Feature Subset Selection with Adaboost and ADTboost

Martin Drauschke

martin.drauschke@uni-bonn.de

TR-IGG-P-2008-04

25th March 2008

Technical Report Nr. 4, 2008

Department of Photogrammetry
Institute of Geodesy and Geoinformation
University of Bonn

Available at
http://www.ipb.uni-bonn.de/~martin/

Feature Subset Selection with Adaboost and ADTboost

Martin Drauschke

martin.drauschke@uni-bonn.de

Abstract

This technical report presents feature subset selection methods for two boosting classification frameworks: Adaboost and ADTboost.

1 Introduction

Feature selection is a challenging task during object recognition and image interpretation. Especially, highly structured objects with a rich diversity and in variable environments are often only separable using high dimensional feature vectors. Then, the explosion of extracted features has to be faced by reduction of the redundant ones to avoid computational costs.

Also, feature selection is used in data mining to extract useful and comprehensible information from data, cf. [Liu & Motoda, 1998]. One classical approach, the principal component analysis (PCA), reduces the dimension of the feature space by projecting all features, cf. [Bishop, 2006]. The resulting feature set of a PCA is not necessarily a subset of all candidate features, but combinations of the original features. Thus, the PCA is no appropriate tool, if one wants to obtain a real subset of features for further investigations.

Structure of the Technical Report. In this technical report, we first give a review on feature subset selection principles and methods in section 2. The commonly used techniques can be combined with many single classifications, but we are interested to combine many weak classifiers to build one strong classifier. This framework is presented in section 3, where we discuss the functionality of two boosting methods: Adaboost and ADTboost. In section 4, we present our feature subset selection scheme. An evaluation of these strategies and an application on detecting buildings and building parts is given in further publications.

Notation. Now, we want to formalize the problem and introduce our notation: Given is a data set of N training samples (x_n, y_n), where $x_n = [f_1^n, .., f_D^n]$ is a real-valued feature vector with dimension D, and y_n is the class membership, e. g. $y_n \in \{-1, +1\}$ in the binary case, or $y_n \in \{1, .., K\}$ if K classes are given. For now, we only considered the binary case, but all applied algorithms have already been introduced for the multi-class case, cf. [Schapire & Singer, 1999] and [Holmes *et al.*, 2002], respectively. Our final goal is not only the classification of these feature vectors, we additionally want to select the best features to reduce the dimension of the feature space and to eliminate redundant features.

2 Feature Subset Selection Principles

If we want to select a subset of appropriate features from the total set of features with cardinality D, we have a choice between 2^D possibilities. If we deal with feature vectors with more than 100 components, the exhaustive search takes too much time. Thus, we must find other ways to select a subset of features. The principle techniques are discussed in the following subsection.

There are mainly two strategies for avoiding the complete search: random-based and deterministic greedy algorithms. According to [Mladenić, 2006], the following basic search strategies are used for selecting feature subsets.

- **Forward selection.** This technique starts with an empty set and greedily adds the best of the remaining features to this set. This process is called **stepwise**, if only one feature is added in each step.
- **Backward elimination.** Here, we start with the full set containing all features. Then we greedily remove the most useless features from this set. This process is called **stepwise**, if only one feature is added in each step.
- **Random mutation.** This strategy starts with a randomly selected feature set and adds randomly selected features or removes them from the set.

2.1 Deterministic approaches

Figure 1 shows the forward stepwise selection search paths for an example with four features. The top node, where the search starts, contains the empty set. In the first step, we can chose between all four features and select the

best feature. Afterwards, we check, if the selection of an additional feature will lead to a better subset, and so on. The backward stepwise selection would take the inverted paths and starts with the complete set of features. Obviously, both strategies can miss the global optimum, if a wrong path has been taken in an earlier step.

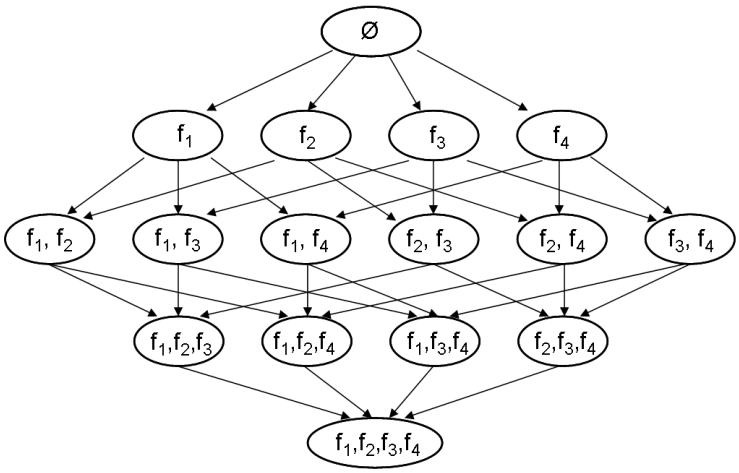

Figure 1: Forward stepwise selection search paths for selecting the best of four features. Start is the top node, the empty set, and then one feature is added to the set until e. g. a local optimum has been found.

Alternatively, feature selection methods can be divided into **filters**, **wrappers** and **embedded approaches**, cf. [Mladenić, 2006]. For analyzing the relevance of feature subsets, filters use evaluation functions that are independent from the learning method, while wrappers evaluate the feature subsets in respect to the learning result. In embedded approaches, the feature subset selection and the learning method are interleaved.

As filter techniques, [Guyon & Elisseeff, 2003] propose feature subset selection methods which are based on variable ranking. The ranking can be done by e. g. squared Pearson's Correlation Coefficient. Other ranking criteria, as listed in [Guyon & Elisseeff, 2003], are on the mutual information between each variable and the class index or on the predictive power of each single variable. Each of these ranking methods is followed by the classification step itself, which is done by only one classifier performing a e. g. linear discriminant analysis or Naive Bayes classification.

These classification methods do not work, if the features are highly correlated, and the training samples do not form compact clusters in feature space. In such cases, it is much better to use a feature subset selection scheme which

combines several classifiers. This leads us to the concept of adaptive boosting (Adaboost) where a strong (or highly accurate) classifier is found by combining several weak (or less accurate) classifiers, cf. [Schapire, 1990].

2.2 Random approaches

Genetic algorithms are commonly used to select the best feature subset randomly, cf. [Martin-Bautista & Vila, 1999]. The principle of genetic algorithms is as follows:

> [...] a population of individuals representing possible space solutions is maintained through several generations. Each individual is evaluated in each generation based on its fitness with respect to the considered function to minimize or maximize. The selection, crossover and mutation of the fittest ones produce new individuals. Through the generations, the population is led to the space of the better solutions in the given domain.
>
> [Martin-Bautista & Vila, 1999]

Genetic algorithms are relatively insensitive to noise, and there is no domain knowledge required. This makes genetic algorithms so favored in many applications. But we want to learn the relevance of features from only a few samples (if possible). Then, the creation of new individuals within the generation development might lead to wrong solutions. Furthermore, the computational time of genetic algorithms in combination with a wrapped classification method is not efficient, cf. [Martin-Bautista & Vila, 1999].

