Multi-class ADTboost

Martin Drauschke

`martin.drauschke@uni-bonn.de`

TR-IGG-P-2008-06

8th August 2008

Technical Report Nr. 6, 2008

Department of Photogrammetry
Institute of Geodesy and Geoinformation
University of Bonn

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

Multi-class ADTboost

Martin Drauschke
martin.drauschke@uni-bonn.de

Abstract

This technical report gives a short review on boosting with alternating decision trees (ADTboost), which has been proposed by Freund & Mason (1999) and refined by De Comité *et al.* (2001). This approach is designed for two-class problems, and we extend it towards multi-class classification. The advantage of a multi-class boosting algorithm is its usage in scene interpretation with various kinds of objects. In these cases, two-class approaches will lead to several one class versus background (the other classes) classifications, where we must solve unappropriate results like "always background" or "two or more valid classes" for a sample.

1 Introduction

Adaptive boosting (Adaboost) is a very successful classification framework for two-class problems, where a strong classifier H is built by combining several weak classifiers h_t. The first weak classifier h_1 is trained on equally weighted samples. Then, the second weak classifier h_2 focuses on the previously misclassified samples, i. e. the the weights of the samples have been adjusted with respect of the classification success of the previous weak classifier. This process continues until T weak classifiers h_t have been trained. We obtain the final strong classifier H by a majority vote of all weak classifiers.

The original Adaboost framework has been extended in several approaches and for various applications. Freund & Mason (1999) proposed a boosting scheme with alternating decision trees (ADTboost), which has been refined by De Comité *et al.* (2001). Fig. 1 shows an alternating decision tree with four weak classifiers. The tree contains two different kind of nodes, *decision nodes* and *prediction nodes*. First represent the decisions of the weak classifiers h_t and the second are used for the predictive values α_t. Both kind of nodes alternate on all paths from the root to a leaf.

ADTboost has three major advantages over Adaboost:

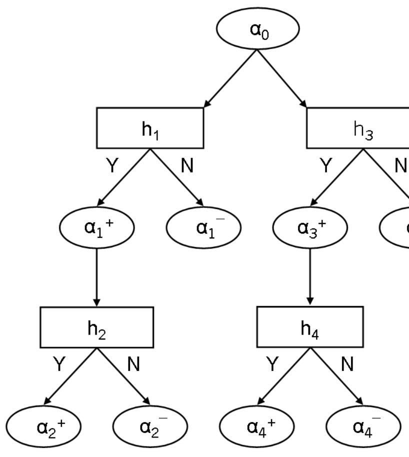

Figure 1: Alternating Decision Tree: Rectangular nodes represent the weak classifiers, elliptic nodes the predictive values.

1. The weak classifiers h_t can be aranged in a hierarchical order - the *alternating decision tree*. The weak classifiers near the root are used to define preconditions for weak classifiers underneath them. In fig. 1 the weak classifiers h_1 and h_3 define preconditions for h_2 and h_4, respectively.

2. There are two predictive values α_t^+ and α_t^-. Thus, the (binary) decisions of a weak classifier h_t can be ranked depending on the direction of the decision instead of finding one criterion for both directions.

3. With the root node, we model an additional predictive value α_0, which is derived from the ratio of the number of samples between both classes and, therefore, it can be interpreted as a a prior classifier.

Notation. Before discussing the specific algorithms in more detail, we want to formalize the problem and introduce our the most important entities of our notation.

Given are N samples (x_n, y_n), where x is a real-valued feature vector $x = [f_1, .., f_D]$ of dimension D, and y encodes the class membership. Regarding the two-class problems $y \in \{-1, +1\}$, and regarding the multi-class case $y \in \{1, .., K\}$ with K given classes.

For the boosting algorithms, we will use the following entities:

- $\mathbb{R}^{\mathrm{D}}$ is the domain of x and $\mathbb{K}$ is the domain of y,
- t is the index of the iteration, $t = 1 \ldots T$,

- w_t^n is the weight of n-th sample in t-th iteration,
- $h_t : \mathrm{I\!R}^{\mathrm{D}} \rightarrow \mathrm{I\!K}, \boldsymbol{x} \mapsto \mathrm{h_t}(\boldsymbol{x})$ is the best weak classifier that has been selected from the set of classifier candidates c_j, $j = 1 \ldots J_t$,
- h_t^+ and h_t^- are used for describing predicates which are based on the weak classifiers: h_t^+ means $h_t(\boldsymbol{x}) = +1$ and h_t^- means $h_t(\boldsymbol{x}) = -1$,
- α_t is the predictive value of h_t
- H is the strong classifier, $H : \mathrm{I\!R}^{\mathrm{D}} \rightarrow \mathrm{I\!K}, \boldsymbol{x} \mapsto \mathrm{H}(\boldsymbol{x})$

Structure of the Technical Report. In this technical report, we first give document the Adaboost and ADTboost algorithms for the two-class problem. Then, we show, how the Adaboost algorithm has been modified for multi-class problems by Zhu *et al.* (2006). Afterwards, we present our strategy for an analogous modification of ADTboost. Finally, we demonstrate our approach on a small example.

2 Two-class Adaboost and ADTboost

In this section we introduce the notion of Adaboost and ADTboost algorithms for classifiaction of two-class problems and discuss their major steps. Therefore, we skip the literary review on the early concept of boosting as proposed by Schapire (1990) and the first proposal for Adaboost by Freund & Schapire (1996) and present the improved version of Adaboost as proposed by Schapire & Singer (1999). Regarding ADTboost, we refer to the publications of Freund & Mason (1999) and De Comité *et al.* (2001).

2.1 Adaboost

The principle Adaboost algorithm is consists of a initialization step and an iterative process, where the T weak classifiers h_t are determined. The major components of the algorithm are sketched in alg. 1.

