIE, p. 1331.

[7] Frank, A., Asuncion, A., 2010. UCI machine learning repository.

[8] Fuernkranz, J., Petrak, J., 2001. An evaluation of landmarking variants, in: Proceedings of the ECML/PKDD Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning (IDDM-2001), pp. 57–68.

[9] Hanczar, B., Dougherty, E.R., 2010. On the comparison of classifiers for microarray data. Current Bioinformatics 5, 29–39.

[10] Ho, T.K., Basu, M., 2002. Complexity measures of supervised classification problems. IEEE Trans. Pattern Anal. Mach. Intell. 24, 289–300.

[11] Hoekstra, A., Duin, R.P.W., 1996. On the nonlinearity of pattern classifiers. Pattern Recognition, International Conference on 4, 271.

[12] Isaksson, A., Wallman, M., Göransson, H., Gustafsson, M.G., 2008. Cross-validation and bootstrapping are unreliable in small sample classification. Pattern Recogn. Lett. 29, 1960–1965.

[13] Kalousis, A., Theoharis, T., 1999. Noemon: Design, implementation and performance results of an intelligent assistant for classifier selection. Intelligent Data Analysis 3, 319 – 337.

[14] Kohavi, R., 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection, in: Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. pp. 1137–1143.

[15] Krijthe, J.H., Ho, T.K., Loog, M., 2012. Improving cross-validation based classifier selection using meta-learning, in: International Conference on Pattern Recognition. Forthcoming.

[16] Liang, P., Abernethy, J., 2009. mlcomp.org.

[17] Lindner, G., Ag, D., Studer, R., 1999. Ast: Support for algorithm selection with a cbr approach, in: Recent Advances in Meta-Learning and Future Work, pp. 418–423.

[18] Macià, N., 2011. Data complexity in supervised learning: A far-reaching implication. Ph.D. thesis. La Salle — Universitat Ramon Llull.

[19] Macià, N., Ho, T.K., Orriols-Puig, A., Bernadó-Mansilla, E., 2010. The lansdcape contest at icpr'10, in: Devrim Ünay, S.A., Çataltepe, Z. (Eds.), Contests in ICPR 2010, Springer, Berlin, Heidelberg. pp. 29–5.

[20] Mansilla, E., Ho, T.K., 2004. On classifier domains of competence, in: Proceedings of the 17th ICPR, pp. 136 – 139 Vol.1.

[21] Ong, C.S., Braun, M.L., Sonnenburg, S., Henschel, S., Hoyer, P.O., 2009. mldata.org.

[22] Peng, Y., Flach, P.A., Brazdil, P., Soares, C., 2002. Decision tree-based characterization for meta-learning, in: Bohanec, M., Kasek, B., Lavrac, N., Mladenic, D. (Eds.), ECML/PKDD'02 workshop on Integration and Collaboration Aspects of Data Mining, Decision Support and Meta-Learning, University of Helsinki. pp. 111–122.

[23] Pfahringer, B., Bensusan, H., Giraud-Carrier, C.G., 2000. Meta-learning by landmarking various learning algorithms, in: Langley, P. (Ed.), ICML, Morgan Kaufmann. pp. 743–750.

[24] Schaffer, C., 1993. Selecting a classification method by cross-validation. Machine Learning 13, 135–143.

[25] Vilalta, R., Drissi, Y., 2002. A perspective view and survey of meta-learning. Artificial Intelligence Review 18, 77–95.

Additional Notes

Chapter 2

Introduction to the Notes

The topic of the main thesis is whether the accuracy of classifier selection using a simple, static rule can be improved using techniques from meta-learning. The work is the culmination of my research over a nine month period focussing on the justification of untrained classification rules and the role of meta-learning.

The original goal, which took up the first 4 months, was to use data complexity measures to study whether the simple, untrained, combination rules used in classifier combination are justified. This was done by comparing the data complexity measures for the original dataset, with the data complexity of the outputs of the classifiers that were to be combined. My hypothesis was that as the static combination rule is so much simpler than the classifiers used, data complexity in the latter case should reflect this. Once we figure out how this complexity changes given a set of classifier, we may be able to understand and predict whether to use a more complex combiner or, for instance, add an extra layer in a neural network.

Unfortunately, data complexity measures turned out not to be suitable (yet), for this kind of analysis. This was mostly due to high variability in estimation, making (non-parametric) testing of the differences difficult, but also a lack of interpretation of the data complexity measures themselves. After coming up with the idea presented in the main text, I abandoned this line of research, but my personal notes on this work can be found in this appendix.

The text and figures in these notes come directly from my research log with some added structure. As such, it may have random comments dispersed through the text. It is provided to give some indication of the topics that were explored during these nine months and contains the results of some experiments that did not end up in the final thesis.

Chapter 3 gives some background and thoughts on data complexity measures, also referred to in the main text. Chapter 4contains some results and notes on using these data complexity measures to characterize the changes caused by classifier combination.

Chapter 5 provides some additional text and results related to the thesis topic that did make it into the main text. In chapter 6 some comments are provided on the implementation of the experiments. Chapter 7 contains the original thesis topic proposal, which is clearly different from the direction this work ultimately took.

The final chapter of this appendix contains a short, four-page paper on this thesis work. It was submitted and accepted to be presented at the International Conference on Pattern Recognition 2012.

Chapter 3

Notes on Data Complexity measures

3.1 Introduction

The measurement constructs used in here are the Data Complexity measures first introduced by [7]. They are a set of 14 measurements (originally 12) on the data that describe different geometrical complexities of the data. Ho considers three types of intrinsic complexity of a classification problem. The first is class ambiguity were we are unable to distinguish between two classes for certain objects because they map to the same objects in the given representation. One clear example is the letter 'l' and number '1' (one) using the same image in many fonts. Given only this image (and not the surrounding context, for example) we are unable to distinguish between the two classes.

The second difficulty is the sample sparsity, especially in high dimensional spaces, which makes it difficult to train any classifier. Interesting but not surprising, the sample sparsity also affects our ability to estimate the complexity of a problem accurately. We will study this problem further below, since the number of points available to estimate the data complexity will be reduced when training transformations.

The third difficulty is the complexity of the boundary, this can range from linear with large margins between classes to highly non-linear with small margins. Data complexity tries to estimate these geometrical properties of the problem.

3.2 Description of the original data complexity measures

For this study I used the implementation of the data complexity measures by [10]. They are sightly different from the original measures by [1] to improve performance on large datasets while keeping with the original purpose of the measures. She also added two extra measures, F1v and F4, which are slightly more advanced versions of F1 and F3 to

take into account the collective effect of the features instead of looking at each feature separately.

An overview of the measures is given in 3.1. A short overview is given below for each of the four types (these types account for how they are constructed, not what they measure). For a thorough description of the measures I refer the reader to [10].

3.2.1 F: Feature measures

The five feature measures try to estimate the complexity introduced by overlap of the classes. The Fisher Discriminant Ratio (F1) does this by dividing the difference between the class means by the sum of the class variances. Calculating this ratio for all features and taking the maximum gives a high value when at least one dimension gives a separation of the classes. Note that we do not take into that multiple dimensions combined might give good separation. Think of a diagonal decision boundary in a 2 dimensional space. The directional-vector maximum Fisher discriminant does allow for this case. The other three measures (F2-F4) describe the overlap of the classes on each dimension.

3.2.2 L: Linear classifier measures

The measures describe properties of a linear classifier trained on the dataset.

3.2.3 N: Non-linearity measures

The measures describe properties of a non-linear classifier trained on the dataset.

3.2.4 T: Descriptive measures

Measures describing the density of the sampling.

3.3 Potential issues

1. Can we expect to predict *generalization* performance from features of the *data*? Does the complexity of the data (mind you, not the problem) tell us something about solving the problem? One problem is that to determine whether a given problem is complex we need a large number of objects to find this out. If we have few samples, we can only say that we do find the problem is complex.

Do we have enough information contained in the samples to make this prediction. A simple experiment can shows why this might be a problem. Suppose we take a

#	Label	Desciption	Range	Expected
1	F1	Maximum Fisher s discriminant ratio	$[0, \mu_k]$	+
2	F1v	Directional-vector maximum Fisher s discriminant	$[0, \mu_k]$	+
3	F2	Overlap of the per-class bounding boxes	$[0, 1]$	−
4	F3	Maximum (individual) feature efficiency	$[0, 1]$	+
5	F4	Collective feature efficiency ratio	$[0, 1]$	+
6	L1	Minimised sum of the error distance of a linear classifier	$[0, \frac{n}{n_t}]$	−
7	L2	Training error of a linear classifier	$[0, 1]$	−
8	L3	Nonlinearity of a linear classifier	$[0, 1]$	−
9	N1	Fraction of points on the class boundary	$[0, 1]$	−
10	N2	Ratio of average intra/inter class nearest neighbour distance	$[0, +\infty]$	−
11	N3	Leave-one-out error rate of the one-nearest neighbour classifier	$[0, 1]$	−
12	N4	Nonlinearity of the one-nearest neighbour classifier	$[0, 1]$	−
13	T1	Fraction of maximum covering spheres	$[0, 1]$	−
14	T2	Average number of points per dimension	$[0, n]$	Controlled

Table 3.1: Overview of the 14 data complexity measures adapted from [10] with the minor addition of adding the last column containing the expectation of the behaviour of different measures as we simplify the problem using a transformation. + means the measure is expected to increase in value as the problem becomes less complex, − a decrease.