Another technique based on random decisions are random forests, cf. [Breiman, 2001, Ho, 1998]. The classification with random forests is based on a majority vote in respect to the results of weak randomly constructed decision trees. In [Ho, 1998], a random selection of a predefined small number of features is used for constructing decision trees. This procedure is equivalent to projecting the feature vectors into a space with much lower dimensionality, but the projections differ from one decision tree to the others. The decision trees in [Breiman, 2001] randomly choose a feature from the whole feature set and takes it for determining the best domain split. Both methods only work well, if the number of random decision trees is large, especially if there are only features which are nearly uncorrelated with the classes. In his experiments, [Breiman, 2001] uses five times more decision trees than weak learners in Adaboost. Furthermore, if the number of decision trees is high, the number of (randomly) selected features is also quite high. Then, we do

not benefit on feature subset selection, because almost every feature has been selected.

[Rogers & Gunn, 2006] show that "the lack of implicit feature selection within random forest can result in a loss of accuracy and efficiency, if irrelevant features are not removed". They use the expected information gain as a criterion for removing such irrelevant features, thus their feature selection is not based on classification results. We would prefer wrapper methods instead of filter methods to assure the get nearly optimal classification results. That's why we studied the boosting methods in more detail.

3 Adaboost and ADTboost

3.1 Adaboost

In this subsection, we introduce the adaptive boosting (Adaboost) framework. For better readability, we firstly introduce our notation in table 1, and then we explain the principle of Adaboost.

Table 1: Notation for Boosting algorithms.

x_n	D-dimensional feature vector, $n = 1 \dots N$
y_n	binary target $\{+1, -1\}$, $n = 1 \dots N$
t	iteration index of boosting algorithm, $t = 1 \dots T$
c_j	classifier candidate, $j = 1 \dots J_t$
W_t^n	Weight of n-th sample in t-th iteration
$W_+(c_j)$	sum of weights of all samples with target $+1$, where c_j predicts $+1$
$W_-(c_j)$	sum of weights of all samples with target -1, where c_j predicts $+1$
$W_+(\neg c_j)$	sum of weights of all samples with target $+1$, where c_j predicts -1
$W_-(\neg c_j)$	sum of weights of all samples with target -1, where c_j predicts -1
$W_*(\neg c_j)$	sum of weights of all samples, where c_j predicts -1
h_t	best weak classifier at iteration t, $h_t : x \mapsto \{+1, -1\}$
α_t	predictive value of h_t
H	strong classifier, $H : x \mapsto \{+1, -1\}$

The basic Adaboost algorithm as presented by [Freund & Schapire, 1996] is sketched in fig. 2. The first weak classifier (or best hypothesis) is learnt on equally treated training samples (x_n, y_n). Then, before training the second weak classifier, the influence of all misclassified samples gets increased by adjusting the weights of the feature vectors. So, the second classifier will focus especially on the previously misclassified samples. In the third step, the weights are readjusted once more depending on the classification result

of the second weak classifier before training the third weak classifier, and so on.

Input: $(x_1, y_1),\ldots, (x_N, y_N)$.
Initialization: $W_1^n = 1/N$.
Repeat $t = 1,\ldots, T$:

- Determine best weak hypothesis h_t using W_t.
- Determine α_t
- Determine distribution W_{t+1}

Output: H with $H(x) = \text{sign}\ (\sum_t \alpha_t h_t(x))$.

Figure 2: Algorithm of Adaboost (binary case) by [Schapire & Singer, 1999]

In more detail: Let there be a set of J_t classifier candidates c_j in the t-th iteration of Adaboost. Then, we determine its discriminative power for each classifier candidate c_j by

$$Z_j = 2\left(\sqrt{W_+(c_j)W_-(c_j)} + \sqrt{W_+(\neg c_j)W_-(\neg c_j)}\right), \tag{1}$$

where $W_+(c_j)$ is the sum of all weights from samples of the positive class which have been classified to the positive class, and $W_-(\neg c_j)$ is the sum of all weights from samples of the negative class which have been classified to the negative class etc. Then, the best weak classier is that candidate which minimizes Z:

$$h_t = c_t \ \ with\ t = \text{argmin}_j Z_j. \tag{2}$$

In the next step, the weak classifier h_t receives a predictive value α_t which depends on its success rate. According to [Schapire & Singer, 1999], the predictive value α_t is derived from the classification result r_t of the weak classifier h_t:

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1 + r_t + \epsilon}{1 - r_t + \epsilon}\right) \tag{3}$$

where

$$r_t = \sum_{n=1}^{N} W_t^n y_n h_t(x_n). \tag{4}$$

The product $y_n h_t(x_n)$ is $+1$, if x_n is correctly classified by h_t, otherwise the product is -1. Since W_t is a distribution, e. g. $\sum_n W_t^n = 1$, we derive as a range for the classification result:

$$-1 \leq r_t \leq +1. \tag{5}$$

r_t is $+1$, if all samples have been correctly classified, and it is -1, if all samples have been misclassified. So, the range for α_t is

$$-\infty < \alpha_t < +\infty, \tag{6}$$

and

$$\alpha_t \begin{cases} > 0, \text{ if } r_t > 0 \\ = 0, \text{ if } r_t = 0 \\ < 0, \text{ if } r_t < 0 \end{cases} \tag{7}$$

The updating of the weights is based on the classification of the last chosen weak classifier h_t:

$$W_{t+1}^n = W_t^n \exp\left(-\alpha_t y_n h_t(x_n)\right), \tag{8}$$

and afterwards W_{t+1} is normalized again. Thus, the weight $W_{t+1}(n)$ of the n-th sample increases if it has been misclassified by h_t, otherwise the weight of the sample decreases, cf. [Schapire & Singer, 1999].

Finally, after T weak classifiers have been chosen, the result of the strong classifier H can be depicted as the sign of the weighted sum of the results of the weak classifiers:

$$H(x) = \text{sign}\left(\sum_t \alpha_t h_t(x)\right). \tag{9}$$

The discriminative power of the resulting classifier H can be expected to be much higher than the discriminative power of each weak classifier h_t, cf. [Rätsch *et al.*, 2001].

The original Adaboost algorithm has been adapted in various ways. One important improvement is ADTboost, where the weak classifiers can additionally be arranged in a hierarchical order. We introduce it in the next subsection.

3.2 ADTboost

The boosting with Alternating Decision Trees (ADTboost) as proposed by [Freund & Mason, 1999] is an extension of Adaboost which has been clearly re-formulated by [De Comité *et al.*, 2001]. The extensions of Adaboost towards ADTboost are the following: Primarily, the weak classifiers are put into a hierarchical order - the *alternating decision tree*. The tree consists of two different kind of nodes which alternately change on a path through the tree, cf. fig. 3. Secondly, each *decision node* contains a weak classifier and has two *prediction nodes* as its children. These two predictive values α_t^+ and α_t^- depict the performance of the weak classifier, similarly to the α_t in Adaboost. The weak classifiers in upper levels of the tree work as preconditions on those classifiers below them. If not all preconditions are fulfilled

for a given sample, the response of a weak classifier is set to 0. And third, the root node contains the predictive value α_0 of the `true`-classifier, which always returns +1. Thus, α_0 is derived from the ratio of the number of samples between both classes and, therefore, it can be interpreted as a a prior classifier.

