Boosting Threshold Classifiers for High–Dimensional Data in Functional Genomics

Ludwig Lausser[1], Malte Buchholz[3], and Hans A. Kestler[1,2,⋆]

[1] Department of Internal Medicine I, University Hospital Ulm, Germany
[2] Institute of Neural Information Processing, University of Ulm, Germany
[3] Internal Medicine, SP Gastroenterology, University Hospital Marburg, Germany
ludwig.lausser@uni-ulm.de, malte.buchholz@staff.uni-marburg.de, hans.kestler@uni-ulm.de

Abstract. Diagnosis of disease based on the classification of DNA microarray gene expression profiles of clinical samples is a promising novel approach to improve the performance and accuracy of current routine diagnostic procedures. In many applications ensembles outperform single classifiers. In a clinical setting a combination of simple classification rules, such as single threshold classifiers on individual gene expression values, may provide valuable insights and facilitate the diagnostic process. A boosting algorithm can be used for building such decision rules by utilizing single threshold classifiers as base classifiers. AdaBoost can be seen as the predecessor of many boosting algorithms developed, unfortunately its performance degrades on high-dimensional data. Here we compare extensions of AdaBoost namely MultiBoost, MadaBoost and AdaBoost-VC in cross-validation experiments on noisy high-dimensional artifical and real data sets. The artifical data sets are so constructed, that features, which are relevant for the class distinction, can easily be read out. Our special interest is in the features the ensembles select for classification and how many of them are effectively related to the original class distinction.

1 Introduction

The onset and progress of many human diseases, including most if not all human cancers, is associated with profound changes in the activity status of large numbers of genes. DNA Microarrays are high–throughput molecular biology devices capable of monitoring the expression levels of up to several thousand genes simultaneously. One important goal in biomedical research is to make use of this biological principle to develop novel approaches for the accurate differential diagnosis of diseases based on microarray analyses of clinical samples (e.g. tissue biopsy samples). Often single classifiers trained are not able to fulfill certain tasks satisfactorily. If this is the case, better results might be achieved by integrating the results of a whole ensemble of classifiers. Meta algorithms, which do so, are called ensemble methods. They use a basic learing algorithm, generate

⋆ Corresponding author.

a set of base classifiers and combine them in order to get an improved classifier. Boosting algorithms are a subgroup of these methods. Boosting methodes have the characteristic to be able to combine classifiers with moderate accuracy (weak classifiers) to an ensemble with high accuracy [1]. One of the most popular Boosting algorithms is AdaBoost from Freund and Schapire [2]. The use of AdaBoost was evaluated for many different data sets, but it has also been shown that its performance did not match the expectations on high–dimensional data [3]. The concept of combining several, sometimes weak categorization rules, resembles in some aspects human medical decision making. This makes the representation more suitable for further investigation of functional dependencies. We here investigate its use in the context of expression profile classification. We apply several variants of AdaBoost to published gene expression profile data from different tumor types. Furthermore we investigate the performance on artificial high-dimensional data with different noise levels and a varying number of discriminating features among many irrelevant, which reflects the current belief (of biologists) of only a small number of genes (among all) being relevant for categorization.

2 AdaBoost

A pseudocode description of AdaBoost can be seen in Algorithm 2. AdaBoost iteratively creates an ensemble of T members. The algorithm receives an sample S of N training examples $(\boldsymbol{x}_i, y_i)$, where $\boldsymbol{x}_i$ is an element of the input space and $y_i \in \{-1, 1\}$ is its label. Before starting the iterations, an N-dimensional weight vector $D_1 = (1/N, \dots, 1/N)^T$ is initialized. This vector influences the training of the weak classifier h_t in one of two ways. If AdaBoost is used as a Boosting by resampling algorithm, D_t will be used as a distribution for choosing the weak classifier's training examples. If AdaBoost is used as a Boosting by reweighting algorithm, the whole training set and the weight vector are used as input arguments for a weak learning algorithm which can deal itself with weighted training examples. This means that an example with an high weight influences the training of the weak classifier more than an example with an low weight (step 1). After h_t has been chosen, D_t is used to compute a weighted training error ϵ_t (step 2). With ϵ_t the parameter α_t is calculated (step 3), which determines the influence of h_t on the final ensemble h_f. According to α_t the weight vector D_t is updated as well (step 4). The weight of an example will be decreased if it was classified correctly and increased otherwise. In this way the training of new ensemble members will always concentrate such misclassified examples.