2. Computational complexity Phahringer actually argues for using maximum complexity of $nlogn$ to be useful for meta-learning

3. Lack of discrimination Another potential problem is that two problems that are clearly different map to the same point in the complexity space. The reason can be structural or happen after generalization. The first is best exemplified by the data complexities of two linearly separable problems but where the separating hyperplane has a different orientation. For instance, the data complexity of a horizontal boundary and vertical boundary gives the same data complexity. This is not necessarily a problem, as long as we do not expect to be able to train our classifier on one of the problems and use it on the other problem, even tough it has the same data complexity. The second problem is might be more immediate. Given the no free lunch theorem, if we are given a training set, performance beyond this training set is arbitrary if we take into account all possible generalizations. So, if we find that for given datasets a certain data complexity is associated with good performance of a certain classifier, there will always be datasets with the same complexity that do not show this result. This problem becomes especially important when generating datasets to match complexities with performance of classifiers: the artificial datasets might not represent problems that we encounter in the real world. So even though particular generalizations work well for certain complexities on this artificial data, the processes that generated them might not resemble the real problems encountered in practice.

3.4 Stability and bias of the measures

Because these measures have to be estimated from the data. Especially important because we cannot use all the data in our experiment to estimate the data complexity, we need a training set for the transformations.

3.5 Implementation problems

When we have 0/1 outputs the overlap measures do not make much sense, but this might actually reflect the behavior of certain classifiers.

Randomness N4 measure?

3.6 Discussion

3.6.1 What to remove

Selecting the most relevant measures from the set of data complexity measures The large number of data complexity measures (14 in all) make the tool not only expensive to compute but also hard to interpret. It would therefore be useful to remove those measures that are least effective to our cause or find replacements that are more readily interpretable. This could be seen as a feature selection problem with an added twist. The twist is that we are not only interested in the performance gain on a particular classifier but also have to take into account interpretability.

3.6.2 What to add

One avenue I have been exploring is turning the assumptions made by different classification algorithms into measures using well known statistical tests. One example is the assumption of Gaussian classes made by (for instance) a linear discriminant classifier. Using Mardia's test for multivariate normality as a meta-descriptor I found that it does allow for some discrimination between cases where a classifier based on these assumptions works well or not. However, several measures contained in the data complexity also allow for this discrimination. Testing for multivariate normality is therefore likely already contained in the data complexity measures but a direct measure might be easier to interpret. It seems likely, however, that the hypothesis test needs larger samples to allow for discrimination than some of the data complexity estimates. Perhaps trying to measure to what extend assumptions of classifiers hold is also not descriptive enough to determine where we lack good classifiers. We will likely only be able to determine that the space described by classifier assumptions is well covered by the classifiers who make

these assumptions, but will be unable to find assumptions that we might have missed in building classifiers.

The problem is that any measure that relies on the difference between two classes could be used as a classifier. So in a sense, we will never get around using classifiers as the descriptors, unless we forget about the labels altogether. Maybe based on statistical notions. For instance based on parametric tests. Maybe based on common assumptions underlying classifiers? Problem might be you will not catch the case when we have not considered certain useful assumptions. We will not be able to show this gap in our set of available classifiers.

Chapter 4

Notes on Data Complexity in Classifier Combination

In supervised learning, classifier combination has proven to be an effective way to improve performance on many classification tasks. This idea of combining the output of many classifiers to form a new classifier that outperforms any of its component classifiers has been further developed into ensemble methods such as bagging. In these approaches we generally use the same classifier but train multiple times by resampling our training data. However in all these approaches we have to solve the problem of combining the classifiers. Since each classifier in the ensemble generates an output for a training sample, our original input data for the problem has been transformed from the original d dimensions to c dimensions, where c is the number of classifiers in the ensemble. The combination is often done using very simple rules, such as a majority vote. One way of interpreting this approach is that we assume that our transformation has decreased the difficulty of the problem. So much so, in fact, that we can use a simple untrained rule instead of a trained classifier.

It is clear that when this analysis can be done for the transformed spaces defined by training classifiers, we can use the same approach to study the effect of any transformations. The total classification task could be interpreted as a transformation of the original d dimensional input vector to a 1 dimensional space (or k dimensional, depending on the representation). This is important because many algorithms can be interpreted as applying various transformations on the problem space. Ensemble methods, then, are an example of a two-transform procedure. Neural networks or deep learning architectures use many layers (transforms) of the data. But also preprocessing steps such as feature selection or extraction and kernels are clear examples of transforms of the original problem space to a new space that is, hopefully, easier to learn.

The question now becomes: Can we make a quantitative prediction of the change a transformation is going to have on the data complexity of a problem? If we can, how do

we interpret this shift in qualitative terms?

Unfortunately, determining the inherent difficulty of a classification problem is no trivial task. If fact, a well known statistic to describe this problem difficulty, Kolmogorov complexity, is proven to be indeterminable. This gives some indication that trying to capture the complexity in one statistic is too optimistic. Data Complexity [7] offers an alternative by using a compilation of various statistics to estimate different types of complexity inherent in the data. This set of measures has mostly been applied to determining the domains of competence of different classifiers: towards what types of difficulties in the data are different classifiers well suited. In this work I use these Data Complexity measures (DC-measures) to estimate the complexity of problems measured on the dataset.

Although a promising instrument, the data complexity measures have been recently introduced and therefore not been applied broadly and their behavior is not thoroughly described. Although a fine implementation has been developed [13] some features are still lacking. Most notably, it currently handles only two-class real valued problems.

First, introduce the data complexity measures. Then I discuss how to evaluate whether a transformation caused any changes to data complexity by discussion various methods from parametric to non-parametric methods such as bootstrapping and permutation tests. After we have derived a usable test we get back to the analysis of various transformations of the original problem, such as combining multiple classifiers or using a multi layered approach.

4.1 Bagging

In this section I describe bagging, a way of building an ensemble of a single classifier using bootstrapping. Traditionally, the resulting ensemble is combined to reach a single prediction for each input using averaging over the classifiers in the ensemble. Although this has a nice intuition behind it, it is not immediately clear this is the best way to do the combination. The main questions is then whether other combiners could improve predictions and for what base-classifier which combiner works best. Since this behavior is likely dependent on the problem itself, I use data complexity to characterize when which combination technique works. Moreover, by looking at the data complexity of the *outputs* of the base-learners we might understand *why* a particular combiner works in that situation. In other words: what transformation in the data complexity does do the base-learners cause that leads to one combiner being better than another?

Bagging, or bootstrap aggregation is a technique to improve the accuracy of a *single* classifier first introduced by [3]. The intuition is that if we would be able to resample our given data D from the original distribution many times, we would be able to estimate any classifier accurately. However, all we have is the given data D. Breiman's idea is to approximate this resampling of the theoretical distribution $P(x, y)$ by resampling with

substitution from the available data D (bootstrapping). We train our single classifier multiple times on each of the$B$ bootstrap samples. By averaging over the resulting B classifiers, we have a better estimate of the classification boundary.

As Wolpert [16] notes, however, its effectiveness is purely based on an intuitive argument. We do not know, for example, how much the real distribution and the emprical distribution differ. In other words, the no free lunch theorem applies equally to bagged classifiers as it does to its unbagged counterparts. The main argument for its effectiveness,then, is an empirical one. Various studies [16, 3] have shown bagging leads to significant improvement of classifier accuracy, while performance degrades in very few cases.

This empirical basis does conjure the question when the approach is effective. Can we know, beforehand, for which base-classifiers this approach is most effective? [3] suggests the approach is only useful when dealing with unstable classifiers. In these cases, the classifier has high variance and while training the classifier on the whole dataset we might be 'unlucky' and get bad performance. Bootstrapping offers a way of generating many datasets. The variance of the mean performance of the classifiers trained on these new datasets will thus be lower than the variance when only training the classifier once. However, as Breiman notes, there is a trade-off since we use less unique data in each of the bootstrapped classifiers than we have available in our complete dataset. Therefore if we have a stable classifier, performance might actually degrade because we are unable to reduce the variance of our classifier but at the same time effectively use less data to train it.

Even if this explanation holds, this merely provides a motivation for the how the mean combiner works by reducing the variance of the classifier. It might be very well posible other ways of combining the bootstrapped base-classifiers improve accuracy even for fairly stable classifiers. Others have studied this opportunity in more detail. [16] develops several methods to extend bagging with stacking. [15] compares different combiner rules to use when bagging and boosting. The default combiner choice for classification, the majority vote, has the worst performance of all combiners used in the study. They find that it is mostly outperformed by the weighted majority rule. Other rules, such as the product rule

Duin [5] gives an intuitive discussion of the issue of trained vs. untrained (fixed) combiners. He concludes fixed combiners can only be expected to work well under strict conditions. In particular it is important the base-learners are not overtrained and have correct confidences as their output. One way to achieve this is by normalizating the base-learner outputs using a seperate training set, after which we can apply a fixed combiner with some confidence. For instance, we can use the training set to find the optimal parameter of a logistic function that maps the classifier output to a confidence. Another possibility is to use a trained learner. This trained combiner might correct for misguiding confidences of the base-learners. This does present the question how we train this trained combiner. Duin offer two alternatives: use the whole training set for trainging

both the base-learners and the combiners or split the training set up. In the former we need to be careful not to overtrain the learners, while in the latter, we do not have to worry about these issues.

