LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

INSTITUT FÜR STATISTIK

Manuel J. A. Eugster, Friedrich Leisch & Carolin Strobl

(Psycho-)Analysis of Benchmark Experiments – A Formal Framework for Investigating the Relationship between Data Sets and Learning Algorithms

Technical Report Number 078, 2010
Department of Statistics
University of Munich

http://www.stat.uni-muenchen.de

(Psycho-)Analysis of Benchmark Experiments

A Formal Framework for Investigating the Relationship between Data Sets and Learning Algorithms

Manuel J. A. Eugster, Friedrich Leisch and Carolin Strobl

Department of Statistics, LMU München, München, Germany

Firstname.Lastname@stat.uni-muenchen.de

Abstract

It is common knowledge that certain characteristics of data sets – such as linear separability or sample size – determine the performance of learning algorithms. In this paper we propose a formal framework for investigations on this relationship.

The framework combines three, in their respective scientific discipline well-established, methods. *Benchmark experiments* are the method of choice in machine and statistical learning to compare algorithms with respect to a certain performance measure on particular data sets. To realize the interaction between data sets and algorithms, the data sets are characterized using *statistical and information-theoretic measures*; a common approach in the field of meta learning to decide which algorithms are suited to particular data sets. Finally, the performance ranking of algorithms on groups of data sets with similar characteristics is determined by means of recursively partitioning *Bradley-Terry models*, that are commonly used in psychology to study the preferences of human subjects. The result is a tree with splits in data set characteristics which significantly change the performances of the algorithms. The main advantage is the automatic detection of these important characteristics.

The framework is introduced using a simple artificial example. Its real-word usage is demonstrated by means of an application example consisting of thirteen well-known data sets and six common learning algorithms. All resources to replicate the examples are available online.

1 Introduction

In machine and statistical learning, benchmark experiments are empirical investigations with the aim of comparing and ranking learning algorithms with respect to certain performance measures. In particular, performance is investigated on a collection of data sets, e.g., from the UCI Machine Learning Repository [1]. It is well known that the characteristics of the data sets have an influence on the performance of the algorithms – almost every publication that proposes a new algorithm presents its performance on data sets in relation to different characteristics (even though often only the number of observations and attributes vary). Nonetheless, in most publications differences of the data sets are noted but not used for further well-founded analyses; perhaps the best known study is STATLOG [13], newer ones are e.g. [15] and [4].

In psychology and related disciplines the pairwise comparative choice model suggested by Bradley and Terry [2] is the most widely used method to study preferences of *subjects* (e.g. consumers or patients) on some *objects* (e.g. a set of chocolate bars or different pain therapies). The preference scaling of a group of subjects may not be homogeneous, but different groups of subjects with certain characteristics may show different preference scalings. A newly developed semi-parametric approach for recursive partitioning of Bradley-Terry models [19] takes this circumstance into account – it identifies groups of subjects with homogeneous preference scalings in a data-driven and statistically correct way. This approach is an extension of the classical algorithms for classification and regression trees (CART) [3, 16] and results in a tree where the subjects are divided into groups according to their characteristics, and in each terminal leaf a Bradley-Terry model shows the preference scaling within this group.

This approach can also be applied to benchmark studies; now the data sets are the subjects and the algorithms are the objects. The interpretation is the following: a data set expresses its preferences for an algorithm by the performance of the algorithm. In other words, the algorithm that performs best on a certain data set is considered to be most preferred by the data set. Using statistical and information-theoretic measures to characterize the data sets, the approach of recursive partitioning of Bradley-Terry models enables us to determine the influence of data set characteristics on the performance of the algorithms. It provides a framework to either investigate the influence exploratorily or test particular hypothesis of interest. This article defines all parts of the framework and, as a first step, presents its prospects for the exploratory analysis.

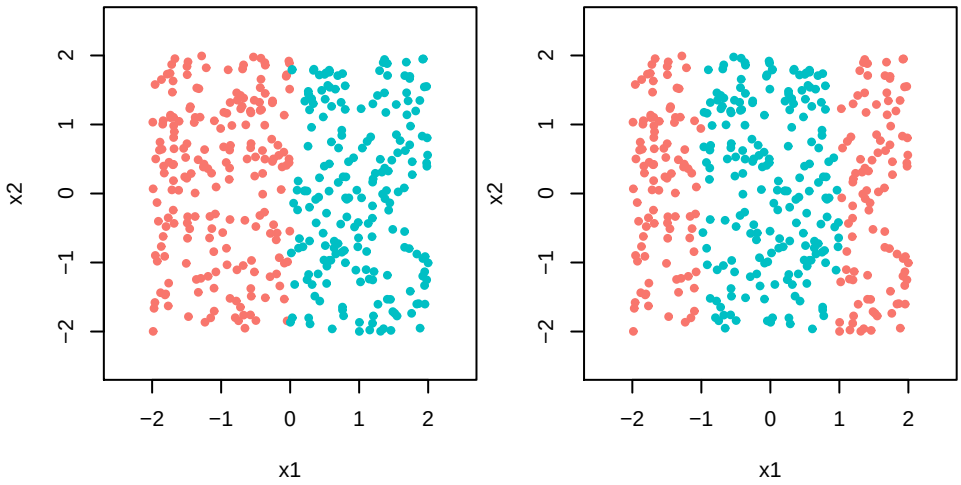

Figure 1: Exemplary classification problems: (a) `ds1` where the feature space is linearly separable; (b) `ds2` with non-linearly separable feature space.