2.1 Base Classifier

The base classifier used in these experiments is chosen from the class $h_{c,d,e}(\boldsymbol{x})$:

$$h_{c,d,e}(\boldsymbol{x}) = \begin{cases} sign(\mathbb{1}_{[e \le x_d]} - 0.5), & \text{if } c = 1 \\ sign(\mathbb{1}_{[e \ge x_d]} - 0.5), & \text{otherwise} \end{cases} \tag{1}$$

```
Algorithm 1. AdaBoost($S$,$WeakLearn$,$T$)
Input:
  – sequence $S$ of $N$ labeled examples $\langle(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\rangle$
    where $\boldsymbol{x}_i \in X$ and $y_i \in \{-1, 1\}$
  – weak learning algorithm WeakLearn
  – integer $T$ specifying number of iterations

Init:
  distribution $\boldsymbol{D_1}$ with $D_1^i = \frac{1}{N}$ for all $i \in \{1, \ldots, N\}$

Procedure:
  Do for $t = 1, 2, \ldots, T$
    1. Call WeakLearn, providing it with the distribution $\boldsymbol{D}_t$;
       get back a hypothesis $h_t : X \to \{-1, 1\}$.
    2. Calculate the error of $h_t : \epsilon_t = \sum_{i=1}^{N} D_t(i) \mathbb{1}_{[h_t(\boldsymbol{x}_i) \neq y_i]}$.
    3. Set $\alpha_t = \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
    4. Update weights vector
       $D_{t+1}^i = \frac{D_t^i \exp\left(-\alpha_t \mathbb{1}_{[h_t(\boldsymbol{x}_i) = y_i]}\right)}{Z_t}$
       where $Z_t$ is a normalization factor

Output:
  A hypothesis $h_f$
  $h_f(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}) > 0 \\ -1, & \text{otherwise} \end{cases}$
```

This class contains all simple threshold classifiers working on only one feature dimension. Here d is the chosen dimension and e is the chosen threshold. The parameter c determines the kind of inequation used by the classifier. In each iteration t the best classifier h_t is chosen:

$$h_t = \arg\min_{h_{c,d,e}} \sum\nolimits_{i=1}^{N} D_t \mathbb{1}_{[h_{c,d,e}(\boldsymbol{x}) \neq y_i]} \tag{2}$$

2.2 Tested Algorithms

Most of the Boosting algorithms proposed after 1995 are more or less based on the AdaBoost algorithm. Differences appear most often in weighting schemes or in the used error formula. In this section the algorithms, which were used in these tests shall be described and their differences to AdaBoost shall be highlighted.

MultiBoost. In this methode the boosting idea is coupled with wagging [4]. Wagging is a derivate from Breiman's Bagging approach [5]. In the original Bagging methode the base classifiers are trained on bootstrap replicates from the original training data. For each example of the training set it is randomly chosen, if an example is placed into the replicate set or not. This process is continued until the replicate set is from the same size than the original training set. In this way some examples will be in the bootstrap replicate more than once and others won't be in there at all. The trained base classifiers are combined to an unweighted sum. Wagging is a variante of Bagging for base classifiers, which can deal with weighted examples. Here each base classifier is trained on a weighted version of the original data. The single weights are chosen after an distribution, which simulates the bootstrapping process. In the MultiBoost algorithm a continuous Poisson distribution is used for this simulation. The MultiBoost [6] algorithm itself builds a wagging ensemble of AdaBoost ensembles. This means that after a certain number of boostingsteps, all example weights are reset like in the Wagging approach. The next ensemble members will again be chosen as in the AdaBoost algorithm with the reseted weights as its initial weight vector. The final ensemble is a sum of all weighted base classifiers. Normally the MultiBoost algorithm receives a number of timesteps determining the sizes of the AdaBoost ensembles. In this work MultiBoost's default settings are used, which determine the size of a single AdaBoost ensemble to $\left\lfloor \sqrt{T} \right\rfloor$.