More detailed, in each iteration we have set of J_t classifier candidates c_j. Based on the actual weights of the samples w_t, we select the best classifier candidate as the t-th weak classifier h_t. Therefore, Schapire & Singer

Algorithm 1 Adaboost algorithm, cf. (Schapire & Singer, 1999).

```
1: function ADABOOST($T, (\boldsymbol{x}_1, y_1) \dots (\boldsymbol{x}_N, y_N)$)
2:     $w_1^n = \frac{1}{N}$
3:     for $t = 1,\dots, T$ do
4:         Determine best weak hypothesis $h_t$ using $w_t$
5:         Determine $\alpha_t$
6:         Determine distribution $w_{t+1}$
7:     end for
8:     return $H$ with $H(\boldsymbol{x}) = \text{sign}\left(\sum_t \alpha_t h_t(\boldsymbol{x})\right)$.
9: end function
```

(1999) showed that the best criteria is discriminative power of the classifier candidates c_j, if it is defined by

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

with

$$W_+(c_j^+) = \sum_n w_t^n \text{ where } c_j(\boldsymbol{x}_n) = y_n = +1 \tag{2}$$

is the sum of weights of the *true positive* samples and

$$W_-(c_j^+) = \sum_n w_t^n \text{ where } c_j(\boldsymbol{x}_n) = +1 \neq -1 = y_n \tag{3}$$

is the sum of weights of the *false positive* samples. Then, the best weak classier is that candidate which minimizes Z:

$$h_t = c_t \text{ with } t = \text{argmin}_j Z_j. \tag{4}$$

When building the strong classifier H, is classification result is obtained by a weighted sum of the results of the weak classifiers, and the weights are the predictive values α_t. We determine one predictive value α_t for each weak classifier h_t. According to Schapire & Singer (1999), the optimal predictive value α_t is derived from the classification result r_t of the weak classifier h_t and, therefore, it depends on the classifier's success rate:

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1 + r_t + \epsilon}{1 - r_t + \epsilon}\right) \tag{5}$$

with

$$r_t = \sum_{n=1}^{N} w_t^n y_n h_t(\boldsymbol{x}_n). \tag{6}$$

The product $y_n h_t(\boldsymbol{x}_n)$ is $+1$, if $\boldsymbol{x}_n$ is correctly classified by h_t, otherwise the product is -1. If the error rate is 0.5 or higher, what is equivalent to $r_t \leq 0$, then Adaboost stops. So it is guaranteed that the predictive value α_t is always positive. Updating the weights, Schapire & Singer (1999) propose:

$$w_{t+1}^n = w_t^n \exp\left(-\alpha_t y_n h_t(\boldsymbol{x}_n)\right), \tag{7}$$

and afterwards w_{t+1} is normalized again, so that w_t forms a discrete distibution.

The nature of the considered weak classifiers is not further specified, and might be chosen with respect to the distribution of the samples. Furthermore, it is still an unanswered question, how to set the number T of weak classifiers, that should get combined to build the strong classifier H.

2.2 ADTboost

The principle boosting algorithm with alternating decision trees, ADTboost, is sketched in alg. 2. It is similar to the proposal by Freund & Mason (1999).

Algorithm 2 Principle ADTboost algorithm.

```
 1: function ADTBOOST($T, (\boldsymbol{x}_1, y_1) \dots (\boldsymbol{x}_N, y_N)$)
 2:     Initialize set of preconditions $\mathcal{P} = \{\mathbf{T}\}$
 3:     Determine $\alpha_0$
 4:     $w_1^n = \frac{1}{N}$
 5:     for $t = 1, \dots, T$ do
 6:         Determine best weak hypothesis $h_t : \mathbb{R}^D \rightarrow \{-1, 0, +1\}$ using $w_t$
 7:         Determine the predictive values $\alpha_t^+$ and $\alpha_t^-$
 8:         Update set of preconditions $\mathcal{P}$
 9:         Determine $w_{t+1}$
10:     end for
11:     return $H$ with $H(\boldsymbol{x}) = \text{sign}\left(\sum_t \alpha_t h_t(\boldsymbol{x})\right)$.
12: end function
```

Each weak classifier h_t consists of precondition p_t and a classifier c_t. Therefore, we decompose $h_t = p_t \wedge c_t$ and redefine the weak classifiers:

$$h_t(\boldsymbol{x}_n) = \left\{ \begin{array}{rl} +1 & \text{if } p_t(\boldsymbol{x}_n) = \textbf{true} \text{ and } c_t(\boldsymbol{x}_n) = +1 \\ 0 & \text{if } p_t(\boldsymbol{x}_n) = \textbf{false} \\ -1 & \text{if } p_t(\boldsymbol{x}_n) = \textbf{true} \text{ and } c_t(\boldsymbol{x}_n) = -1 \end{array} \right\} \tag{8}$$

The preconditions are used to model the hierarchical structure of the alternating decision tree. Therefore, we define a set of preconditions $\mathcal{P}$ which initially contains the **T**-condition which is always *true*. If a weak classifier contains the **T**-condition as its precondition, the decision node with the weak classifier is a direct child of the root. In each iteration, where the weak classifier h_t has been selected, the update of the set of preconditions $\mathcal{P}$ is done by

$$\mathcal{P}_{t+1} = P_t \cup \left\{h_t^+\right\} \cup \left\{h_t^-\right\}. \tag{9}$$

