Stat Comput (2016) 26:1039–1057
DOI 10.1007/s11222-015-9588-z

Improved nearest neighbor classifiers by weighting and selection of predictors

Gerhard Tutz[1] · Dominik Koch[1]

Received: 15 June 2014 / Accepted: 23 June 2015 / Published online: 5 July 2015

Abstract Nearest neighborhood classification is a flexible classification method that works under weak assumptions. The basic concept is to use the weighted or un-weighted sums over class indicators of observations in the neighborhood of the target value. Two modifications that improve the performance are considered here. Firstly, instead of using weights that are solely determined by the distances we estimate the weights by use of a logit model. By using a selection procedure like lasso or boosting the relevant nearest neighbors are automatically selected. Based on the concept of estimation and selection, in the second step, we extend the predictor space. We include nearest neighborhood counts, but also the original predictors themselves and nearest neighborhood counts that use distances in sub dimensions of the predictor space. The resulting classifiers combine the strength of nearest neighbor methods with parametric approaches and by use of sub dimensions are able to select the relevant features. Simulations and real data sets demonstrate that the method yields better misclassification rates than currently available nearest neighborhood methods and is a strong and flexible competitor in classification problems.

Keywords Nearest neighborhood methods · Classification · Lasso · Boosting · Logit model · Random forests · Support vector machine

✉ Gerhard Tutz
tutz@stat.uni-muenchen.de;
gerhard.tutzg@stat.uni-muenchen.de

[1] Ludwig-Maximilians-Universität München, Akademiestraße 1, 80799 Munich, Germany

1 Introduction

Classification by nearest neighbors is an established non-parametric tool, which does not assume a specific form of the boundaries between classification regions as linear or quadratic discrimination does. Therefore it is very adaptive and can be used in situations where no reasonable assumption on the distribution of variables is available. The basic concept to use the class labels in the neighborhood of the target value to be classified goes back to Fix and Hodges (1951). The k-NN method, which uses the k nearest neighbors, has been shown to be asymptotically optimal for increasing k if the number of observations increases at a proper rate (Stone 1977).

Since then various extensions of the basic k-NN method have been proposed. Instead of using a fixed number of nearest neighbors one can assign weights to the labels of the neighbors as proposed by Morin and Raeside (1981), Parthasarthy and Chatterji (1990) and Silverman and Jones (1989). Friedman (1994) proposed local flexible weights for the predictors to account for their local relevance. Ghosh (2007) used spatially adaptive selection of the number of nearest neighbors, Ghosh (2012) showed how to extract useful information also from unlabeled test cases.Alternative approaches which choose the metric adaptively were given by Hastie and Tibshirani (1996) and Domeniconi et al. (2002). A connection between these adaptively chosen metrics and random forests was derived by Lin and Jeon (2006). A detailed overview on further methods of nearest neighborhoods is found in Ripley (1996), for basic concepts see also Hastie et al. (2009).

A key issue in nearest neighbor classification is the selection of the number of nearest neighbors or, in weighted nearest neighbor methods, the selection of the corresponding tuning parameter. Another important topic, in particular in higher dimensions, is the selection of the relevant dimensions, since the computation of distances may suffer from the curse of dimensionality yielding poor performance in high dimensional settings. These problems are tackled by a novel approach to compute the weights on nearest neighbors. Instead of using weights that are determined by distances we use the labels of the nearest neighbors as predictors in a regression model and estimate the corresponding parameters by regularized estimation procedures that enforce selection of predictors, for example lasso (Tibshirani 1996) or boosting (Bühlmann and Yu 2003); (Bühlmann and Hothorn 2007). One strength of the approach is that the relevant neighbors are automatically selected. The main advantage, however, is that one can, in addition to the labels of the nearest neighbors include the original variables and/or transformations of them. The selection generated by regularized estimation lets the data decide if the nearest neighbors or the original predictors are the most relevant features for classification purposes. A further extension is the inclusion of nearest neighbors computed for single or subsets of predictors, which allows to select the relevant nearest neighbor dimensions. The method is an extension of the nearest neighbor approach proposed by Holmes and Adams (2003). The authors also use a logit representation of the nearest neighbors and consider the option to include original variables in the predictor. However, they rely on a stepwise selection procedure to obtain a final model and do not include distances built on single predictors.

The method proposed here can also be seen as a specific ensemble method. In general, ensemble methods as bagging (Breiman 1996a) and stacking (Breiman 1996c) combine a collection of simple base models. An overview on ensemble learning is found, for example, in Hastie et al. (2009). The ensemble considered here combines original variables and nearest neighbors on subsets of the predictor space. Alternative ensemble methods including nearest neighbors have been considered before. Paik and Yang (2004) introduced a method called adaptive classification by mixing (ACM). They used combinations of k-NN classifiers with different values for k and subsets of predictors, but used a quite different way to weight the candidate models, namely by sub-sampling and evaluation. Domeniconi and Yan (2004) investigated ensembles of nearest neighbor classifiers based on random subsets of predictors while performing adaptive sampling. The approach is a two-step procedure, because in a first step the relevance of all the feature needs to be determined by an adequate technique. Gertheiss and Tutz (2009) considered a linear model for the probability and minimized a specific loss function to obtain weights. A common feature of these ensemble methods is that they combine neighborhood methods but never include original variables, which is a strength of the method proposed here. A further combination of nearest neighborhood methods has been considered by Ghosh and Godtliebsen (2012).

In Sect. 2 we consider the basic nearest neighbor methods and introduce the basic estimation concept. In Sect. 3 it is extended to include original predictors and nearest neighbors defined by single predictors. Section 4 is devoted to the multi-class problem.

2 Classical and extended nearest neighborhood methods

2.1 Basic nearest neighborhood methods

k-Nearest-neighbor method

Let the learning sample be given by $(y_i, \boldsymbol{x}_i)$, $i = 1, \ldots, n$, $y_i \in \{0, 1\}$, where y_i represents the class and $\boldsymbol{x}_i$ a vector of predictors. Moreover, let $d(\boldsymbol{x}, \tilde{\boldsymbol{x}})$, $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in R^p$ denote a distance in feature space, which is used to define the nearest neighbors. The basic procedure is simple. One determines for a new observation $\boldsymbol{x}_0^T = (x_{01}, \ldots, x_{0p})$ the k observation points that are closest in distance to $\boldsymbol{x}_0$. This means that one seeks the nearest neighbors $\boldsymbol{x}_{(1)}, \ldots, \boldsymbol{x}_{(k)}$ with

$$d(\boldsymbol{x}_0, \boldsymbol{x}_{(1)}) \leq \cdots \leq d(\boldsymbol{x}_0, \boldsymbol{x}_{(k)}),$$

where $\boldsymbol{x}_{(1)}, \ldots, \boldsymbol{x}_{(k)}$ are values from the learning sample. With $y_{0(1)}, \ldots, y_{0(k)}$ denoting the corresponding classes, one classifies, using the majority vote rule, by

$$\hat{\delta}(\boldsymbol{x}_0) = r \Leftrightarrow \text{class } r \text{ is the most frequent class in} \\ \times \{y_{0(1)}, \ldots, y_{0(k)}\}.$$

Thus, for a given $\boldsymbol{x}_0$, one looks within the learning sample for almost perfect matches of $\boldsymbol{x}_0$ and then classifies by using the class labels of these observations. If ties occur, they are broken at random. The resulting classifier is called the *k-nearest-neighbor* (k-NN) classifier. The number of neighbors k is a tuning parameter that can be chosen by cross validation. Although the simple next neighbor rule 1-NN often performs remarkably well, typically performance is improved by increasing the number of neighbors. This is supported by large sample behavior. Stone (1977) showed that the risk for the k-NN rule converges in probability to the Bayes risk under the assumptions $k \to \infty$ and $k/n \to 0$. Further asymptotic results were given by Ripley (1996). More recently, Hall et al. (2008) derived the optimal asymptotic order of nearest neighbors.

Weighted nearest neighbor method

An obvious extension of nearest neighbor methods that ameliorates the dependence on the choice of the number of nearest neighbors is the weighted k-nearest-neighbor method. Let

$K(.,.)$ denote a symmetric kernel function. Simple examples are the tri-cube or Gaussian kernel.

Define weights based on the kernel function by

$$w(\boldsymbol{x}_0, \boldsymbol{x}_{0(j)}) = K(d(\boldsymbol{x}_0, \boldsymbol{x}_{0(j)})/\tau) \Big/ \sum_{j=1}^{k} K(d(\boldsymbol{x}_0, \boldsymbol{x}_{0(j)})/\tau),$$

where τ is the window width. One obtains the weighted nearest neighbor classifier

$$\hat{\delta}(\boldsymbol{x}_0) = 1 \Leftrightarrow \hat{\pi}(\boldsymbol{x}_0) = \sum_{j=1}^{k} w(\boldsymbol{x}_0, \boldsymbol{x}_{0(j)}) y_{0(j)} \geq 0.5.$$

The estimate $\hat{\pi}(\boldsymbol{x}_0)$ of $\pi(\boldsymbol{x}_0) = P(y = 1|\boldsymbol{x}_0)$ is a loess-type estimate, which corresponds to a Nadaraya–Watson estimate for binary responses, see Nadaraya (1964), Watson (1964) or Simonoff (1996).

The method uses two tuning parameters, the number of next neighbors and the smoothing parameter τ. But if k is chosen large, in the extreme case by $k = n$, the smoothing parameter τ becomes the only tuning parameter that determines how many neighbors are actually used. Of course, if one uses kernels with a finite support, the number of the nearest neighbors is automatically determined by τ, see also Ghosh and Godtliebsen (2012).

An alternative form of the estimate uses the transformed data $\tilde{y}_i = 2y_i - 1$, which replaces the class labels 0, 1 by $-1, 1$. With $\tilde{y}_{0(j)} = 2y_{0(j)} - 1$ denoting the transformed data of the nearest neighbors the score that can be used is

$$s(\boldsymbol{x}_0) = \sum_{j=1}^{k} w(\boldsymbol{x}_0, \boldsymbol{x}_{0(j)}) \tilde{y}_{0(j)},$$

which yields the equivalent classifier with cut point zero

$$\hat{\delta}(\boldsymbol{x}_0) = 1 \Leftrightarrow s(\boldsymbol{x}_0) \geq 0.$$

The representation is in particular useful for extended versions to be considered in the following.

2.2 Alternative forms of weighted nearest neighbors

The weighted nearest neighborhood classifier considered in the previous section uses a linear score with weights that are solely determined by the distances. In this section we consider alternative ways to obtain a score with weights, in particular, by explicitly estimating the weights.

An alternative representation of the classifier uses the logit model. Although a linear model could be used if only nearest neighbors are included, the logit model has the advantage that further variables can be included without violating the restriction that probabilities have to be from the interval $(0, 1)$. Moreover, in generalized linear models terminology, the logit model uses the canonical link which has some computational advantages. With the score from the previous section a logit representation of the classifier is

$$\hat{\delta}(\boldsymbol{x}_0) = 1 \Leftrightarrow \hat{\pi}(\boldsymbol{x}_0) = \frac{\exp(s(\boldsymbol{x}_0))}{1 + \exp(s(\boldsymbol{x}_0))} \geq 0.5.$$

The score $s(\boldsymbol{x}_0)$ is linear in the nearest neighbor dummy variables $\tilde{y}_{0(1)}, \ldots, \tilde{y}_{0(k)}$. Therefore, one might estimate directly the logit model $\pi(\boldsymbol{x}_i) = \exp(\eta_i)/(1 + \exp(\eta_i))$ with linear predictor

$$\eta_i = \gamma_0 + \sum_{j=1}^{k} \tilde{y}_{i(j)} \gamma_j = \gamma_0 + \tilde{\boldsymbol{y}}_{i,\mathrm{NN}}^T \boldsymbol{\gamma}, \qquad (1)$$

where $\tilde{\boldsymbol{y}}_{i,NN}^T = (\tilde{y}_{i(1)}, \ldots, \tilde{y}_{i(k)})$ is the vector of the k nearest neighbors. It is easily shown that for $k = 1$ and positive weight $\gamma > 0$ (and additionally omitted constant γ_0) the rule yields the same classifier as the 1-NN classifier.