The article is organized as follows: Section 2 introduces all related methods in detail. First, we introduce a formal notation for benchmark experiments; then we define a sound and flexible framework for data set characterization and introduce a common set of data set characteristics; finally we outline preference scaling using recursive partitioning of Bradley-Terry models. A toy example is used to demonstrate each part. Section 3 then applies the proposed method to a real-world example based on classification problems from the well-known UCI Machine Learning Repository. The article is concluded with a summary and an outlook in Section 4 and computational details for reproducibility in Section 5.

2 Methods

In this section the individual methods are introduced and merged into a framework for analyzing the influence of data set characteristics on the performance of algorithms.

For illustration purposes, an artificial toy example is used. It consists of two 2-dimensional classification problems with 400 observations in each case: `ds1` is linearly separable, `ds2` is not linearly separable; Figure 1 shows scatter plots of the data. The algorithms of interest are support vector machines (`svm`), linear discriminant analysis (`lda`) and quadratic discriminant analysis (`qda`) (see e.g. [10] and Section 5 for computational details).

2.1 Benchmark experiments

Following [11], a benchmark experiment is defined as follows: Given is a data set $\mathfrak{L} = \{z_1, \ldots, z_N\}$. In case of supervised learning problems, any observation $z \in \mathfrak{L}$ is of the form $z = (y, x)$ where y denotes the response variable and x describes a vector of input variables (note that for readability we omit in case of $z = (y, x)$ the index i with $i = 1, \ldots, N$). We draw $b = 1, \ldots, B$ learning samples of size n using a resampling scheme, such as sampling with replacement (bootstrapping, usually of size $n = N$) or subsampling without replacement ($n < N$):

$$\mathfrak{L}^b = \{z_1^b, \ldots, z_n^b\}$$

Furthermore we assume that there are $K > 1$ candidate algorithms a_k, $k = 1, \ldots, K$, available for the solution of the underlying problem. For each algorithm a_k the function $a_k(\cdot \mid \mathfrak{L}^b)$ is the fitted model based on the sample $\mathfrak{L}^b$. This function itself has a distribution $\mathcal{A}_k$ as it is a random variable depending on $\mathfrak{L}^b$:

$$a_k(\cdot \mid \mathfrak{L}^b) \sim \mathcal{A}_k(\mathfrak{L}),\ k = 1, \ldots, K$$

The performance of the candidate algorithm a_k when provided with the training data $\mathcal{L}^b$ is measured by a scalar function p:

$$p_{bk} = p(a_k, \mathfrak{L}^b) \sim \mathcal{P}_k = \mathcal{P}_k(\mathfrak{L})$$

The p_{bk} are samples drawn from the distribution $\mathcal{P}_k(\mathfrak{L})$ of the performance measure of the algorithm k on the data set $\mathfrak{L}$. The aim of a supervised learning task is to construct a learner $\hat{y} = a_k(x \mid \mathfrak{L}^b)$ for any observation $z = (y, x) \in \mathfrak{L}$.

The discrepancy between the true response y and the predicted response $\hat{y}$ for an arbitrary observation $z \in \mathfrak{L}$ is measured by a scalar loss function $l(y, \hat{y})$. The performance measure p is then defined by some functional μ of the distribution of the loss function over all observations:

$$p_{bk} = p(a_k, \mathfrak{L}^b) = \mu(l(y, a_k(\mathbf{x} \mid \mathfrak{L}^b))) \sim \mathcal{P}_k(\mathfrak{L})$$

With regard to the toy example, an exemplary loss function is the misclassification error which incurs loss 1 if an observation is wrongly classified

$$l(y, \hat{y}) = \begin{cases} 0 & y = \hat{y} \\ 1 & \text{otherwise,} \end{cases}$$

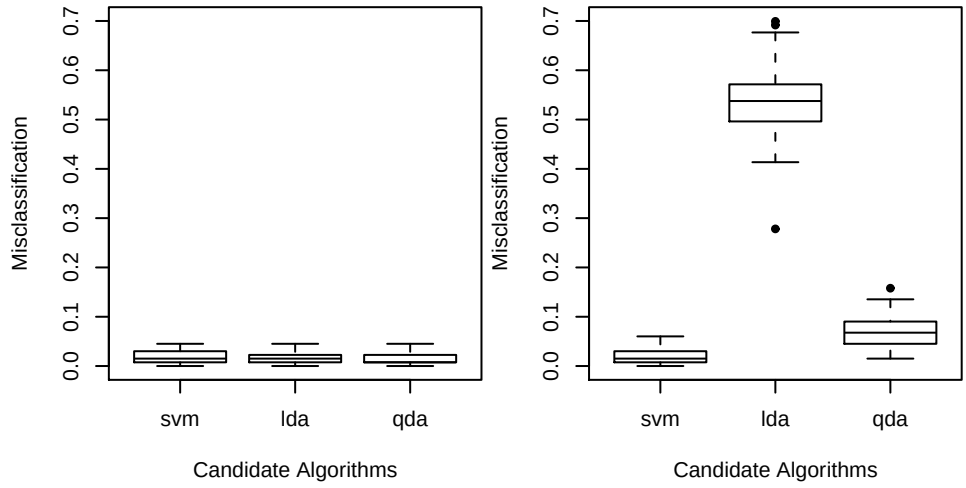

Figure 2: Performance measures of the candidate algorithms on (a) the linearly separable (`ds1`) and (b) the non-linearly separable (`ds2`) classification problems.

and a reasonable choice for the functional μ is the expectation. See [9] for further common loss functions and adequate choices for the functional μ. With a view to practicability in real-world applications, further interesting performance measures are execution time and memory requirements of the learning algorithms.

