Network Game and Boosting

Shijun Wang and Changshui Zhang

State Key Laboratory of Intelligent Technology and Systems,
Department of Automation, Tsinghua University, Beijing 100084, China
wsj02@mails.tsinghua.edu.cn, zcs@mail.tsinghua.edu.cn

Abstract. We propose an ensemble learning method called Network Boosting which combines weak learners together based on a random graph (network). A theoretic analysis based on the game theory shows that the algorithm can learn the target hypothesis asymptotically. The comparison results using several datasets of the UCI machine learning repository and synthetic data are promising and show that Network Boosting has much resistance to the noisy data than AdaBoost through the cooperation of classifiers in the classifier network.

1 Introduction

With the rapid development of various business and scientific research based on information technology, the number and the size of distributed databases increase continuously. For the massiness of the distributed datasets, the conditional data mining techniques based on central process on a single computer are not appropriate for today's need. New technologies are required for distributed applications.

In recent 10 years, the ensemble learning methods become a hot topic in the machine learning community. Two of the most popular techniques for constructing ensembles are bootstrap aggregation ("bagging" [1]) and the Adaboost family of algorithms ("boosting" [2-4]). Both of these methods operate by taking a base learning algorithm and invoking it many times with different training sets.

The Bagging algorithm (Bootstrap aggregating) [1] uses bootstrap samples to build the base classifiers. Each bootstrap sample is formed by uniformly sampling from the training set with replacement. The accuracy can be improved through building multiple versions of base classifier when unstable learning algorithms (e.g. neural networks, decision trees) are used.

The AdaBoost algorithm [3], calls a given base learning algorithm repeatedly and maintains a distribution of weights over the training set in a series of rounds $t = 1, ..., T$. During the training progress, the weights of incorrectly classified examples are increased so that the weak learner is forced to focus on the hard examples in the training set.

While the overall success of AdaBoost, there is increasing evidence that boosting algorithms are not quite as immune from overfitting [5]. Krieger et al. [6] introduced a new ensemble learning strategy called BB algorithm based on the careful application of both bagging and boosting. They demonstrated experimentally that the performance of this algorithm is superior to boosting when the training set is noisy. Other noisy tolerant boosting algorithms include AdaBoost$_{Reg}$ [7], AveBoost2 [8], et al.

To satisfy the requirement of rapid developing distributed applications, Fan [9] and Lazarevic [10][11] proposed the distributed versions of boosting for parallel and distributed data mining. The distributed boosting algorithms put more efforts on the disjoint partitions of the data set (d-sampling) and the mechanism is designed for such purpose.

In this paper, we propose a new ensemble learning method called Network Boosting (NB) which combines classifiers on the basis of a network. Through the communication between classifiers, the misclassified samples' weights increase during the training progress. The final classification decision is made through the majority voting of all the hypotheses learned in the training progress. Under the game theory framework, we prove that the Network Boosting algorithm can learn target hypothesis asymptotically. The difference between distributed boosting algorithm and Network Boosting lies in the cooperation mechanism between distributed sites (classifiers).

In chapter 2 we propose the classifier network and the mechanism of Network Boosting (NB). Chapter 3 gives the proof of the convergence of the NB algorithm. Results of comparisons on UCI data sets and the comparisons on noisy data are shown in chapter 4. Chapter 5 concludes with a short summary.

2 Classifier Network and Network Boosting

The idea of Network Boosting comes from our recent research [13, 14] on complex network [15]. In a complex system, the complexity comes from the self-organization or emergence of structure and function from the interaction between the constituent parts of the system. So we introduce the cooperation of classifiers based on a network and expect high accuracy and noise resistance as emergent functions of the classifier network and network boosting scheme.

From biology over computer science to sociology the world is abundant in networks [15]. In the present work, we use random graph as the topology of our classifier network for it has much resistance to the targeted attack [16] and is more suitable for the distributed applications. A random graph is a collection of points, or vertices, with lines, or edges, connecting pairs of them at random. Starting with the influential work of Erdos and Renyi in the 1950s and 1960s [17], the study of random graphs has a long history.

Based on the communication structure, we construct a classifier network in which the nodes are classifiers and links between nodes represent the relationship between classifier pairs. If there is a link between node i and node j, then classifier i will exchange information with classifier j during the training progress.