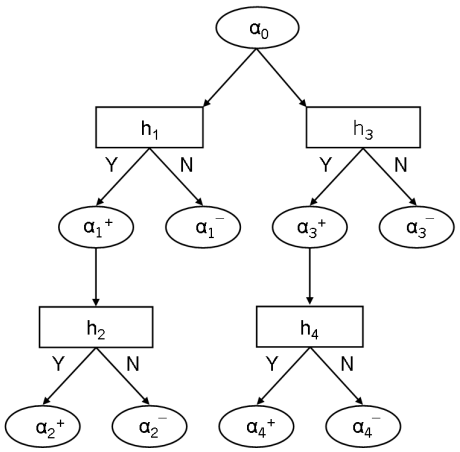

Figure 3: Alternating Decision Tree: Rectangular nodes represent a weak classifier, elliptic nodes the predictive values. The root node contains the predictive value α_0 of the `true`-classifier. Due to the hierarchy of the classifiers, h_1 and h_3 are pre-classifiers of h_2 and h_4, respectively.

Before starting ADTboost, we have to set the samples' weights, and, additionally, we declare a set of preconditions $\mathcal{P}$. Initially, this set only contains a tautological `true`-classifier. At the end of each iteration step, $\mathcal{P}$ is extended by two additional preconditions h_t^+ and h_t^-. (Again, h_t is the currently chosen weak classifier). So, in the t-th iteration step, we select the best weak classifier from the combinations of all $2 \cdot (t-1) + 1$ preconditions with all unselected classifier candidates. This quadratic search can be reduced to a nearly linear search by a heuristic search, cf. [Pfahringer *et al.*, 2001]. If the best classifier candidate is c_j with its precondition h_p, then the new best weak classifier h_t is set $h_t = h_p \wedge c_j$, and the set of preconditions will be extended by $\mathcal{P} = \mathcal{P} \cup \{h_p \wedge c_j, h_p \wedge \neg c_j\}$. Technically, we determine the predictive values with $\epsilon = 1$ by

$$\alpha_t^+ = \frac{1}{2} \ln \left(\frac{W_+(h_p \wedge c_j) + \epsilon}{W_-(h_p \wedge c_j) + \epsilon} \right) \tag{10}$$

and

$$\alpha_t^- = \frac{1}{2} \ln \left(\frac{W_+(h_p \wedge \neg c_j) + \epsilon}{W_-(h_p \wedge \neg c_j) + \epsilon} \right), \tag{11}$$

and the update of the sample's weights at the end of each iteration is done by

$$W_{t+1}(n) = W_t(n) \exp\left(-r_t(x_n) y_n\right), \tag{12}$$

where

$$r_t(x_n) = \begin{cases} = \alpha_t^+, \text{ if } h_p(x_n) = +1 \text{ and } c_j(x_n) = +1 \\ = \alpha_t^-, \text{ if } h_p(x_n) = +1 \text{ and } c_j(x_n) = -1 \\ = 0, \text{ if } h_p(x_n) = -1. \end{cases} \tag{13}$$

If a sample does not fulfill the precondition h_p, i. e. $h_p(x_n) = -1$, then the sample's weight remains the same. In the initialization step, we start with equal weights $W_1(n) = 1$, and there is no normalization of the weights neither within the initialization step, nor after each update of the weights. But in our point of view, this is only a minor change which effects in more significant changes of the weights.

The determination of the best classifier candidate has also been slightly adapted. Considering Adaboost, the Z-value describes the separability of a classifier candidate c_j. Now we have to integrate the separability of the precondition h_p as well:

$$\begin{aligned} Z_{pj} &= 2\left(\sqrt{W_+(h_p \wedge c_j) W_-(h_p \wedge c_j)} + \sqrt{W_+(h_p \wedge \neg c_j) W_-(h_p \wedge \neg c_j)}\right) \\ &\quad + W_*(\neg h_p) \end{aligned} . \tag{14}$$

The best weak classifier minimizes Z over all h_p and all c_j.

Since ADTboost is an extension of Adaboost, we can visualize the resulting Adaboost-classifier by an equivalent tree structure. Then, we have to fill the two prediction nodes beneath the decision node of the classifier with the positive and negative of the weak classifier's weight. Furthermore, all decision nodes are fastened onto the prediction root node which contains a 0, since Adaboost has not integrated any prior decision. Fig. 4 shows the ADT-like visualization of a strong Adaboost classifier with four weak classifiers.

3.3 A synthetic example: Adaboost vs. ADTboost

We implemented Adaboost and ADTboost using the formulations presented here which are taken from the algorithms in [Schapire & Singer, 1999] and [De Comité *et al.*, 2001], respectively. Both algorithms are optimized with respect to find the best weak classifier, to determine the classifier's weight or the predictive values, respectively, and to update each sample's weight.

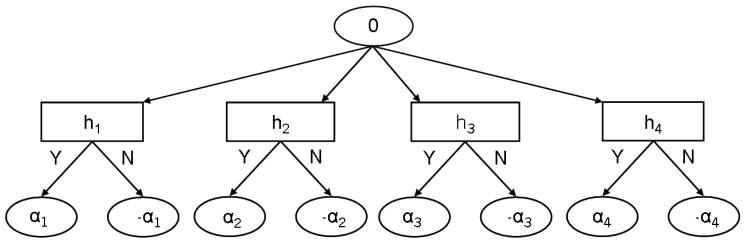

Figure 4: ADT-like visualization of classification with Adaboost.

There are only two unsolved questions that are left to the user. The first question refers to the maximum number of iterations. We stop the training either after T steps or earlier, if the error rate on the training samples is below threshold θ. The other question deals with the generation of the set of classifier candidates. Since we are interested in the subset selection of the given features, we consider only threshold classification on single features. Then, we avoid the combinatorial explosion which would occur, if we would also consider linear separation with 2 or more features. Thus, our decision trees will be k-d-trees and no oblique decision trees or other domain partitioning structures.

We want to discuss the functionality of both algorithms by presenting their workflows on a synthetic data set, which we visualized in fig. 5. It consists of points which belong to two classes (red and blue). The target of the red class is $y_n = 1$, and the membership to blue class is encoded by $y_n = -1$. We work with a 2-dimensional feature vector, and its domain is $x_n \in [0,1] \times [0,1]$. The class membership of a given data point x_n is defined by

$$y_n = \begin{cases} +1\,(\text{red}) & - & \text{if } f_1^n < 0.4 \,\text{and}\, f_2^n > 0.4 \,\text{or} \\ & & f_1^n > 0.6 \,\text{and}\, f_2^n < 0.6 \\ -1\,(\text{blue}) & - & \text{else.} \end{cases} \tag{15}$$

Besides the four domain bounds, we only need four other discriminative lines for describing the feature set. Therefore, we limit our set of classifier