Using the performance on the learning data as a basis for further analyses is not a good idea, however, because it would reward overfitting. Thus – as in most cases we are not able to compute μ analytically – we use the empirical functional $\mu_{\mathfrak{T}}$ based on a test sample $\mathfrak{T}$:

$$\hat{p}_{bk} = \hat{p}(a_k, \mathfrak{L}^b) = \hat{\mu}_{\mathfrak{T}}(l(y, a_k(x \mid \mathfrak{L}^b))) \sim \hat{\mathcal{P}}_k(\mathfrak{L})$$

This means we compute the performance of the model (fitted on the learning sample $\mathfrak{L}^b$) for each observation in the test sample $\mathfrak{T}$ and apply the empirical functional $\hat{\mu}$ to summarize over all observations. The most common example for the empirical functional $\hat{\mu}$ is the mean, that is the empirical analogue of the expectation.

Most further analysis methods require independent observations of the performance measure, therefore we define the test sample $\mathfrak{T}$ in terms of out-of-bag observations: $\mathfrak{T} = \mathfrak{L} \setminus \mathfrak{L}^b$. Based on $\hat{p}_{bk}$ the empirical performance distribution $\hat{\mathcal{P}}_k(\mathfrak{L})$ for each candidate algorithm a_k on data set $\mathfrak{L}$ is estimated. In [9] we introduced a toolbox with exploratory and inferential tools to compare these empirical performance distributions and, finally, compute a statistically correct order of the candidate algorithms on the data set $\mathfrak{L}$.

Now, generally we are interested in the performance of the algorithms within a domain of problems. This domain is specified by a collection of data sets $\mathfrak{L}_1, \ldots, \mathfrak{L}_M$. A benchmark experiment is executed on each data set $\mathfrak{L}_m$, $m = 1, \ldots, M$ and an order of the algorithms is calculated. However, the performance-rankings of the candidate algorithms will vary over the different data sets in all realistic benchmark scenarios and one common assumption is that the performance depends on particular characteristics of the data sets. This paper proposes a first approach to answer the question which data set characteristics make the algorithms perform differently on certain data sets.

In case of the toy example ($\mathfrak{L}_1$ = `ds1`, $\mathfrak{L}_2$ = `ds2`, $B = 100$ with 2/3-subsampling, i.e. $n = 267$, as resampling scheme and p being the misclassification error), Figure 2 shows a boxplot of the misclassification error: As expected, a_1 = `svm` and a_2 = `qda` solve both problems very well; a_3 = `lda` solves the linearly separable problem very well, but has huge problems with the non-linearly separable one. The goal is now to provide a method which detects that `lda` has problems with data set `ds2` because of the non-linearly separable feature space.

2.2 Data set characterization

The question why certain algorithms perform better on particular data sets than others requires a possibility to describe the data sets. One common approach is to extract statistical and informative measures from the data sets; the STATLOG project [13] was the first one which defined such a set of structural characteristics. Newer approaches, e.g. [12] and [6], extend this set of data set characteristics and use them in terms of meta-learning – an approach to learn which algorithm is best suited for an unknown learning problem (see, e.g. [22]).

Given some user-specified characteristics, data set characterization can be seen as a two-step process: (1) *map* each data set into its individual characterization space; (2) *reduce* the individual characterization spaces into one common characterization space where all data sets are comparable, i.e. a metric can be defined. More formal, let $\mathcal{L}$ be the space of all data sets and $\mathfrak{L} \in \mathcal{L}$. The function

$$map\colon \mathcal{L} \to \mathbb{R}^* \quad \text{with} \quad \mathfrak{L} \mapsto x^*$$

computes one specific characteristic of a data set. $\mathbb{R}^*$ indicates that the dimension of the vector x^* can depend on the data set. For example, computing the skewness of each continuous input variable results in a vector

Characteristic	Description	ds1	ds2
obs.n	number of observations	400	400
var.n	number of variables	2	2
nvar.n	number of nominal variables	0	0
nvar.entropy	mean nominal variable entropy		
nvar.bin	number of binary variables	0	0
cvar.n	number of continuous variables	2	2
cvar.mac	mean multiple attribute correlation	0.06	0.06
cvar.skew	mean skewness	-0.08	-0.08
cvar.kurt	mean kurtosis	-1.20	-1.20
resp.cl	number of response classes	2	2
resp.entropy	mean response entropy	5.93	5.93
i2r.fcc	**first canonical correlation**	**0.86**	**0.04**
i2r.frac1	variation from first linear discriminant	1.00	1.00
i2r.mi	mean mutual information		
i2r.envar	equivalent number of variables		
i2r.nsratio	noise to signal ratio		

Table 1: Description of the characteristics used in this paper and its realization for the linearly separable (`ds1`) and non-linearly separable (`ds2`) classification problems.

with dimension equal to the number of continuous input variables of the given data set; on the other hand, computing the number of observations results in one number for every data set. The dimension of x^* can even be zero if, for example, a data set has no continuous input variables so that the characteristic is missing. This first step does not guarantee that all data sets are comparable, therefore another function red is defined as

$$red\colon \mathbb{R}^* \to \mathbb{R}^d \quad \text{with} \quad x^* \mapsto x^d$$

which reduces the dimension of characteristic vector x^* to dimension d identical for all data sets. Examples for such reduction functions are: the mean or a quantile (for which $d = 1$) or a histogram representation (for which d is chosen according to the number of bins) for characteristics like the skewness, that provide a value for each continuous variable in the data set; or the identity function (for which $d = 1$) for characteristics like the number of observations, that provide exactly one value for each data set in the first place.