In the general case, with k next neighbors in the predictor the classifiers are not equivalent but typically show similar performance. The approach has several advantages. With k chosen large as in weighted nearest neighbor methods it is not necessary that a tuning parameter has to be selected by cross validation. The weights on nearest neighbors are automatically determined by the fitting of a logit model. Therefore, probabilities are estimated in a one step procedure without the choice of tuning parameters. Moreover, the intercept γ_0 automatically accounts for the mixing proportions of different classes in the training set. In this section it is treated as an alternative to weighted nearest neighbor methods, the real potential of the approach is exploited in later sections.

Since it is not known how many neighbors should be used, k is typically chosen large in weighted NN methods. In estimation a large number of predictors can raise problems. Although estimates are easily stabilized by simple ridging, it is much more attractive to use regularized estimates that select predictors. One candidate is the lasso, which was introduced by Tibshirani (1996) for linear models, but is also available for the logit model, see, for example, Park and Hastie (2007). The lasso maximizes the penalized log-likelihood

$$l_p(\boldsymbol{\beta}) = l(\boldsymbol{\beta}) - \frac{\lambda}{2} J(\boldsymbol{\beta}),$$

where $l(\boldsymbol{\beta})$ is the usual log-likelihood of the logit model, λ is a tuning parameter, and the penalty term $J(\boldsymbol{\beta})$ is given by the L_1-norm $J(\boldsymbol{\beta}) = \sum_{j=1}^{k} |\gamma_j|$. The use of the L_1-norm in the penalty enforces that some of the coefficients are shrunk toward zero and therefore the relevant variables are selected

(Tibshirani 1996). The effect is that the method for appropriately chosen tuning parameter automatically selects the relevant nearest neighbors. As will be demonstrated in the following it is not necessary to select among all the nearest neighbors. There is no loss of accuracy if one uses summary measures that contain the nearest neighbors. Instead of including the whole set of neighbors we use the sums of nearest neighborhood values. For observation $\boldsymbol{x}_i$ the sum

$$s_{i(k)} = \sum_{j=1}^{k} y_{i(j)}$$

is the number of responses "1" among the k nearest neighbors in the training sample. We use the predictor

$$\eta_i = \gamma_0 + \sum_{j=1}^{m} s_{i(j)}\gamma_j,$$

which includes all the sums with a maximal value m. When using the lasso the relevant neighborhoods are automatically selected and weighted. We use a slightly reduced version by including only sums with k odd. This reduces the computation time, which becomes important in extensions considered in later sections where different forms of neighbors are included. The omission of neighborhoods over even numbers is not crucial since asymptotically nearest neighborhood methods with $2k$ and $2k-1$ have equal performance (Ripley 1996). For the classification of a new observation $\boldsymbol{x}_0$ one simply computes the sums $s_{0(k)} = \sum_{j=1}^{k} y_{0(j)}$, which represents the number of responses "1" for the k nearest neighbors of the new observation $\boldsymbol{x}_0$. We chose $m = 45$ as the maximal number of nearest neighbors, which means 23 possible sums of nearest neighbors.

For the specification of the link between the predictors and the class we chose the logit model as did Holmes and Adams (2002). Of course one could use alternative link functions, for example the complementary log-log link. One advantage of the logit model is that theory and programs for penalized estimates as lasso are available for the logit model only. Also alternative penalization approaches have been proposed as the elastic net (Zou and Hastie 2005), SCAD (Fan and Li 2001) or the Dantzig selector (Candes and Tao 2007), which could be used to select predictors.

As already mentioned in the introduction similar concepts have been considered by Holmes and Adams (2002, 2003). While Holmes and Adams (2002) focussed on the choice of the size of the neighborhood and used Bayesian concepts, the model considered by Holmes and Adams (2003) is an equivalent representation of the model used here when using sums in the predictor. The main difference is that in higher dimensional settings Holmes and Adams (2003) rely on stepwise variable selection whereas we prefer selection by penalty terms or boosting methods, which have the advantage of continuously reducing the impact of predictors in contrast to a stepwise procedure, which is a discrete process and therefore less stable, see also Breiman (1996b). Moreover, we also include nearest neighbors defined on single dimensions and, in addition, original variables (Sect. 3).

The concept of Ghosh and Godtliebsen (2012) is quite different. Their method does not combine nearest neighbors and original variables on the variable level but on the model level. They combine different classifiers in an ensemble. The method of Gertheiss and Tutz (2009) uses a linear combination of posterior estimates obtained from nearest neighbors computed for single dimensions of the predictor space. Thus it forms an ensemble of nearest neighborhood methods but does not include the classical nearest neighbors that are computed for a distance measure on the whole predictor space. In the present approach these are included, but also nearest neighbors defined on single dimensions and, in addition, original variables (see Sect. 3).

2.3 Comparison of nearest neighbor methods

In the following we briefly compare the performance of the alternative forms of weighted nearest neighbors methods by use of several real data sets. The glass data, ionosphere data and the Australian credit data are available from the UCI machine learning repository (Bache and Lichman 2013). The glaucoma data are from the R package TH.data (Hothorn 2014). In the following the data sets are described.

Glass data Originally the glass data set was collected for crime detection. It contains 6 different types of glass which can be classified by 9 metric predictors—the refractive index (RI) and the proportion of Sodium (NA), Magnesium (Mg), Aluminium (Al), Silicon (Si), Potassium (K), Calcium (Ca), Barium (Ba) and Iron (Fe). Since we are considering classification for two classes we only distinguish between float processed building windows (70 observations) and non-float processed building windows (76 observations).

Ionosphere data The data set consists of 351 radar returns from the ionosphere. There are 34 normalized metric covariates which can be used to identify "good" and "bad" radar returns. The challenge is to take the very uneven distribution of both classes (bad: 126 / good 225) into account.

Australian credit data The aim is to devise a rule for assessing applications for credit cards. The data set has 14 covariates and 690 observations. Due to confidentiality neither the meaning of the covariates nor the exact meaning of the two classes is known. Some categorical variables consist of up to 14 categories.

Glaucoma data In medical studies one is often confronted with many predictors and a very limited number of samples. To test the new method under these circumstances the glaucoma data set is used. It contains information about 98 patients and 98 controls. Due to the fact that age is known to be a major risk factor for glaucoma, both populations are matched by age and by gender to prevent bias. Altogether 62 covariates describing the eye morphology are available for classification.

Figure 1 shows the misclassification rates for the four data sets. The data set were split 30 times into a learning (70 % of the data) and a test data set (30 % of the data) and misclassification was evaluated in the test data, which were not used to find the classifier. When comparing NN methods in Fig. 1 we use the following abbreviations

knn:	k-th nearest neighbor method, k chosen by cross-validation based on predictive deviance, programm package class (Venables and Ripley 2002)
wknn:	weighted k-th nearest neighbor method with the weights determined by a Gaussian kernel, k was chosen by cross-validation based on the predictive deviance (smoothing parameter is automatically adapted), programm package kknn (Schliep and Hechenbichler 2013)
logit.nnk:	Fitting of a logit model with k nearest neighbor dummies, k chosen by cross-validation based on predictive deviance, programm package stats (R Core Team 2013)
lasso.nnk:	Fitting of a logit model with k nearest neighbor dummies by lasso, cross-validated smoothing parameter, programm package penalized (Goeman 2012)
lasso.sg:	Fitting of a logit with the global sums by lasso, programm package penalized (Goeman 2012)

In Fig. 1 we included only two methods with fixed number of nearest neigbbors (*logit.nn3* and *logit.nn10*) as representatives of this approach. The methods *lasso.nn10* and *lasso.sg* include selection of nearest neighbors; the latter method selects from all available 23 nearest neighbors. It is seen that the weighted nearest neighbor does not always outperform the simple k-NN method. In two of the data sets it showed worse performance, for the ionosphere data it was definitely poor. However, if weights are determined by estimation of coefficients instead of simple weights derived from distances the performance improves in three of the four data sets. Only in the last data set (glaucoma) the performance is slightly worse. It is noteworthy that the performance of all the methods with estimated coefficients is rather similar. In extensions to be considered later we will always use the method with global sums.

As noted in the description of the methods we use available R packages. However, first some data preparation is necessary. In particular we generate a data set that includes the nearest neighbors and their sums and standardize variables to have variance one since lasso is scale sensitive. A function is provided that calculates the nearest neighbors based on the Euclidean distance. When training the models some hyperparameter tuning for the lasso penalization parameter is needed which is done by 10 fold cross-validation. The training of the models is done by adding the sums as additional parameters to the original lasso function. No further modification of the underlying model is necessary.

3 Combining nearest neighbors with alternative distances and predictors

The concept of estimating NN-rules by selection of relevant nearest neighbors or sums of them has even more potential to improve classifiers. The first is that one can include nearest neighbors computed in sub dimensions, the second is that one

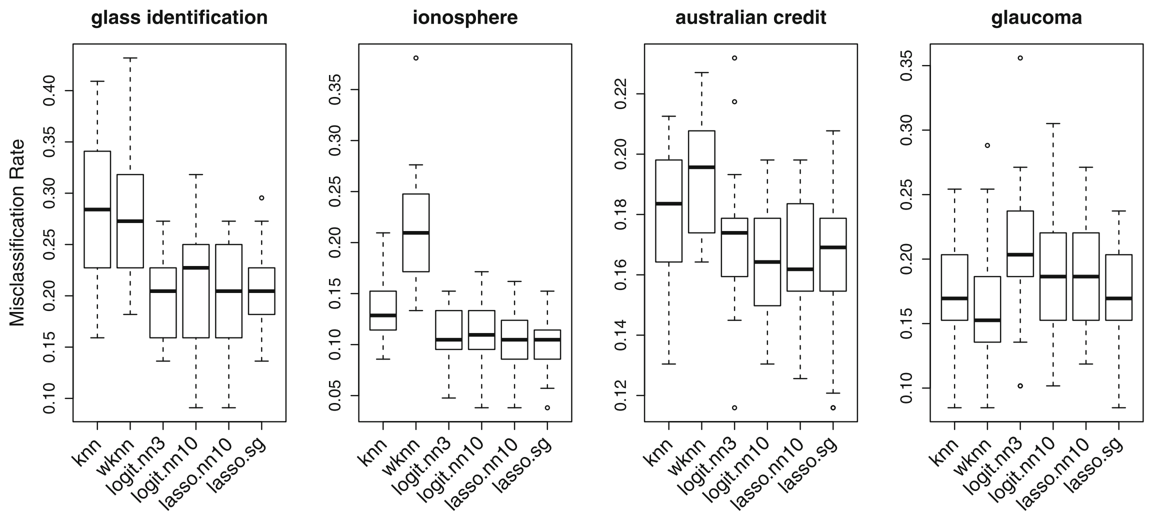


Fig. 1 Classifiers for glass data, ionosphere data, Australian credit data, glaucoma data

can combine nearest neighbors and predictors themselves. By using a selection method as the lasso, the relevant features are automatically selected.