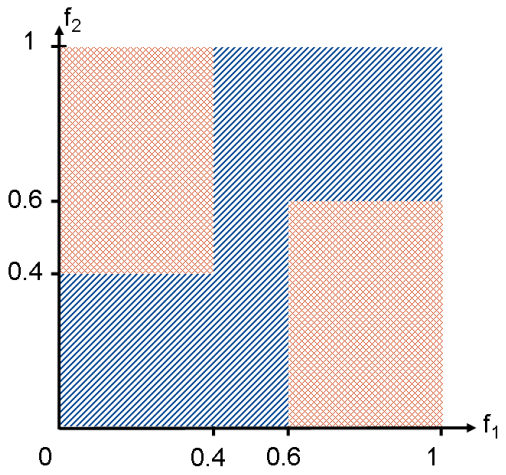

Figure 5: Synthetic data set with 2 features and 2 classes (red and blue).

candidates onto these eight items:

$$\begin{aligned}
c_1 &: \texttt{if } f_1 < 0.4 \texttt{ then red else blue} \\
c_2 &: \texttt{if } f_1 > 0.6 \texttt{ then red else blue} \\
c_3 &: \texttt{if } f_2 < 0.4 \texttt{ then red else blue} \\
c_4 &: \texttt{if } f_2 > 0.6 \texttt{ then red else blue} \\
c_5 &: \texttt{if } f_1 < 0 \texttt{ then red else blue} \\
c_6 &: \texttt{if } f_1 > 1 \texttt{ then red else blue} \\
c_7 &: \texttt{if } f_2 > 0 \texttt{ then red else blue} \\
c_8 &: \texttt{if } f_2 < 1 \texttt{ then red else blue}
\end{aligned} \tag{16}$$

If a candidate is selected, we also determine, if c_j or $\neg c_j$ is the best weak classifier according to their classification errors. We do not list both classifiers in the set of candidates, because we get the same Z-value for both candidates due to the symmetry of its determination. Furthermore, we want to avoid duplicate selections of weak classifiers. Considering Adaboost, we reduce the set of classifier candidates by the chosen weak classifier at the end of each iteration step. In respect to ADTboost, this procedure is a little more complicated, because we ignore a already chosen classifier only, if it part of the precondition. So, a weak classifier may occur twice or more often, if the preconditions of these weak classifiers differ. The calculations can further be reduced, because all four domain bound classifiers c_5-c_8 will produce the same Z-values.

3.3.1 Adaboost

In this part, we document the functionality of Adaboost. Initializing the system, we only have to set up the weights, including their normalization. Since we will not gain an errorless classification, we will stop after the fourth iteration, when only the domain bounds classifiers are left to choose.

1st Iteration. In the first iteration step, the classifier candidates produce the following Z-values:

$$\begin{aligned}
c_1: &\ \begin{matrix} W_+(c_1) = 0.24, W_+(\neg c_1) = 0.24 \\ W_-(c_1) = 0.16, W_-(\neg c_1) = 0.36 \end{matrix} \Rightarrow Z_1 = 0.9798 \\
c_2: &\ \begin{matrix} W_+(c_2) = 0.24, W_+(\neg c_2) = 0.24 \\ W_-(c_2) = 0.16, W_-(\neg c_2) = 0.36 \end{matrix} \Rightarrow Z_2 = 0.9798 \\
c_3: &\ \begin{matrix} W_+(c_3) = 0.28, W_+(\neg c_3) = 0.24 \\ W_-(c_3) = 0.32, W_-(\neg c_3) = 0.16 \end{matrix} \Rightarrow Z_3 = 0.9906 \\
c_4: &\ \begin{matrix} W_+(c_4) = 0.28, W_+(\neg c_4) = 0.24 \\ W_-(c_4) = 0.32, W_-(\neg c_4) = 0.16 \end{matrix} \Rightarrow Z_4 = 0.9906 \\
c_5: &\ \begin{matrix} W_+(c_5) = 0, W_+(\neg c_5) = 0.48 \\ W_-(c_5) = 0, W_-(\neg c_5) = 0.52 \end{matrix} \Rightarrow Z_5 = 0.9992
\end{aligned} \tag{17}$$

The first candidate minimizes Z, so c_1 becomes the first weak classifier $h_1 = c_1$. Then, its predictive value is

$$\alpha_1 = \frac{1}{2} \ln \left(\frac{1 + 0.2}{1 - 0.2} \right) = 0.2027. \tag{18}$$

So far, the total classification will return 40% errors on the training data.

2nd Iteration. Thus, we perform the second iteration step, but without considering c_1 as a candidate:

$$\begin{aligned}
c_2: &\ \begin{matrix} W_+(c_2) = 0.2976, W_+(\neg c_2) = 0.2016 \\ W_-(c_2) = 0.1344, W_-(\neg c_2) = 0.3664 \end{matrix} \Rightarrow Z_2 = 0.9436 \\
c_3: &\ \begin{matrix} W_+(c_3) = 0.2352, W_+(\neg c_3) = 0.2656 \\ W_-(c_3) = 0.3008, W_-(\neg c_3) = 0.1984 \end{matrix} \Rightarrow Z_3 = 0.9912 \\
c_4: &\ \begin{matrix} W_+(c_4) = 0.2992, W_+(\neg c_4) = 0.2016 \\ W_-(c_4) = 0.3648, W_-(\neg c_4) = 0.1344 \end{matrix} \Rightarrow Z_4 = 0.9900 \\
c_5: &\ \begin{matrix} W_+(c_5) = 0, W_+(\neg c_5) = 0.4992 \\ W_-(c_5) = 0, W_-(\neg c_5) = 0.5008 \end{matrix} \Rightarrow Z_5 = 1.0000
\end{aligned} \tag{19}$$

The second candidate minimizes Z, so c_2 becomes the secondly chosen weak classifier $h_2 = c_2$, and its predictive value is

$$\alpha_2 = \frac{1}{2} \ln \left(\frac{1 + 0.328}{1 - 0.328} \right) = 0.3406. \tag{20}$$

After the first two iterations, the strong classifier consists of two weak classifiers, and it performs a classification with still an error of 40%.

3rd Iteration. At this stage, we want to demonstrate, that Adaboost may select a weak classifier twice, if one does not reduce set of candidates. In this case, we would not improve the total classification, because there is no new weak classifier, but its predictive value will change. If this duplicate selection repeats again and again, we would only suffer from computational costs.

So, we obtain for all five listed candidates

$$\begin{aligned}
c_1 &: \begin{matrix} W_+(c_1) = 0.3, W_+(\neg c_1) = 0.2232 \\ W_-(c_1) = 0.1488, W_-(\neg c_1) = 0.3280 \end{matrix} \Rightarrow Z_1 = 0.9638 \\
c_2 &: \begin{matrix} W_+(c_2) = 0.2232, W_+(\neg c_2) = 0.3 \\ W_-(c_2) = 0.2, W_-(\neg c_2) = 0.2768 \end{matrix} \Rightarrow Z_2 = 0.999 \\
c_3 &: \begin{matrix} W_+(c_3) = 0.2768, W_+(\neg c_3) = 0.2 \\ W_-(c_3) = 0.3744, W_-(\neg c_3) = 0.1488 \end{matrix} \Rightarrow Z_3 = 0.9888 \\
c_4 &: \begin{matrix} W_+(c_4) = 0.2256, W_+(\neg c_4) = 0.2512 \\ W_-(c_4) = 0.3232, W_-(\neg c_4) = 0.2 \end{matrix} \Rightarrow Z_4 = 0.9882 \\
c_5 &: \begin{matrix} W_+(c_5) = 0, W_+(\neg c_5) = 0.5232 \\ W_-(c_5) = 0, W_-(\neg c_5) = 0.4768 \end{matrix} \Rightarrow Z_5 = 0.9989
\end{aligned} \tag{21}$$

Since we discard the candidates which have been selected so far, c_3 is the the best candidate left. Without discarding, we would enter a loop where the candidates c_1 and c_2 would be reselected again and again. Since the total Adaboost classification would perform better, we favor the discarding of these classifier candidate set after their first selection. The predictive value of the third weak classifier $h_3 = \neg c_4$ is

$$\alpha_3 = \frac{1}{2} \ln \left(\frac{1 + 0.1488}{1 - 0.1488} \right) = 0.1499. \tag{22}$$

Now, the strong classifier which relies on the predictions of the three weak classifiers has a classification error of 32%.