A data set characterization then consists of a set of characteristics $\{(map_1, red_1), \ldots, (map_J, red_J)\}$, and for a given data set $\mathfrak{L}$, its characterization is the vector $c = (c_1, \ldots, c_J)$ with

$$c_j = red_j(map_j(\mathfrak{L})),\ j = 1, \ldots, J.$$

This framework allows a sound and flexible definition of data set characterization (and a simple implementation in software, see Section 5). Common characteristics, like those defined in [13], [12] and [6] can be formulated in terms of this map/reduce framework. As already noted, the STATLOG project defined the first characteristics which are broadly established nowadays. To simplify matters we make use of most of their characteristics together with some simple additional ones. Table 1 provides a list of typical data set characteristics (for a detailed description we refer to the original paper). With respect to our notation, map_j, $j = 1, \ldots, J$, corresponds to the different characteristics in Table 1 and red_j was chosen to be the mean for all characteristics. Columns `ds1` and `ds2` of Table 1 show the characterization of the two data sets in case of the toy example. As the data sets are constructed in way that they match in all characteristics except the linearly separability, the first canonical correlation (i2r.fcc) is the only characteristic that differs; the first canonical correlation ranges between $[0, 1]$, whereas 1 means linearly separable and 0 not linearly separable.

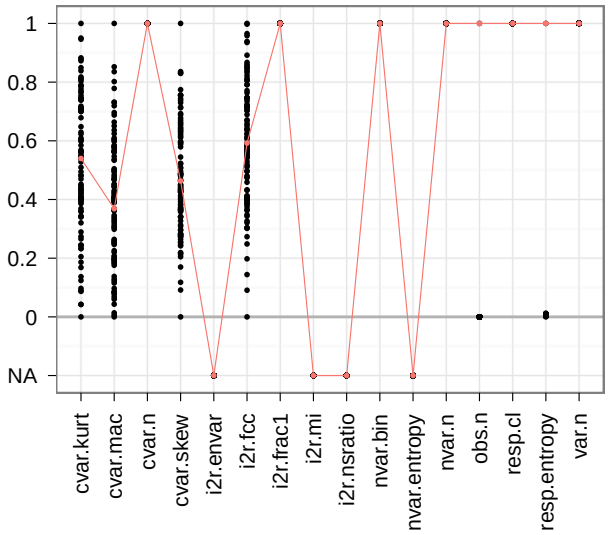

Figure 3: Relative variation of the characterizations of the 100 drawn samples in case of `ds1`. The red line marks the characterization of the original data set; NA means that this characteristic is not available on this data set.

Now, extending the benchmark experiment framework with the calculation of data set characteristics al-

lows us to determine the influence of data set characteristics on the performance of algorithms: for each sample b drawn from the original data set m, the benchmark experiment provides (1) the performance of the candidate algorithms p_{mbk} $(k = 1, \ldots, K)$, and (2) the characterization of the sample $c_{mb} = (c_{mb1}, \ldots, c_{mbJ})$ with $c_{mbj} = red_j(map_j(\mathfrak{L}_m^b))$. Note that some characteristics could vary between samples drawn from the same data set, while others definitely stay constant. For example, the mean response entropy (resp.entropy) could vary depending on how many observations are drawn from each class, whereas the number of variables (var.n) always stays constant. Figure 3 shows the relative variation of the characterization of the 100 drawn samples in case of `ds1`.

The result of such an experiment is a collection of tuples of the performance of each algorithm on each learning sample and the characterizations of each sample. The corresponding data matrix is of the form of a tabular with $B \cdot M$ rows (for B samples drawn from M original data sets) and $K + J$ columns (for the performances measures of K algorithms and the J data set characteristics):

	K algorithms			J characteristics		
B samples from data set 1	p_{111}	$\cdots$	p_{11K}	c_{111}	$\cdots$	c_{11J}
	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	p_{1B1}	$\cdots$	p_{1BK}	c_{1B1}	$\cdots$	c_{1BJ}
	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
B samples from data set M	p_{M11}	$\cdots$	p_{M1K}	c_{M11}	$\cdots$	c_{M1J}
	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	p_{MB1}	$\cdots$	p_{MBK}	c_{MB1}	$\cdots$	c_{MBJ}

2.3 Preference scaling

The basic model for preference scaling, that is used here for comparing the performance of different candidate algorithms on a variety of data sets, is the Bradley-Terry model [2]. Here, we consider the formulation by [5], that allows for ties so that each comparison of the performance of two algorithms has three possible outcomes: (1) the first algorithm wins, (2) the second algorithm wins, or (3) both algorithms perform equally (i.e., a tie).

Here, we conduct paired comparisons of the performance measures for the K algorithms: For K algorithms there are $\frac{K\cdot(K-1)}{2}$ paired comparisons. Accordingly, the left part of the data matrix illustrated above is now replaced by a $B \cdot M \times \frac{K\cdot(K-1)}{2}$ table with an entry for each paired comparison (columns) on each data set (rows):

	$K \cdot (K-1)/2$ comparisons		
B samples from data set 1	$R(p_{111}, p_{112})$	$\cdots$	$R(p_{11K-1}, p_{11K})$
	$\vdots$	$\ddots$	$\vdots$
	$R(p_{1B1}, p_{1B2})$	$\cdots$	$R(p_{1BK-1}, p_{1BK})$
	$\vdots$	$\ddots$	$\vdots$
B samples from data set M	$R(p_{M11}, p_{M12})$	$\cdots$	$R(p_{M1K-1}, p_{M1K})$
	$\vdots$	$\ddots$	$\vdots$
	$R(p_{MB1}, p_{MB2})$	$\cdots$	$R(p_{MBK-1}, p_{MBK})$