For selecting a weak classifier h_t in the t-th iteration, we need to search for the best one over all preconditions $p_i \in \mathcal{P}, i = 1 \ldots |\mathcal{P}| = 2t-1$, and over all classifier candidates $c_j, j = 1 \ldots J_t$. Freund & Mason (1999) propose to select that classifier candidate c_j under precondition p_i which minimizes the function

$$\begin{aligned} Z_{i,j} \;=\; & 2\sqrt{W_+(p_i \wedge c_j^+) W_-(p_i \wedge c_j^+)} + \\ & 2\sqrt{W_+(p_i \wedge c_j^-) W_-(p_i \wedge c_j^-)} + W(\neg p_i) \end{aligned} \tag{10}$$

where W_+ are the summed weights of the *true positive* samples etc. and $W(\neg p_i)$ is the sum of weights of all samples which do not fulfill p_i. Then the weak classifier h_t is defined by

$$h_t = p_{t_1} \wedge c_{t_2} \text{ with } (t_1, t_2) = \text{argmin}_{i,j} Z_{i,j}. \tag{11}$$

Besides the hierarchy of the weak classifiers, the usage of two predictive values is another advantage of ADTboost. Both decisions of the weak classifier h_t^+ and h_t^- are evaluated separately. Then we obtain

$$\alpha_t^+ = \frac{1}{2} \log \frac{W_+(p_{t_1} \wedge c_{t_2})}{W_-(p_{t_1} \wedge c_{t_2})} \text{and} \tag{12}$$

$$\alpha_t^- = \frac{1}{2} \log \frac{W_+(p_{t_1} \wedge \neg c_{t_2})}{W_-(p_{t_1} \wedge \neg c_{t_2})}. \tag{13}$$

The update of the weights at the end of each iteration is done analogously to Adaboost. If the sample does not fulfill the precondition of the weak classifier, so $h_t(x_n) = 0$, then its weight w_{t+1}^n remains unchanged, otherwise it gets in- or decreased, depending on the determined predictive value and the sample's target.

2.3 A small example

In (Drauschke, 2008), we studied the steps of the Adaboost and ADTboost algorithms, respectively. Therefore, we generated a synthetic data set with two features, and chose some threshold classifier candidates for building the strong classifiers with both methods. After four iterations, ADTboost returned a strong classifier with 0% classification error, while the strong Adaboost classifier did not fall below 32% classification error. Due to the choice of the weak classifiers, the resulting alternating decision tree of ADTboost can be interpretated as a $k - 2$-tree. We visualized these results in fig. 2.

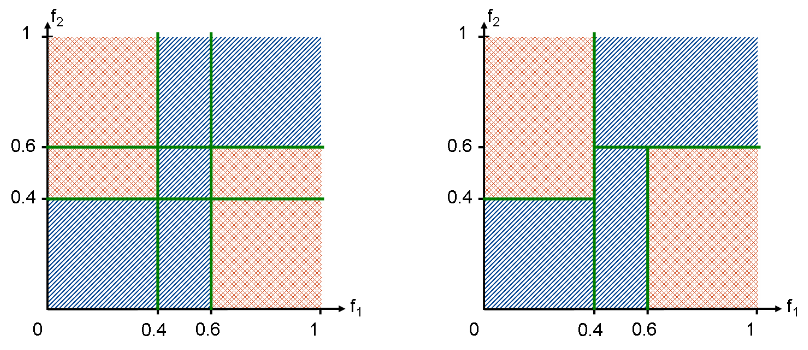

Figure 2: Feature space partitioning by weak classifiers (left: Adaboost, right: k-d-tree of ADTboost).

3 Multi-class Adaboost

In the previous section, we introduced the Adaboost algorithm as designed for binary classification problems. In practise, there are many problems with several classes, so there is an demand to extend Adaboost to multi-class problems. Schapire & Singer (1999) proposed to solve the problem by reducing it to a binary problem again.

Therefore, the number of samples is enlarged in the following way. If there are given N different samples with targets in K different classes with labels $k = 1 \ldots K$, there are constructed $N * K$ many new samples (x', y') with the feature vectors

$$\boldsymbol{x}'_{nk} = [\boldsymbol{x}_n, k] \tag{14}$$

and targets

$$y'_{nk} = \left\{ \begin{array}{ll} +1 & \text{if } y_n = k \\ -1 & \text{if } y_n \neq k \end{array} \right\} \tag{15}$$

This approach leads to explosion of number of samples. If you have a data set with $10,000$ samples from 100 classes, then you will have to deal with 1 million newly constructed samples. Furthermore, the following results

$$H([\boldsymbol{x}_n, k]) = -1 \forall k$$

and

$$H([\boldsymbol{x}_n, k_1]) = +1 = H([\boldsymbol{x}_n, k_2])$$

lead to discussions about the usability of the approach.

In this section we will present the multi-class Adaboost algorithm of Zhu *et al.* (2006), and then we discuss and interprete its procedures.

3.1 Zhu's multi-class Adaboost algorithm

In a technical report, Zhu *et al.* (2006) proposed a multi-class Adaboost algorithm, which is presented in alg. 3.

Algorithm 3 Multi-class Adaboost algorithm, cf. (Zhu *et al.*, 2006).

```
1: function MultiClassAdaboost($T, (\boldsymbol{x}_1, y_1) \dots (\boldsymbol{x}_N, y_N)$)
2:     $w_1^n = \frac{1}{N}$
3:     for $t = 1, \dots, T$ do
4:         Determine best weak hypothesis $h_t$ using $w_t$
5:         Determine error $r_t$
6:         Determine $\alpha_t$
7:         Determine distribution $w_{t+1}$
8:     end for
9:     return $H$ with $H(\boldsymbol{x}) = \arg\max_k \sum_t \alpha_t \mathbb{I}\left(\mathrm{h_t}(\boldsymbol{x}) = \mathrm{k}\right)$.
10: end function
```

Zhu *et al.* (2006) demand to obtain only positive predictive values α_t, what is, from their point of view, the major difference to the original algorithm for two-class problems. Therefore, they re-define the error rate r_t by

$$r_t = \sum_n w_t^n \mathbb{I}\left(\mathrm{y_n} \neq \mathrm{h_t(x_n)}\right) / \sum_{\mathrm{n}} \mathrm{w_t^n} \tag{16}$$

determine the predictive value α_t by

$$\alpha_t = \log \frac{1 - r_t}{r_t} + \log(K - 1). \tag{17}$$

Furthermore, each weak classifier h_t is only accepted, if it reaches an error rate that is below $(K-1)/K$.