3.1 Selection of directions and use of covariates

When the set of predictors is large and contains predictors that do not contribute to classification, simple classifiers without selection of predictors fit noise and performance suffers. This also holds for the nearest neighborhood method that uses a simple distance like the Euclidean distance based on all dimensions of the predictor space. With selection in mind one can include various distance measures in the linear predictor of the logit model. A simple choice is to include all one-dimensional distances.

The procedure is obtained by using the augmented data

$$y_i, y_{(1)}^{(1)}(\boldsymbol{x}_i), \ldots, y_{(k)}^{(1)}(\boldsymbol{x}_i), \ldots, y_{(1)}^{(p)}(\boldsymbol{x}_i), \ldots, y_{(k)}^{(p)}(\boldsymbol{x}_i), \boldsymbol{x}_i,$$

where $y_{(l)}^{(j)}(\boldsymbol{x}_i)$ is the class of the l-th nearest neighbor to $\boldsymbol{x}_i$ based on the distance $d(x_{ij}, x_{lj})$, that is, the distance of the j-th component. Then one uses the predictor

$$\eta_i = \gamma_0 + \sum_{j=1}^{p} (\boldsymbol{y}_{i,\text{NN}}^{(j)})^T \boldsymbol{\gamma}_j,$$

where $\boldsymbol{y}_{i,NN}^{(j)} = (y_{i(1)}^{(j)}, \ldots, y_{i(k)}^{(j)})^T$ with $y_{i(k)}^{(j)} = y_{(k)}^{(j)}(\boldsymbol{x}_i)$. Now the predictor includes all nearest neighbors built for single dimensions of the predictor space. If it is high-dimensional the number of predictors strongly increases and the choice of the direction is even more important. Therefore, we again replace the whole predictor $\boldsymbol{y}_{i,\text{NN}}^{(j)}$ by the corresponding sums of the j-th dimension $s_{i(k)}^{(j)} = \sum_{l=1}^{k} y_{i(l)}^{(j)}$ for all odd values k that are smaller than 46. With lasso-type methods the directions most useful for classification should be selected.

Based on the augmented data one can also use the predictor

$$\eta_i = \gamma_0 + \sum_{j=1}^{p} (\boldsymbol{y}_{i,\text{NN}}^{(j)})^T \boldsymbol{\gamma}_j + \boldsymbol{x}_i^T \boldsymbol{\beta},$$

which, in addition to the neighbors, includes a linear predictor $\boldsymbol{x}_i^T \boldsymbol{\beta}$ that contains the original variables. Selection procedures like the lasso or boosting methods are used to determine which of the predictors have discriminatory power.

The augmented data given previously also contain the original predictors themselves. The linear predictor is built by sums of nearest neighbors in various directions and the original predictors. Then, in the fitting procedure it is decided how much weight is given to the next neighbors and how much to the original predictors.

3.2 Illustration of the working of the classifier

For simplicity we consider a benchmark scenario that has only two dimensions. It is used to demonstrate how selection of relevant features works. The predictors were generated by use of mlbench.threenorm from the R package mlbench of Leisch and Dimitriadou (2010). The distributions of the predictors are visualized in Fig. 2, which shows one data set.

We use the following abbreviations:

lasso.cov:	Fitting of a logit model by lasso with the available original covariates,
lasso.cov.sg:	Fitting of a logit model by lasso with available covariates and global sums,
lasso.cov.sd:	Fitting of a logit model by lasso with available covariates and all one-dimensional (directional) sums,
lasso.cov.sg.sd:	Fitting of a logit model by lasso with available covariates, global sums, and all one-dimensional (directional) sums.

Figure 3 shows the variable importance measured as the proportion of the absolute value of the parameter under consideration and the absolute value of the largest parameter for 30 repetitions. Thus if the proportion is 1 the variable had the largest parameter value, if the proportion is 0 it was deleted. For ease of presentation here we use only the sums over 5, 10 and 25 variables, in all the other investigations we use all sums over odd numbers of nearest neighbors neighbors up to 45 as specified before. It is seen from Fig. 3 that if one uses the original variables only (*lasso.cov*) both predictors were selected in most of the simulated data sets and variable x_2 showed slightly stronger impact. Selection of both variables

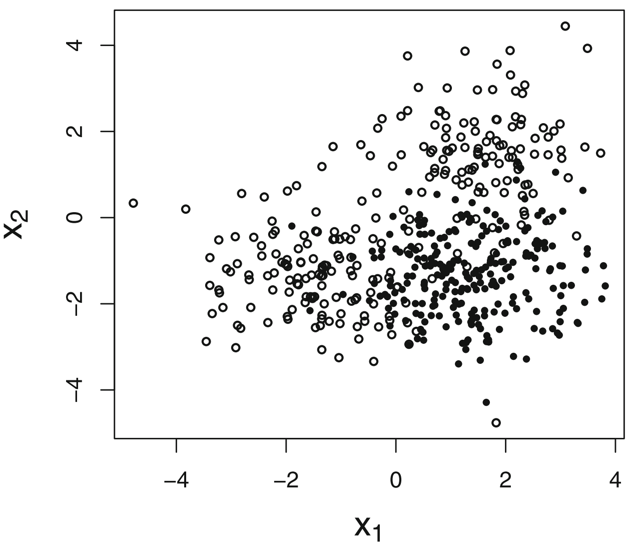


Fig. 2 Distribution of predictors in benchmark data set, points are observations from class one, *circles* are observations from class two

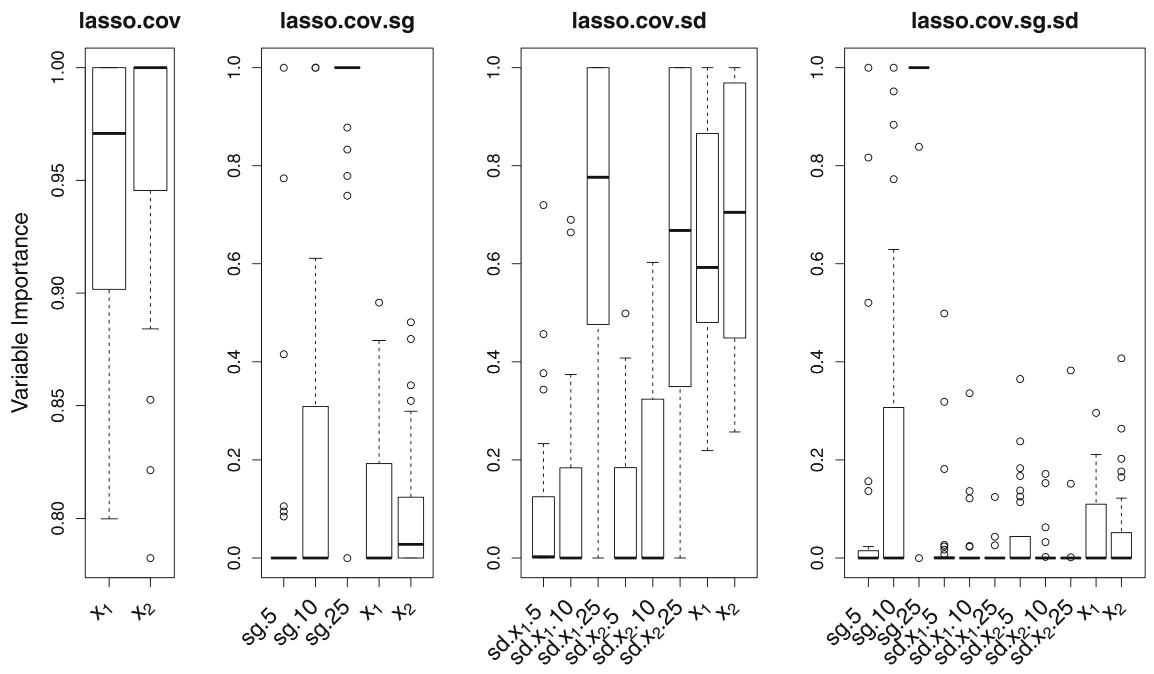


Fig. 3 Variable importance for selection by lasso (benchmark data). Variable importance is measured as the proportion of the absolute value of the parameter under consideration and the absolute value of the largest parameter. x_1, x_2 refer to to the parameter weights on the original variables, $sg.k$ and $sd.k$ refer to the parameters linked to global and directional sums for k nearest neighbors

was to be expected because both covariates contribute to the separation of classes. If in addition the sums of neighborhoods are included (*lasso.cov.sg*), the sum over the 25 nearest neighbors carries the most information followed by the sum over the 10 nearest neighbors. If one used only one dimensional distances and the original variables (*lasso.cov.sd*) the sums over the 25 nearest neighbors are the most important variables but also the original variables and some nearest neighbor distances for variable x_2 show strong values. If one adds the global sums the dominating feature is again the sum over the 25 nearest neighbors since single directions are less important as is seen from Fig. 2. The mean misclassification errors were 0.163 (*lasso.cov*), 0.106 (*lasso.cov.sg*), 0.122 (*lasso.cov.sd*) and 0.106 (*lasso.cov.sg.sd*), which supports that for these data the nearest neighbors built for the whole feature space yield the strongest tool to separate classes. Consequently, they had strong impact if they were available as predictors as in the methods *lasso.cov.sg* and *lasso.cov.sg.sd*.

3.3 Comparison of methods: simulations

In the following the performance of the methods is investigated by use of simulation scenarios that have been used before in classification problems. We use 200 observations in the training sample and 500 observations in the test sample and consider the following settings.

Easy The example is taken from Hastie et al. (2009) and describes another classification problem with difficulties arising from noise variables. Altogether there are 10 independent predictors X_j ($j = 1, \dots, 10$), each uniformly distributed on $[0, 1]$. In this setting the binary response y only depends on the first predictor X_1 and is defined by $y = I(x_1 > 0.5)$.

Difficult As in the easy classification problem each predictor is uniformly distributed between 0 and 1. However, in this setting the response y depends on the combination of the predictors X_1, X_2 and X_3 and is specified by

$$y = I\left(\text{sign}\left\{\left(x_1 - \frac{1}{2}\right)\cdot\left(x_2 - \frac{1}{2}\right)\cdot\left(x_3 - \frac{1}{2}\right)\right\} > 0\right)$$

Unstructured This example refers to the unstructured problem of Hastie and Tibshirani (1996) and possesses an extremely disconnected class structure. We use a slightly modified version. Instead of 4 classes the problem is reduced to 2 classes. These are defined as a mixture of 5 spherical bivariate normal subclasses with a standard deviation of 0.25. The means of the 10 subclasses are chosen at random (without replacement) from the 25 possible combinations between the integers $[1, 2, \dots, 5] \times [1, 2, \dots, 5]$. Therefore each subclass has its own distribution, which is used to draw 20 observations per subclass.

Unstructured noise The data generating mechanism is the same as in the previous example, but augmented by 8 noise variables taken from a standard normal distribution.