The entries of the table are the relations $R(p_{mbk}, p_{mbk'})$ describing one of the outcomes (1), (2) or (3) of the comparison of algorithms k and k' on sample b drawn from data set m. Since this new data matrix still has one row for each sample b drawn from data set m, it is compatible with the table of data set characteristics from the above illustration. Thus, the complete data matrix used for the following analysis consists of a $B \cdot M \times \frac{K\cdot(K-1)}{2}$ table for the paired comparisons and a $B \cdot M \times J$ table for the data set characteristics, that will be used to identify groups of data sets between which the performance comparisons of the algorithms differ.

When we now consider the paired comparisons, according to the Bradley-Terry model the three possible outcomes have the probabilities:

$$P\left(R(p_{mbk}, p_{mbk'}) = 1\right) = \frac{\pi_k}{\pi_k + \pi_{k'} + \nu\sqrt{\pi_k \pi_{k'}}},$$

$$P\left(R(p_{mbk}, p_{mbk'}) = 2\right) = \frac{\pi_{k'}}{\pi_k + \pi_{k'} + \nu\sqrt{\pi_k \pi_{k'}}},$$

$$P\left(R(p_{mbk}, p_{mbk'}) = 3\right) = \frac{\nu\sqrt{\pi_k \pi_{j'}}}{\pi_k + \pi_{k'} + \nu\sqrt{\pi_k \pi_{k'}}},$$

where $\pi_k \geq 0$, $k = 1, \ldots, K$, are the parameters indicating the strength of each algorithm, and $\nu \geq 0$ is a discrimination constant governing the probability of ties. For parameter estimation via maximum likelihood, one restriction is necessary: usually, either one parameter is fixed to zero or the sum of all parameters constrained to 1, as in the following illustrations.

In order to assess whether there are groups of data sets with certain characteristics, for which the performance-rankings of the candidate algorithms – and thus the parameters indicating the strength of each algorithm in the Bradley-Terry model – differ systematically, the model-based partitioning approach of [19] is used. The algorithm for model-based recursive partitioning is an extension of the popular CART algorithm for classification and regression trees [3] and consists of the following consecutive steps:

1. Fit a Bradley-Terry model for the paired comparisons of the algorithms based on all data sets in the current node (starting with the root node including all data sets).

2. Assess the stability of the Bradley-Terry model parameters with respect to each characteristic of the data sets.

3. If there is significant instability in the model parameters, split the data sets in two nodes along the characteristic with the strongest instability, and use the cutpoint with the highest improvement of the model fit.

4. Repeat steps 1–3 recursively in the resulting nodes until there are no more significant instabilities (or the number of data sets left in a node falls below a given stopping value).

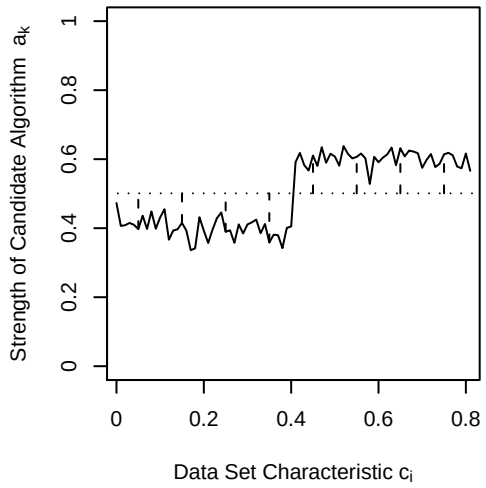

Figure 4: Parameter instability in the strength of a candidate algorithm (simplified illustration).

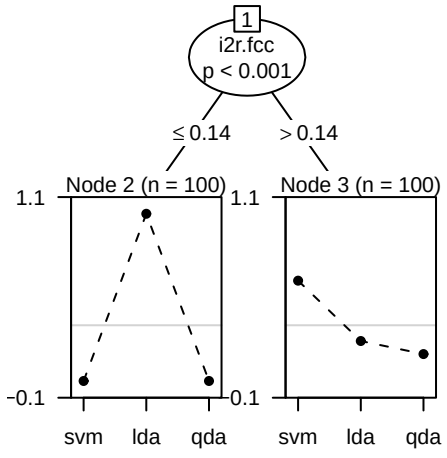

Figure 5: Partitioned paired comparison model.

The statistical framework employed for testing the significance of instabilities in the model parameters is described in detail in [19] and [24]. The rationale of the underlying tests for parameter instability is that the individual deviations from a joint model are considered over the range of each potential splitting variable: As illustrated in Figure 4, the strength parameter for algorithm a_k may show a systematic change when considered over the range of the characteristic c_j – such as the first canonical correlation, that indicates linear separability – while over the range of other characteristics the parameter may vary only randomly. Statistical tests are available to detect such systematic parameter instabilities and select the splitting variable or characteristic inducing the strongest instability [19, 24].

When the model-based partitioning approach of [19] is employed to detect groups of data sets with certain characteristics for which the performance-rankings of the candidate algorithms differ, the resulting partition can be displayed as a tree, as illustrated for the artificial toy example in Figure 5: From all available characteristics in Table 1, the first canonical correlation i2r.fcc – that indicates whether the data set is linearly separable – is correctly identified as the characteristic that induces a significant change in the performance-ranking of the algorithms. For the 100 samples from data set `ds1`, that is not linearly separable, the values of the characteristic i2r.fcc are low and `lda` performs poorly (left node in Figure 5), while for the 100 samples from data set `ds2`, that is linearly separable, the values of the characteristic i2r.fcc are high and `lda` performs well (whereas `svm` performs slightly worse when compared to the other algorithms, as displayed in the right node in Figure 5).