Kuncheva & Whitaker [9] study the effectiveness of using various diversity measures of an ensemble to explain en predict the accuracy improvement. The diversity measures correlate well, meaning the different researchers that have come up with these measures do share a similar concept for the defintion of diversity. The measures also explain the accuracy improvements well in various simulation experiments. However, when using the measures on real world data, their usefulness is less aparent. Correlation between these measures and and the accuracy improvement is only ????. One explanation is that in this real dataset, the possible improvement was very small. However, one does wonder wether current measures of diversity

4.2 Exploring bagging accuracy

The purpose of this experiment is to give us an idea which classifiers and combination rules are interesting to study further. To do this we evaluate a large number of classifier and combiner combinations and study how often each combiner improves the accuracy of the base-learner. It also provides a general sense of the behavior of the combiners: does it always increases accuracy slightly or does it exhibit large deviations in its performance depending on the problem? Especially in the latter case the questions of when to apply a specific combiner becomes relevant.

4.2.1 Setup

To do this type of analysis for a large number of base-classifiers and combiners we have to make concessions in terms of thoroughness to keep the computational demands in check. This means that for each of the 300 problems, instead of doing cross validation, we randomly split the problem once in a training (70%) and testing (30%) part. For each base-classifier we use the training set to build 50 bootstrapped classifiers. For the trained combination rules we then use the same training set to train the rule. For example, when using a linear discriminant as the combination rule, we take the output of the bootstrapped classifiers when given the training set as input and use the this output as the training set for the linear discriminant. Even though the single split might provide high variance in the result for a single experiment, the hope is that when done on each of the 300 test problems, these problems cancel each other out. So, even though the results from single experiment should be considered fairly unreliable, the qualitative conclusions (whether a classifier or combiner is worthy of further study) will hopefully be useful to guide us in further analysis using the data complexity.

4.2.2 Results

No degradation when using naive bayes as the base-classifier or when using the product rule as the combiner. Trained fails less often than untrained.

4.3 First experiments

In this section I describe the initial results from some initial experiments to study whether we observe any changes in data complexity when we use the output of simple classifiers to form a new transformed space, as opposed to the original space that the simple classifiers work on.

4.3.1 Implementation

The problems used in these experiments are the 300, two-class, problems that form the *S1* set of problems of the ICPR2010 classifier landscape competition [11]. They are a diverse selection of problems that were generated from 5 original problems (4 artificial and one real-world dataset) using the techniques discussed in [12] to span the a large part of the complexity space. I chose this set of problems because they are easy to use (only real-valued attributes), they supposedly span the complexity space and there are sufficiently many to draw some conclusions.

During the initial stages I used all samples to train the transformation (classifiers) and transformed these same samples using the trained transformations. However, this might make the results overly optimistic since using the trained transform on the training data will make the problem seem less complex than it will be if the transform is used on new data. The results presented here therefore use 50% of the data of each problem to train the transform and the data complexity is calculated on the transform of the other half of the data. However, more thought should be put into this training vs. test set: for one the datasets were generated by selecting samples from the original problem so as to attain the right complexity. So sampling might affect the data complexity, causing it to be different in the training set than in the test set.

The code used to calculate the data complexity measures on a given dataset is the open source implementation by [13] using a custom interface to Matlab. This interface seems to work except for one dimension (N4) where subsequent runs give slight fluctuations in the measure. These fluctuations do not occur when using the command line interface. Given the number of datasets used here, this does not cause big problems, but it should be resolved.

Since the data shows very extreme behaviour, it is unwarranted to make the assumption the complexity measures follow a normal disribution. To test whether there is a change between the complexiy measures between the original problem and the transformed

problem, we therefore use a sign test which assumes the data comes from any arbitrary continuous distribution. Note that even though the measures are continuous, it is still worth checking whether they all also behave continuously. Some might only be able to assume certain values by construction. Also the assumption of independence of datasets might not hold here, since some datasets were originally constructed usign the same dataset. The sign test, evaluates the null hypothesis that the median of the differences between the complexity measures is 0. To do this we count the number of samples where the difference is positive and use the binomial distribution to calculate the significance.

4.3.2 Random Forest spaces

To evaluate the data complexity of the space spanned by the output of the individual trees in a random forest I used Matlab's TreeBagger function. This implementation is based on Breiman's desciption of a random forest (see [3]).

Description

The random forest is an ensemble of decision trees, combined using a majority vote considering all these trees. In Breiman's approach each tree is trained using a bootstrapped sample from the trainset.

Complexity on a large tree

To find out whether these transformation have any effect I first trained 100 decision trees one each dataset and then transformed said dataset by taking the output of these trees for each example. The mean data complexities of these transformed sets can be found in table 4.1. On 9 of the 13 relevant dimensions (the 14th is simply controlled by choosing the number of trees), we find the expected effect. For example, on the first dimension, 300 of the 300 datasets showed an increase in this measure after the transformation, which for this measure means a lower complexity.

One dimension is undecided (dimension 6) since, for about half of the datasets it increases and for about half it decreases. This may not be a problem if the increases are negligible while the decreases are large. However, the means change does not suggest this is the case. It is interesting that this dimension often shows insignificant changes in other experiments as well.

Three of the dimensions (3-5) actually show the reversed effects what we would hope to do. These show that the problem has become more complex. The reason seem to be that these three measures all depend on the class overlap. Since the outputs of the trees in this experiment can only attain one of two values (one of the two classes) a tree only has to make one wrong classification for it for the classes to fully overlap on this

dimension. The outcomes for these dimension might therefore be because of the nature of the outputs of our transform. This may be a motivation for using classifier confidences to make the problem of classifier combination easier.

Training vs. Test set

In the previous experiment, we have used all the objects in a dataset to train the transform. We than use this to transform these points used for training. This will likely produce too optimistic outcomes: of course the transformation is going to be effective on the training data, but what about new data points. To tackle this, we split the datasets in half to form a trainingset and a testset. The training set is only used to train the classifiers used for transformation, while the testset contains the points that will be transformed and on which we calculate the complexity measures. The results are shown in 4.2. We find that indeed, the previous experiment was too optimistic: the results are much less clear in this case. However, the only dimension that has a qualitatively different interpretation is dimension 10, were we are less sure whether it indeed becomes simpler. Even though these estimates are much more conservative, it is does not represent how we usually treat these small datasets. When evaluating accuracy, for example, a better approach is to use cross-validation to use the most examples as possible. The problem might even be bigger here since we know leaving out many objects lead to different complexity measures (this is how the datasets were generated) and some measures even contain the number of objects! It shows that we should get a better understanding of how the apparent complexity that we measure depends on the sample size and what technique is most suitable to get a realistic estimate of the real data complexity.

Increasing the number of trees

The results for the data complexity using 2, 10, 25 and 50 trees are shown in 4.1. For 2, N2 might not be stable.

4.3.3 Layering

Found that when using the training sample complexity, layering helped, but effect seemed to fade when splitting test and training set.

4.3.4 Open Problems

An important problem for the evaluation is how the data complexity of the new problem should be estimated: can we use all data to train the transformation and evaluate the data complexity estimate on this transformation. It seems more realistic to use a different part of the data to calculate the transform to avoid being overly optimistic. There are

Original	Transformed	Difference	Number Positive
1.1704	28.9736	27.8032	300
2.5567	942.3629	939.8062	293
0.07337	0.81	0.73663	243
0.33591	0.13287	-0.20304	38
0.61272	0.18711	-0.42561	30
0.48402	0.48986	0.0058352	176
0.21865	0	-0.21865	0
0.42531	0	-0.42531	0
0.33895	0.0056746	-0.33327	0
0.70484	0.24747	-0.45737	0
0.21804	7.4571e-06	-0.21803	0
0.17789	0	-0.17789	0
0.99703	0.81116	-0.18587	26
28.7778	3.8985	-24.8794	0

Table 4.1: The mean data complexity measures from the original problem and the transformed problem when using 100 decision trees. The same data used for training as also the data that is transformed.

Original	Transformed	Difference	Number Positive
1.2678	3.4256	2.1578	203
3.0237	222.046	219.0223	295
0.040751	0.69	0.64925	207
0.40373	0.048657	-0.35507	4
0.7605	0.20384	-0.55666	13
0.49652	0.39552	-0.101	127
0.23556	0.075939	-0.15963	0
0.45136	0.17322	-0.27814	0
0.36472	0.24655	-0.11817	35
0.77025	0.74542	-0.024832	109
0.24633	0.15438	-0.091951	32
0.15436	0.03807	-0.11629	21
0.99575	0.99116	-0.004583	56
14.3472	1.944	-12.4031	0

Table 4.2: The mean data complexity measures from the original problem and the transformed problem when using 100 decision trees. Half the data is used for training the classifiers and half to transform and calculate the complexity measures.