The update of the weights is done by

$$w_{t+1}^n = w_t^n \mathbb{I}\left(\mathrm{y_n} \neq \mathrm{h_t(x_n)}\right). \tag{18}$$

In the two-class algorithm by Schapire & Singer (1999), the weights have always been normalized with the constraint $\sum_n w_t^n = 1 \forall t$, so that they form a discrete distribution. Here, Zhu *et al.* (2006) have relaxed this constraint and do not perform a normalization of the weights afterwards. Therefore, they need a normalization term in equ. 16. In our point of view, there is no argument, why the weight should not form a discrete distribution, and since our weak classifier (as presented in section 5.1) will use this normalization, we propose to normalize the weights additionally.

3.2 Discussion of the Zhu's proposal

In this part, we want to discuss some of the steps of the multi-class Adaboost algorithm by Zhu *et al.* (2006).

3.2.1 Strong classifier H

The first point is the construction of the strong classifier H. Its definition in the two-class variant of Adaboost by Schapire & Singer (1999)

$$H(\boldsymbol{x}) = \text{sign}\left(\sum_t \alpha_t h_t(\boldsymbol{x})\right) \tag{19}$$

has been consequently generalized by Zhu *et al.* (2006). The sign-function is limited to the binary case, where the one class is encodes by $+1$ and the other by -1. Grouping the α_t with respect to their classification result, we obtain a reformulation for equ. 19 with

$$H(\boldsymbol{x}) = \arg\max_k \sum_t \alpha_t \mathbb{I}\left(\mathrm{h_t}(\boldsymbol{x}) = \mathrm{k}\right), \mathrm{k} = \{-1, +1\}, \tag{20}$$

where $\mathbb{I}$ is the indicator function which returns 1, if its argument is *true*, and 0, if its argument is *false*. Now, the generalization towards multi-class classification is very simple, because we only have to extend the domain of k.

3.2.2 Selecting the weak classifier h_t

Zhu *et al.* (2006) do not mention anything about how they select the best weak classifier. They only document that they "fit a classifier [...] to the training data using" w_t^n. So, we assume that they have a deterministic procedure that delivers exactly one possible classifier candidate c_j.

Thereby, the generalization of the Z-function is not so difficult. Equ. 1 contains a sum of two square roots, and each root-term can be interpretated as geometric mean. The first one is the geometric mean of the *true* and *false* *positives*, and the second one of the *true* and *false negatives*. We define the generalized term by

$$\sqrt{W_k(c_j^k)W_{\neq k}(c_j^k)} \tag{21}$$

where

$$W_k(c_j^k) = \sum_n w_t^n \mathbb{I}\left(\mathrm{h_t}(\boldsymbol{x}) = \mathrm{k} = \mathrm{y_n}\right) \tag{22}$$

and

$$W_{\neq k}(c_j^k) = \sum_n w_t^n \mathbb{I}\left(\mathrm{h_t}(\boldsymbol{x}) = \mathrm{k} \neq \mathrm{y_n}\right). \tag{23}$$

So, we replace the classification result (*true* or *false*) by the class representative k and determine the correctly and misclassified samples with respect to k. Then, the Z-function should consider all classes, and therefore, we define it by

$$Z_j = 2\left(\sum_k \sqrt{W_k(c_j^k)W_{\neq k}(c_j^k)}\right). \tag{24}$$

3.2.3 Predictive value α_t

Zhu *et al.* (2006) demand all predictive values to be positive. If we have given a very bad classifier with $r = (K-1)/K - \epsilon$ with $\epsilon > 0$, then we obtain together with equ. 17

$$\lim_{\epsilon \to 0} \alpha = \log \frac{K-1}{K^2} + \log(K-1) = \log \frac{(K-1)^2}{K^2} < 0, \tag{25}$$

so that the predicitive value should additionally be set by

$$\alpha_t = \max\{0, \alpha_t\}. \tag{26}$$

4 Multi-class ADTboost

In the previous section, we presented the approach of Zhu *et al.* (2006) for a multi-class Adaboost algorithm, discussed it, and proposed some minor improvements. Now we want to discuss its adaptation for generalizing the two-class ADTboost algorithm.

4.1 Preliminary considerations

In section 1, we listed the three major advantages of ADTboost over Adaboost. When generalizing ADTboost towards multi-class problems, these advantages must persist. These are the hierarchical structure in an alternating decision tree, the distinction into two predictive values, and the "a priori" classification by the predictive value α_0.

4.1.1 Defining the hierarchical structure

The alternating decision tree by Freund & Mason (1999) used binary conditions to model a tree using the weak classifiers h_t. Originally, these conditions were simple predicates like $f_1 < 5$. Since each of these conditions is treated as a weak classifier, the more general formulation (including the preconditions) has been made in equ. 8. So, we can easily adapt it regarding multiple decisions:

$$h_t(\boldsymbol{x}_n) = \left\{ \begin{array}{ll} 0 & \text{if } p_t(\boldsymbol{x}_n) = \textbf{false} \\ k & \text{if } p_t(\boldsymbol{x}_n) = \textbf{true} \text{ and } c_t(\boldsymbol{x}_n) = k \end{array} \right\} \tag{27}$$

This requires that no class is represented by 0. And K different outputs of h_t besides 0 should yield in K different predictive values α_t, so that the second advantage of ADTboost remains.