The dynamic integration approach contains two phases. Assume there are K nodes (classifiers) in the network and the training round is T. In the learning phase, given training set $Z = \langle (x_1, y_1), (x_2, y_2), ..., (x_l, y_l) \rangle$, each classifier on the classifier network is provided with the same training instances and maintains a weight record $w_{k,t}(i)$ for $k = 1, ..., K$, $t = 1, ..., T$, $i = 1, ..., l$ of the instances respectively. Then the classifier in the classifier network is built by the training set sampled from the training data according to the weights record of the training data it holds. After that, the weights of the instances of every node are updated according to the classification results of the node and its neighbors. The classifier network is trained T rounds in such way.

In the application phase, the final classification of the committee is formed by all the hypotheses the classifier network learned during the training progress so that for a new instance its label is decided by the voting. The algorithm is listed in Fig. 1.

```
Algorithm Network Boosting

     Input: Examples $Z = \langle (x_1, y_1), (x_2, y_2), ..., (x_l, y_l) \rangle$
            Network $N$
            Training rounds $T$
            Sampling parameter $\rho$
            Weight update parameter $\beta$

Initialize: $w_{k,1}(x_i) = 1$ for all sample $i = 1, ..., l$ and node $k = 1, ..., K$

    Do for: 1. Generate a replicate training set $T_{k,t}$ of size $l\rho$, by weighted
            sub-sampling with replacement from training set $Z$ for $k = 1, 2, ..., K$.
            2. Train the classifier (node) $C_k$ in the classifier network with respect
            to the weighted training set $T_{k,t}$ and obtain hypothesis
            $h_{k,t} : x \mapsto \{-1, +1\}$ for $k = 1, ..., K$ .
            3. Update the weight of instance $i$ of node $k$ :

            $w_{k,t+1}(i) = w_{k,t}(i)\, \beta^{I\left(h_{k,t}(x_i)=y_i\right)+\sum_n I\left(h_{n,t}(x_i)=y_i\right)} / Z_{k,t},$   (1

            where node $n$ is neighbor of node $i$. $I$ is indication function and $Z_{k,t}$
            is a normalization constant, such that $\sum_{i=1}^{l} w_{k,t+1}(x_i) = 1$.

    Output: Final hypothesis by majority voting using the learned hypotheses
            $h_{k,t} : x \mapsto \{-1, +1\}$ for $k = 1, ..., K$ and $t = 1, ..., T$.
```

Fig. 1. Algorithm Network Boosting

For convenience, we use $NB(K, T, \rho, \beta)$ denoting the parameters used by the Network Boosting when the network is given.

3 Network Game and Boosting

Freund and Schapire [18] showed that boosting can be cast in a game-theoretic framework [19][20][21]. They treated the problem of learning as a repeated game and refer to the row player as the learner and the column player as the environment. Let M be the mistake matrix in which entry $M(i, j)$ is the loss suffered by the row player (learner). The game is played from the row player's perspective and leave the column player's (environment) loss or utility unspecified. Mixed strategies are used by the row and column players in each round. That is, the row player chooses a distribution P over the rows of M and the column player chooses a distribution Q over columns. The row player's expected loss in the round t is computed as

$$M(P_t, Q_t) = \sum_{i,j} P_t(i) M(i,j) Q_t(j) = P_t^T M Q_t.$$

If the column player uses a mixed strategy but the row player chooses a single row i (pure strategy), then the (expected loss) is $\sum_j M(i,j) Q_j$ which we denote by $M(i,Q)$. The notation $M(P,j)$ is defined similarly.

Given a weak learning algorithm, the goal of boosting is to run the weak learning algorithm many times in the repeated play with the environment (instances), and to combine the learned hypotheses into a final hypothesis with error rate as small as possible. Von Neumann's well-known minmax theorem states that

$$\max_Q \min_P M(P,Q) = v = \min_P \max_Q M(P,Q), \tag{2}$$

for every matrix M. The common value v of the two sides of the equality is called the value of the game M. Freund and Schapire [18] proved that by using the LW algorithm, the average loss in the repeated game is not much lager than the game value v . Here we extend the proof of Freund and Schapire for boosting algorithm to the Network Boosting algorithm.

Let X be a finite set of instances and H be finite set of hypotheses $h : X \to \{-1, 1\}$. Let $c : X \to \{-1, 1\}$ be an unknown target concept, not necessarily in H .

The mistake matrix M has rows and columns indexed by instances and hypotheses, respectively.

$$M(x,h) = \begin{cases} 1, & if\ h(x) = c(x) \\ 0, & otherwise \end{cases} . \tag{3}$$

Assuming (H, c) is γ learnable (so that there exists a γ-weak learning algorithm). On each round t , the learner k in the classifier network computes mixed strategy $P_{k,t}$ by normalizing the weights:

$$P_{k,t}(i) = \frac{w_{k,t}(i)}{\sum_{i=1}^{l} w_{k,t}(i)}.$$

Given $M(i, Q_{k,t})$ for each node k at round t , the environment updates the weights by the simple multiplicative rule:

$$w_{k,t+1}(i) = w_{k,t}(i) \beta^{M(i,Q_{k,t}) + \sum_n M(i,Q_{n,t})} \tag{4}$$

where node n is neighbor of node k and $\beta \in [0, 1)$.