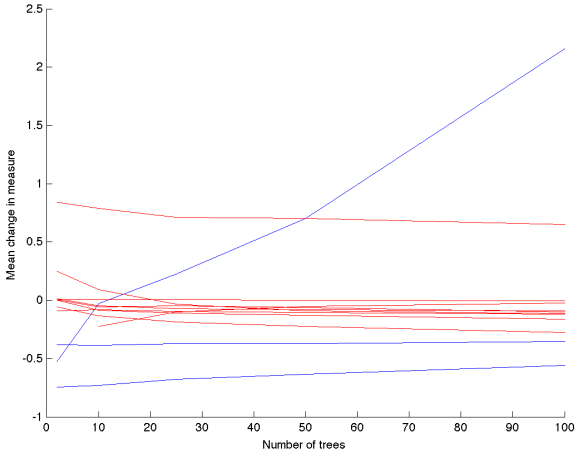
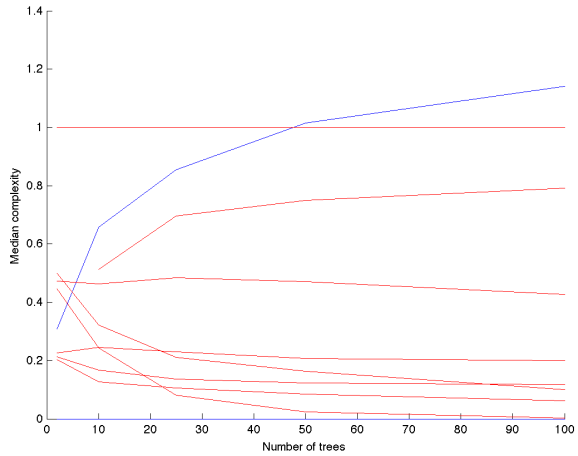


Figure 4.1: The mean change is the different complexity measures. F1 has been left out because of the different scaling. Red lines are measures that should become lower as complexity decreases, and blue lines are measures where an increase corresponds to lower complexity. Note: the lines are sampled at: 2, 10, 25, 50, 100 trees.

Figure 4.2: The means of different complexity measures for a different number of trees. F1 has been left out because of the different scaling. Red lines are measures that should become lower as complexity decreases, and blue lines are measures where an increase corresponds to lower complexity. Note: the lines are sampled at: 2, 10, 25, 50, 100 trees.



some possible alternatives to the current split: using leave-one-out or cross-validation posteriors as the new space or using cross-validations splits to get different estimates for the data-complexity. The former seems more promising since we get to keep the same number of data points

As noted earlier, the N4 measure seems to show some random fluctuations when using the Matlab interface. While this might not be a problem for the results, it is still worth investigating why this is happening. Another improvement to the Matlab interface would be to change the underlying C code to deal with the problem matrices directly. Currently the problems are first written to memory in ARFF format and then loaded by the C-code. However, the performance savings might not be worth the time needed to change the code.

More classifiers cause F4 to become insignificantly different from zero.

When layering classifiers the complexity seem to get better even after the first level, effect becomes unstable if we add layers. Now we need to study what happens if we move from using the transformation on the trainingset to using a test set. After doing this, layering does not seem to work.

Out of sample approach: does this also hold for unsupervised feature reduction? Studying effect of N. Is lower complexity really better?

Chapter 5

Notes on Improving Cross-validation

5.1 Introduction

One comment that is often mentioned to question the usefulness of meta-learning is that it is easier to train all the classifiers you have at your disposal and use cross-validation to select the best one. At first it seems hard to rebut this statement. Better understanding of the behavior of classifiers could be another reason for doing meta-learning, but this has not been very successful as of yet. But what if we could actually do better than cross-validation? Assuming we have all the information of previously encountered problems as well as the given dataset we should be able to outperform cross-validation which relies on just the given dataset. Doing an experiment resampling many times from a gaussian and a non-gaussian problem, I find that meta-learning does outperform 10-fold-cross-validation. One reason is that cross-validation does not take into account its performance on previous problems. In a sense, cross-validation is an untrained rule while meta-learning is a trained rule. In the figure (see figure) we can see the 'cross-validation classifier' forming a diagonal line in the space (not shown) does worse than a trained rule (shown) on the red class. The trained rule using the cross-validation estimates for the two different classifiers as inputs gives equal performance to the meta-learner, however, the latter uses merely two non-classifier dependent complexity measures as input.

There are several reasons why we cannot conclude we can outperform cross-validation in general. First of all, in this example, the choice is merely between two classifiers, a nearest mean classifier and k-nearest neighbor. Secondly, in this example we could use an accurate estimate of the real error of the classifiers in the meta-training phase, because we have access to the original problem to generate more testing data. In practice we do not have this. In effect the approach outlined above works because we know the distribution of problems in the universe (which is very small in this example, just one type of gaussian and one type of non-gaussian problem). Meta-learning might thus be expected to work well when we are able to get reliable estimates of distribution of

problems out there in the real universe. Another interesting observation is that the meta-learning approach does show large improvement over cross-validation in selecting the best classifier for a dataset but that the difference in average performance by the classifiers chosen by both approaches is very small. This suggests cross-validation does at least work for many cases were the difference between competing classifiers is large.

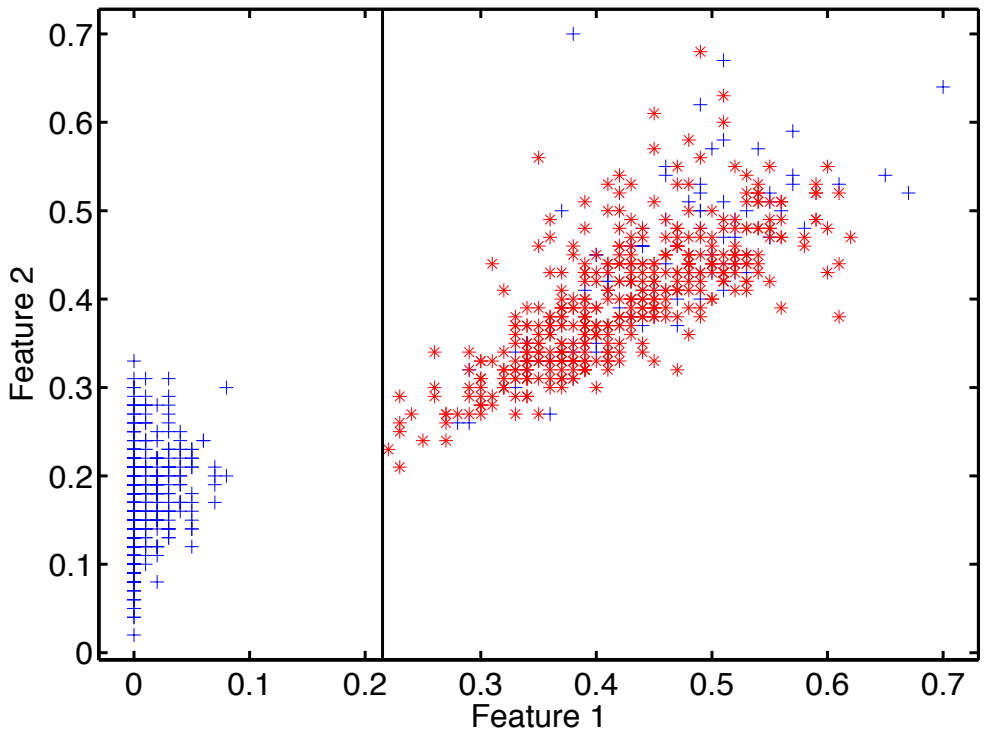


Figure 5.1: An example universe where meta-learning can outperform cross-validation. The features are cross-validation errors of two classifier, while the solid line is a Support Vector Machine trained on this meta–problem. This clearly offers a better separation of the meta-classes than the diagonal line representing cross-validation selection (not shown)

The first experiment is an illustration of the existence of a way to improve cross-validation under controlled conditions. The second experiment studies the effect on real world data.

5.2 Small simulated universes

Suppose we have a universe containing two kinds of problems, gaussian and non-gaussian, split evenly over both cases (see figure for an example for each case). Our toolbox

contains merely two classifiers: the nearest mean classifier and the k-nearest neighbor rule (where k is chosen using cross-validation). Now given a small dataset, which of the two classifiers should we choose to get the best performance? One approach is to use 10-fold cross-validation, and use the classifier with the lowest estimated error, a technique often used in practice. The alternative that I propose here is meta-learning and I will compare the properties of its predictions of the classifier to use with that of cross-validation. Half of the problems are generated using a base of two gaussian classes where the distance between the classes is uniformly chosen between [0,1]. The other half is similar except that the classes do not follow a gaussian distribution but rather a banana shape (see figure). We generate 500 problems from each of the types and select a small subsample of these

5.3 Experiments pseudo-real world data

Although the introduced ideas are interesting in their own right, their practical significance can only be tested on real world data. We need to determine the improvements by meta-learning and additional meta-features do not just occur in a number of contrived universes but in the real-world as well. To do this we would ideally need a large number of datasets that form a representative sample of real-world datasets.

There are several problems however. The first is conceptual. How do we determine whether this sample is representative of all real world data? The second is practical. Where do we find this large number of datasets.

Data sources

Although a truly representative sample of datasets is beyond our reach, our goal is to at least determine whether the effects occur on some datasets that represent real data. For this we consider two sources.

The first is the collection of UCI datasets, a repository of publicly available datasets that is widely used in the machine learning community. At time of writing there are 150 classification datasets available. An advantage of this set is that they are well known and studied. The main disadvantage is that it is unclear whether these datasets provide a varied set of problems and complexities. In terms of the data complexity measures, introduced earlier, we can not be sure these datasets cover all different values of these measures. Another disadvantage of the UCI datasets is the heterogeneity in the formats of the datasets.

A second possible source is a set of 300 datasets generated by [11] to study domains of competence of different classifiers in terms of the data complexity measures. These datasets were generated from 5 different seed problems specifically to span the complexity space. Unlike the UCI datasets, therefore, these datasets, by construction do cover a

large number of different complexities that we could come across in the real-world. However, we do pay the price for this diversity: these new datasets are altered versions of the original seed problems (of which most are constructed as well!). The result is a set of different datasets that exhibit different complexities, but are far from 'real data'.