4.1.2 Defining the predictive values

If we define a alternating decision tree with multiple decisions in each decision node, we need to define multiple predictive values as well. In the two-class case, cf. equ. 12, we interpretated α_t^+ as measure, how good is the classifier to recognize the positive samples in the data set. This will lead to K predictive values α_t^k which we may define by

$$\alpha_t^k = \frac{1}{2} \log \frac{W_k(h_t^k) + \epsilon}{W_{\neq k}(h_t^k) + \epsilon}, \tag{28}$$

where h_t^k is the short notation of the condition $h_t(\boldsymbol{x}) = k$ and W_k and $W_{\neq k}$ are defined as in equ. 22 and 23. This formula does not guarantee positive α_t^k, so we must adapt the determination of the predictive values once more.

$$\alpha_t^k = \frac{1}{2} \log \frac{W_k(h_t^k) + W_{\neq k}(h_t^k) + \epsilon}{W_{\neq k}(h_t^k) + \epsilon} \tag{29}$$

ensures that $\alpha_t^k > 0 \forall k$.

4.1.3 Defining $\boldsymbol{\alpha}_0$

In the two-class case, we interpreted the value α_0 in the root node of the alternating decision tree as a "a priori" classification, since it is defined, according to Freund & Mason (1999) by

$$\alpha_0 = \frac{1}{2} \log \frac{W_+(\mathbf{T})}{W_-(\mathbf{T})}. \tag{30}$$

Similar to the other predictive values $\alpha_t, t \geq 1$, α_0 should be defined as a K-dimensional vector. The probability that a sample belongs to class k will correspond the the k-th value of $\boldsymbol{\alpha}_0$, if we define it by

$$\alpha_0^k = \frac{1}{2} \log \frac{W_k(\mathbf{T}) + W_{\neq k}(\mathbf{T})}{W_{\neq k}(\mathbf{T})}. \tag{31}$$

4.1.4 Strong classifier H

$$H(\boldsymbol{x}) = \arg\max_k \boldsymbol{\alpha}_0 + \sum_t \boldsymbol{z} \tag{32}$$

where $\boldsymbol{z}$ is a K-dimensional vector with

$$\boldsymbol{z}_k = \alpha_t^k \mathbb{I}\left(\mathrm{h_t}(\boldsymbol{x}) = \mathrm{k}\right). \tag{33}$$

4.2 Proposal for a multi-class ADTboost algorithm

Now, everything is prepared, so that we can present our algorithm for a multi-class ADTboost, cf. alg. 4.

Algorithm 4 Multi-class ADTboost algorithm.

```
 1: function MultiClassADTboost($T, (\boldsymbol{x}_1, y_1) \dots (\boldsymbol{x}_N, y_N)$)
 2:     Initialize set of preconditions $\mathcal{P} = \{\mathbf{T}\}$
 3:     Determine $\boldsymbol{\alpha}_0$
 4:     $w_1^n = \frac{1}{N}$
 5:     for $t = 1, \dots, T$ do
 6:         Determine best weak hypothesis $h_t : \mathbb{R}^{\mathrm{D}} \rightarrow \mathbb{K} \cup \{0\}$ using $w_t$
 7:         Determine the predictive values $\alpha_t^k$
 8:         Update set of preconditions $\mathcal{P}$
 9:         Determine $w_{t+1}$
10:     end for
11:     return $H$ according to equ. 32.
12: end function
```

In each iteration, we select the best weak classifier by choosing the combination of precondition p_i and classifier candidate c_j that cause the minimal value for the Z-function, which is defined by

$$Z_{i,j} = 2 \left(\sum_k \sqrt{W_k(p_i \wedge c_j^k) W_{\neq k}(p_i \wedge c_j^k)} \right) + W(\neg p_i) \tag{34}$$

We adopted the factor 2 in front of the sum, because we did not see any reason why we should change the weights between both components of Z.

Since we insert K decisions with each new decision node (weak classifier h_t) in the tree, we also have to extend the set of preconditions $\mathcal{P}$ by the K new elements h_t^k. The update of the weights is taken from the multi-class Adaboost algorithm as stays unchanged as proposed in equ. 18. Again, if the precondition p_t of the weak classifier h_t is not fulfilled by $\boldsymbol{x}$, then h_t returns a 0 and the weight of the sample remains unchanged.

Last, we want to mention the updating of the weights. On the one side, we need normalized weights due to our choice of weak classifier, cf. section 5.1. On the other side, the weights of the samples that do not fulfill the chosen weak classifiers precondition keep their amount. This leads to a normalization constraint after updating the weights, where only the changed weights should be considered. We start the updating process with the condition $\sum_n w_t^n = 1$. The weak classifier h_t divides the set of samples into two subsets, the one with the samples that fulfill the classifiers precondition, and the other with the samples that do not. The weights of the samples from the first subset get changed according to equ. 18, and the weights of the others remain unchanged. Then we re-normalize the weights of the samples of the first set (the one where the weights have been changed) such, that the weights w_{t+1}^n form a discrete distribution with $\sum_n w_{t+1}^n = 1$.

5 Realization and Demonstration

In this section, we want to document how we realized ADTboost and demonstrate its performance at the same example where we have studied the two-class algorithm.

5.1 Weak classifiers

The ADTboost algorithm has already been formulated in the previous section. There is only one detail open, which we now want to present: the nature of our weak classifiers.