Theorem 1. For any node k in the classifier network, the accumulative loss suffered by the instances with parameter $\beta \in [0,1)$ satisfies:

$$\sum_{t=1}^{T} M\left(P_{k,t}, Q_{k,t}\right) \leq c_\beta \ln l + \alpha_\beta \min_P \sum_{t=1}^{T} \left\{M\left(P, Q_{k,t}\right)\right\} +$$
$$\alpha_\beta \left(\min_j \sum_{t=1}^{T} \sum_n M\left(j, Q_{n,t}\right) - \sum_{t=1}^{T} \min_i \sum_n M\left(i, Q_{n,t}\right) \right) \tag{5}$$

where $\alpha_\beta = \frac{\ln(1/\beta)}{1-\beta}$ and $c_\beta = \frac{1}{1-\beta}$.

The proof of Theorem 1 is given in the appendix.

From Theorem 1 it is clear that when β approaches 1, α_β also approaches 1 and the accumulative loss of row player in the T rounds repeated play will not be much greater than the loss of the best strategy for node k.

Corollary 2. Under the conditions of Theorem 1 and with β set to

$$\frac{1}{1+\sqrt{\frac{2\ln l}{T}}}$$

the average per-trial loss suffered by the instances in node k when T is large enough is

$$\frac{1}{T} \sum_{t=1}^{T} M\left(P_{k,t}, Q_{k,t}\right) \leq \min_P \frac{1}{T} \sum_{t=1}^{T} \left\{M\left(P, Q_{k,t}\right)\right\} + \Delta_T \tag{6}$$

where

$$\Delta_T = \sqrt{\frac{2\ln l}{T}} + \frac{\ln l}{T}.$$

.

Proof: See Section 2.2 in Freund and Shapire [2] and note that

$$\frac{1}{T} \left(\min_j \sum_{t=1}^{T} \sum_n M\left(j, Q_{n,t}\right) - \sum_{t=1}^{T} \min_i \sum_n M\left(i, Q_{n,t}\right) \right)$$

approaches 0 when T is large enough.

Corollary 3. Under the conditions of corollary 2, the average expected loss of the instances over K nodes in the T trainings when T is large enough is

$$\frac{1}{KT} \sum_{k=1}^{K} \sum_{t=1}^{T} M\left(P_{k,t}, Q_{k,t}\right) \leq v + \Delta_T \tag{7}$$

where v is the value of game M .

Proof: Let P^* be a minmax strategy for M so that for all column strategies $Q_{k,t}$, for all $k = 1, ..., K$ and $t = 1, ..., T$, $M(P^*, Q_{k,t}) \leq v$. According to Corollary 2,

$$\begin{aligned}&\tfrac{1}{KT}\sum_{k=1}^{K}\sum_{t=1}^{T} M(P_{k,t}, Q_{k,t}) \leq \tfrac{1}{KT}\sum_{k=1}^{K}\sum_{t=1}^{T} M(P^*, Q_{k,t}) + \varDelta_T \\ &\leq v + \varDelta_T\end{aligned}$$

Proof end.

On each round t at the node k , $Q_{k,t}$ may be a pure strategy $h_{k,t}$ and should be chosen to maximize

$$M(P_{k,t}, h_{k,t}) = \sum_{i=1}^{l} P_{k,t} M(i, h_{k,t}) = \Pr_{x \sim P_{k,t}} [h_{k,t}(x) = c(x)]$$

In other words, $h_{k,t}$ should have maximum accuracy with respect to distribution $P_{k,t}$. It's just the goal of weak learner. According to the minmax theorem:

$$\begin{aligned}&\min_x \max_Q M(x, Q) = \min_P \max_Q M(P, Q) \\ &= v \\ &= \max_Q \min_P M(P, Q) = \max_h \min_P M(P, h)\end{aligned},$$

each classifier on the network will learn the target hypothesis asymptotically. So the combined hypotheses learned by the classifier network will compute a final hypothesis h_{fin} identical to target c for sufficiently large T.