Setup

Unlike the simulation studies, where we set the data generating process ourselves, when using real world data we can not simply generate more data to run our experiments. This causes two problems. First of all, the datasets in our set of real world datasets may be too small to extract a large test set on which to estimate the real error of each classifier accurately. In the simulation studies we simply generated a large enough dataset (20000 objects) to get to the desired accuracy. Secondly, we might not have enough datasets to train and test the meta-learner. So beside the base-problems, the meta-problem might also contain too few objects.

Several available strategies and how they relate to these two problems are outlined below:

1. Split each dataset in a small

2. Similar to (1) except reuse splits as both training and test sets, as we do in cross-validation. Each dataset is split in k folds. Each fold is used as the training set, using the rest of the data for testing. Each datasets therefore becomes k different problems, increasing the number of objects in our meta-problem. However, these problems are likely correlated. Another possibility is to vary k for each dataset in order to get similar sample sizes for the training sets. However, this removes the variability in complexity introduced by different sample sizes.

3. Estimate the density of the problem en generate new data from this estimated density. However, the problems are not generated to be easy, so a density estimate might be far from any real world problem.

To establish the sample size for which meta-learning can be expected to give possible improvements we ran the experiment described above using a smaller set of classifiers for different sample sizes. The meta-learning errors for choosing between 1NN and a Fisher classifier can be found in Table 5.1. For large sample sizes, meta-learning will not outperform the accurate cross-validation errors. For extremely small sample sizes, effects are likely small as well. When N_{train} is around 50, improvement seems possible. Given these results we tested the larger set of classifiers on problems with $N_{train} = 50$.

On a more conceptual level, we believe effective classifier selection depends on the assumption that there exist measures characterizing each dataset that we can rely on to determine when one classifier outperforms another. They needs to be discriminative in the domains of competence of different classifiers. In other words, datasets with similar values for these measures should preferably have the same optimal classifier choice.

n	Most Common	CV selection	LDA	K-NN
10	0.292	0.342	0.301	0.341
20	0.197	0.326	0.197	0.199
40	0.471	0.418	0.396	0.336
60	0.389	0.393	0.285	0.263
80	0.339	0.354	0.226	0.213
100	0.345	0.301	0.234	0.215
200	0.405	0.228	0.254	0.228
500	0.451	0.142	0.153	0.153
1000	0.352	0.111	0.144	0.120
2000	0.267	0.094	0.243	0.115

Table 5.1: Leave-one-set out errors of different meta-learning for different sample sizes. The base-classifiers these meta-learners have to choose between are 1NN and a Fisher classifier. Most Common is a strategy that always selects the classifier that is most common in the training set of the meta-problem. Note cross-validation selection becomes hard to improve as sample sizes increase and cross-validation becomes more accurate.

Aside from this, we believe *good* parameterizations need to have some additional properties. (1) Preferably the domains of competence of the classifiers are compact in the chosen parameterization. This means the datasets a classifier works well on are close to one another in terms of the chosen parameters, and are locally continuous, hence some separation of the domains is possible. (2) They can be efficiently computed, which has been the focus of most research in meta-learning and an important prerequisite for their practical appeal. (3) Finally, it would be helpful if the parameters are interpretable. This property helps diagnose problems when performance is worse than required, as well as gives insight into the properties of different algorithms. Combined with compactness in (1), this allows for an understanding of what specific type of problems a classifier works well for.

5.3.1 More complex meta-features

I would possibly like to add an experiment that studies whether more complex meta-features such as bootstrap errors or data complexity measures can improve meta-learning performance.

Alternatively I could run the same analysis as in 4.3 on a single large real-world dataset. This would show that the effects occur on real-world data as well, not just the pseudo real-world example I showed before.

As we have seen from these attempts, meta-learning is considered to be a time efficient alternative to the presumably more accurate cross-validation. In fact, cross-validation selection is used to establish the ground truth against which meta-learning classifiers

are compared. This is based on the assumption that the cross-validation error gives is an accurate estimate of the real error. In the large sample cases considered in the literature, this assumption may well hold. In our research, however, we study the small sample case, in which cross-validation errors become unreliable. This opens up the opportunity to improve upon cross-validation selection. In the next section we will mold cross-validation selection into the meta-learning framework to compare both approaches. Because we can no longer rely on cross-validation to establish a ground truth, we will assume we have a large test dataset with which we can accurately estimate the real error. To our knowledge, this approach has not been taken before.

Another idea I came up with might not effect the meta-learning problem but I find it interesting to consider nonetheless. Given a set of problems (which can all be described as sets of vectors in some space), we calculate the DC measures and performance of different classifiers. These descriptions of the datasets give us a set of vectors forming a new problem, the meta-problem. This meta-problem itself is therefore a set of vectors and, therefore, can be described using the data complexity measures, giving us a new vector. This vector however, is also a vector in the meta-problem space. Therefore, the meta-problem contains itself as an object! Now two questions present themselves: 1. If the number of vectors in the meta-problem is not very large, adding this vector to the meta-problem changes the meta-problem itself, giving us a new meta-problem vector describing this new problem. Adding this vector to the problem again causes a change to the meta-problem description... etcetera. How does this process converge? 2. Because the vector forms an object in the meta-problem we can use a meta-classifier trained on the meta-problem to predict which meta-classifier to use on the problem. Now, using this new meta-classifier, we can again make a prediction giving possibly another meta-classifier. What does convergence in these iterations of predicting meta-classifiers mean for the final classifier that is selected and how does it depend on the initial meta-classifier chosen?

The first source comes from the internal variance $var(\hat{e}_{cv}|D_{train})$, the variance of our estimator given one particular dataset. This is caused by the random assignment of samples to different folds [2]. So, given the dataset repeating the cross-validation procedure gives different estimates $\hat{e}_{cv}$. An estimate of this internal variance is what we added as a meta-feature in the second experiment. A second source is the variance introduced by error-counting, meaning that the estimate can only change in increments, since the error estimates can only take on different values at increments of $\frac{1}{n}$. The third source the dependency between training sets in each fold, since many of the objects in $D \setminus D_i$ where also in $D \setminus D_{i-1}$, especially when k is large. This may be the reason for the large variance of leave-one-out-estimation, where $k = n$.

Whether the cross-validation errors contain all of the information relevant to classifier selection almost looks similar to sufficiency of a statistic, where in this case sufficiency is with respect to the optimal classifier choice)

During the construction of the simulated universes in sections 4.2 and 4.3, we were sur-

prised that interesting problem universes turned out not to be trivial to construct. The main issue is that for many distributions over datasets, often one classifier dominates the whole meta-problem. Even though meta-learning gives good results in these cases, they are not particularly interesting because learning is trivial: choosing the most common classifier serves as a very good selection rule. Of course, we can only speculate whether collections of real-datasets share this property as well. But, given that it is relatively difficult to construct examples of universes where all classifiers have a similar prior probability, we would be quite surprised if real datasets do not have some classifiers that have best performance on a large fraction of the these datasets.

As [14] points out, cross-validation selection represents an inductive bias, just as any individual classifier. He concludes that "cross-validation and prior knowledge are best seen as complementary. Little has been done to date to help us understand how to apply them together in classification work, and this appears to be an important area for future work." Our results can be seen as a step in this direction. The prior knowledge, in our experiments, comes from the assumption our dataset is part of a particular universe of datasets. Results from other datasets in this universe can inform our selection decision. As an example of prior information [14] uses the example of knowing whether a problem is parallel or sequential (has equally important features vs. a few important features) and preferring connectionist resp. symbolic methods as classifier. In our experiments we assume our prior information is the former, the type of problem we are dealing with, but we learn the latter, which methods are preferred for these problem types.

5.4 Conclusion

What meta-learning does is learn the relationship between features of the dataset and *generalization performance* of different algorithms. However, we can not say anything theoretically about this relationship, a fact used to reach the No Free Lunch theorem. Now it is my strong belief that meta-learning therefore, can *only* be effectively trained on real data. The experiment shows what happens when we live in a universe of which we know what the true empirical function is and the meta-learner learns the relationship within this universe. In other words we train a meta-learner on a characteristics, generalization performance relationship in a universe. Therefore, I believe meta-learning, although possible, is only *useful* if we train on a set of real data. Using (semi-)artificially generated data will not yield a useful meta-learner unless there is a very close overlap between the generalization distribution in both universes.

Although this puts up a barrier for meta-learning it is by no means insurmountable. Using data repositories such as (Kaggle, UCI and others), we might be able to contract meta-training sets that reasonably approximate 'reality'. This is especially true when we focus on a specific domain.

Chapter 6

Notes on Implementation

Once the conceptual ideas have been thought out, the experiments needed in this research do not seem overly complex in terms of implementation. It turns out, however, that you run into limitations of many software libraries because of the extremely small sample sizes as well as the problems with scaling the approach. This chapter overs some of the things I learned during the implementation of the project.

The main tools using in this project were Matlab combined with the PRTools library [4] and R. The former was chosen due to the large number of classifiers that were available for testing and the clear and concise notation. The latter was used during the simulation studies to quickly test many different universes due to the easy and inexpensive ways to paralellize the computations. In comparison, Matlab code was generally faster and slightly more legible than R code. The main advantage of R was its huge collection of libraries. Unfortunately, there is little consistency in the interfaces of difference classifier implementation. This makes meta-learning somewhat more cumbersome than a unified package like PRTools.