4th Iteration. In the fourth iteration, we can only choose between the original third classifier candidate c_3 and the domain bounds:

$$\begin{aligned}
c_3 &: \begin{matrix} W_+(c_3) = 0.2928, W_+(\neg c_3) = 0.2216 \\ W_-(c_3) = 0.3640, W_-(\neg c_3) = 0.1216 \end{matrix} \Rightarrow Z_3 = 0.9814 \\
c_5 &: \begin{matrix} W_+(c_5) = 0, W_+(\neg c_5) = 0.4856 \\ W_-(c_5) = 0, W_-(\neg c_5) = 0.5144 \end{matrix} \Rightarrow Z_5 = 0.9996
\end{aligned} \tag{23}$$

Then, in the fourth iteration, the third candidate is also chosen as a weak classifier $h_4 = \neg c_3$ with its predictive value

$$\alpha_4 = \frac{1}{2} \ln \left(\frac{1 + 0.1712}{1 - 0.1712} \right) = 0.1729. \tag{24}$$

Unfortunately, the strong classifier is not improved by this additional weak classifier, so we remain with a final classification error of 32%. The final predictions of the Adaboost classifier are obtained for a given $x = [f_1, f_2]$ by the sign of the weighted sums of the predicitions of all weak classifiers:

$\sum \alpha_t h_t(x)$	$f_1 < 0.4$	$0.4 \leq f_1 \leq 0.6$	$f_1 > 0.6$
$f_2 > 0.6$	-0.1149	-0.5203	0.1609
$0.4 \leq f_2 \leq 0.6$	0.1849	-0.2205	0.4607
$f_2 < 0.4$	-0.1609	-0.5663	0.1149

. (25)

Classification errors only occur, if $f_1 < 0.4 \wedge f_2 > 0.6$ or if $f_1 > 0.4 \wedge f_2 > 0.6$. In all other subdomains, Adaboost predicts correctly. The strong Adaboost classifier is visualized by the tree representation from ADTboost in fig. 6. We do not consider further iterations, because the domain bounds will always have an error of at least 48%. In reality, Adaboost could choose several other axis-parallel classifiers as $f_1 < 0.5$, and so the classification can furtherly get improved.

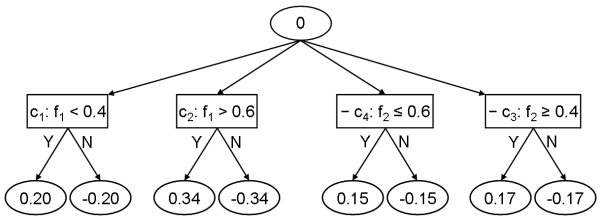

Figure 6: Visualization of Adaboost's result within the tree-drawing framework of ADTboost.

3.3.2 ADTboost

In this part, we document the functionality of ADTboost. First, we initialize the system, and then we discuss each iteration step until the total classification error is decreased to 0% after the 4th step.

Initialization. The initialization ADTboost contains the three steps. Firstly, we define the first weight per sample, and we define the set of preconditions,

containing only the `true`-classifier. We initially set all weights to 1 without a normalization step afterwards. The domain of our synthetic data set can be divided in 9 subdomains, according to first four classifier candidates in eq. 25. When presenting the weights and their sums in respect to the success of a classifier candidate, we use 100 samples distributed according to the area of these subdomains. Only the amounts of the summed weights are effected by this decision. The ratios of the weights stay similar and the minimum detection in respect to Z remains the same.

In figs. 7 and 8, we present the weight of one sample of each subdomain. The upper left corner stands for the subdomain with the bounds $0 \leq f_1 \leq 0.4$ and $0.6 \leq f_1 \leq 1$, etc. Secondly, we determine the prior classifier

1	1	1
1	1	1
1	1	1

Figure 7: Initialization of weights.

0.96	1.04	1.04
0.96	1.04	0.96
1.04	1.04	0.96

Figure 8: Weights after integrated prior classification.

$$\alpha_0 = \frac{1}{2} \ln \left(\frac{W_+(\texttt{true})}{W_-(\texttt{true})} \right) = -0.04, \tag{26}$$

which is the predictive value of the `true`-classifier. In the third step, we update the weights in respect to the results of the prior classification, see fig. 8.

1st Iteration. The first iteration of ADTboost starts with only one precondition $h_0 : \texttt{true}$ and we may select each of the eight given classifier candidates from eq. 25. We determine the following Z-values for these classifier candidates:

$$\begin{aligned}
c_1 &: \begin{array}{l} W_+(c_1) = 23.04, W_+(\neg c_1) = 16.64 \\ W_-(c_1) = 23.04, W_-(\neg c_1) = 37.44 \end{array} \Rightarrow Z_{01} = 97.90 \\
c_2 &: \begin{array}{l} W_+(c_2) = 23.04, W_+(\neg c_2) = 16.64 \\ W_-(c_2) = 23.04, W_-(\neg c_2) = 37.44 \end{array} \Rightarrow Z_{02} = 97.90 \\
c_3 &: \begin{array}{l} W_+(c_3) = 15.36, W_+(\neg c_3) = 24.96 \\ W_-(c_3) = 30.72, W_-(\neg c_3) = 29.12 \end{array} \Rightarrow Z_{03} = 98.98 \\
c_4 &: \begin{array}{l} W_+(c_4) = 15.36, W_+(\neg c_4) = 24.96 \\ W_-(c_4) = 30.72, W_-(\neg c_4) = 29.12 \end{array} \Rightarrow Z_{04} = 98.98 \\
c_5 &: \begin{array}{l} W_+(c_5) = 0, W_+(\neg c_5) = 46.08 \\ W_-(c_5) = 0, W_-(\neg c_5) = 54.08 \end{array} \Rightarrow Z_{05} = 99.84
\end{aligned} \tag{27}$$

Again, Z is a function for evaluating the separability of a classifier. A classifier is more discriminative, if it has a less Z-value. So, obviously, the first chosen weak classifier is $h_1 = c_1$ with the `true`-classifier as its precondition. The predictive values of the first weak classifier h_1 are:

$$\alpha_1^+ = \frac{1}{2} \ln \left(\frac{1 + 23.04}{1 + 16.64} \right) = 0.15 \tag{28}$$

and

$$\alpha_1^- = \frac{1}{2} \ln \left(\frac{1 + 23.04}{1 + 37.44} \right) = -0.23. \tag{29}$$

And similar to Adaboost, we obtain a classification error of 40% for the strong classifier. Finally, the updated weights are presented in fig. 9.