4 Experiment Results

4.1 Experiments on UCI Repository

In this section we present experiments of C4.5, Bagging, AdaBoost.M1, BB and Network Boosting on the UCI data sets. We use the implementations of C4.5, Bagging, AdaBoost.M1 and BB provided by Weka in our experiments [22]. The experimental setting is described before the results of the experiments. We employed 16 domains drawn from the UCI Repository [12]. Previously comparisons of AdaBoost, Bagging and other dynamic classifier integration methods were made in [23] [5] [24] [7]. The main characteristics of the 16 data sets are presented in Table 1.

We performed statistical tests to compare the five algorithms. For all the domains we generate 100 random partitions into training and test set with proportion 60 : 40. Then we train a classifier and compute its test set error on each partition. C4.5 is employed as base classifier in all the ensemble methods without the pruning.

In our experiments, we found that sampling parameter 33% achieves better performance. The same conclusion is drawn in BB algorithm [6]. About the size and the connection probability of the classifier network, we found that the accuracy can be improved if the size of the network becomes large (with the expense of more calculation demand); the smaller connection probability often leads to better results for higher connection probability makes the environment more chaostic.

In order to play fairly, 300 C4.5 base classifiers were used in these four ensemble methods. For AdaBoost, we constructed ensembles of at most 300 classifiers. However, if the AdaBoost algorithm terminated early (because a classifier had weighted error greater than 0.5 or unweighted error equal to zero), then a smaller ensemble was necessarily used. For BB algorithm, $BB\,(30, 10, 1/3)$ was used. That is, the aggregate BB classifier is a combination of 300 base classifiers, resulting from the combination of 30 subsamples of 10 boosting iterations. For Network Boosting, a classifier network contains 30 classifiers and based on a random graph with connection probability 0.07 for each pair of nodes and 10 training steps was built. The sampling parameter $\rho = 1/3$ and $\beta = 0.5$ were used for all the data sets.

Table 1. Description of UCI datasets

Dataset	Instances	Classes	Features	
			Discrete	Continuous
audiology	226	24	69	-
breast-w	699	2	-	9
colic	368	2	15	7
credit-a	690	2	9	6
diabetes	768	2	-	8
glass	214	6	-	9
heart-c	303	2	7	6
heart-h	294	2	7	6
hepatitis	155	2	13	6
iris	150	3	-	4
labor	57	2	8	8
lymph	148	4	15	3
soybean	683	19	35	-
vehicle	846	4	-	18
vote	435	2	16	-
waveform	300	3	-	40

Table 2 shows the average error rate (err) and standard deviation (std) of each data set tested by every algorithm and results of significance tests of Bagging, AdaBoost, BB with NB. "+" and "-" mean that there is significant difference between the results of the two algorithms. From the table we can find that NB can improve the accuracy significantly compared to Bagging, AdaBoost and BB.

Learning curve analysis provides a powerful tool to inspect the dynamics of an ensemble learning method [25]. In Figure 2 we show the performance of NB algorithm on different training steps. Due to lack of space, we do not include the results for all 16 datasets, but present 5 representative datasets. Every point in

Table 2. Comparisons of C4.5 and four ensemble methods on UCI datasets. The columns S1, S2 and S3 show the results of significance tests (at 0.05 the significance level) of Bagging, AdaBoost.M1 and BB with Network Boosting respectively.

Name	C4.5	Bagging	AdaBoost	BB	NB			
	$err \pm std$	$err \pm std$	$err \pm std$	$err \pm std$	S1	S2	S3	$err \pm std$
audiology	.2818 ± .0463	.2332 ± .0489	.2126 ± .0420	.2671 ± .0500		-	+	.2264 ± .0461
breast-w	.0610 ± .0125	.0412 ± .0101	.0319 ± .0074	.0339 ± .0101	+	-		.0356 ± .0082
colic	.1792 ± .0263	.1559 ± .0244	.1955 ± .0348	.1561 ± .0229		+		.1598 ± .0242
credit-a	.1695 ± .0176	.1400 ± .0169	.1391 ± .0158	.1337 ± .0173				.1368 ± .0166
diabetes	.2753 ± .0242	.2427 ± .0186	.2647 ± .0201	.2374 ± .0188		+	-	.2439 ± .0221
glass	.3353 ± .0459	.2852 ± .0443	.2572 ± .0493	.3003 ± .0444	+		+	.2662 ± .0410
heart-c	.2413 ± .0355	.2089 ± .0357	.1984 ± .0285	.1780 ± .0281	+	+		.1793 ± .0309
heart-h	.2155 ± .0338	.2008 ± .0282	.2028 ± .0307	.1906 ± .0315	+	+		.1897 ± .0303
hepatitis	.2200 ± .0479	.1829 ± .0387	.1697 ± .0402	.1700 ± .0383	+	+	+	.1563 ± .0359
iris	.0618 ± .0305	.0582 ± .0271	.0647 ± .0257	.0602 ± .0313				.0583 ± .0229
labor	.2139 ± .0899	.1687 ± .0790	.1430 ± .0733	.1652 ± .0865	+	+	+	.1096 ± .0548
lymph	.2500 ± .0484	.2150 ± .0459	.1790 ± .0372	.1840 ± .0485	+	+	+	.1618 ± .0460
soybean	.1259 ± .0223	.0936 ± .0205	.0843 ± .0171	.0927 ± .0200	+	+	+	.0742 ± .0144
vehicle	.2968 ± .0231	.2622 ± .0205	.2322 ± .0203	.2553 ± .0226	+		+	.2373 ± .0195
vote	.0518 ± .0164	.0403 ± .0136	.0541 ± .0158	.0415 ± .0129		+		.0435 ± .0132
waveform	.3142 ± .0395	.2108 ± .0417	.1758 ± .0337	.1727 ± .0303	+	+	+	.1621 ± .0286
average	.2058	.1712	.1628	.1649				.1526

