Prediction of Classifier Performance Based on Meta Characteristics

Bernd Bischl, Max Wornowizki, Gero Szepannek, Claus Weihs

Abstract

1 Introduction

One of the most time-consuming problems in machine learning today is choosing an appropriate method for a classification task. Over the last decades a multitude of algorithms has been published, strongly varying fitted model types, discriminant function, runtime complexity and accuracy. While the general practise for a specific data set at hand is to select the best classifier through cross-validation, this becomes very laborious because of the large number of available methods. Especially for non-experts or classifications task, where only a good but maybe not the best choice of an algorithm is needed, a recommendation system would be beneficial. Even though some methods like support vector machines [**?**] are very popular and have proven to be extremely useful powerful with regard to prediction performance, it is known that no classifier will outperform all others over all possible data sets [**?**]. For this reason, Hornik and Meyer proposed some consensus ranking [**?**] of a set of classifiers meaningfully harmonizes with observed results on a group of data sets.

The question arises, whether it is actually possible to learn associations between between data sets $D_1, \ldots, D_n$ and at least reasonably well-performing respective classification methods. This is known as the 'meta learning' problem which has generated considerable interest in research

cite 3-4 wichtigste papers ($x_i.y_j$)

Probably the most crucial decisions when casting the problem into a statistical model is, what functional realtionship X -> Y one actually tries to learn. For X one has to decide upon a set of numerical values, which hopef. Here two approaches have been used in the past: Either one generates a certain set of statistics for every data set (DCT approach), which include information theoretic measurement like...,

or one directly evaluates a couple of weak and fast learners on all data sets, and uses their empirical performances in order to predict the behaviour of more complex and costly algorithms (landmarking approach)
was gibts kurz, metaL, landmarking, etc wir machen klassif. in sigif. klassen
The rest of the paper is organized as follows: Section 2 describes the benchmark framework that we use to compare algorithm and its connection to the meta learning problem. Section 3, Finally, our results and conclusion are presented in section 5 and 6.

2 Benchmark Analysis

When comparing classification methods with regard to a specific data set, it is generally not enough to report only mean values of the considered performance measure - e.g. misclassification error - and produce an according absolute ranking. A much more reasonable approach is to try to partition the algorithms into groups, where one cannot statistically prove significant different performances for any pair of algorithms. Hothorn et al. have published a theoretical framework for dealing with such questions in a concise manner. In a nutshell, one generates generates bootstrap samples of the data set and repeatedly fits and evaluates the models (e.g. the out-of-sample error). This result in a (conditional) sample from the performance measure for every classifier. Now standard statistical tests can be employed to compare the locations of the underlying distributions.

3 Suggested Approach to Meta Learning

3.1 Main idea

The key idea of the suggested new approach to meta learning is the question *whether it is meaningful to use a specific classifier for a specific classification problem or not.* For this purpose we try to build a prediction model to answer this question based on a 'training" group of data sets. In a first step, we try to solve this question by checking whether a specific classifier is significantly outperformed by any of the other classifiers on any of the training data sets (see Section 3.2). Based on so called *DCT* features that can be directly derived from any data set (see Section 3.3) a prediction model (based on the results for all training data sets) will be derived that automatically decides whether a classifier is within the group of the "significantly best classifiers" or not. This results in a second "meta" classification problem (see

Section 3.4).

3.2 Testing significance

- Referenz
- Test beschreiben

3.3 Data set characteristics

The fifty meta-level attributes used in this study can be divided in three main groups: simple measures like the number of attributes and examples, statistical measures like skewness and kurtosis and information theory-based measures like entropy of classes. All measures that have to be calculated for each attribute of a data set separately are described through a minimum, a maximum and an average value. Most attributes arise from the METAL project [**?**], the remaining attributes were proposed in (REFERENZ:am ende tabelle). Table 2 gives an overview about all characteristics used in this study.

3.4 Implementing the meta learner

As it has already been mentioned before in Section **??** the final meta learning problem consists in finding a (meta-)classification model for each classifier based on the "DCT data set characteristics" that allows to decide automatically whether a classifier is outperformed by any other classifier ($\sim$ class 0) or not ($\sim$ class 1). For this purpose two different classification algorithms have been implemented. Some first results are obtained using decision trees. The reason for this choice is the interpretability of the tree models: it is of interest to identify which of the data set characteristic features are of relevant influence for the performance of the different classifiers.
To further improve the prediction the algorithm is extended by boosting using the `AdaBoost` Algorithm [**?**].
To measure the performance of the obtained meta learning models leave one out cross validation is used: i.e. for *each single* classifier a meta-classification model is trained based on *all* performances of *all* classifiers on *all* data sets except one. The latter is used for testing the performance prediction (i.e. whether an algorithm is significantly outperformed by another one or not) of the model. As a performance measure the average test prediction accuracy over all classifiers and all data sets is tabled.