2nd Iteration. In the second iterative step, we must evaluate the classifier candidates in combination with three preconditions: h_0, $h_1^+ = h_1$ and $h_1^- = \neg h_1$. The summed weights of the samples which do not fulfill a precondition are, respectively,

$$\begin{array}{l} W_*(\neg h_0) = 0 \\ W_*(\neg h_1^+) = 58.92 \\ W_*(\neg h_1^-) = 39.28. \end{array} \tag{30}$$

Now, we obtain for all possible classifier candidates and each precondition the following Z-value:

$$\begin{array}{rl} h_0 \wedge c_2: & Z_{02} = 92.80 \\ h_0 \wedge c_3: & Z_{03} = 97.34 \\ h_0 \wedge c_4: & Z_{04} = 97.20 \\ h_0 \wedge c_5: & Z_{05} = 98.20 \\ h_1^- \wedge c_2: & Z_{22} = 78.56 \\ h_1^- \wedge c_3: & Z_{23} = 91.96 \\ h_1^- \wedge c_4: & Z_{24} = 73.30 \\ h_1^- \wedge c_5: & Z_{25} = 98.19 \\ h_1^+ \wedge c_2: & Z_{12} = 98.20 \\ h_1^+ \wedge c_3: & Z_{13} = 58.92 \\ h_1^+ \wedge c_4: & Z_{14} = 81.60 \\ h_1^+ \wedge c_5: & Z_{15} = 98.20 \end{array} \tag{31}$$

The minimal Z is obtained when we combine the third classifier candidate with the precondition h_1 to the secondly chosen weak classifier $h_2 = h_1^+ \wedge \neg\, c_3$. Its predictive values are

$$\alpha_2^+ = \frac{1}{2} \ln \left(\frac{1 + 19.92}{1 + 0} \right) = 1.52 \tag{32}$$

and

$$\alpha_2^- = \frac{1}{2}\ln\left(\frac{1+0}{1+19.36}\right) = -1.51. \tag{33}$$

At this stage, ADTboost performs with a total classification error of only 24%, which is a value Adaboost never reaches in our scenario. The updated weights are shown in fig. 10.

0.83	0.83	0.83
0.83	0.83	1.21
1.21	0.83	1.21

Figure 9: Weights after 1st iteration.

0.18	0.83	0.83
0.18	0.83	1.21
0.27	0.83	1.21

Figure 10: Weights after 2nd iteration.

3rd Iteration. In the third iteration, the search will include five possible preconditions, and the two added preconditions are $h_2^+ = h_2$ and $h_2^- = \neg h_2$. Their partial input to the according Z-values is

$$\begin{aligned} &W_*(\neg h_0) = 0 \\ &W_*(\neg h_1^+) = 58.92 \\ &W_*(\neg h_1^-) = 8.64 \\ &W_*(\neg h_2^+) = 63.24 \\ &W_*(\neg h_2^-) = 63.24. \end{aligned} \tag{34}$$

When searching for the combination of precondition and classifier candidate which minimize Z, we start with the precondition with the lowest impact on Z. If a combination produces a Z-value which is lower than the impact of other preconditions, we may stop without considering the left preconditions. Here, we obtain the following intermediate results:

$$\begin{aligned} h_0 \wedge c_2: &\quad Z_{02} = 58.30 \\ h_0 \wedge c_3: &\quad Z_{03} = 65.21 \\ h_0 \wedge c_4: &\quad Z_{04} = 56.86 \\ h_0 \wedge c_5: &\quad Z_{05} = 67.55 \\ h_1^- \wedge c_2: &\quad Z_{22} = 47.92 \\ h_1^- \wedge c_3: &\quad Z_{23} = 61.32 \\ h_1^- \wedge c_4: &\quad Z_{24} = 42.65 \\ h_1^- \wedge c_5: &\quad Z_{25} = 67.55. \end{aligned} \tag{35}$$

The current best combination of $\neg h_1$ and c_4 is also the globally best combination, because its separability-value is less than the impacts of the other

preconditions. Thus, the third best weak classifier consists of $h_3 = h_1^- \wedge \neg c_4$ and its predictive values are

$$\alpha_3^+ = \frac{1}{2} \ln \left(\frac{1 + 29.04}{1 + 9.96} \right) = 0.50 \tag{36}$$

and

$$\alpha_3^- = \frac{1}{2} \ln \left(\frac{1 + 0}{1 + 19.92} \right) = -1.52. \tag{37}$$

At this stage of the process, we have a total classification error of only 12%. The updated weights are shown in fig. 11.

4th Iteration. Now, we have to consider seven possible preconditions, and the two new ones are $h_3^+ = h_3$ and $h_3^- = \neg h_3$. Their partial input to the according Z-values is

$$\begin{array}{l} W_*(\neg h_0) = 0 \\ W_*(\neg h_1^+) = 38.28 \\ W_*(\neg h_1^-) = 8.64 \\ W_*(\neg h_2^+) = 42.60 \\ W_*(\neg h_2^-) = 42.60 \\ W_*(\neg h_3^+) = 12.96 \\ W_*(\neg h_3^-) = 42.60. \end{array} \tag{38}$$

Again, we only have to consider those preconditions which have less impact on Z than the currently best combination. The Z-values for the combinations are:

$$\begin{array}{rl} h_0 \wedge c_2: & Z_{02} = 33.79 \\ h_0 \wedge c_3: & Z_{03} = 46.68 \\ h_0 \wedge c_4: & Z_{04} = 46.73 \\ h_0 \wedge c_5: & Z_{05} = 46.81 \\ h_1^- \wedge c_2: & Z_{22} = 22.85 \\ h_1^- \wedge c_3: & Z_{23} = 46.40 \\ h_1^- \wedge c_5: & Z_{25} = 46.78 \\ h_3^+ \wedge c_2: & Z_{52} = 12.96. \end{array} \tag{39}$$

The last combination is the globally best one, thus we set the fourth best weak classifier to be $h_4 = h_3^+ \wedge c_2$ and its predictive values are

$$\alpha_4^+ = \frac{1}{2} \ln \left(\frac{1 + 17.52}{1 + 0} \right) = 1.46 \tag{40}$$

and

$$\alpha_4^- = \frac{1}{2} \ln \left(\frac{1 + 0}{1 + 16.44} \right) = -1.43. \tag{41}$$

The weighted predictions of all weak classifiers are presented in fig. 12. We derive classification results of the strong classifier with no error at all, since the signs of all predictions are correct. The visualized alternating decision tree is shown in fig. 13.

0.18	0.18	0.18
0.18	1.37	0.73
0.27	1.37	0.73

Figure 11: Weights after 3rd iteration.

1.63	-1.79	-1.79
1.63	-1.20	1.69
-1.40	-1.20	1.69

Figure 12: Predictions of weighted weak classifiers.