MadaBoost. The MadaBoost[1] algorithm was proposed by Carlos Domingo and Osamu Watanabe [7]. The algorithm differs from the original AdaBoost in its sample weighting scheme. If the weight of an single data point x_i $D_t^i \geq D_0^i$ this weight is reset to D_0^i. In this way MadaBoost counteracts the fact, that noisy data receives very high weights in the original AdaBoost weighting scheme.

AdaBoost-VC. The main difference between AdaBoost and AdaBoost-VC [9] is the new error formular ϵ_{VC} that is used by this algorithm:

$$\epsilon_t^{VC} = \epsilon_t + \frac{d}{N}\left(\log N + \sqrt{1 + \frac{\epsilon_t N}{d}}\right) \tag{3}$$

Here N denotes the number of training examples and ϵ_t the weighted training error as used in the original AdaBoost algorithm. The parameter d regulates the influence of the additional term. This term is inspired by theoretical foundings of Vapnik about an upper bound to the expectation error of classifiers [10]. Within these experiments d is set to $(1, 2, 3, 4)$. Another difference to the original AdaBoost is that each feature can only be used once in an ensemble. If a feature exists, which seperates the training data perfectly, a classifier using it will always minimize a weighted training error and will be selected in each iteration.

[1] Actually Carlos Domingo and Osamu Watanabe proposed different sub-versions of their algorithm [7] [8]. The version, which is talked about here is the batch learning version of MB1.

3 Data Sets

For the shown tests both artificial and real data sets are used. The main tests were done under the controlled conditions of the artificial data sets. How these sets are built is shown in 3.1. The real data sets are described in 3.2.

3.1 Artificial Data

An data set contains 200 data points $\boldsymbol{x} \in [-1,1]^{100}$. Each entry of $\boldsymbol{x}$ is drawn independently after a uniform distribution. The labels $y \in [-1,1]$ are given by a function $f_j(\boldsymbol{x})$ which depends on the first j dimensions of $\boldsymbol{x}$:

$$f_j(\boldsymbol{x}) = \begin{cases} +1, & \text{if } \sqrt{\sum_{i=1}^{j}(x_i-1)^2} < \sqrt{\sum_{i=1}^{j}(x_i-(-1))^2} \\ -1, & \text{else} \end{cases} \tag{4}$$

The function $f_j(\boldsymbol{x})$ signals if the distance from the subvector $(x_1,\cdots,x_j)^T$ to the j-dimensional unit vector is smaller than the distance from $(x_1,\cdots,x_j)^T$ the the j-dimensional negative unit vector. In this way the number of real features can easily be regulated by changing the parameter j. This methode will create approximately the same number of positiv and negativ examples. In the experiments perturbed labels y' are used. These labels are generated according to a noise rate ρ. For each $\boldsymbol{x}$ an random variable $\tau \in [0,1]$ is drawn after a uniform distribution. The new label is built as follows:

$$y' = \begin{cases} +y, & \text{if } \tau >= \rho \\ -y, & \text{else} \end{cases} \tag{5}$$

Tests were made for $j \in \{2,10,20\}$ and $\rho \in \{0, 0.1, 0.2\}$.

3.2 Real Data

ALL-AML-Leukemia. The ALL-AML-Leukemia data set presented by Golub et al. [11] contains data from a microarray experiment concerning accute Leukemia. The data set contains examples for two different subtypes of the disease called ALL (acute lymphoblastic leukemia) and AML (acute myeloid leukemia). The 47 ALL and 25 AML examples contain 7129 probes for 6817 human genes.