3 Application example

This section demonstrates the framework by means of well-known problems from the UCI Machine Learning Repository [1] and common learning algorithms. Computational details of the application example are described in Section 5.

The application example consists of $M = 13$ data sets, that are all binary classification problems but cover a wide area of data set characteristics. Figure 6 names all data sets and shows the relative variation of their characteristics (again the characteristics listed in Table 1 are used, i.e. $J = 16$). The $K = 6$ algorithms of interest are linear discriminant analysis (`lda`), k-nearest neighbor classifier (`knn`), classification trees (`rpart`), support vector machines (`svm`), neural networks (`nnet`) and random forests (`rf`) (see, e.g. [10]). The performance measure p is the misclassification error and we draw $B = 100$ samples using 2/3-subsampling without replacement.

The results of [18] and our preliminary experiments have shown that subsampling – rather than bootstrap sampling – is the resampling method of choice for this kind of benchmark experiments. The reason for this is that bootstrap sampling with replacement can induce artifacts in measures of the association between attributes, such as the entropy or the χ^2-statistic [18].

On each data set the benchmark experiment is computed and the result is a 1300×15 table with paired comparisons of the algorithms and the corresponding 1300×16 table with data set characteristics. To fit the recursively partitioned Bradley-Terry model, categorical and ordinal characteristics are employed directly, while continuous characteristics are discretized based on their quantiles. (This discretization discards the or-

der information, but allows to treat missing values as an extra category in a straightforward way – and if the order is important the model will find it nevertheless). We require a minimum of 200 observations per node; here the idea is to create nodes that contain in average the samples of at least two data set, so that the resulting partition is well interpretable. Based on this setup, Figure 7 shows the resulting tree.

Focusing first on the rightmost node (Node 11) in Figure 7, it becomes clear that the performance-rankings of the 6 algorithms is conditional on the domain specified by the data sets: Node 11 consists of big (obs.n > 667) and linearly well separable (i2r.fcc > 0.520) data sets. On these data sets, all algorithms perform well except classification trees. This is plausible from the method of operating of classification trees, where a linear separation can only be roughly approximated by means of stepwise binary partitioning. Another example where one algorithm is clearly inferior is displayed in Node 9, where the data sets are hardly linearly separable (i2r.fcc ≤ 0.520) and the dimensionality is low (var.n ≤ 2). In this case, linear discriminant analysis performs poor for obvious reasons, while the remaining algorithms perform well. With increasing dimensionality (var.n > 2, Node 10) it appears that support vector machines perform best, followed by random forests and neural networks; k-nearest neighbor classifiers perform equal to classification trees and again linear discriminant analysis comes last.

The data sets that are grouped in the remaining three nodes on the left hand side of the tree (Nodes 4, 6 and 6) based on their smaller samples sizes (obs.n ≤ 667) also show clear differences in the performance-rankings, but here the tree provides no reasonable explanation: the only characteristic used for splitting in this part of the tree is the number of observations obs.n. However, this characteristic may only serve as a proxy for other differences between the data sets, that are not yet covered by our list of characteristics.

This application example nicely illustrates the main challenge of the proposed method: the selection of the "right" data set characteristics. However, the benefit of the proposed method is that – at least from the set of characteristics provided – the relevant ones are selected for splitting automatically. In further consequence this allows to compute a huge set of characteristics, (e.g. join all characteristics available in the three papers cited in the introduction); the relevant ones are chosen automatically.

4 Summary and outlook

This paper proposes a formal framework to determine the influence of data set characteristics on the performance of learning algorithms. The framework is a combination of the three methods, benchmark experiments, data set characterization and recursively partitioning Bradley-Terry models. Benchmark experiments are used to generate performance information for algorithms and relevant data set characteristics on various data sets. The recursively partitioning Bradley-Terry model then employs splits in characteristics which induce a significant change in the performance-ranking of the algorithms. Advantages of the resulting trees are that (1) they are easy to interpret by means of visualization and (2) from a potentially large number of data set characteristics those that correspond to a significant change in the performances of the algorithms are automatically detected.

The approach can be used both for exploring the impact of characteristics of a given sample of data sets, like the ones from UCI Machine Learning Repository [1] used in our example, and for analyzing the results of simulation experiments, where certain characteristics of data sets are systematically varied and the aim is to test their effect on the performance of candidate algorithms.

In either case, it is important to note that – due to the fact that the number of boostrap- or subsamples drawn form given data sets, and the number of samples drawn from a data generating process in a simulation study is arbitrary – one can detect very small performance differences with very high power when the number of learning samples B is large (see also [11]).

Future work will also include investigations on the stopping criteria for the recursive partitioning algorithm. For example, the number of minimum observations per node can be chosen greater than B as in our example, but it could also be chosen smaller than B, which would result in more than one node for a data set and could uncover different performance-rankings on sub-samples from one data set with different characteristics.

5 Computational details

All computations and graphics have been done using R 2.10.1 [17] and the package **ggplot2** 0.8.5 [23].

The proposed method is implemented in the package **benchmark** 0.1 [8]; it relies on the package **psychotree** 0.9-1 [19] for recursively partitioning Bradley-Terry models.