In addition to the extended weighted neighborhood methods we include the following procedures:

lda:	Linear discriminant analysis, package MASS (Venables and Ripley 2002)
qda:	Quadratic discriminant analysis, package MASS (Venables and Ripley 2002)
logit:	Logistic discrimination glm(.) from package stats (R Core Team 2013)
mboost:	Gradient boosting, package stats (Hothorn et al. 2013)
svm:	Support vector machine, package e1071 (Meyer et al. 2014)
rf:	Random forest, package randomForest (Liaw and Wiener 2002)

Gradient boosting is an alternative to lasso. Boosting methods as proposed by Friedman et al. (2000), Bühlmann and Yu (2003), Tutz and Binder (2006) and Bühlmann and Hothorn (2007) also select predictors in an efficient way. While boosting is often used as a general method to combine classifiers the so-called componentwise boosting focusses on variable selection. If one uses as base learner just one linear component of a predictor and selects the best one, only one variable is updated within one iteration. When the iteration is stopped typically not all variables have been selected yielding a predictor that contains a reduced number of variables. Support vector machines are described in Cortes and Vapnik (1995), random forests in Breiman (2001). For the support vector machine (package e1071) we used radial basis functions, type = "C-Classification" and the hyperparameters were chosen by grid.search (tune.svm). For RF we used mtry = sqrt (number of predictors) and ntree = 1000. We use throughout the same notation as for lasso, the ending *cov* means that covariates are used, *sg* means global sums over nearest neighbors and *sd* directional sums over nearest neighbors. For lasso we used the package penalized and for boosting the package mboost.

Figures 4, 5, 6 and 7 show the boxplots of the resulting misclassification rates. For the setting *easy*, where only one variable is relevant and nine variables have no discriminatory power it is seen that simple nearest neighbors, represented by *lasso.sg*, perform poorly. If variables are included and predictors are selected, the nearest neighbor methods (mboost and lasso, right part of the panel) improve but performance is not yet satisfying. If one allows for nearest neighbors built by distances for single variables (directional nearest neigh-

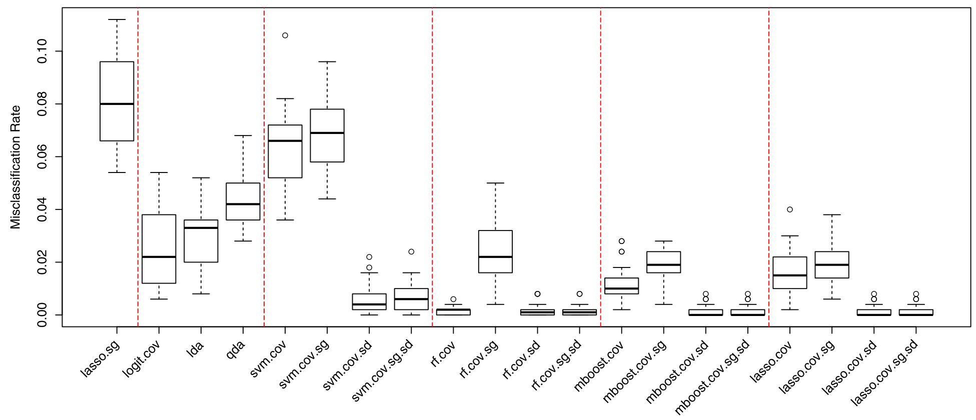


Fig. 4 Misclassification rates for easy data

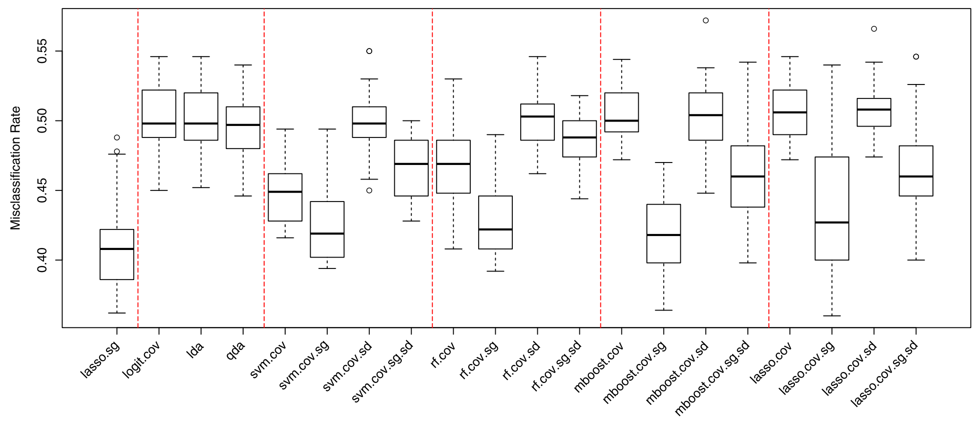


Fig. 5 Misclassification rates for difficult data

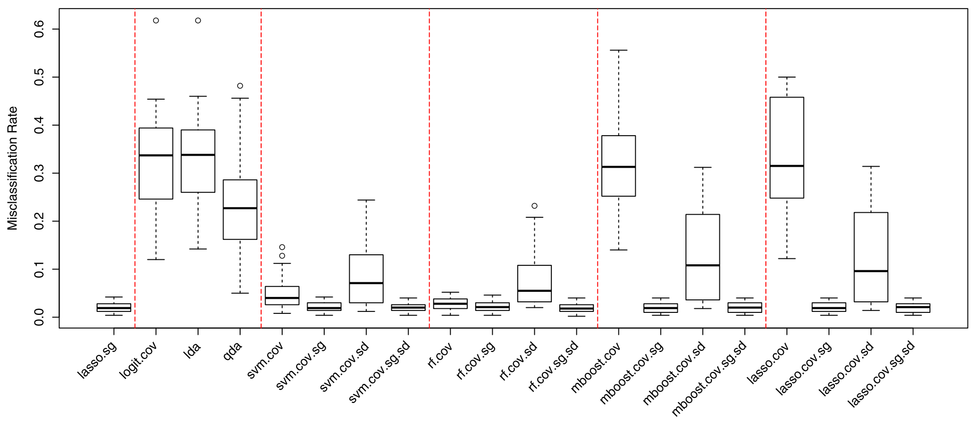


Fig. 6 Misclassification rates for unstructured data

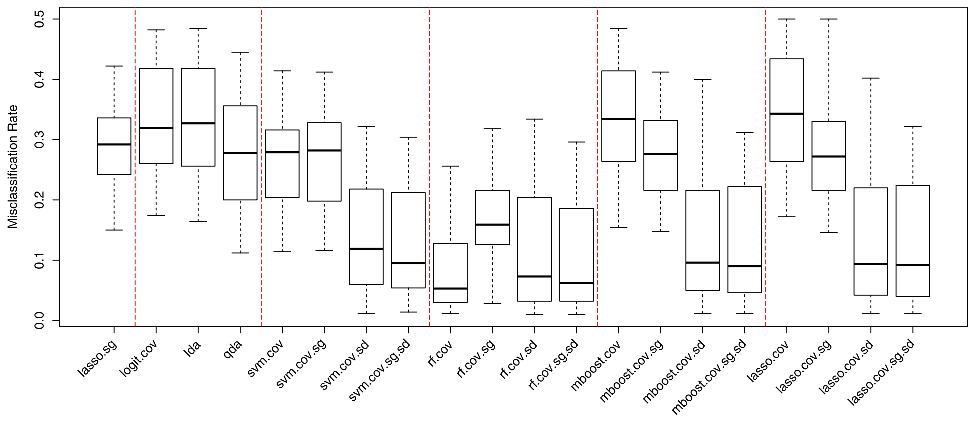


Fig. 7 Misclassification rates for unstructured noise data

bors represented by the form .sd) and the procedure can also select among the predictors the discrimination is much better. The example shows that the inclusion of directional nearest neighbors can select relevant variables yielding much better performance than simple NN methods. The method distinctly outperforms classical linear and quadratic discrimination. It is also seen that the strong competitors support vector machine (SVM) and random forests (RF) also profit from the inclusion of directional nearest neighbors. In particular for the SVM the performance is much better if these sums are included. Random forest as one of the strongest tools in classification shows good performance also without these sums.

The second setting, *difficult*, is characterized by an interaction of three variables. Consequently, classical linear and quadratic discrimination yield poor results while the methods that use nearest neighbors work well. The best classifier in this case was the simple 1-NN method (misclassification error 0.391) closely followed by *lasso.sg* (misclassification error 0.408). All the methods that include global sums show good performance. Also SVM and RF, which do not perform well with covariates only, profit from the inclusion of nearest neighbors. Due to the interaction effect methods that include only directional sums show poor performance.

Also the third setting, *unstructured*, with the extremely disconnected class structure favors the nearest neighbor approach. The qualitative results are very similar to the *difficult* setting with a strong performance of methods that include nearest neighbors in the predictor.

The fourth setting, *unstructured noise* is the same as *unstructured*, but now with additional noise variables. It is seen that the methods with selection of neighbors in different directions outperform all the other methods with the exception of random forests, which have a built-in variable selection procedure. All other methods strongly profit from the implicit selection of variables by using directional sums. It makes the methods much stronger than methods that build nearest neighbors over the whole feature space.

Table 1 shows the resulting misclassification rates including the 1-NN, k-NN and weighted nearest neighbor method. The two best methods are given in boldface and standard errors are given in brackets.

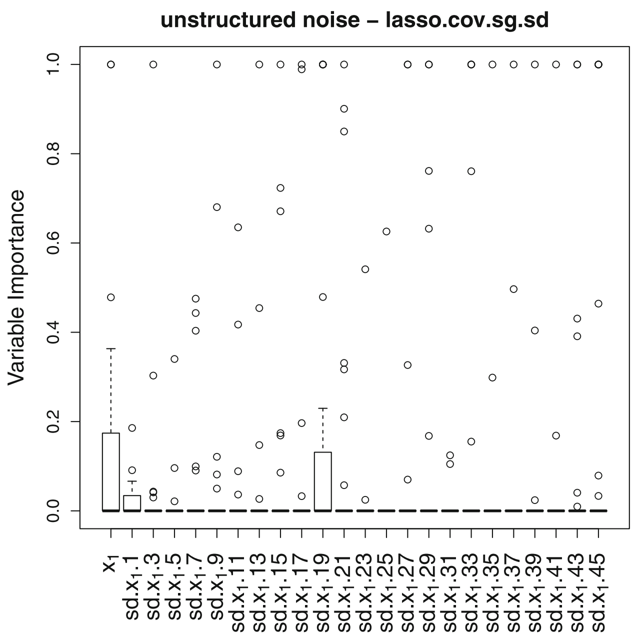
From Table 1 and Figs. 4, 5, 6 and 7 it is not seen whether the differences are significant. Therefore we include test results. Table 2 shows the p values for McNemar tests under the null hypothesis that the error rates of two approaches are equal. Since comparing all of the differences yields too many values we choose *lasso.cov* as the reference. Thus the p value shows if the method differs significantly from the performance of *lasso.cov*. It is seen, for example, that for the unstructured noise data (Fig. 7) all the methods differ significantly from the reference method.