Breast cancer. The breast cancer data set presented by van't Veer et al. [12] contains microarray data from patients who had developed distant metastases within 5 years (relapsed patients) and patients who remained healthy from the disease for at least 5 years (non-relapsed patients). The 34 relapse and the 44 non-relapse examples contain data from 24481 gene fragments. If the value of an attribute is not availble in a single example the mean value is calculated over al values from the same class.

Colon cancer. The colon cancer data set presented by Alon et al. [13] contains 40 biopsis of tumor tissues (negative examples)and 22 of normal tissues (positive examples). Each expression profile consists of 2000 genes.

4 Results and Conclusion

We performed 10–fold cross–validation tests on the previously described gene expression profiles. The ensemble size was stepwise increased by 10 from 10 to 100. The results are given in Table 1.

Table 1. Error rates on gene expression data form different tumor types [11,12,13] (AdaBoost-VC: $d = 1 \dots 4$). The number of ensemble members was increased from 10 to 100 by a stepsize of 10.

	breast cancer			colon cancer			leukemia		
	min	mean	var	min	mean	var	min	mean	var
AdaBoost	0.27	0.31	$6.0 \cdot 10^{-4}$	0.16	0.21	$6.0 \cdot 10^{-4}$	0.03	0.06	$3.0 \cdot 10^{-4}$
MadaBoost	0.27	0.31	0.001	0.20	0.23	$4.0 \cdot 10^{-4}$	0.03	0.04	$2.0 \cdot 10^{-4}$
MultiBoost	0.25	0.30	$8.0 \cdot 10^{-4}$	0.15	0.19	$5.0 \cdot 10^{-4}$	0.04	0.06	$1.0 \cdot 10^{-4}$
Adaboost-VC1	0.25	0.28	$6.0 \cdot 10^{-4}$	0.15	0.17	$4.0 \cdot 10^{-4}$	0.025	0.038	$1.9 \cdot 10^{-4}$
Adaboost-VC2	0.28	0.29	$1.7 \cdot 10^{-4}$	0.13	0.14	$1.3 \cdot 10^{-4}$	0.038	0.04	$8.5 \cdot 10^{-5}$
Adaboost-VC3	0.27	0.56	0.012	0.23	0.61	0.018	0.0375	0.0379	$1.7 \cdot 10^{-6}$
Adaboost-VC4	0.66	0.68	$3.2 \cdot 10^{-4}$	0.70	0.72	$8.8 \cdot 10^{-4}$	0.0625	0.2733	0.0272

Table 2. Minimal errors on the artificial training set. Here ρ and j determine the noise rate and the number of real features used for building the artificial data set. The labels *err* and *iter* determine the minimal test error and the iteration when it was achieved.

	$j = 2$						$j = 10$						$j = 20$					
	$\rho = 0$		$\rho = 0.1$		$\rho = 0.2$		$\rho = 0$		$\rho = 0.1$		$\rho = 0.2$		$\rho = 0$		$\rho = 0.1$		$\rho = 0.2$	
	err	*iter*	*err*	*iter*	*err*	*iter*	*err*	*iter*	*err*	*iter*	*err*	*iter*	*err*	*iter*	*err*	*iter*	*err*	*iter*
AdaBoost	0.09	14	0.14	3	0.26	3	0.15	72	0.21	130	0.31	30	0.21	118	0.28	13	0.34	9
MadaBoost	0.08	7	0.13	3	0.22	8	0.14	30	0.21	16	0.30	11	0.21	127	0.27	13	0.33	49
MultiBoost	0.07	134	0.13	86	0.21	16	0.13	150	0.18	139	0.29	150	0.25	138	0.27	125	0.31	116
AdaBoost-VC	0.20	3	0.22	100	0.24	12	0.17	9	0.21	13	0.30	26	0.27	48	0.26	16	0.34	24