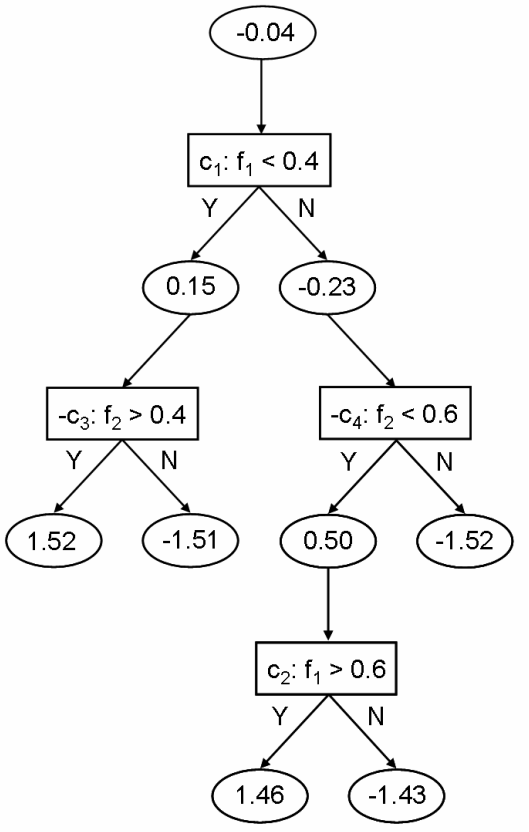

Figure 13: Visualization of Alternating Decision Tree on synthetic data set.

3.3.3 Comparison of Adaboost and ADTboost

Although we have demonstrated the functionality of Adaboost and ADTboost on a very simple synthetic data set of two non-overlapping classes,

we are able to assert one major difference between both approaches. We designed our weak classifiers such that each classification within Adaboost is a interpretable as a division of the $2D$ feature space into two half spaces. Alternatively, the hierarchical order of these weak classifiers builds a multiple k-d-tree, see fig. 14. Multiple, because we could have several weak classifiers with the same preconditions. In logical terms, the hierarchical compositions of classifiers describe `AND` relations. If several classifiers have the same precondition, they are composed by a `OR` relation.

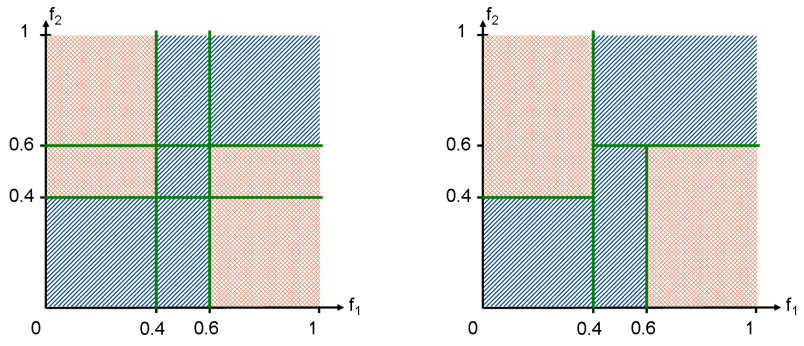

Figure 14: Feature space partitioning by weak classifiers (left: Adaboost, right: k-d-tree of ADTboost).

ADTboost favors to expand the alternating decision tree in depth. Therefore, we assume ADTboost to select less weak classifiers than Adaboost, if the error bounds of both approaches have to be fallen below a certain threshold. Then, we also might have fewer features involved in the classification progress. Unfortunately, this behavior of ADTboost also leads to overfitting. For avoiding such overfitting, one could either restrict the depth of the alternating decision tree, or the splitting must be done in respect to significant large subsets. So far, we have not implemented such restrictions.

4 Feature Subset Selection with Boosting

The goal of our work was to classify samples and to select the subset of their most relevant features. In worst case, the boosting techniques have a similar performance as random forests, i. e. we obtain a long list of weak classifiers and from a certain stage on, all features are involved in the classification process. In this section, we discuss a greedy strategy to select the most appropriate features, if (nearly) all features are used by the weak classifiers.

4.1 Two Greedy Strategies for Adaboost

In our point of view, we have two ways to derive these most appropriate features:

1. In the first variant, we would perform the selection process when training the weak classifiers. This will lead to a constricted training, where we may learn the weak classifiers on all features until the number of features used by the weak classifiers reaches the limit of appropriate features we have set. Then we only may precede further choosing weak classifiers on the features that have been selected so far.

2. In the second variant, we would select the features from the strong classifier after the training by removing the most ineffective weak classifiers again. This strategy is adequate to pruning the Adaboost tree.

First, we want to study the effects of the first proposal, where we consider the synthetic data from the preview section again. The first two chosen weak classifiers use f_1, the last two use f_2. When selecting a subset containing only one feature, this strategy will lead to the selection of the first feature. If the order of the weak classifiers turns around, this strategy leads to the selection f_2. Should the chronology of the weak classifiers be relevant for selecting the feature subset, or do the predictive values tell more about the significance of a weak classifier?

Equ. 9 describes the impact of each weak classifier on the strong classifier. The chronology of the weak classifiers does not have any influence on the strong classifier, but their predictive values α_t do. If these α_t for each weak classifier would decrease monotonously, the influence of further weak classifiers would also decrease, and then this strategy would be very sensible. But there is no proof that the predictive values fulfill such order. Thus, it seems to be more sensible to evaluate the weak classifiers when all α_t are known.

This condition is fulfilled in the second proposal, where we evaluate the weak classifiers and their associated features after the training of the strong classifier has finished. This will lead to the ranking of weak classifiers and their features, respectively: The impact of each weak classifier h_t depends on the absolute value of α_t. Then, the impact of a feature on the classification result can be measured by its contributive value $\mathcal{C}$ as

$$\mathcal{C}(f_i) = \sum_t |\alpha_t| \text{ where } h_t \text{ works on } f_i. \tag{42}$$

Considering the synthetic data set example, we have four weak classifiers with the predictive values:

$$\alpha_1 = 0.2027,\ \alpha_2 = 0.3406,\ \alpha_3 = 0.1499,\ \alpha_4 = 0.1729. \quad (43)$$

This leads two the two contributive values:

$$\begin{array}{l} \mathcal{C}(f_1) = |\alpha_1| + |\alpha_2| = 0.5433 \\ \mathcal{C}(f_2) = |\alpha_3| + |\alpha_4| = 0.3228. \end{array} \quad (44)$$

Both strategies for feature selection lead to the same result in respect to the synthetic data set, but the second method always and definitely selects the most relevant features concerning the impact on the classification with Adaboost. In the next subsection, we will adapt these strategies on ADTboost.

4.2 Adapted Greedy Strategies for ADTboost

ADTboost produces a partial order of the weak classifiers due to their hierarchy. This order has to be taken into account when selecting the most relevant classifiers and most relevant features. Obviously, the first variant as introduced in the previous subsection does not have to get modified. And again, this selection scheme has influence on the selection of the weak classifiers without evaluating their relevance.