nr. attributes	Wilks lambda
nr. sym. attributes	Bartlett statistic
nr. num. attributes	avg. of gini sym
nr. examples	min. of gini sym
nr. classes	max. of gini sym
missing values relative	avg. of relevance
lines with missing values relative	min. of relevance
mean absolute skew	max. of relevance
avg. of skew	avg. of gFunction
min. of skew	min. of gFunction
max. of skew	max. of gFunction
mean kurtosis	class entropy
avg. of kurt	entropy attributes
min. of kurt	mutual information
max. of kurt	joint entropy
nr. attrs. with outliers	equivalent nr. of attrs.
avg. of multi correl	noise signal ratio
min. of multi correl	nr. sym. attrs./nr.attrs.
max. of multi correl	nr. num. attrs./nr .attrs.
M-stat	nr.examples/nr. attrs.
M-stat df	nr. classes/nr. attrs.
M-stat chi sq	nr. attrs. with outliers/nr. num. attrs.
sd ratio	equivalent nr.of attrs./nr. sym. attrs.
fract	log of nr. examples
cancor	max. of classvalue freq

Table 2: Meta characteristics of the datasets

4 Design of the Study

4.1 Overview

The aim of our study is to investigate the benefit of meta learning as a supervised guide for the classifier choice based on data set meta characteristics. For this purpose we evaluated and tuned a set of different classifiers (see Section **??**) on several different data sets **??**. The results were compared and related to the data set meta characteristics as it is described in Section **??**.

4.2 Classifiers

The five classifiers used in the analysis were k Nearest Neighbor (kNN), Linear Discriminant Analysis (LDA) [**?**], Naive Bayes [**?**], Recursive Partitioning Classification Trees (rpart) [**?**] and Support Vector Machines (SVM) [**?**]. Their performance is measured as follows taking into account that in daily application hyper parameters have to be optimized: At first 30 bootstrap samples are drawn from each dataset. On each training sample the misclassification error of each classifier except the LDA using each evaluated hyper parameter combination is estimated in a separate five fold cross validation. Hereafter, the parameter set with the lowest average misclassification rate for each classifier is used for training on the whole training sample to predict the out of bag (oob) - observations for each bootstrap sample. Note that the best hyper parameters usually are not be the same for every bootstrap sample. For LDA i is not necessary to set any additional hyper parameters hence training and prediction is directly done here. For each dataset this approach generates 30 misclassification errors of each tuned classifier. The considered hyper parameter combinations are shown in table3.
A typical problem using the programming language `R` for classification purposes is the heterogenity of the different implemented algorithms. A framework has been developed [**?**] in order to easily enable optimizing and benchmarking different classifiers in `R`. Its features are: an object oriented `S4` interface to `R` classification methods, easy extension to new methods, it provides a unified call of different methods, bootstrapping, cross-validation, train/test splits, parameter tuning and benchmarking of different classification algorithms are possible (e.g. by 'double cv' with tuning on an inner cv).

Classifier	Hyperparameters
kNN	k = 1, 2, ..., 10
LDA	-
Naive Bayes	laplace = $0, 2^{-6}, 2^{-5}, \ldots, 2^{6}$
rpart	minsplit = $1, 5, 10, 20, 40$; cp = $0.005, 0.01, 0.05, 0.1$
SVM	kernel = polydot ; C = $2^{-6}, 2^{-5}, \ldots, 2^{6}$

Table 3: Overview of the classifiers and hyper parameters that are used for tuning.

4.3 Data sets

For our analysis 20 data sets were taken from the UCI Machine Learning Repository [**?**]. Almost all of them contain both numerical and categoric attributes and are medium-sized meaning from 500 to 10000 observations (for more details see Appendix). This selection was made on purpose as it leads to a realistic and more challenging setting than e.g. problems with only numerical predictors.
Unfortunately a lot of preprocessing was inevitable: As some data sets on UCI are subdivided in train and test part they first had to be merged so that samples could be drawn from the whole data sets. Also obvious variables without information for the classification task like IDs for example and on the other hand variables with many NAs were eliminated. Additionally, the predictors in particular the integer-valued were checked to have an appropriate data class which ensured factor values not to be interpreted as numerics and vice versa. At last all observations with NAs were removed from each data set. Table 4 gives an overwiev of the datasets and some important characteristics.

5 Results

6 Summary

Name	Nr. of observations [1]	Nr. of attributes [2]	Nr. of classes
abalone	4117	8 / 7 / 1	28
ann	7200	21 / 6 / 15	3
australian	690	14 / 6 / 8	2
car	1728	6 / 0 / 6	4
contraceptive method choice	1473	9 / 2 / 7	3
credit approval	653	15 / 6 / 9	2
german credit	1000	20 / 7 / 13	2
image segmentation	2310	19 / 19 / 0	7
image segmentation (statlog version)	2310	19 / 19 / 0	7
landsat satellite	6435	36 / 36 / 0	6
mammographic mass	831	4 / 2 /2	2
mushroom	5644	22 / 0 / 22	2
musk (version 2)	6598	166 / 166 / 0	2
optical recognition of handwritten digits	5620	64 / 64 / 0	10
pen-based recognition of handwritten digits	10992	16 / 16 / 0	10
spambase	4601	57 / 57 / 0	2
splice-junction gene sequences	3190	61 / 0 / 61	3
teaching assistant evaluation	151	5 / 1 / 4	3
tic-tac-toe endgame	958	9 / 0 / 9/	2
vehicle	846	18 / 18 / 0	4

Table 4: Properties of the used datasets

[1] after removing the observations with NAs
[2] total number / number of numerical attributes / number of factor attributes