In case of the application example the following functions and packages are used: Function `lda` (package **MASS** [21]) for linear discriminant analysis. Function `knn` (package **class** [21]) for k-nearest neighbor classifiers; the hyperparameter k (the number of neighbors) is determined between 1 and $\sqrt{N}$, N the number of observations, using 10-fold cross-validation (function `tune.knn`, package **e1071** [7]). Function `nnet` (package **nnet** [21]) for neural networks; the number of hidden units is determined between 1 and $\log(N)$ using 10-fold cross-validation (function `tune.nnet`, package **e1071**). Function `rpart`(package **rpart** [20]) for classification trees; the 1-SE rule is used to prune the trees. Function `svm` (package **e1071**) for C-classification support vector machines; the two hyperparameters γ (the cost of constraints violation) and C (the kernel parameter) are determined using a grid search over the two-dimensional

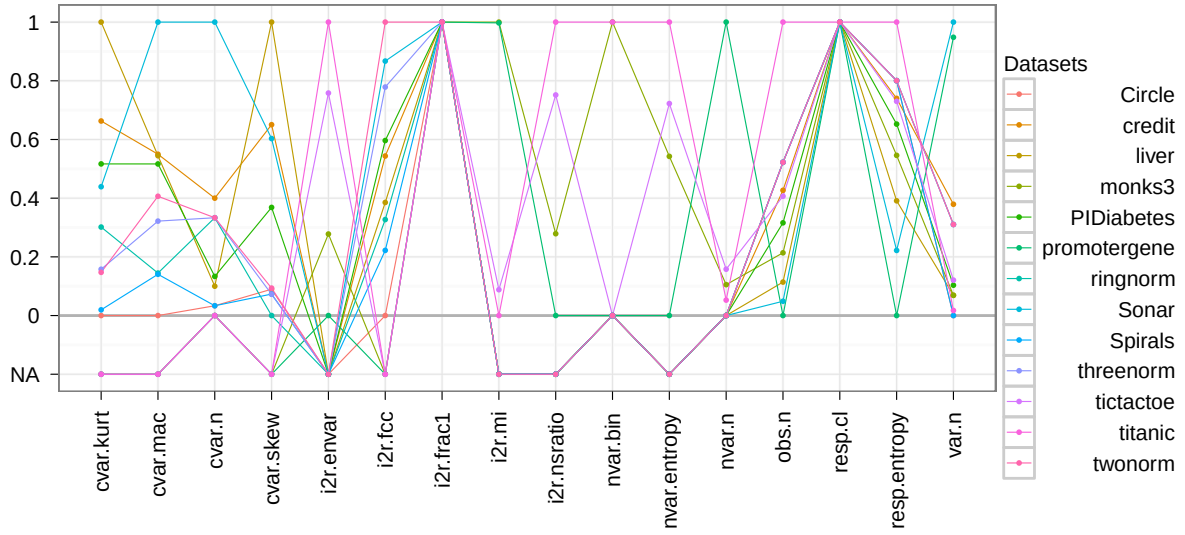

Figure 6: Relative characterization of the UCI machine learning repository data sets.