On the artificially generated data we estimated the expected classification error by 10-fold cross-validation tests. Ensembles of different sizes from 1 to 150 were trained for the algorithms, AdaBoost, MultiBoost and MadaBoost. Because AdaBoost-VC discards on feature-dimension per iteration, AdaBoost-VC can only be evaluated up to an ensemble size of 100. The algorithms are tested on artificial data sets differing in their number of real features $j \in \{2, 10, 20\}$ and the used noise level $\rho \in \{0, 0.1, 0.2\}$, see Figure 1. The minimal test errors are shown in Table 2. Reference experiments with other standard classifiers are listed in Table 3.

Table 3. Reference experiments on the artificial data set. The classifiers, which were used for these experiments are 1-nearest neighbour ($1NN$), 5-nearest neighbour ($5NN$), support vector machine (SVM) with linear kernel and $C = 1$, nearest centroid ($NCen$) and nearest shrunken centroid classifier ($SCen$) with $\Delta = 0.1$ and 30 steps. The parameters ρ and j denote the noise rate and the number of real features respectively.

	ρ														
	$1NN$			$5NN$			SVM			$NCen$			$SCen$		
j	0	0.1	0.2	0	0.1	0.2	0	0.1	0.2	0	0.1	0.2	0	0.1	0.2
2	0.41	0.35	0.42	0.41	0.33	0.39	0.15	0.22	0.32	0.23	0.20	0.24	0.20	0.17	0.22
10	0.37	0.43	0.41	0.39	0.39	0.37	0.23	0.31	0.35	0.24	0.25	0.29	0.25	0.29	0.28
20	0.37	0.34	0.43	0.36	0.33	0.37	0.18	0.24	0.32	0.23	0.23	0.28	0.22	0.23	0.29

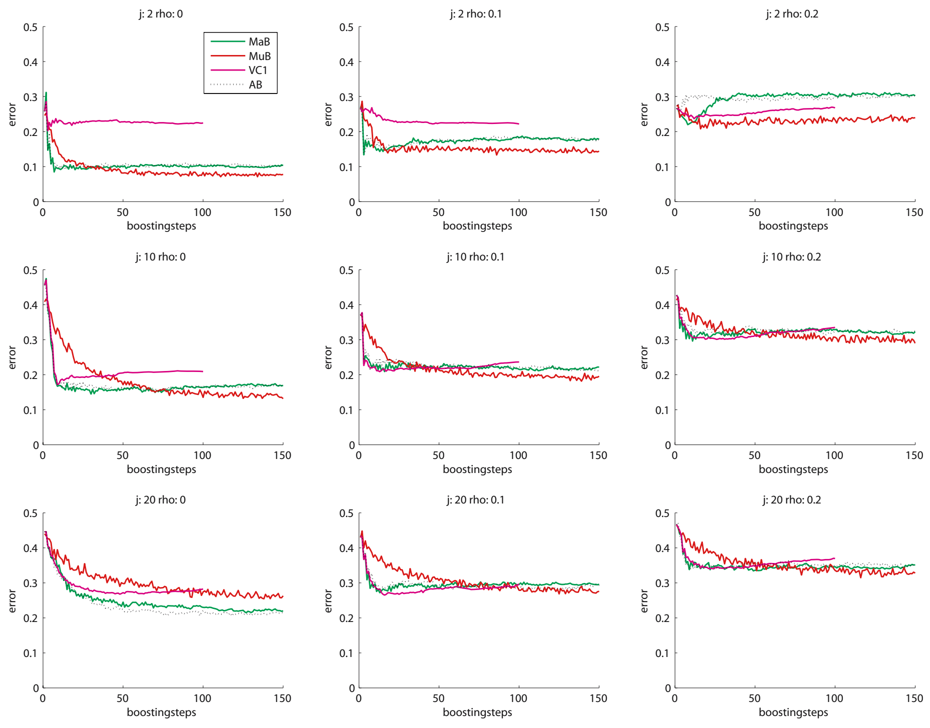

Fig. 1. Results of AdaBoost (AB, black line), MadaBoost (MaB, green), MultiBoost (MuB, red), AdaBoost-VC (VC1, pink) on the artifical data set. Please note that AdaBoost-VC can use at most 100 weak classifiers in this setting.