6.1 Interacting with C code

For many of the experiments in the initial phases of my research project, I needed to calculate the set of data complexity measures on many (at least 300) datasets. These measures are rather computationally intensive, so an efficient implementation was needed. Luckily, this was provided for in the form of the DCoL library by [13]. This library provides fast C implementations of all the measures. To interface with Matlab and R I set out to build interfaces to be able to call the library directly from Matlab/R. Unfortunately, the concept of an ARFF formatted data file is very much ingrained in the design of the library. It was therefore hard to calculate these measures directly on the Matlab/R dataset objects. As not to waste time, I used the second best alternative, writing the Matlab/R objects to memory as ARFF files and calling the C code using a

Matlab/Interface. Although arguably not the most elegant solution, I believe the time saved not having to reimplement most of the library was worth the slight degradation in performance.

The C code itself was a time-saver as well. Comparing my own Matlab implementations of the data complexity measures to the C versions, proved to be more than three times faster for some measures. Over many datasets and many iterations, these time savings add up.

6.2 Parallelization

Parallelization in Matlab: ten though Matlab offers a toolbox for this, this requires an expensive license to operate. Using the free multicore package which provides a bit of a hack to paralellization, it is still possible. In practice I mostly ran multiple Matlab sessions and experiments in parallel. Advantage of open source tools, implemented several classifiers in R but you ran into the problem of insufficient testing.

Chapter 7

Thesis proposals

7.1 Preliminary proposal

Research into the areas of competence of different classifiers has yielded interesting results such as those presented in [1,3]. For my MSc research project I would like to build on these results by finding general patterns and gaining better insight into some questions raised in the literature. My interest in this area is two-fold. First of all, Im interested in the possibilities of automated pattern recognition offered by a more comprehensive view of the data complexity space and the behaviour of different classifiers therein. Secondly, the areas of competence give insight into where classifiers work well or fail to work well and understand this behaviour better. In the remainder of this proposal I explain what I mean by general patterns and gaining better insight, mentioned earlier.

First, general patterns: the data complexity space offers a different way of studying the behaviour of different classifiers. Previous studies have found interesting results for particular classifiers. It would be interesting to see whether we can find more general behaviours of different types of classifiers. One example is given in [1] where the authors find decision forests subsume a single decision tree (i.e. perform better in the whole complexity space). Do we also find this behavior for combinations of other classifiers (for example, linear classifiers)? In general, how does the behavior of a combination of classifiers relate to the behavior of the single classifiers in the complexity space and why (also relates to [2])? And are there other classifiers that are inferior over the whole data complexity space?

Secondly, regarding gaining better insight into questions raised in the literature, one example is the extreme behaviour of linear and nearest neighbour classifiers found in [1]. It seems this is mostly caused by a lack of performance of the ensemble classifiers. Can we gain a better understanding why and where in the complexity space this happens? Is this a weakness in the design of the ensemble methods or intrinsic in selecting a very specific classifier.

Overall, my main interests are in the areas of competence of classifiers for use in matching classifiers with problems. The two areas presented here are two aspects of the data complexity approach that interest me. Other questions, such as how the problem of approximating apparent complexity relates to approximating the apparent error, how to extend the data complexity space to semi-supervised problems or in what problems the area of competence can be most helpful in selecting a good classifier also interest me. It is however less clear to me how these questions relate to the available literature, while the two topics presented earlier came up during my initial literature search.

References

1. Mansilla, E.B.; Tin Kam Ho; , "On classifier domains of competence," Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on , vol.1, no., pp. 136- 139 Vol.1, 23-26 Aug. 2004
2. Tin Kam Ho. 2001. Data Complexity Analysis for Classifier Combination. In Proceedings of the Second International Workshop on Multiple Classifier Systems (MCS '01), Josef Kittler and Fabio Roli (Eds.). Springer-Verlag, London, UK, 53-67.
3. Mitra Basu and Tin Kam Ho. 2006. Data Complexity in Pattern Recognition (Advanced Information and Knowledge Processing). Springer-Verlag New York, Inc., Secaucus, NJ, USA.

7.2 Narrow proposals

7.2.1 Enhancing the complexity approach

Macia's study shows there are different problems that each learner performs particularly well at: golden problems. However, it is unclear whether the problems around these golden problems are also best classified using this learner. Do we actually find clusters in the data, either in the space of the first two principal components or in the 12 dimensional complexity space? Or does the measure not contain enough information to describe these problems. In this respect it would also be interesting to find which dimensions could be dropped from the definition and what measures we could add to improve the dataset characterization. We might look at measures generated by a decision tree such as those proposed by Peng et. al. 2002. Another problem that might require more study is that data complexity forms a lower dimensional representation of the problem, so problems might overlap. This overlap has not been taken into account in Macia's study: here just one dataset for each point in the complexity space was considered. However, if there are different problems that map to this point, the performance on that single problem might not be representative of the real world. This could be done using

simulated micro-experiments to construct a problem were data complexity is the same but classifier performance is different.

7.2.2 Behavior of classifier combinations

Data complexity has promise as a tool to study classifier behavior. One possible research direction would be to study the behavior of different types of classifier combinations and how their areas of competence depend on the areas of competence of the base classifiers. This could either be done using an empirical study on a large number of datasets or by constructing problems that span the whole space of, for example, the two most descriptive complexity dimensions. Using the latter approach we might be able to find a connection between the data generating processes (which we know because they have been artificially constructed) and the behavior of the classifier combinations. This might shed some light on how the shape of the area of competence of the combiner relates to the areas of competence of the base classifiers. Another interesting question would be whether we can find problems on which decision trees outperform decision forest (or in general, were the base classifier outperforms the ensemble) and how this relates to the data complexity measures.

7.2.3 Clustering tendency

One way to study the effectiveness of the domains of competence is to look at clustering tendency. This might be done by treating the area of competence as a sample from the distribution of the real area of competence. By computing the likelihood of this sample originating from various distributions, for example the uniform distribution, we might be able to determine whether what we find are real clusters of competence or that they merely show up by random chance.

7.2.4 Manifold learning/Kernels/Dissimilarity approach

Another interesting question is how a transformation of the dataset or mapping to a kernel space affects the data complexity measures and whether we can use this to guide appropriate transformations of a dataset to an 'easier' representation. This might include selecting combinations of transformations. An interesting aspect to this is whether it is possible to calculate data complexity estimates on all possible kernels. For example, a RBF kernel maps to an implicit infinite dimensional Hilbert space: how can we calculate the complexity measures from this. It might be that the approach is only possible on manifold learning or the dissimilarity approach. In the latter case for example, we might look at the effect of transformations with increasing numbers of prototypes (chosen with some systematic procedure) and the resulting data complexity measures of the transformed dataset.

7.2.5 Predicting classifier accuracy

Instead of predicting where a classifier will outperform other classifiers it might be informative to predict the possible accuracy of classification based on the data complexity measures. From examples were we are fairly sure that current algorithms give optimal performance we might infer what the possible performance on other parts of the complexity space should be. If real performance is different in some areas, there might be room for developing new methods that perform well on these types of problems.

7.2.6 Website to accumulate large number of real world datasets

Using a website where users can either submit datasets or allow users to easily upload both data complexity measures and performance of a set of learners on the dataset, would allow us to better map learner behavior on data complexity measures. The resulting areas of competence for each learner would allow us to offer a recommendation on what learner would be most effective based on data complexity characteristics. From an research perspective, this large number of datasets would offer a more complete picture of areas of competence on real world datasets and might lead to insights on where there is room for improvement to develop new or adept existing classifiers.

7.3 Proposal notes

[6] stops and ask the question how combination methods actually work. She wonders whether the methods have a solid basis or are merely a desperate attempt to increase performance. Instead of selecting the best features and classifier we now seem to select the best classifiers and combiner. We could fall into the trap of taking this on higher and higer levels.

Also interesting how this relates to Stochastic discriminants, in which we combine a very large amount of weak learners to get a good coverage of the space and the problem. By subsequently adding weak classifiers the result is proven to converge to perfect classification on the training set, and depending on the project-ability of the weak learners, good performance on off-training set samples. It is interesting what the effect would be of subsequently adding weak classifiers to the space on the data complexity measure of the space.

Sequential vs. Parallel (many classifiers in one layer vs. many layers) This approach is different from a many layered network, since we basically have only one transformation from the original features to the classifier predictions. After adding enough learners, the set will become linearly separable on one of the dimensions. This convergence happens faster is we use stronger (more enhancing) learners. In multi-layer approaches we use

less classifiers for the transformation but do transformations sequentially so each layer deals with a simpler problem.

Advantage of stochastic discriminant is that its easier to paralellize.

We could think of the a neural network, or the more complex variant of the deep belief network as a number of transformations taking place sequentially. Each layer in the network produces a higher level, more abstract representation of the features. The layering is used explicitly as an argument for the success of Deep Learning architectures. How many layers are used in classification? 2 layers in support vector machines. The above hypothesis would suggest deep learning architectures work by reducing the complexity of the problems through applying transformations at each layer. If some structure is found in how these transformations effect the complexity of the problem, we could use this to guide the choice of the number and types of layers.

Using such simple, even static, classifiers we assume that the transformed space must be much simpler than the original space. Static classifiers would even assume how the objects are spread over this space. Is there a difference in the effect on the resulting space when using diverse vs. similar classifiers.Does this explain why one does better than the other or what combiners should be used? A theory of transformations: Can we characterize the space of transformations in terms of their effect on the complexity of the resulting space.

Bibliography

[1] Mitra Basu and Tin Kam Ho. *Data Complexity in Pattern Recognition (Advanced Information and Knowledge Processing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[2] Ulisses Braga-Neto and Edward R. Dougherty. Is cross-validation valid for small-sample microarray classification? *Bioinformatics*, 20(3):374–380, 2004.