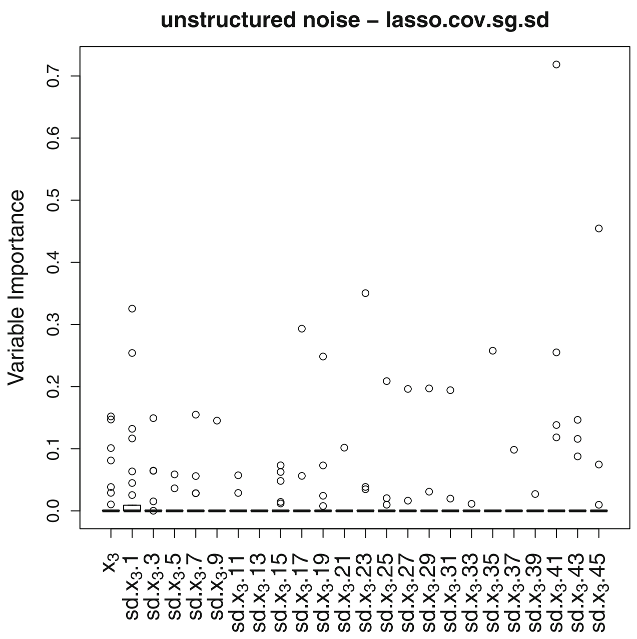
Table 1 Misclassification rates of classifiers for simulated data sets (computed on test data); standard errors are given in brackets

	Easy	Difficult	Unstructured	Unstructured noise
nn1	0.18 (0.018)	**0.391** (0.024)	0.027 (0.013)	0.346 (0.059)
knn	0.124 (0.022)	0.439 (0.041)	**0.019** (0.011)	0.308 (0.068)
wknn	0.094 (0.021)	0.433 (0.026)	**0.019** (0.011)	0.298 (0.072)
lasso.sg	0.08 (0.017)	**0.408** (0.030)	**0.019** (0.011)	0.292 (0.073)
logit.cov	0.022 (0.014)	0.498 (0.025)	0.337 (0.104)	0.319 (0.092)
lda	0.033 (0.011)	0.498 (0.025)	0.338 (0.104)	0.327 (0.095)
qda	0.042 (0.01)	0.497 (0.025)	0.227 (0.104)	0.278 (0.093)
svm.cov	0.066 (0.014)	0.449 (0.022)	0.04 (0.035)	0.279 (0.082)
svm.cov.sg	0.069 (0.014)	0.419 (0.029)	**0.019** (0.011)	0.282 (0.081)
svm.cov.sd	0.004 (0.006)	0.498 (0.024)	0.071 (0.074)	0.119 (0.094)
svm.cov.sg.sd	0.006 (0.006)	0.469 (0.023)	0.020 (0.010)	0.095 (0.087)
rf.cov	0.002 (0.002)	0.469 (0.028)	0.028 (0.013)	**0.053** (0.067)
rf.cov.sg	0.022 (0.011)	0.422 (0.03)	0.021 (0.012)	0.159 (0.07)
rf.cov.sd	0.001 (0.002)	0.503 (0.021)	0.055 (0.062)	0.073 (0.088)
rf.cov.sg.sd	0.001 (0.002)	0.488 (0.02)	**0.018** (0.011)	**0.062** (0.084)
mboost.cov	0.01 (0.007)	0.500 (0.021)	0.313 (0.095)	0.334 (0.09)
mboost.cov.sg	0.019 (0.006)	0.418 (0.029)	**0.019** (0.01)	0.276 (0.076)
mboost.cov.sd	**0.000** (0.002)	0.504 (0.025)	0.108 (0.092)	0.096 (0.107)
mboost.cov.sg.sd	**0.000** (0.002)	0.460 (0.031)	0.020 (0.010)	0.090 (0.099)
lasso.cov	0.015 (0.009)	0.506 (0.023)	0.315 (0.112)	0.343 (0.106)
lasso.cov.sg	0.019 (0.007)	0.427 (0.046)	**0.019** (0.01)	0.272 (0.083)
lasso.cov.sd	**0.000** (0.002)	0.508 (0.020)	0.096 (0.095)	0.094 (0.109)
lasso.cov.sg.sd	**0.000** (0.002)	0.460 (0.036)	0.021 (0.01)	0.092 (0.103)

Table 2 *P* values for McNemar tests that compare the classification outcomes in the test sets with reference to *lasso.cov*

	Easy	Difficult	Unstructured	Unstructured noise
nn1	0.000	0.000	0.000	0.004
knn	0.000	0.000	0.000	0.000
wknn	0.000	0.000	0.000	0.000
lasso.sg	0.000	0.000	0.000	0.000
logit.cov	0.000	0.231	0.000	0.010
lda	0.000	0.230	0.001	0.001
qda	0.000	0.007	0.000	0.000
svm.cov	0.000	0.000	0.000	0.000
svm.cov.sg	0.000	0.000	0.000	0.000
svm.cov.sd	0.000	0.112	0.000	0.000
svm.cov.sg.sd	0.000	0.000	0.000	0.000
rf.cov	0.000	0.000	0.000	0.000
rf.cov.sg	0.000	0.000	0.000	0.000
rf.cov.sd	0.000	0.194	0.000	0.000
rf.cov.sg.sd	0.000	0.000	0.000	0.000
mboost.cov	0.000	0.557	0.000	0.000
mboost.cov.sg	0.034	0.000	0.000	0.000
mboost.cov.sd	0.000	0.609	0.000	0.000
mboost.cov.sg.sd	0.000	0.000	0.000	0.000
lasso.cov.sg	0.000	0.000	0.000	0.000
lasso.cov.sd	0.000	0.923	0.000	0.000
lasso.cov.sg.sd	0.000	0.000	0.000	0.000





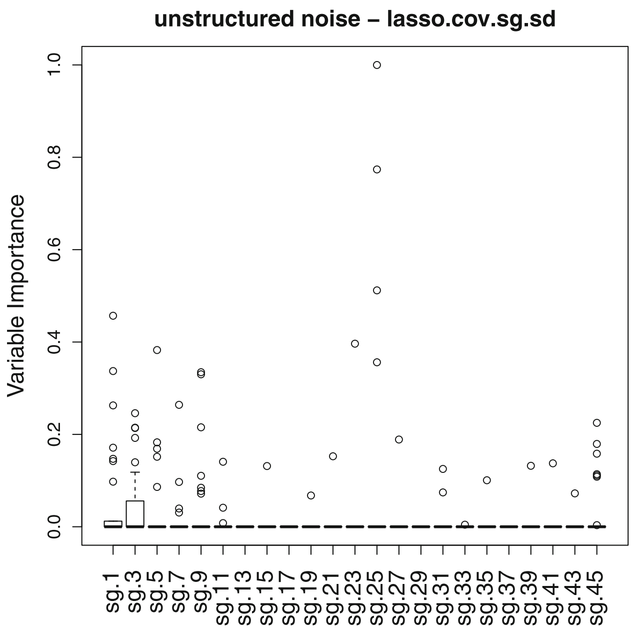


Fig. 8 Variable importance for variables x_1 (carries information) and x_3 (irrelevant). Variable importance is measured as the proportion of the absolute value of the parameter under consideration and the absolute value of the largest parameter. x_1, x_3 refer to to the parameter weights on the original variables, $sg.k$ and $sd.k$ refer to the parameters linked to global and directional sums for k nearest neighbors

In simulations one can investigate if the right terms are selected. We illustrate the selection for the unstructured noise data, where only the first two variables carry information and the others are irrelevant. Although all variables were included in the fitting process in Fig. 8 the selection of predictors is visualized for only two variable namely x_1, which carries information and the corresponding directional sums (first part of figure) and x_3, which is one of the irrelevant variables and corresponding directional sums (second part of the figure). In the third part of the figure selection of the global sums is shown. It is seen that most of the irrelevant terms are set to zero. The variable x_1 and in particular the sum over 19 neighbors is selected from the x_1 based distances and for the global sums the three nearest neighbors are selected. Thus in the selection procedure the irrelevant terms connected to x_3 are rarely selected and one obtains a sparse representation that contains the relevant terms. The pictures for the other irrelevant terms were very similar to the picture for the variable x_3.

3.4 Comparison of methods: real data sets

For the comparison of methods we use one more data set, which represents the “small n large p” case.

DLBCL data The gene expression data on lymphomas was collected from 77 patients on 6817 genes. The classes are determined by *diffuse large B-cell lymphomas* (DLBCL) with n=58 patients and *follicular lymphomas* (FL) with n=19 patients. The analysis was carried out using the data for all 77 patients. We used the 100 most significant genes that were found by using the method to identify differentially expressed genes provided by the `samr` package in `R` (Tibshirani et al. 2011).

Figures 9, 10, 11, 12, 13 and Table 3 show the results for the real data sets. They show the misclassification in the test samples over 30 splits of the data sets. As in Sect. 2.3 the data sets were split into learning (70 % of the data) and test data sets (30 % of the data) and misclassification was evaluated in the test data, which were not used to find the classifier. For the *glass data* the random forest and the SVM perform best. While the RF does not profit from the inclusion of directional distances the SVM obtains the good performance only