When studying the two-class algorithms, cf. (Drauschke & Förstner, 2008), we used simple threshold classifiers. Therefore, we determined the best possible threshold θ_d for each feature f_d, and then we searched for the best classifier as described in the previous sections. This is only a successful technique, if the data is separable binarily, otherwise it will lead to too many classification errors. Fig. 3 shows histograms of a single feature's occurrences, where the data is taken from the UCI machine learning repository, cf. (Asuncion & Newman, 2007). The example from the *ringnorm* data set shows two feature frequencies where each can get approximated well by a

Gaussian function. If we could use both intersections for the determination of class borders, then we would obtain significantly better classification results. The example from the *image* data set shows distributions which could be modeled by mixtures of Gaussian functions, where we would prefer even more thresholds.

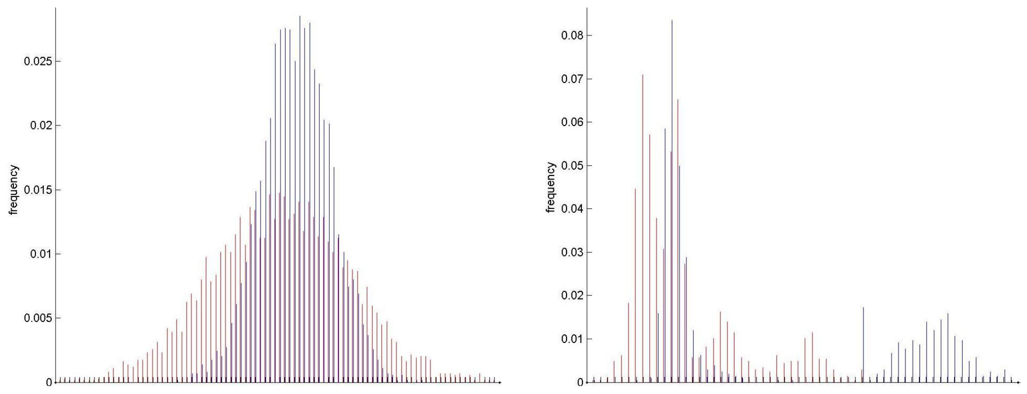

Figure 3: Histograms of a single feature with respect to two classes (red and blue), taken from the data sets *ringnorm* and *image*.

For our study of the multi-class ADTboost algorithm, we choose classifier candidates c_j which are different from the single thresholds θ_d. Now, we determine a MAP classification for each feature with respect to the samples weights. The principle of the classifier is as follows. First, we determine class-specific histograms $_k$ with respect to the samples weights, where the bins of all K histograms have the same ranges. So, we may determine the superior class of each histogram's bin. The algorithms has as inputs the number of bins B, N samples (f_n, y_n) (only a single feature per sample) and the weight of each sample w_n. Then the algorithm returns list with B class indices. The algorithm is presented in alg 5.

If we would normalize each histogram $_k$, so that $\sum_b {}_k(b) = 1$, then the histograms $_k$ represent Likelihood functions $p(f|k)$. Then we only would obtain a real MAP classification, if we determine the *a priori* probabilities $p(k)$ for each of the K classes. We obtain the Likelihood function by

$$p(f_d|k) = \frac{_k}{\sum_{y_n=k} w_n}, \tag{35}$$

and the divisor is equal to the *a priori* probability $p(k)$. We obtain

$$p(k) = \sum_{y_n=k} w_n, \text{ if } \sum_n w_n = 1. \tag{36}$$

Algorithm 5 MAP classification algorithm.

```
1: function MAPClassiciation($B, (f_{d1}, y_1) \dots (f_{dN}, y_N), (w_1 \dots w_N)$)
2:     Initialize field $F$ with $B$ elements
3:     for $k = 1, \dots, K$ do
4:         Initialize histogram $_k$ for each class with $B$ bins
5:     end for
6:     for $n = 1, \dots, N$ do
7:         Determine class index $k$ of sample $(f_{dn}, y_n)$
8:         Determine histogram's bin $b$ of sample $(f_{dn}, y_n)$
9:         Add $w_n$ to $_k(b)$
10:    end for
11:    for $b = 1, \dots, B$ do
12:        Determine maximally occured class $\kappa = \arg\max_k {}_k(b)$
13:        Set $F(b) = \kappa$
14:    end for
15:    return $F$
16: end function
```

So, our classification scheme is based on data that is proportional to the *a posteriori* probability $p(k|f)$, and therefore we call it MAP.

In fig. 4, we present again a histogram of a feature with respect to the two classes. For each bin, we determined the superior class, and we encoded this result by a colored dot on top of the according bar.

We use histograms with B bins, so we obtain a classifier which is equivalent to the classification with maximally $B - 1$ thresholds, which could be modeled by a decision tree.

Although, we presented the MAP classification with respect to two class problems, it is easily adaptable to multi-class problems. We demonstrate an example with three classes in the next part.

5.2 Demonstration at synthetic dataset

We changed our synthetic data set which we briefly introduced in section 2.3 to a multi-class example. Therefore, we defined the samples (x_n, y_n) by

$$y_n = \left\{ \begin{array}{lll} 1\,(\text{red}) & - & \text{if}\, f_1^n < 0.4\,\text{and}\, f_2^n > 0.4 \\ 3\,(\text{green}) & - & f_1^n > 0.6\,\text{and}\, f_2^n < 0.6 \\ 2\,(\text{blue}) & - & \text{else} \end{array} \right\} \qquad (37)$$

and visualized the data set in fig. 5. We generated 10000 training samples and 5000 test samples by an uniform distribution over $[0,1] \times [0,1]$. The