In a second round of experiments on the artificial data it was tested which features are selected by the different algorithms. Ensembles of 2 up to 50 members are trained as described for the cross-validation experiment. It was recorded how many relevant features were used. The training set contained all 200 points. The

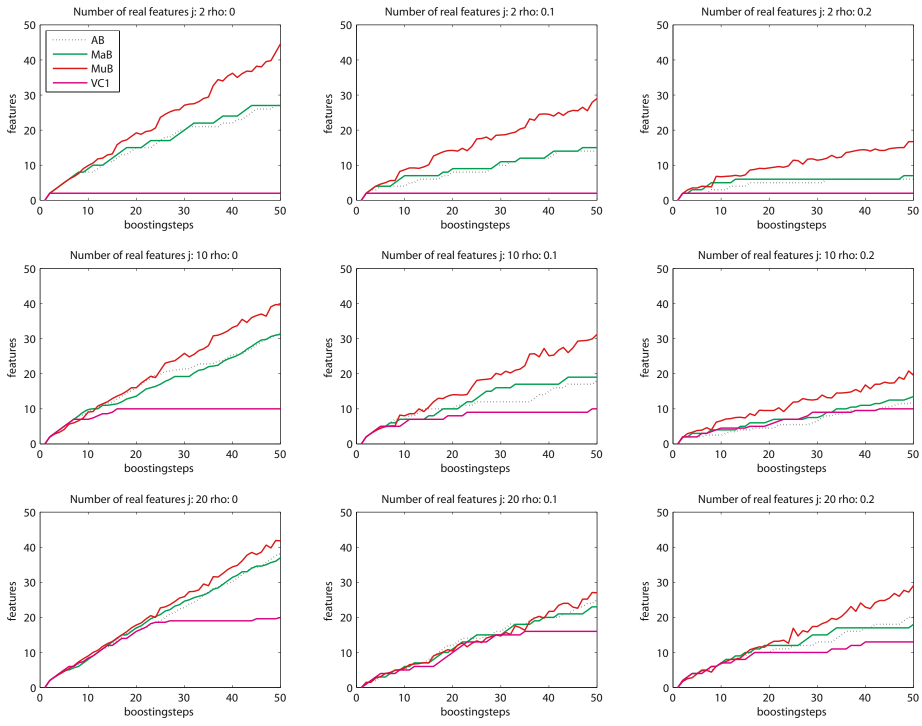

Fig. 2. Results of the feature selection test on the artifical data set

experiment was repeated 10 times with different permutations of the training set. The mean of these 10 experiments is given in Figures 2.

4.1 Results

The results for the experiments on the artificial data sets can be seen in Fig. 1. In the noise free case ($\rho = 0$), AdaBoost and MadaBoost achieve in the earlier iterations lower error rates than the other algorithms. This effect increases if the number of real features increases ($j \in \{10, 20\}$). If more noise is added, the effect is reduced. In the case of few real features MultiBoost achieves lowest error rates, if large ensembles are used. This effect decreases if j raises. The AdaBoost-VC experiments show, that the use of an higher value for d also increases the error rate. In the best tested case ($d = 1$), the error rate of AdaBoost-VC is the highest one in case $\rho = 0$ and $j = 2$. As the ensemble increases the error rate of AdaBoost-VC tends to increase.

The results on the real data sets can be seen in Table 1. For the ALL-AML-Leukemia data set and the colon cancer data set the accuracy of AdaBoost-VC outperformes the other algorithms. In all real data experiments, the error rates of MultiBoost outperform the error rates of AdaBoost if large ensembles are used. The results given for the ALL-AML-Leukemia and the colon cancer data set are comparable to those given in [9]. For the classifier, published with the breast cancer data set, a classification error of 17% is reported [12]. Note that for this classifier manually set cut-off parameters are used.