[3] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[4] R. P. W. Duin. Prtools - version 3.0 - a matlab toolbox for pattern recognition. In *Proc. of SPIE*, page 1331, 2000.

[5] Robert P W Duin and Marina Skurichina. A Discussion on the Classifier Projection Space for Classifier Combining. pages 137–148, 2002.

[6] Tin Kam Ho. Data complexity analysis for classifier combination. In *Proc. Int. Workshop on Multiple Classifier Systems (LNCS 2096*, pages 53–67. Springer, 2001.

[7] Tin Kam Ho and Mitra Basu. Complexity measures of supervised classification problems. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(3):289–300, 2002.

[8] Jesse H. Krijthe, Tin Kam Ho, and Marco Loog. Improving cross-validation based classifier selection using meta-learning. In *International Conference on Pattern Recognition*, 2012. Forthcoming.

[9] Ludmila I. Kuncheva and Christopher J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2):181–207, 2003.

[10] Núria Macià. *Data complexity in supervised learning: A far-reaching implication.* PhD thesis, La Salle — Universitat Ramon Llull, October 2011.

[11] Núria Macià, Tin Kam Ho, Albert Orriols-Puig, and Ester Bernadó-Mansilla. The lansdcape contest at icpr'10. In Selim Aksoy Devrim Ünay and Zehra Çataltepe, editors, *Contests in ICPR 2010*, volume 6388 of *Lecture Notes in Computer Science*, pages 29–5, Berlin, Heidelberg, August 2010. Springer.

[12] Núria Macià, Albert Orriols-Puig, and Ester Bernadó-Mansilla. Beyond homemade synthetic data sets. In *HAIS'09: Proceedings of the 4th international conference on hybrid artificial intelligence systems*, volume 5572 of *Lecture Notes in Artificial Intelligence*, pages 605–612. Springer-Verlag, Berlin, Heidelberg, June 2009.

[13] Albert Orriols-Puig, Núria Macià, and Tin Kam Ho. Documentation for the data complexity library in C++. Technical report, La Salle - Universitat Ramon Llull, 2010.

[14] Cullen Schaffer. Selecting a classification method by cross-validation. *Machine Learning*, 13:135–143, 1993.

[15] Marina Skurichina and Robert P. W. Duin. The role of combining rules in bagging and boosting. In Francesc J. Ferri, José Manuel Iñesta Quereda, Adnan Amin, and Pavel Pudil, editors, *SSPR/SPR*, volume 1876 of *Lecture Notes in Computer Science*, pages 631–640. Springer, 2000.

[16] David H. Wolpert and William G. Macready. Combining stacking with bagging to improve a learning algorithm. Technical report, 1996.

Appendix A

ICPR2012 Submission

Improving Cross-validation Based Classifier Selection using Meta-Learning

Jesse H. Krijthe
Delft University of Technology
jkrijthe@gmail.com

Tin Kam Ho
Bell Laboratories, Alcatel-Lucent
tin.ho@alcatel-lucent.com

Marco Loog
Delft University of Technology
mloog@tudelft.nl

Abstract

In this paper we compare classifier selection using cross-validation with meta-learning, using as meta-features both the cross-validation errors and other measures characterizing the data. Through simulation experiments we demonstrate situations where meta-learning offers better classifier selections than ordinary cross-validation. The results provide some evidence to support meta-learning not just as a more time efficient classifier selection technique than cross-validation, but potentially as more accurate. It also provides support for the usefulness of data complexity estimates as meta-features for classifier selection.

1 Introduction

In selecting a classifier for a classification problem, a standard procedure is to determine the cross-validation errors of a wide range of classifiers, and to choose one with the lowest cross-validation error [9]. However, problems with this procedure have been observed before. For small samples, cross-validation has been shown to be unreliable [1, 4]. Yet with large training samples, this approach to selection can become very expensive. Research in meta-learning has given us measures that are less computationally intensive but can still aid the classifier choice [2]. There meta-learning is considered to be a time efficient alternative to the presumably more accurate cross-validation.

In this paper, we demonstrate that meta-learning can provide not only efficiency, but also potential improvement to the accuracy of classifier selection over cross-validation. We show this by treating the classifier selection problem itself as a classification problem, where the cross-validation errors are used as features describing the datasets, and the goal is to predict which classifier works better for each dataset. Viewing cross-validation from this perspective, we can compare it to other meta-learning procedures, and compare the roles of these features and decision rules to their counterparts in ordinary classification problems. We also show that additional meta-features describing properties of the dataset can improve selection accuracy, providing additional evidence for the potential utility of so-called data complexity measures [3, 6, 7] in guiding classifier selection.

The rest of the paper is structured as follows. Section 2 presents cross-validation classifier selection in the form of a meta-classifier. Section 3 discusses a situation where this meta-classifier fails and how to improve upon it. Section 4 demonstrates the usefulness of adding new features to this meta-problem. We end with a discussion of our results and a conclusion.

2 Classifier Selection Problem

The premise of effective classifier selection depends on two assumptions. First, the classifier domains of competence [7], meaning the types of problems a classifier provides the best performance on, do not overlap completely. Hence a careful selection can potentially optimize performance. Secondly, there exist measures characterizing each dataset that we can rely on to determine when one classifier outperforms another. These measures can be considered as a parametrization of the space of all classification problems, S. An example of such a parametrization is cross-validation errors of several classifiers, which characterize each dataset by the difficulties it causes to the concerned classifiers. Another example from the meta-learning literature is the 'data complexity' measures [3, 6, 7] describing geometrical and topological properties of a dataset.

We believe *good* parameterizations of S need to have some additional properties. (1) Preferably the domains

of competence of the classifiers are compact in the chosen parameterization. This means the datasets a classifier works well on are close to one another in terms of the chosen parameters, and are locally continuous, hence some separation of the domains is possible. (2) They can be efficiently computed, which has been the focus of most research in meta-learning and an important prerequisite for their practical appeal. (3) Finally, it would be helpful if the parameters are interpretable. This property helps diagnose problems when performance is worse than required, as well as gives insight into the properties of different algorithms. Combined with compactness in (1), this allows for an understanding of what specific type of problems a classifier works well for.

Note that the assumptions mentioned earlier are the same as for any classification problem. In fact we can treat the classifier selection problem as another classification problem, or a "meta-problem". This meta-problem has as its meta-features the measures characterizing the dataset and the meta-classes are the classifiers that perform best on each dataset.

Seen through this framework, classifier selection using cross-validation is a meta-classifier that makes an additional assumption: not only is it possible to distinguish between the domains of competence, but we can do this by a simple static decision rule. The selection rule is in fact an untrained meta-classifier: we *always* select the classifier with the lowest cross-validation error, regardless of our experience on other datasets.

In section 3 we will demonstrate that there exist universes of problems where the space formed by the cross-validation errors does not support the superiority of this decision rule. Instead, using a different meta-classifier allows for an improvement in accuracy over cross-validation based selection. In section 4 we study how adding extra meta-features to the meta-problem may allow us to further improve accuracy, demonstrating cross-validation errors do not necessarily carry all information useful in selecting a classifier.

3 Cross-Validation in Meta-Learning

Cross-validation based selection employs a static decision rule: the classifier with the lowest cross-validation error gets selected regardless of experience obtained on selecting classifiers on previous problems. In case of a choice between two classifiers, this rule can be visualized by the diagonal in the 2D cross-validation error space (see Figure 1). To illustrate how a situation may arise where this rule is too rigid and we can improve upon it, we sample classification problems from a chosen universe that is generated systematically. The goal is to visualize the potential issue of a static selection procedure using a simplified but intuitive example. We believe similar results also occur for more complex universes.

The example universe of problems we consider consists of instances of two problems: a problem with strongly overlapping, two dimensional Gaussian classes (call it $\mathbf{G}$) and a more complex problem with 'banana-shaped' classes with Gaussian noise and small overlap (call it $\mathbf{B}$) (see Figure 1). We introduce some differences to the instances of $\mathbf{G}$ by selecting the separation of the class means from a uniform distribution. For instances in $\mathbf{B}$, we change the variance of the Gaussian noise. In this way we create a family of 500 instances for each problem: $\mathbf{G} = \{G_1, G_2, ..., G_{500}\}$, and $\mathbf{B} = \{B_1, B_2, ..., B_{500}\}$. From each of these problems B_i or $G_i (i = 1, ..., 500)$, we extract a small sample of size N_{train} (varying uniformly randomly between 20 and 100) for training, call this G_i^{train} and B_i^{train}, and use the rest ($N_{test} = 20000 - N_{train}$) for testing, G_i^{test} and B_i^{test}. The extremely small training set size is chosen to cause cross-validation error to have high variance. In real world problems, with complex boundaries, this effect will occur for larger sample sizes. The large test set size is used to get an accurate estimate of the real error of the classifier for the problem instance.

The goal is to choose between two classifiers: a nearest mean classifier (NM) and a 1-nearest neighbor classifier (1-NN). Casting this as a meta-learning problem, we create the meta-features and meta-classes as follows. For each G_i, we obtain a 10-fold cross validation error for the NM classifier using G_i^{train}. That is, for each of ten passes, NM is trained using $N_{train} * 9/10$ samples, and tested using the remaining $N_{train}/10$ samples. The total number of errors over the ten passes, divided by N_{train}, is used as meta-feature 1 for G_i. Doing the same for the 1-NN classifier, we obtain the value of meta-feature 2 for G_i, and similarly for each B_i.