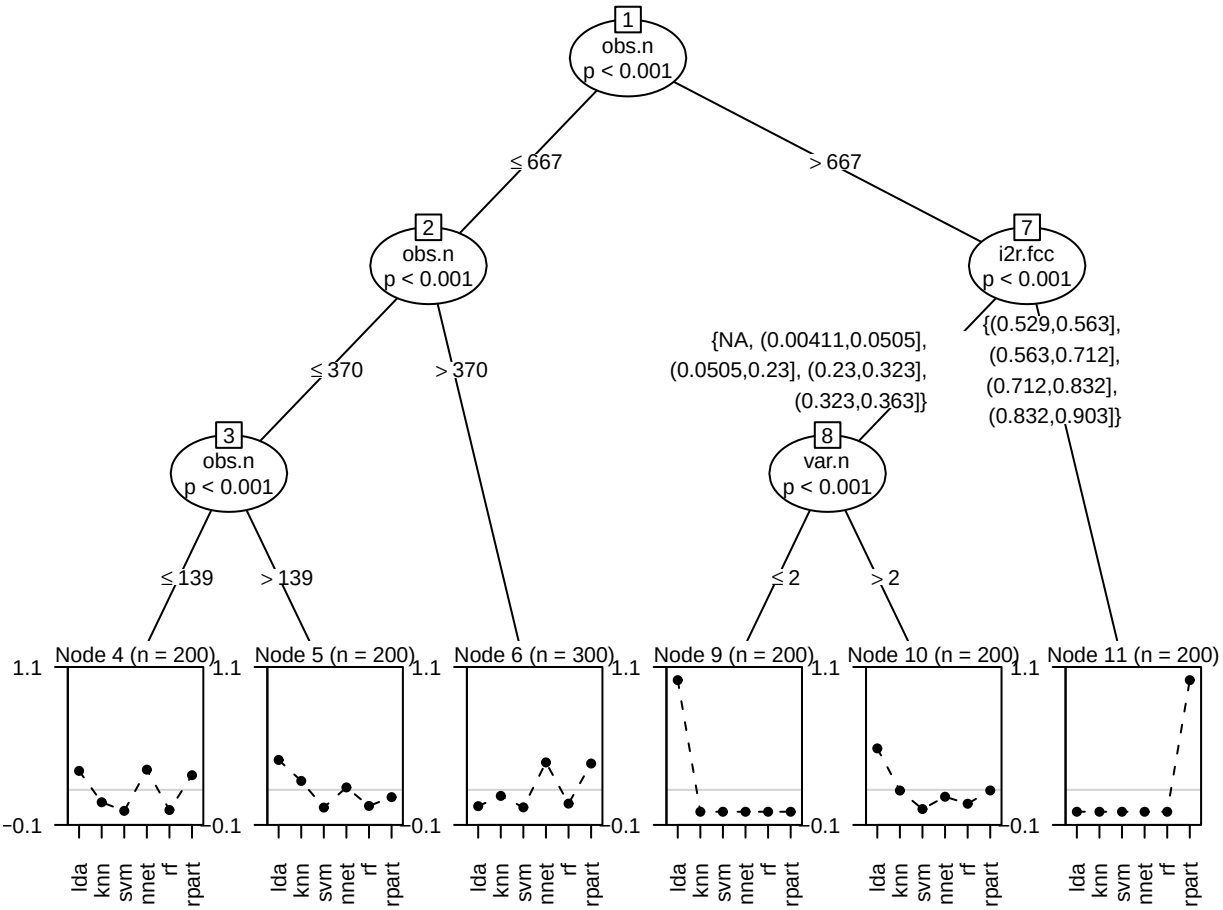

Figure 7: Partitioned paired comparison model for the thirteen UCI machine learning repository data sets and the six candidate algorithms.

parameter space (γ, C) with $\gamma \in [2^{-5}, 2^{12}]$ and $C \in [2^{-10}, 2^5]$ (function `tune.svm`, package **e1071**). Function `randomForest` (package **randomForest** [14] for random forests.

R itself and all packages used are freely available under the terms of the General Public License from the Comprehensive R Archive Network at `http://CRAN.R-project.org`. Code and precalculated results for replicating our analysis is available from the first author's website at `http://www.statistik.lmu.de/~eugster/benchmark/`.

6 Acknowledgments

The authors would like to thank Achim Zeileis for valuable discussions and helpful comments. Carolin Strobl is supported by grant STR1142/1-1 ("Methods to Account for Subject-Covariates in IRT-Models") from the German Research Foundation (Deutsche Forschungsgemeinschaft).

References

[1] A. Asuncion and D. Newman. UCI machine learning repository, 2007.

[2] R. A. Bradley and M. E. Terry. Rank analysis of incomplete block designs. I. the method of paired comparisons. *Biometrica*, 39(3/4):324–345, 1952.

[3] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman and Hall, New York, 1984.

[4] R. Caruana, N. Karampatziakis, and A. Yessenalina. An empirical evaluation of supervised learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

[5] D. E. Critchlow and M. A. Fligner. Paired comparison, triple comparison, and ranking experiments as generalized linear models, and their implementation on GLIM. *Psychometrika*, 56(3):517–533, 1991.

[6] M. C. P. de Souto, R. B. C. Prudencio, R. G. F. Soares, D. A. S. Araujo, I. G. Costa, T. B. Ludermir, and A. Schliep. Ranking and selecting clustering algorithms using a meta-learning approach. In *International Joint Conference on Neural Networks*, 2008.

[7] E. Dimitriadou, K. Hornik, F. Leisch, D. Meyer, , and A. Weingessel. **e1071***: Misc Functions of the Department of Statistics (e1071), TU Wien*, 2009. R package version 1.5-22.

[8] M. J. A. Eugster. **benchmark***: Benchmark Experiments Toolbox*, 2010. R package version 0.1, development version available from `http://r-forge.r-project.org/projects/benchmark`.

[9] M. J. A. Eugster, T. Hothorn, and F. Leisch. Exploratory and inferential analysis of benchmark experiments. Technical Report 30, Institut für Statistik, Ludwig-Maximilians-Universität München, Germany, 2008. Submitted.

[10] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, second edition, 2009.

[11] T. Hothorn, F. Leisch, A. Zeileis, and K. Hornik. The design and analysis of benchmark experiments. *Journal of Computational and Graphical Statistics*, 14(3):675–699, 2005.

[12] A. Kalousis and M. Hilario. Model selection via meta-learning: A comparative study. In *Proceedings of the 12th International IEEE Conference on Tools with Artificial Intelligence*, 2000.

[13] R. D. King, C. Feng, and A. Sutherland. STATLOG: Comparison of classification algorithms on large real-world problems. *Applied Artifcial Intelligence*, 9, 1995.

[14] A. Liaw and M. Wiener. Classification and regression by **randomForest**. *R News*, 2(3):18–22, 2002.

[15] T.-S. Lim, W.-Y. Loh, and Y.-S. Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40(3), 2000.

[16] J. R. Quinlan. *C4.5: Programms for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, 1993.

[17] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2009. ISBN 3-900051-07-0.

[18] C. Strobl, A.-L. Boulesteix, A. Zeileis, and T. Hothorn. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics*, 8(25), 2007.

[19] C. Strobl, F. Wickelmaier, and A. Zeileis. Accounting for individual differeneces in Bradley-Terry models by means of recursive partitioning. *Journal of Educational and Behavioral Statistics*, 2010. Accepted for Publication.

[20] T. M. Therneau and B. A. R. port by Brian Ripley. **rpart***: Recursive Partitioning*, 2010. R package version 3.1-46.

[21] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. ISBN 0-387-95457-0.

[22] R. Vilalta and Y. Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2), 2002.

[23] H. Wickham. **ggplot2***: Elegant Graphics for Data Analysis*. Springer New York, 2009.

[24] A. Zeileis, T. Hothorn, and K. Hornik. Model-based recursive partitioning. *Journal of Computational and Graphical Statistics*, 17(2):492–514, 2008.