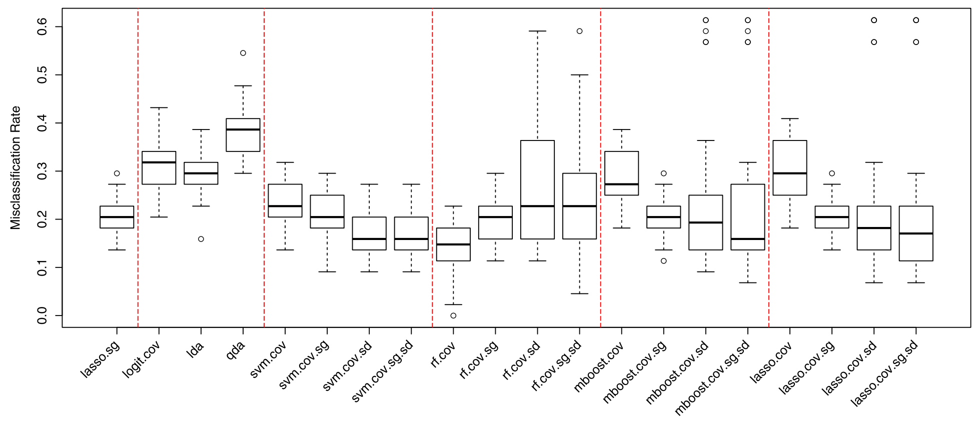


Fig. 9 Misclassification rates for glass data

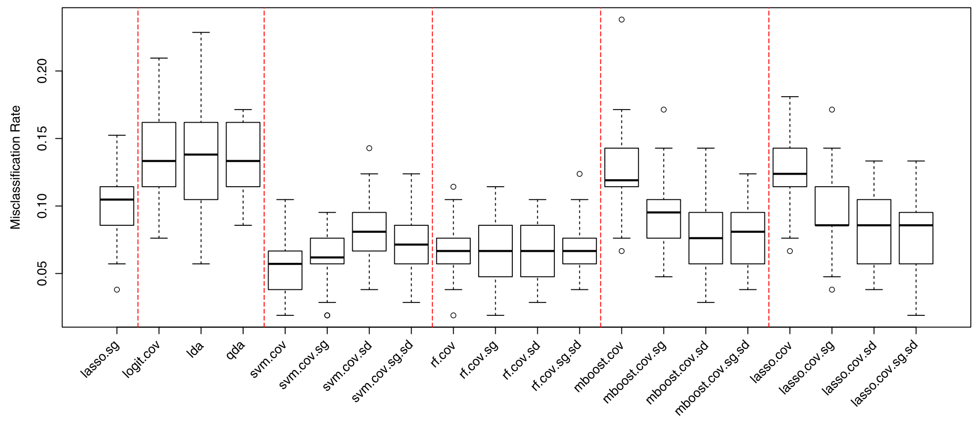


Fig. 10 Misclassification rates for ionosphere data

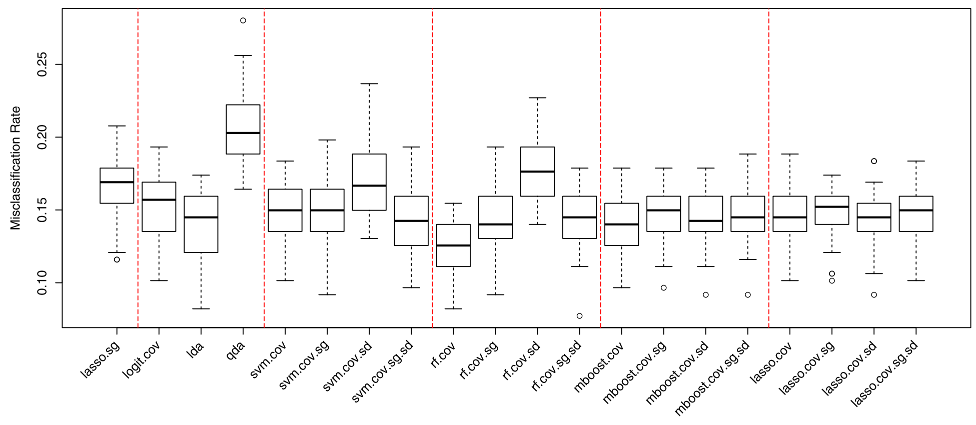


Fig. 11 Misclassification rates for Australian credit data

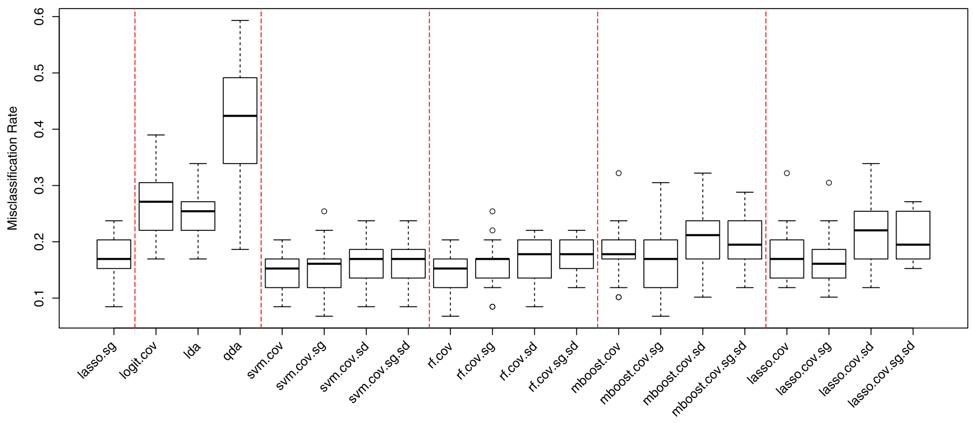


Fig. 12 Misclassification rates for glaucoma data

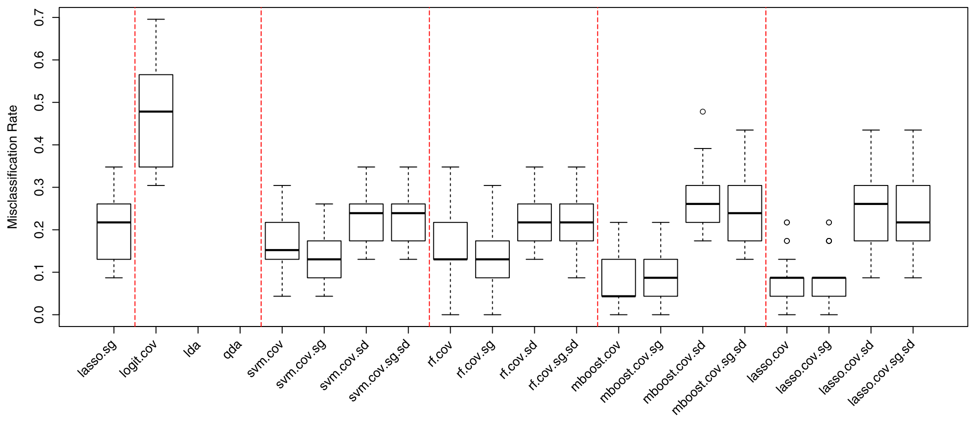


Fig. 13 Misclassification rates for DLBCL data

Table 3 Misclassification rates of classifiers for real data sets (computed on test data); standard errors are given in brackets

	Glass	Ionosphere	Credit	Glaucoma	DLBCL
nn1	0.205 (0.053)	0.129 (0.031)	0.222 (0.019)	0.203 (0.049)	0.217 (0.075)
knn	0.284 (0.061)	0.129 (0.031)	0.184 (0.02)	0.169 (0.04)	0.174 (0.06)
wknn	0.273 (0.062)	0.210 (0.052)	0.196 (0.018)	**0.153** (0.045)	0.217 (0.071)
lasso.sg	0.205 (0.039)	0.105 (0.027)	0.169 (0.024)	0.169 (0.041)	0.217 (0.075)
logit.cov	0.318 (0.053)	0.133 (0.032)	0.157 (0.023)	0.271 (0.056)	0.478 (0.11)
lda	0.295 (0.048)	0.138 (0.038)	0.145 (0.025)	0.254 (0.041)	–
qda	0.386 (0.057)	0.133 (0.029)	0.203 (0.029)	0.424 (0.102)	–
svm.cov	0.227 (0.047)	**0.057** (0.02)	0.15 (0.022)	**0.153** (0.036)	0.152 (0.074)
svm.cov.sg	0.205 (0.049)	**0.062** (0.019)	0.150 (0.024)	0.161 (0.042)	0.130 (0.058)
svm.cov.sd	**0.159** (0.049)	0.081 (0.026)	0.167 (0.026)	0.169 (0.032)	0.239 (0.067)
svm.cov.sg.sd	**0.159** (0.051)	0.071 (0.023)	0.143 (0.022)	0.169 (0.033)	0.239 (0.067)
rf.cov	**0.148** (0.057)	0.067 (0.02)	**0.126** (0.018)	**0.153** (0.037)	0.130 (0.074)
rf.cov.sg	0.205 (0.051)	0.067 (0.024)	**0.140** (0.023)	0.169 (0.038)	0.130 (0.067)
rf.cov.sd	0.227 (0.136)	0.067 (0.022)	0.176 (0.023)	0.178 (0.034)	0.217 (0.067)
rf.cov.sg.sd	0.227 (0.126)	0.067 (0.021)	0.145 (0.022)	0.178 (0.033)	0.217 (0.073)
mboost.cov	0.273 (0.058)	0.119 (0.032)	**0.140** (0.021)	0.178 (0.044)	**0.043** (0.069)
mboost.cov.sg	0.205 (0.038)	0.095 (0.026)	0.150 (0.019)	0.169 (0.054)	**0.087** (0.07)
mboost.cov.sd	0.193 (0.169)	0.076 (0.027)	0.143 (0.02)	0.212 (0.049)	0.261 (0.071)
mboost.cov.sg.sd	**0.159** (0.171)	0.081 (0.024)	0.145 (0.021)	0.195 (0.044)	0.239 (0.072)
lasso.cov	0.295 (0.068)	0.124 (0.026)	0.145 (0.021)	0.169 (0.047)	**0.087** (0.056)
lasso.cov.sg	0.205 (0.037)	0.086 (0.028)	0.152 (0.019)	0.161 (0.045)	**0.087** (0.061)
lasso.cov.sd	0.182 (0.17)	0.086 (0.027)	0.145 (0.021)	0.220 (0.049)	0.261 (0.075)
lasso.cov.sg.sd	0.170 (0.177)	0.086 (0.028)	0.150 (0.021)	0.195 (0.042)	0.217 (0.085)

if they are included. The extended nearest neighbor methods with selection come close in terms of misclassification rate, in particular if directional distances are included. Parametric approaches perform poorly. For the *ionosphere data* the performance of the NN methods and the parametric approaches is very similar to the performance for the glass data. However, the SVM now dominates the RF and is overall the best classifier. The extended NN methods do not attain the same level of performance but strongly improve if directional distances are included. For the *credit data* also the linear approaches work well. The NN approaches are slightly better than the SVM. SVM again profits from directional distances while RF does not, but performs best. For the *glaucoma data* NN methods perform distinctly better than linear and quadratic discrimination methods. The weighted NN method is, together with RF, the best classifier. Nevertheless, one of the NN methods with selection is second best. For the very high dimensional data set *DLBCL data* boosting and lasso with global sums and covariates perform best and even outperform RF and the SVM, although the latter tend to profit from the inclusion of global sums. However, all methods suffer when directional sums are included. It seems that in the the “small n large p” case the inclusion of directional sums, which in our case enlarges the sum of available predictors by 2300, selection of predictors does not work well. This is also supported by simulations of the “small n large p” case, which are not presented. Nevertheless, NN methods with covariates and global sums perform well. The parametric methods lda and qda are not available, logit.cov was computed by adding a small ridge penalty. Table 4 shows the corresponding p values for McNemar tests that compare the classification outcomes in the test sets with reference to *lasso.cov*.

In summary, considering simulations and the analysis of real data, it is seen that the combination of covariates and global sums combined with selection typically yields improved NN classifiers. The performance is much better than for classical NN methods in almost all of the considered simulation scenarios and data sets. Exceptions are the difficult scenario and the glaucoma data set, for which classical NN methods perform very well, but also for these data sets the lasso NN with covariates and global sums performs not much worse. The inclusion of directional sums can strongly or moderately improve the performance (easy, unstructured noise, glass, ionosphere) but in some cases can also be counterproductive (difficult, glaucoma, DLBCL). In particular, if the number of predictors is very large when compared to the number of observations (glaucoma, DLBCL), one should be cautious with the inclusion of directional sums. Support vector machines and random forests are always strong competitors in classification problems and show good performance, in particular in the real data sets. NN methods with selection showed better performance only for the DLBCL

Table 4 *P* values for McNemar tests that compare the classification outcomes in the test sets with reference to *lasso.cov*

	Glass	Ionosphere	Australian credit	Glaucoma	DLBCL
nn1	0.000	0.290	0.000	0.011	0.000
knn	0.067	0.290	0.000	0.637	0.000
wknn	0.080	0.000	0.000	0.012	0.000
lasso.sg	0.000	0.000	0.000	0.548	0.000
logit.cov	0.841	0.202	0.001	0.000	0.000
lda	0.649	0.042	0.116	0.000	–
qda	0.000	0.461	0.000	0.000	–
svm.cov	0.000	0.000	0.377	0.016	0.000
svm.cov.sg	0.000	0.000	0.315	0.045	0.000
svm.cov.sd	0.000	0.000	0.000	0.451	0.000
svm.cov.sg.sd	0.000	0.000	0.482	0.320	0.000
rf.cov	0.000	0.000	0.000	0.013	0.000
rf.cov.sg	0.000	0.000	0.964	0.142	0.001
rf.cov.sd	0.025	0.000	0.000	0.712	0.000
rf.cov.sg.sd	0.000	0.000	0.594	0.709	0.000
mboost.cov	0.090	0.841	0.101	0.374	1.000
mboost.cov.sg	0.000	0.000	0.556	0.108	0.345
mboost.cov.sd	0.000	0.000	0.720	0.002	0.000
mboost.cov.sg.sd	0.000	0.000	0.910	0.008	0.000
lasso.cov.sg	0.000	0.000	0.329	0.210	0.850
lasso.cov.sd	0.000	0.000	0.809	0.000	0.000
lasso.cov.sg.sd	0.000	0.000	0.623	0.001	0.000

data set. In the simulation studies the results were less in favor of support vector machines and random forests with covariates only. They dominate the NN methods with selection only for the unstructured noise scenario. Moreover, also support vector machines and random forests can profit from the inclusion of nearest neighbors. In almost all of the scenarios, with the exception of the ionosphere and the glaucoma data sets, one of the versions of the support vector machine with nearest neighbors shows better performance than the support vector machine with covariates only. The effect is less pronounced for random forests, which can suffer from the inclusion of nearest neighbors despite their in-built selection procedure.