The meta-class for each of G_i or B_i is the classifier that performs the best for the testing data G_i^{test} or B_i^{test}. This is determined by, say, using all of G_i^{train} for training the NM classifier, estimating the error rate of the resultant classifier on G_i^{test}, and doing the same for the 1NN classifier. If the NM gives a lower error, G_i is labeled with the true meta-class NM, and vice versa. Similarly we obtain the true meta-class for B_i.

For the meta-learning problem, a random half of the 500 instances from $\mathbf{G}$ or $\mathbf{B}$ are used for meta-training (denoted by $\mathbf{G}^{TRAIN}$ and $\mathbf{B}^{TRAIN}$) and the rest are reserved for meta-testing ($\mathbf{G}^{TEST}$ and $\mathbf{B}^{TEST}$). Recall that the cross validation rule corresponds to a static classifier, and hence requires no training. We compare this to a trainable classifier that is a support vector machine, trained with the meta-features for $\mathbf{G}^{TRAIN}$ and

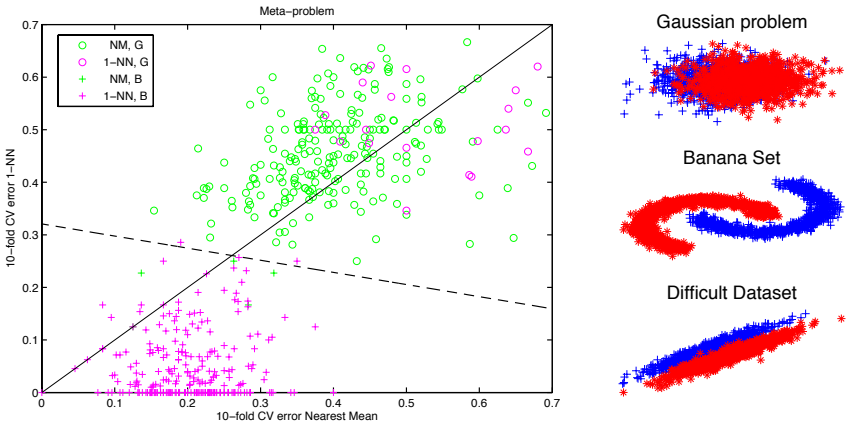


Figure 1. On the left: an example of a universe of problems where static cross-validation selection does not perform well. On the right: the problems 'Gaussian', 'Banana Set' as discussed in Section 3, and the 'Difficult Dataset', a 2D version of the problems discussed in Section 4.

$\mathbf{B}^{TRAIN}$. The comparison is shown in Figure 1.

In Figure 1, the points plotted represent problem instances from $\mathbf{G}^{TEST}$ and $\mathbf{B}^{TEST}$. Their coordinates are given by the two meta features, and they are marked with symbols denoting their source family (o for the Gaussian problems, or $+$ for the Banana problems), and colors denoting their true meta-class (green for NM, and pink for 1-NN).

We can observe that 1-NN almost always gives better performance for the cluster of Banana problems (almost all points in the cluster of $+$ are marked pink). The other cluster formed by the Gaussian problems primarily favors the NM classifier (most o's are marked green). Note that in this particular universe, the cross-validation rule does not give the best separation: the decision boundary crosses through the cluster of Gaussian problems (o's), forcing one to choose the 1-NN classifier for the points below the diagonal, while performance in this cluster is in fact almost always better with the NM classifier. The selection error, defined as the proportion of datasets in $\mathbf{B}^{TEST} \cup \mathbf{G}^{TEST}$ assigned by the meta-rule to a classifier different from their true meta-class, is 0.1583 for cross-validation. The support vector machine reduces the selection error to 0.0601, offering a more accurate alternative.

4 Adding Meta-Features

The example in the previous section has shown that learning the selection rule can improve performance in some cases where the cross validation errors have high variance. In more elaborate universes of problems, the meta-classes may overlap heavily, making it difficult to select a classifier based solely on the cross-validation errors. Other measures characterizing the dataset may reduce this overlap by introducing extra features that can aid the selection task.

Consider a universe of 100 dimensional linearly separable two-class problems where most of the class variance is along the class boundary (the 'Difficult Dataset' from the PRTools library (prtools.org)), see Figure 1 (right). We introduce some variability to the universe by randomly selecting the class-priors using a uniform distribution between 0 and 1. Similar to Section 3, we estimate cross-validation errors with different sample sizes (N_{train} is now between 20 and 100). The classifiers we considered are now the nearest mean classifier and the Fisher classifier. Cross-validation selection gives a selection error of 0.265. Using a nearest neighbor classifier as a meta-learner, we improve the error to 0.234. Adding to the meta-features an extremely simple characterization of the sampling density of the dataset, the number of objects divided by the number of dimensions, causes a further reduction in error to 0.204.

5 Discussion

We consider classifier selection as a classification problem in its own right. Recall that the reason for doing classifier selection in the first place is an assumption that different problems may need different classifiers. If we really believe this to be true, we should assume the same for the meta-problem, thereby allowing different meta-classifiers. Yet cross-validation based selection is a static, universal rule. We have shown that it could indeed be suboptimal for this meta-problem, and could be

inferior to another rule obtained by meta-learning.

The reason meta-learning can outperform cross-validation is that it leverages experience from other previously seen instances from the same source problem, whereas the cross-validation rule remains static. During our experiments, we have often come across situations where one classifier outperforms all others on most instances of the same problem family. A meta-learner might learn that this classifier should be preferred for all problems from this family. This could be a better choice than deciding, for each instance, solely based on cross-validation error, as that may be unreliable when small samples are used to train a complex classifier, which is exactly the case where cross-validation selection frequently fails. An interesting observation that follows from this is that the belief by a practitioner working on a specific application domain that a certain classifier should always be preferred is not necessarily irrational and may be a more accurate rule than using cross-validation.

The problem universe where extra measures characterizing the data improve selection accuracy shows that these measures can not only serve as computationally efficient proxies to cross-validation errors, but actually improve performance. This provides support for the goal of approaches like data complexity measures. To make this approach more useful, however, further research should focus on finding measures that take into account the properties a good parameterization should have. Especially interpretability will likely help adoption of these extra measures, because this not only tells the user when each classifier has good performance, but also helps explain why this is the case.

We have chosen to use extremely small training sizes and large test sets to highlight the issue of variance in cross-validation error estimation. In real world data, this could happen when complex class boundaries must be inferred from relatively limited training samples. Another situation where we expect cross-validation selection to become unreliable is when choosing between a large number of similar classifiers. As this number increases, at least one of the cross-validation errors will likely be low by chance. Meta-learning allows us to correct for this case. Even for a task where large samples are eventually available, the procedure could be used to select a classifier early on, and before the large samples arrive. As more data becomes available we will be able to evaluate whether this choice is correct.

The key question that remains is whether one can get this improved selection strategy to work not just in some artificial universe but in a domain of real-world datasets as well. This can only be established through an empirical study. Unfortunately, the lack of availability of large numbers of standardized real-world datasets is a recurring problem in meta-learning research. Some projects [5, 8] supporting the publication of datasets might help alleviate this problem, but currently offer still too few real-world datasets. For specific restricted application domains, the possibility of finding a reasonably large set of datasets is more plausible.

6 Conclusion

The purpose of this paper has been to show that the effectiveness of the cross-validation selection could be impacted by factors such as the variance of the cross-validation error estimates, not unlike those factors affecting the accuracy of an ordinary classification problem. As we have demonstrated, in some simple universes, cross-validation selection is not the best strategy. In these cases, treating the cross-validation errors as meta-features in a meta-classification problem offers a chance to increase selection accuracy using other decision rules. Adding other features characterizing the dataset may also increase selection accuracy, which supports the idea of data complexity measures and similar approaches to meta-learning.

References

[1] U. Braga-Neto and E. R. Dougherty. Is cross-validation valid for small-sample microarray classification? *Bioinformatics*, 20(3):374–380, 2004.

[2] P. Brazdil, J. Gama, and B. Henery. Characterizing the applicability of classification algorithms using meta-level learning. In F. Bergadano and L. De Raedt, editors, *Machine Learning: ECML-94*, volume 784 of *Lecture Notes in Computer Science*, pages 83–102. Springer Berlin / Heidelberg, 1994.

[3] T. K. Ho and M. Basu. Complexity measures of supervised classification problems. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(3):289–300, 2002.

[4] A. Isaksson, M. Wallman, H. Göransson, and M. G. Gustafsson. Cross-validation and bootstrapping are unreliable in small sample classification. *Pattern Recogn. Lett.*, 29:1960–1965, October 2008.

[5] P. Liang and J. Abernethy. mlcomp.org, 2009.

[6] N. Macià, T. K. Ho, A. Orriols-Puig, and E. Bernadó-Mansilla. The lansdcape contest at icpr'10. In S. A. Devrim Ünay and Z. Çataltepe, editors, *Contests in ICPR 2010*, volume 6388 of *Lecture Notes in Computer Science*, pages 29–5, Berlin, Heidelberg, August 2010. Springer.

[7] E. Mansilla and T. K. Ho. On classifier domains of competence. In *Proceedings of the 17th ICPR*, volume 1, pages 136 – 139 Vol.1, aug. 2004.

[8] C. S. Ong, M. L. Braun, S. Sonnenburg, S. Henschel, and P. O. Hoyer. mldata.org, 2009.

[9] C. Schaffer. Selecting a classification method by cross-validation. *Machine Learning*, 13:135–143, 1993.