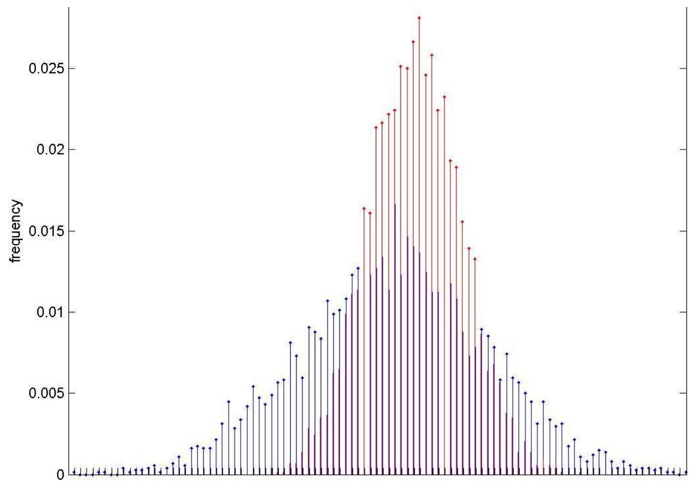

Figure 4: MAP classification based on histogram of a single feature with respect to two classes (red and blue), taken from the data set *ringnorm*.

training set contains 2285 samples from class 1, 5333 samples from the second class and $2,382$ samples from class 3.

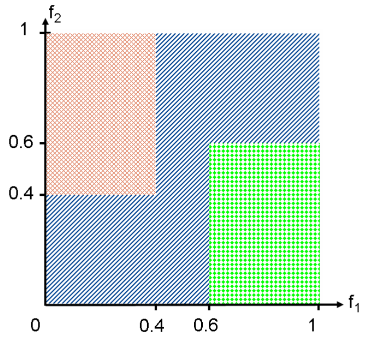

Figure 5: Synthetic dataset with three non-overlapping classes: class 1 is shown in red, class 2 in blue and class 3 in green.

We chose 11 bins for our histograms. The first bin covers $0 \leq f \leq 0.1$, and then the other follow, and we have the borders between the classes, 0.4 and 0.6, exactly between two bins, so that we have minimal overlap of the classes for each bin. For the $\boldsymbol{\alpha}_0$ we obtain

$$\boldsymbol{\alpha}_0 = [0.26, 0.76, 0.27] \tag{38}$$

In the first iteration, we only have one precondition $\mathbf{T}$, which is fulfilled by all samples. So, we perorm a MAP-classification on all samples. The histograms of both features including the visualization of the MAP-classification results are shown in fig. 6. The histogram with respect to feature f_1 shows the two bins with the data $0.4 \leq f_1 < 0.5$ and $0.5 \leq f_1 < 0.6$ filled only with data from class 2, and in the other bins the classes 1 and 3 dominate over the samples of class 2. The histogram with respect to the other feature, f_2, shows a MAP-classification result which is based on relatively close decisions. So, it is no surprise that the first classification becomes the first weak classifier h_1. As predictive values we obtain

$$\alpha_1^1 = 0.59 \quad \alpha_1^2 = 3.00 \quad \alpha_1^3 = 0.61. \tag{39}$$

Totally, 33.10% of the test data got wrongly classified by ADTboost, which combines the prediction $\boldsymbol{\alpha}_0$ with the result of the weak classifier h_1.

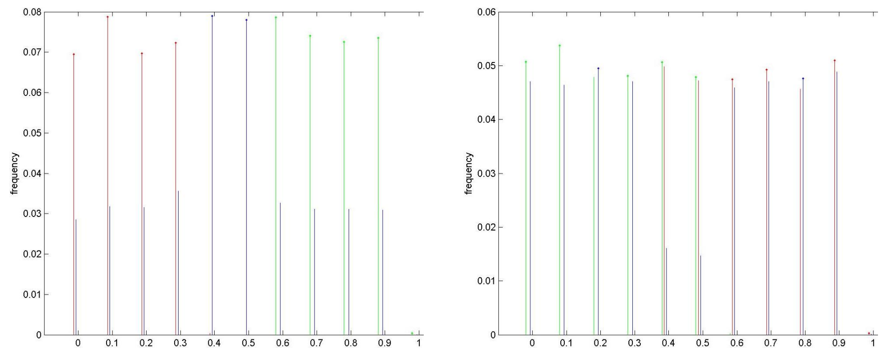

Figure 6: ADTboost: 1st iteration, classifier candidates c_1 and c_2 using features f_1 and f_2, respectively.

In the 2nd iteration, we have four possible preconditions for our new classifier candidates, namely $p_1 : \mathbf{T}$, $p_2 : h_1^1$, $p_3 : h_1^2$ and $p_4 : h_1^3$. At each of these four positions at the alternating decision tree we look for the best MAP-classifier. Therefore, we just select the classifier with the lowest error rate on those samples which fulfill the specific precondition. Then we need to compare the errors with respect to the preconditions. Therefore, we determined the Z-values according to to equ. 34, where we just ignored the second index:

$$Z_1 = 1.69 \quad Z_2 = 1.20 \quad Z_3 = 1.75 \quad Z_4 = 1.15. \tag{40}$$

So, we selected the best classifier candidate c_2 at the fourth precondition and obtain as second weak classifier $h_2 = h_1^3 \wedge c_2$, where c_2 is the MAP-classifier on feature f_2. We show this MAP-classifier in fig. 7, left. Due to the precondition $p_4 : h_1(x) = 3$, which is equivalent to $f_1 > 0.6$, there are no samples which belong to class 1, and the error rate of only the second classifier is 0%. We obtain for the predictive values

$$\alpha_2^1 = 0.0 \quad \alpha_2^2 = 3.69 \quad \alpha_2^3 = 4.99. \tag{41}$$

The total error of the ADTboost, which combines the prediction $\boldsymbol{\alpha}_0$ with the results of the first two weak classifiers h_1 and h_2, is 16.32% of the test data.