3.5 Comparison of methods: computation time

In applications computation time can be an important aspect. To investigate the computation time we generated several data sets with the function mlbench.threenorm. The number of observations was $n = 500, 1000, 2000, 3000$ and the number of variables $nvar = 2, 5, 10, 15, 20, 25$. We split the data sets into a learning (70 % of the data) and a test data set (30 % of the data) and evaluated the time needed for the computation of the classifier on the learning data and the following classification of the test data set. Computation times of all classifiers for every combination averaged over five repetitions are shown in Tables 5 and 6. As was to be expected computation time increases with sample sizes and number of variables for all of the methods. The times of *nn1* and *knn* are similar to the parametric approaches *lda*, *qda*. Variable selection procedures by lasso or boosting are more time consuming, in particular if all the directional distances are included. However, for directional distances the number of parameters from which one selects is obtained by multiplying the number of variables by 23 for the directional distances plus the 23 global sums over nearest neighbors. Even then computation time for 25 variables is manageable in reasonable time. It should be noted that boosting has an advantage over lasso in terms of computation time if many predictors are included. The nonparametric approaches RF and SVM need much more computation time, in particular the fit of SVM is very time consuming. For example, for $n = 3000$, $p = 25$ (cov.sg.sd) 55.882 seconds are needed to fit the SVM as compared to 22.612 for RF and 1.622 for boosting.

4 Extension to multi-class problems

In multi-class problems the proposed method can be applied if one uses strategies like pairwise comparison of classes or one class against all. However, it is more natural to treat the multi-class case directly by using the multinomial logit model. Let $\boldsymbol{x}_{i(1)}, \ldots, \boldsymbol{x}_{i(n)}$ denote the nearest neighbors of observation $\boldsymbol{x}_i$. With m classes the class corresponding to the kth nearest neighbor is now represented by a vector $\boldsymbol{y}_{i(k)}^T = (y_{i(k)1}, \ldots, y_{i(k)m-1})$, where $y_{i(k)j} = 1$ if the observation is from class j and $y_{i(k)j} = 0$ otherwise. The total vector for all the k nearest neighbors is then given by $\boldsymbol{y}_{i,NN}^T = (\boldsymbol{y}_{i(1)}^T, \ldots, \boldsymbol{y}_{i(k)}^T)$. With $Y \in \{1, \ldots, m\}$ denoting the class the corresponding logit model has the form

$$P(Y = r|\boldsymbol{x}_i) = \frac{\exp(\eta_{ir})}{1 + \sum_{s=1}^{m-1} \exp(\eta_{is})}, \quad r = 1, \ldots, m-1,$$

with the predictors

$$\eta_{ir} = \gamma_0 + \boldsymbol{y}_{i,\mathrm{NN}}^T \boldsymbol{\gamma}_r + \boldsymbol{x}_i^T \boldsymbol{\beta}_r, \quad r = 1, \ldots, m-1. \tag{2}$$

The predictor (2) is given in the general form including nearest neighbors and covariates that may contribute to classification. As in the case of two classes the nearest neighbor indicators $y_{i(k)j}$ can be replaced by the sums

$$s_{i(k)j} = \sum_{s=1}^{k} y_{i(s)j},$$

which represent the number of observations from class j among the k nearest neighbors.

Since the multinomial logistic model contains category-specific parameters with one set of parameters for each category selection of the relevant features is even more important than in the case of two classes. Inclusion of too many and irrelevant predictors typically yields unstable estimates. Lasso-type selection procedures that focus on the selection of variables have been proposed recently by Tutz et al. (2015). Their categorically structured lasso uses the penalty

$$J(\boldsymbol{\beta}) = \sum_{j=1}^{p} ||\boldsymbol{\beta}_{.j}|| = \sum_{j=1}^{p} (\beta_{1j}^2 + \cdots + \beta_{k-1,j}^2)^{1/2}, \tag{3}$$

where $||\boldsymbol{u}|| = ||\boldsymbol{u}||_2 = \sqrt{\boldsymbol{u}^T \boldsymbol{u}}$ denotes the L_2-norm and $\boldsymbol{\beta}_{.j}^T = (\beta_{1j}, \ldots, \beta_{m-1,j})$ collects all the parameters linked to the jth predictor. The advantage of this version of a group lasso penalty is that it selects variables not parameters. Straightforward use of the lasso penalty by using

$$J(\boldsymbol{\beta}) = ||\boldsymbol{\beta}||_1 = \sum_{r=1}^{k-1} ||\boldsymbol{\beta}_r||_1 = \sum_{r=1}^{k-1} \sum_{j=1}^{p} |\beta_{rj}|. \tag{4}$$

enforces parameter selection but not variable selection. An elastic net version of the latter penalty was used by Friedman et al. (2010).

Table 5 Computation time of classifiers for mlbench.threenorm data

n	nvar	nn1	knn	wknn	lasso.sg	logit.cov	lda	qda	mboost.cov	mboost.cov.sg	mboost.cov.sd	mboost.cov.sg.sd	lasso.cov	lasso.cov.sg	lasso.cov.sd	lasso.cov.sg.sd
500	2	0.002	0.000	0.016	0.030	0.002	0.006	0.010	0.074	0.082	0.088	0.110	0.010	0.024	0.022	0.036
500	5	0.002	0.002	0.016	0.022	0.006	0.006	0.006	0.076	0.084	0.110	0.118	0.012	0.028	0.108	0.096
500	10	0.002	0.002	0.016	0.024	0.006	0.008	0.008	0.078	0.084	0.184	0.184	0.012	0.028	0.220	0.288
500	15	0.004	0.006	0.018	0.032	0.010	0.012	0.012	0.078	0.086	0.302	0.308	0.016	0.036	0.696	0.852
500	20	0.006	0.014	0.022	0.040	0.010	0.014	0.016	0.090	0.094	0.556	0.584	0.022	0.048	1.834	2.018
500	25	0.006	0.006	0.022	0.034	0.012	0.016	0.018	0.088	0.090	1.052	1.166	0.022	0.052	4.046	4.708
1000	2	0.008	0.004	0.028	0.024	0.006	0.010	0.010	0.094	0.100	0.112	0.124	0.016	0.028	0.050	0.050
1000	5	0.006	0.006	0.028	0.050	0.008	0.014	0.012	0.094	0.100	0.140	0.168	0.012	0.054	0.080	0.138
1000	10	0.010	0.010	0.034	0.162	0.012	0.012	0.016	0.096	0.104	0.206	0.228	0.016	0.062	0.252	0.456
1000	15	0.010	0.008	0.036	0.100	0.012	0.018	0.022	0.096	0.106	0.344	0.372	0.024	0.156	0.758	1.258
1000	20	0.012	0.012	0.040	0.050	0.016	0.022	0.022	0.106	0.112	0.624	0.646	0.026	0.086	1.886	2.166
1000	25	0.016	0.012	0.046	0.210	0.022	0.028	0.024	0.114	0.112	1.106	1.232	0.024	0.088	4.200	5.820
2000	2	0.012	0.016	0.062	0.122	0.018	0.038	0.018	0.128	0.142	0.156	0.166	0.022	0.120	0.084	0.208
2000	5	0.016	0.016	0.070	0.066	0.016	0.022	0.024	0.134	0.140	0.194	0.210	0.024	0.060	0.250	0.734
2000	10	0.020	0.022	0.070	0.240	0.022	0.028	0.028	0.132	0.144	0.296	0.318	0.028	0.152	1.198	0.452
2000	15	0.036	0.034	0.100	0.186	0.046	0.054	0.042	0.200	0.256	0.702	0.824	0.050	0.438	1.610	1.854
2000	20	0.036	0.034	0.090	0.174	0.032	0.038	0.038	0.144	0.150	0.790	0.810	0.036	0.160	2.580	2.490
2000	25	0.038	0.034	0.096	0.084	0.034	0.044	0.044	0.146	0.156	1.344	1.434	0.038	0.200	5.816	6.720
3000	2	0.024	0.028	0.104	0.242	0.028	0.030	0.030	0.166	0.182	0.198	0.230	0.034	0.090	0.104	0.196
3000	5	0.030	0.034	0.116	0.090	0.032	0.034	0.038	0.162	0.180	0.260	0.274	0.040	0.194	0.534	0.502
3000	10	0.040	0.044	0.128	0.104	0.036	0.048	0.044	0.168	0.182	0.384	0.394	0.046	0.128	0.464	0.624
3000	15	0.066	0.056	0.146	0.320	0.068	0.060	0.054	0.190	0.218	0.628	0.660	0.062	0.180	1.222	2.740
3000	20	0.068	0.070	0.156	0.140	0.044	0.060	0.060	0.188	0.194	0.938	0.990	0.052	0.206	3.594	3.534
3000	25	0.080	0.076	0.172	0.258	0.054	0.070	0.070	0.196	0.212	1.506	1.622	0.058	0.182	7.212	6.294

Times were measured in seconds and were averaged over five runs for every classifier

Table 6 Computation time of classifiers for mlbench.threenorm data

n	nvar	svm.cov	svm.cov.sg	svm.cov.sd	svm.cov.sg.sd	rf.cov	rf.cov.sg	rf.cov.sd	rf.cov.sg.sd
500	2	0.824	0.906	1.038	1.196	0.174	0.322	0.412	0.546
500	5	0.926	1.032	1.512	1.620	0.156	0.318	0.636	0.706
500	10	1.316	1.498	3.046	3.338	0.168	0.284	1.028	1.104
500	15	1.652	1.742	5.468	5.984	0.192	0.252	1.534	1.550
500	20	2.038	2.150	10.934	12.026	0.210	0.284	2.324	2.390
500	25	2.426	2.516	21.316	24.024	0.230	0.280	3.652	3.866
1000	2	1.932	2.124	2.272	2.454	0.290	0.712	0.942	1.168
1000	5	2.290	2.472	3.198	3.448	0.310	0.724	1.542	1.744
1000	10	2.992	3.162	5.440	5.856	0.384	0.680	2.542	2.740
1000	15	3.650	3.776	8.304	8.804	0.444	0.596	3.378	3.408
1000	20	4.372	4.498	14.510	15.500	0.522	0.702	4.950	5.142
1000	25	5.078	5.170	25.434	28.120	0.562	0.608	6.576	6.524
2000	2	6.676	6.892	7.476	7.866	0.628	1.898	2.752	3.554
2000	5	6.954	7.590	9.300	9.382	0.730	1.828	4.496	4.958
2000	10	8.412	8.666	12.180	12.536	0.902	1.746	6.962	7.676
2000	15	9.544	9.830	17.402	17.730	1.098	1.716	8.942	8.774
2000	20	10.956	11.188	25.166	25.854	1.298	1.804	11.920	11.812
2000	25	12.258	12.402	37.972	39.754	1.448	1.588	14.820	13.520
3000	2	13.782	15.154	15.160	16.970	0.966	3.080	4.716	6.192
3000	5	14.528	15.326	18.300	18.656	1.130	3.058	8.606	9.210
3000	10	15.956	16.674	23.330	23.738	1.442	3.060	12.206	12.742
3000	15	17.938	18.534	30.646	30.660	1.822	3.062	16.604	16.418
3000	20	19.888	20.124	40.224	39.512	2.116	2.770	21.022	18.730
3000	25	21.940	22.054	55.568	55.882	2.442	2.732	25.842	22.612

Times were measured in seconds and were averaged over five runs for every classifier