The second feature selection scheme rates the features by their contributive values which does not interferes the selection of further weak classifiers. When extending this approach for ADTboost, we should adapt equ. 42, since we determine two predictive values:

$$\mathcal{C}(f_i) = \sum_t |\alpha_t^+| + |\alpha_t^-| \text{ where } h_t \text{ works on } f_i. \quad (45)$$

Checking the effect of this adaptation in respect to the synthetic data set, we determine the contributive values

$$\begin{array}{rcl} \mathcal{C}(f_1) & = & |\alpha_1^+| + |\alpha_1^-| + |\alpha_4^+| + |\alpha_4^-| \\ & = & 0.15 + 0.23 + 1.46 + 1.43 = 3.27 \\ \mathcal{C}(f_2) & = & |\alpha_2^+| + |\alpha_2^-| + |\alpha_3^+| + |\alpha_3^-| \\ & = & 1.52 + 1.51 + 0.50 + 1.52 = 5.05. \end{array} \quad (46)$$

The weak classifiers have high absolute values as predictive values, if their classification performance is excellent. While the first weak classifier has to distinguish from the whole domain, and therefore, has a bad performance, the other weak classifiers only work on a portion of the domain, and therefore,

have better classification results. This disadvantage of the more general classifiers can be overcome, if we weight the predictive values of each weak classifier by the portion of its domain ωt. Then we gain:

$$\begin{aligned}
\mathcal{C}(f_1) &= \omega_1 \cdot \left(|\alpha_1^+| + |\alpha_1^-|\right) + \omega_4 \cdot \left(|\alpha_4^+| + |\alpha_4^-|\right) \\
&= 1.00 \cdot (0.15 + 0.23) + 0.24 \cdot (1.46 + 1.43) \\
&= 0.38 + 0.6936 = 1.0736 \\
\mathcal{C}(f_2) &= \omega_2 \cdot \left(|\alpha_2^+| + |\alpha_2^-|\right) + \omega_3 \cdot \left(|\alpha_3^+| + |\alpha_3^-|\right) \\
&= 0.40 \cdot (1.52 + 1.51) + 0.60 \cdot (0.50 + 1.52) \\
&= 1.212 + 1.212 = 2.424.
\end{aligned} \tag{47}$$

Furthermore, we should integrate the hierarchical structure of the weak classifiers when ranking the influence of the classifiers and the used features, respectively. For ADTboost, a weak classifier consists of two parts, the precondition and the classifier. So, the preconditions should also take benefit from further good classifications. Then, the contributive values should also consist of the predictive values where the feature is taken for getting a precondition:

$$\mathcal{C}(f_i) = \sum_t \omega_t \cdot \left(|\alpha_t^+| + |\alpha_t^-|\right) + \sum_p \omega_p \cdot \left(|\alpha_p^+| + |\alpha_p^-|\right), \tag{48}$$

where h_t is a classifier that uses f_i and h_p is precondition for classifiers that use feature f_i. Regarding the synthetic data set, we obtain the following contributive values

$$\begin{aligned}
\mathcal{C}(f_1) &= 1.00 \cdot (0.15 + 0.23) + 0.24 \cdot (1.46 + 1.43) + \\
&\quad 0.40 \cdot (1.52 + 1.51) + 0.60 \cdot (0.50 + 1.52) + \\
&\quad 0.24 \cdot (1.46 + 1.43) \\
&= 0.38 + 0.6936 + 1.212 + 1.212 + 0.6936 = 4.1912 \\
\mathcal{C}(f_2) &= 0.40 \cdot (1.52 + 1.51) + 0.60 \cdot (0.50 + 1.52) + \\
&\quad 0.24 \cdot (1.46 + 1.43) \\
&= 1.212 + 1.212 + 0.6936 = 3.1176.
\end{aligned} \tag{49}$$

The contributive value of feature f_1 is a sum of five components. The first two summands are the weighted predictive values that belong to the classifiers h_1 and h_4, respectively. The other three summands are added, because h_1 is a precondition of the three classifiers h_2, h_3 and h_4.

In this section, we discussed the feature subset selection strategies using Adaboost and ADTboost. This subset selection leads to a pruning of the original classifiers. A comparison of the strategies between Adaboost and ADTboost is given in [Drauschke & Förstner, 2008] and an application on detecting buildings and building parts is presented in [Drauschke & Förstner, 2008 (to appear)].

References

[Bishop, 2006] Bishop, Chr. M. 2006. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer.

[Breiman, 2001] Breiman, L. 2001. Random Forests. *Machine Learning*, **45**, 5–32.

[De Comité *et al.*, 2001] De Comité, F., Gilleron, R., & Tommasi, M. 2001. Learning Multi-label Alternating Decision Trees and Applications. *Pages 195–210 of: Proc. CAP 2001.*

[Drauschke & Förstner, 2008] Drauschke, M., & Förstner, W. 2008. Comparison of Adaboost and ADTboost for Feature Subset Selection. *In: Proc. 8th PRIS 2008.*

[Drauschke & Förstner, 2008 (to appear)] Drauschke, M., & Förstner, W. 2008 (to appear). Selecting Appropriate Features for Detecting Buildings and Building Parts. *In: 21st ISPRS Congress.*

[Freund & Mason, 1999] Freund, Y., & Mason, L. 1999. The Alternating Decision Tree Learning Algorithm. *Pages 124–133 of: Proc. 16th ICML.*

[Freund & Schapire, 1996] Freund, Y., & Schapire, R. E. 1996. Experiments with a new boosting algorithm. *Pages 148–156 of: Proc. 13th ICML.*

[Guyon & Elisseeff, 2003] Guyon, I., & Elisseeff, A. 2003. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, **3**, 1157–1182.

[Ho, 1998] Ho, T. K. 1998. The Random Subspace Method for Constructing Decision Forests. *PAMI*, **20**(8), 832–844.

[Holmes *et al.*, 2002] Holmes, G., Pfahringer, B., Kirkby, R., Frank, E., & Hall, M. 2002. Multiclass Alternating Decision Trees. *Pages 161–172 of: Proc. 13th ECML.* LNCS 2430. Springer.

[Liu & Motoda, 1998] Liu, H., & Motoda, H. 1998. *Feature Selection for Knowledge Discovery and Data Mining.* Kluwer Academic.

[Martin-Bautista & Vila, 1999] Martin-Bautista, M. J., & Vila, M.-A. 1999 (July). A survey of genetic feature selection in mining issues. *Pages 1314–1321 of: Proc. CEC 1999*, vol. 2.

[Mladenić, 2006] Mladenić, D. 2006. Feature Selection for Dimensionality Reduction. *Pages 84–102 of: SLSFS*. LNCS 3940.

[Pfahringer *et al.*, 2001] Pfahringer, B., Holmes, G., & Kirkby, R. 2001. Optimizing the Induction of Alternating Decision Trees. *Pages 477–487 of: 5th PACKDDM 2001*. LNCS 2035.

[Rätsch *et al.*, 2001] Rätsch, G., Onoda, T., & Müller, K.-R. 2001. Soft margins for AdaBoost. *Machine Learning*, **43**(3), 287–320.

[Rogers & Gunn, 2006] Rogers, J., & Gunn, St. 2006. Identifying Feature Relevance Using a Random Forest. *Pages 173–184 of: SLSFS*. LNCS 3940.

[Schapire, 1990] Schapire, R. E. 1990. The strength of weak learnability. *Machine Learning*, **5**(2), 197–227.

[Schapire & Singer, 1999] Schapire, R. E., & Singer, Y. 1999. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, **37**(3), 297–336.