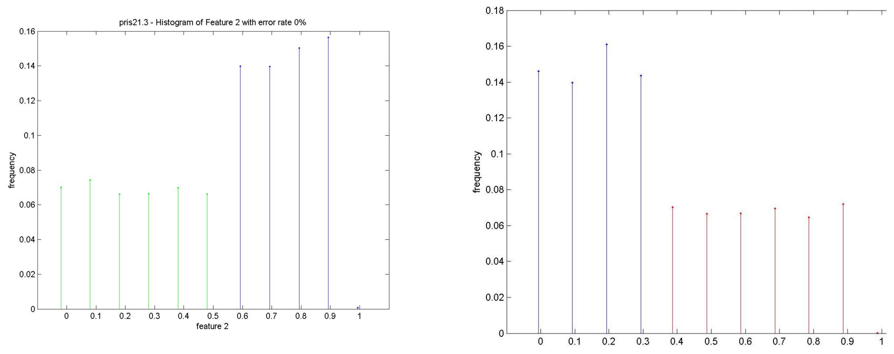

Figure 7: ADTboost: Best MAP-classifiers at 2nd (left) and 3rd (right) iteration, respectively. Both use the feature f_2, but have different preconditions.

In the 3rd iteration, we have seven possible preconditions for our new classifier candidates. The four one from the previous iteration and, additionally, $p_5 : h_2^1$, $p_6 : h_2^2$ and $p_7 : h_2^3$. Now, the best classifier candidate is chosen in combination with precondition $p_2 : h_1^1$, which is visualized in fig. 7, right. We obtain the following predictive values

$$\alpha_3^1 = 4.98 \quad \alpha_3^2 = 5.16 \quad \alpha_3^3 = 0.0, \tag{42}$$

which leads to a total error rate of 0.0% of the test data set, if all three weak classifiers are combined together and with the initial predictive values $\boldsymbol{\alpha}_0$.

The resulting alternating decision tree is visualized in fig. 8. It shows a ternary tree, because we work with three classes and arranged the preconditions with respect to the number of classes. Small predictive values, e. g. $\alpha < 1.0$, sinalize that the specific weak classifier performs with a high error rate; big predictive values, e. g. $\alpha > 2.0$ show that the error rate is very low for a specific decision.

Now, we want to demonstrate, how some feature vectors will get classified. Therefore, we determine the results of ADTboost for nine selected points $P_1 = [0.25, 0.25]$, $P_2 = [0.25, 0.5]$, $P_3 = [0.25, 0.75]$, $P_4 = [0.5, 0.25]$, $P_5 = [0.5, 0.5]$, $P_6 = [0.5, 0.75]$, $P_7 = [0.75, 0.25]$, $P_8 = [0.75, 0.5]$, $P_9 = [0.75, 0.75]$. For a correct classification, the points P_4 and P_7 should get classified as

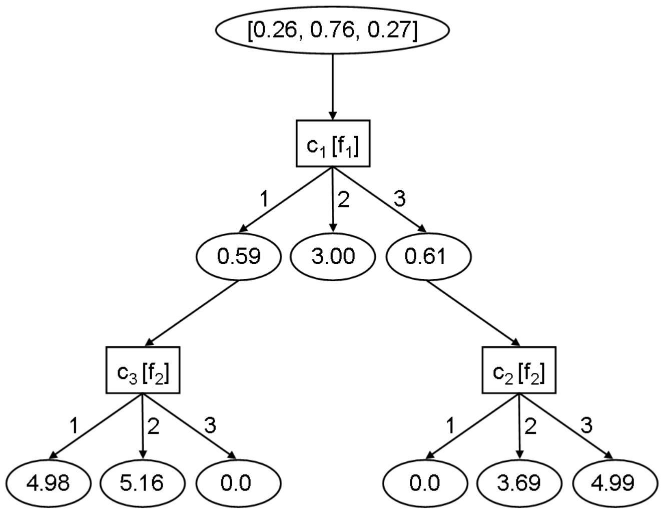

Figure 8: Resulting alternating decision tree after three iterations of ADT-boost on the synthetic data set with three classes.

class 1, the points P_1, P_2, P_5, P_8 and P_9 should be identified as members of class 2, and if the points P_3 and P_6 belong to class 3.

$$\begin{array}{rcllcl}
H(P_1) & = & \arg\max\,[0.26+0.59, 0.76+5.16, 0.27] & = & 2 \\
H(P_2) & = & \arg\max\,[0.26, 0.76+3.00, 0.27] & = & 2 \\
H(P_3) & = & \arg\max\,[0.26, 0.76, 0.27+0.61+4.99] & = & 3 \\
H(P_4) & = & \arg\max\,[0.26+0.59+4.98, 0.76, 0.27] & = & 1 \\
H(P_5) & = & \arg\max\,[0.26, 0.76+3.00, 0.27] & = & 2 \\
H(P_6) & = & \arg\max\,[0.26, 0.76, 0.27+0.61+4.99] & = & 3 \\
H(P_7) & = & \arg\max\,[0.26+0.59+4.98, 0.76, 0.27] & = & 1 \\
H(P_8) & = & \arg\max\,[0.26, 0.76+3.00, 0.27] & = & 2 \\
H(P_9) & = & \arg\max\,[0.26, 0.76+3.69, 0.27+0.61] & = & 2
\end{array} \tag{43}$$

All nine test points are correctly classified by ADTboost after three iterations. In our small example, the test points P_1 and P_9 are the only two points which would get misclassified, if we would stop the training after the first iteration. And after the second iteration, only P_1 would still get misclassified.

References

Asuncion, A., & Newman, D.J. 2007. *UCI Machine Learning Repository.*

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

Drauschke, M. 2008. *Feature Subset Selection with Adaboost and ADT-boost.* Tech. rept. Department of Photogrammetry, University of Bonn.

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

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

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

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

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

Zhu, Ji, Rosset, Saharon, Zou, Hui, & Hastie, Trevor. 2006 (January). *Multi-class AdaBoost.* submitted in 2005, not published yet.