The results of the second round of experiment on the artificial data are given in Fig. 2. In this experiment MultiBoost choses more often real features than the other algorithms. This effect depends on the parameter f. As f raises, the effect is decreased. The maximal number of real features used by AdaBoost-VC is also determined by f. For higher values of f the number of used real features varies among the different varients of these algorithm. In thes case the varients with small d use more real features than the other.

4.2 Conclusion

The behaviour of MultiBoost can be attributed to its internal AdaBoost ensemble of size $\left\lfloor \sqrt{T} \right\rfloor$ which depends on the size T of the whole MultiBoost ensemble. If T is small, a single AdaBoost ensemble will not contain enough weak classifiers according to the number of effective features j. If the single AdaBoost ensembles reaches an appropriate size, MultiBoost outperforms AdaBoost. As no feature is presented twice, AdaBoost-VC can only choose j useful weak classifiers. So after $T \geq j$ iterations an AdaBoost-VC ensemble contains at least $T - j$ useless classifiers. The benefit of including these classifiers is mere chance. Note that this is also the reason why the ensembles outperform most of the mentioned distance based classifiers in this scenario. On the real data sets AdaBoost-VC seems to be more robust in the choise of parameter d. This phenomenon originates possibly from a greater number of informative features in the tumor gene expression profiles which may also be attributed to co-regulation or prior gene selection.

References

1. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. Machine Learning 37(3), 297–336 (1999)
2. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. In: Vitányi, P. (ed.) EuroCOLT 1995. LNCS, vol. 904, pp. 23–37. Springer, Heidelberg (1995)
3. Dudoit, S., Fridlyand, J., Speed, T.P.: Comparison of discrimination methods for the classification of tumors using gene expression data. Journal of the American Statistical Association 97(457), 77–87 (2002)
4. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning 36(1-2), 105–139 (1999)
5. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
6. Webb, G.I.: Multiboosting: A technique for combining boosting and wagging. Machine Learning 40(2), 159–196 (2000)
7. Domingo, C., Watanabe, O.: Madaboost: A modification of adaboost. In: COLT 2000: Proceedings of the Thirteenth Annual Conference on Computational Learning Theory, pp. 180–189. Morgan Kaufmann Publishers Inc., San Francisco (2000)

8. Domingo, C., Watanabe, O.: Experimental evaluation of an adaptive boosting by filtering algorithm. Technical Report C-139, Tokyo Institut of Technology Department of Mathematical and Computing Sciences, Tokyo, Japan (December 1999)
9. Long, P.M., Vega, V.B.: Boosting and microarray data. Mach. Learn. 52(1-2), 31–44 (2003)
10. Vapnik, V.: Estimation of Dependences Based on Empirical Data: Springer Series in Statistics (Springer Series in Statistics). Springer-Verlag New York, Inc., Secaucus (1982)
11. Golub, T.R., Slonim, D.K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J.P., Coller, H., Loh, M.L., Downing, J.R., Caligiuri, M.C., Bloomfield, C.D., Lander, E.S.: Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science 286(5439), 531–537 (1999)
12. van 't Veer, L.J., Dai, H., van de Vijver, M.J., He, Y.D., Hart, A.A., Mao, M., Peterse, H.L., van der Kooy, K., Marton, M.J., Witteveen, A.T., Schreiber, G.J., Kerkhoven, R.M., Roberts, C., Linsley, P.S., Bernards, R., Friend, S.H.: Gene expression profiling predicts clinical outcome of breast cancer. Nature 415(6871), 530–536 (2002)
13. Alon, U., Barkai, N., Notterman, D.A., Gish, K., Ybarra, S., Mack, D., Levine, A.J.: Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. Proc. Natl. Acad. Sci. USA 96(12), 6745–6750 (1999)