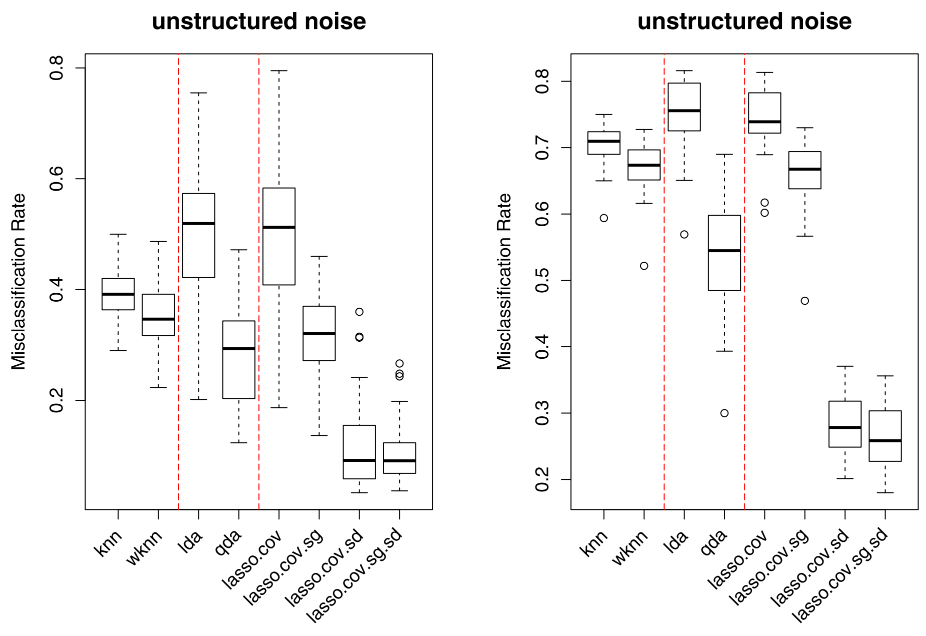


Fig. 14 Misclassification rates for simulated data with four (*left*) and 10 classes (*right*)

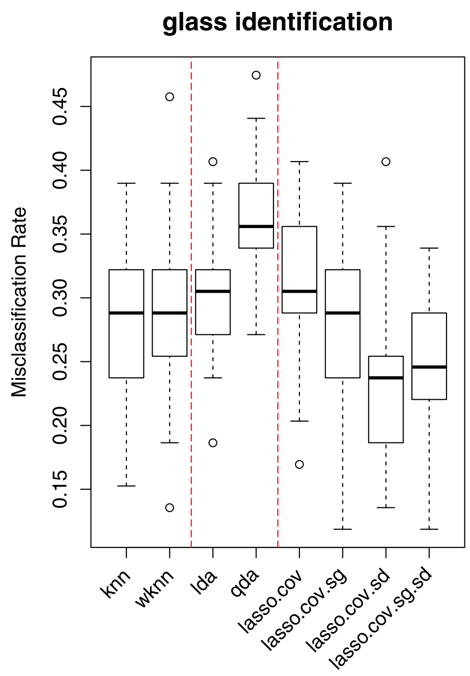


Fig. 15 Misclassification rates for glass data

It should be noted that for simplicity the structured lasso penalty (3) is given for the covariates $\boldsymbol{x}$. For the nearest neighbors included in the predictor (2) a similar form can be used but to obtain variable selection also the grouping of all the dummy variables linked to one nearest neighbor has to be taken into account, which yields a more complicated form of the penalty (see Tutz et al. 2015). We used the R package MLSP (Pößnecker 2014) that uses the structured lasso penalty.

For illustration we first use simulated data that have the structure proposed by Hastie and Tibshirani (1996). There are 4 classes that are defined as a mixture of 3 spherical bivariate normal subclasses with a standard deviation of 0.25. The means of the 12 subclasses are chosen at random (without replacement) from the 25 possible combinations of the integers [1,2,3,4,5] × [1,2,3,4,5]. The training sample is made up of 1200 observations and the test sample contains 1200 observations. In an extended version we consider 10 classes defined as the same mixture of 3 spherical bivariate normal subclasses but with the combinations drawn from $[1, \ldots, 8] \times [1, \ldots, 8]$. In the latter case the training sample and the test data contain 1500 observations each. It is seen from Fig. 14 that the inclusion of distances for single variables distinctly dominates simple NN and parametric methods. As real data example we again consider the glass data from Sect. 2.3 by including the third class of non-window glass with 51 observations. It is seen from Fig. 15 that the dominance of the NN approaches with directional sums is less strong but still yields the best results.

5 Concluding remarks

The proposed method aims at combining the best of two worlds nonparametric classification by nearest neighbors and parametric classification by the inclusion of linear or other parametric terms. Regularization techniques are used to select which parts are the most relevant. The combination is shown to distinctly improve simple nearest neighborhood estimates in most scenarios and real data sets. The performance is close and in some cases better than for SVM and random forests, which are among the strongest available classifiers. The inclusion of directional sums can strongly improve the performance if irrelevant variables are present and not too many variables are present. In the “small n large p” case global sums of nearest neighbors perform well but directional sums seem counterproductive.

References

Bache, K., Lichman, M.: Uci machine learning repository. http://archive.ics.uci.edu/ml *19* (2013)
Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996a)
Breiman, L.: Heuristics of instability and stabilisation in model selection. Ann. Stat. **24**, 2350–2383 (1996b)
Breiman, L.: Stacked regressions. Mach. Learn. **24**(1), 49–64 (1996c)
Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
Bühlmann, P., Hothorn, T.: Boosting algorithms: regularization, prediction and model fitting (with discussion). Stat. Sci. **22**, 477–505 (2007)
Bühlmann, P., Yu, B.: Boosting with the L2 loss: regression and classification. J. Am. Stat. Assoc. **98**, 324–339 (2003)
Candes, E., Tao, T.: The Dantzig selector: statistical estimation when p is much larger than n. Ann. Stat. **35**(6), 2313–2351 (2007)
Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)
Domeniconi, C., Peng, J., Gunopulos, D.: Locally adaptive metric nearest-neighbor classification. IEEE Trans. Pattern Anal. Mach. Intell. **24**, 1281–1285 (2002)
Domeniconi, C., Yan, B.: Nearest neighbor ensemble. In: Proceedings of the 17th International Conference on Pattern Recognition, vol. 1, pp. 228–231 (2004)
Fan, J., Li, R.: Variable selection via nonconcave penalize likelihood and its oracle properties. J. Am. Stat. Assoc. **96**, 1348–1360 (2001)
Fix, E., Hodges, J.L.: Discriminatory Analysis-nonparametric Discrimination: Consistency Properties. US Air Force School of Aviation Medicine, Randolph Field (1951)
Friedman, J., Hastie, T., Tibshirani, R.: Regularization paths for generalized linear models via coordinate descent. J. Stat. Softw. **33**(1), 1–22 (2010)
Friedman, J.H.: Flexible metric nearest neighbor classification. Technical report 113, Stanford University, Statistics Department (1994)
Friedman, J.H., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Ann. Stat. **28**, 337–407 (2000)
Gertheiss, J., Tutz, G.: Feature selection and weighting by nearest neighbor ensembles. Chemom. Intell. Lab. Syst. **99**, 30–38 (2009)
Ghosh, A.K.: On nearest neighbor classification using adaptive choice of k. J. Comput. Graph. Stat. **16**(2), 482–502 (2007)
Ghosh, A.K.: A probabilistic approach for semi-supervised nearest neighbor classification. Pattern Recognit. Lett. **33**(9), 1127–1133 (2012)

Ghosh, A.K., Godtliebsen, F.: On hybrid classification using model assisted posterior estimates. Pattern Recognit. **45**(6), 2288–2298 (2012)

Goeman, J. J.: Penalized: weighted k-nearest neighbors. R package version 0.9–42 (2012)

Hall, P., Park, B.U., Samworth, R.J.: Choice of neighbor order in nearest-neighbor classification. Ann. Stat. **36**, 2135–2152 (2008)

Hastie, T., Tibshirani, R.: Discriminant adaptive nearest-neighbor classification. IEEE Trans. Pattern Anal. Mach. Intell. **18**, 607–616 (1996)

Hastie, T., Tibshirani, R., Friedman, J.H.: The Elements of Statistical Learning, 2nd edn. Springer, New York (2009)

Holmes, C., Adams, N.: A probabilistic nearest neighbour method for statistical pattern recognition. J. R. Stat. Soc.: Ser. B (Stat. Methodol.) **64**(2), 295–306 (2002)

Holmes, C.C., Adams, N.M.: Likelihood inference in nearest-neighbour classification models. Biometrika **90**(1), 99–112 (2003)

Hothorn, T.: TH.data: TH's data archive. R package version 1.0-3 (2014)

Hothorn, T., Buehlmann, P., Kneib, T., Schmid, M., Hofner, B.: Mboost: model-based boosting. R package version 2.2-3 (2013)

Leisch, F., Dimitriadou, E.: mlbench: Machine learning benchmark problems. R package version 2.1-1 (2010)

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

Lin, Y., Jeon, Y.: Random forests and adaptive nearest neighbors. J. Am. Stat. Assoc. **101**, 578–590 (2006)

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F.: e1071: Misc functions of the department of statistics (e1071), TU Wien. R package version 1.6-2 (2014)

Morin, R.L., Raeside, D.E.: A reappraisal of distance-weighted k-nearest neighbor classification for pattern recognition with missing data. IEEE Trans. Syst. Man Cybern. **11**, 241–243 (1981)

Nadaraya, E.A.: On estimating regression. Theory Probab. Appl. **10**, 186–190 (1964)

Paik, M., Yang, Y.: Combining nearest neighbor classifiers versus cross-validation selection. Stat. Appl. Genet. Mol. Biol. **3**(12), 1–19 (2004)

Park, M.Y., Hastie, T.: An l1 regularization-path algorithm for generalized linear models. J. R. Stat. Soc. B **69**, 659–677 (2007)

Parthasarthy, G., Chatterji, B.N.: A class of new knn methods for low sample problems. IEEE Trans. Syst. Man Cybern. **20**, 715–718 (1990)

Pößnecker, W.: MRSP: multinomial response models with structured penalties. R package version 0.4.2 (2014)

R Core Team: R: A language and environment for statistical computing. Vienna, Austria: R Foundation for Statistical Computing (2013)

Ripley, B.D.: Pattern Recognition and Neural Networks. Cambridge University Press, Cambridge (1996)

Schliep, K., Hechenbichler, K.: kknn: Weighted k-nearest neighbors. R package version 1.2-3 (2013)

Silverman, B.W., Jones, M.C.: Commentary on fix and hodges (1951): an important contribution to nonparametric discriminant analysis and density estimation. Int. Stat. Rev. **57**, 233–238 (1989)

Simonoff, J.S.: Smoothing Methods in Statistics. Springer, New York (1996)

Stone, C.J.: Consistent nonparametric regression (with discussion). Ann. Stat. **5**, 595–645 (1977)

Tibshirani, R.: Regression shrinkage and selection via the lasso. J. R. Stat. Soc. B **58**, 267–288 (1996)

Tibshirani, R., Chu, G., Narasimhan, B., Li, J.: samr: SAM: Significance analysis of microarrays. R package version 2.0 (2011)

Tutz, G., Binder, H.: Generalized additive modeling with implicit variable selection by likelihood-based boosting. Biometrics **62**, 961–971 (2006)

Tutz, G., Pössnecker, W., Uhlmann, L.: Variable selection in general multinomial logit models. Comput. Stat. Data Anal. **82**, 207–222 (2015)

Venables, W.N., Ripley, B.D.: Modern Applied Statistics with S (Fourth ed.). New York: Springer. ISBN 0-387-95457-0 (2002)

Watson, G.S.: Smooth regression analysis. Sankhyā, Ser. A **26**, 359–372 (1964)

Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. J. R. Stat. Soc. B **67**, 301–320 (2005)