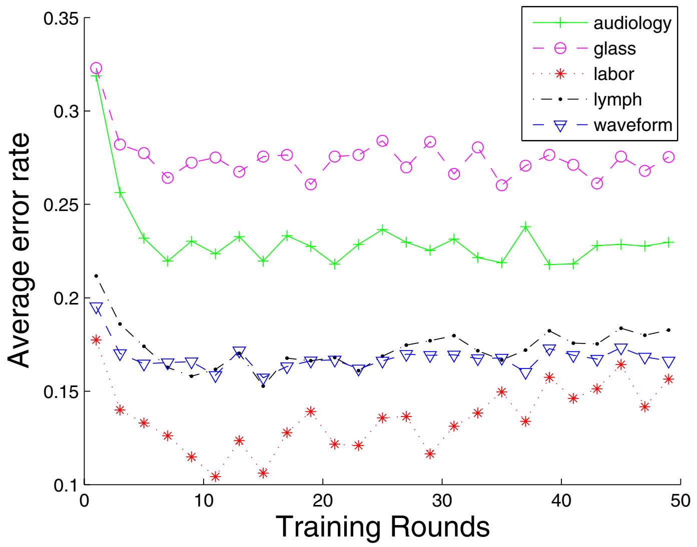

Fig. 2. Learning curves of NB on several UCI data sets

the graph is the average error rate of 50 tests at the given training round on that dataset. From the figure we can find that NB can learn the target hypothesis quickly after only about 10 training rounds.

4.2 Experiments on Synthetical Data Set

First we begin by specifying a simple model used by Krieger [6]. We suppose that there are five dependent feature variables, $X_1, ..., X_5$. We generate each feature i.i.d. from a uniform distribution on the unit interval. The class label Y is binary and determined only by X_1 and X_2 according to the following rule:

$$Y = \begin{cases} 1, X_1 \leq X_2 \\ 0, otherwise. \end{cases} \tag{8}$$

Furthermore, we assume that addition of noise which randomly and independently flips the observed class label with some fixed probability $1-p$. Under this distortion we have that $P(Y = 1|X_1 > X_2) = 1 - p$ and $P(Y = 0|X_1 \leq X_2) = 1 - p$. The special case where $p = 1$ corresponds to noiseless data. We generate a training set consisting of 1000 pairs (Y, X) from the Unit Square model. To test our classifier we provide a noiseless test data set of 10000 points. For bagging and AdaBoost, 300 base classifiers are used; for BB, BB(30,10,1/3) is used; for Network Boosting, we use NB(30,10,1/3,β) with different β. The connection probability of network is set 0.07.

Table 3. Comparisons on UnitSquare model

	p=1	p=0.9	p=0.8	p=0.7
C4.5	0.0333±0.0037	0.0615±0.0098	0.0881±0.0169	0.1282±0.0291
Bagging	0.0243±0.0035	0.0388±0.0069	0.0576±0.0108	0.0909±0.0232
AdaBoost	0.0236±0.0030	0.0757±0.0171	0.0878±0.0144	0.1278±0.0244
BB(30,10,1/3)	0.0227±0.0039	0.0405±0.0057	0.0596±0.0107	0.0935±0.0231
NB(30,10,1/3,0.5)	0.0204±0.0025	0.0434±0.0070	0.0693±0.0134	0.1286±0.0316
NB(30,10,1/3,0.7)	0.0204±0.0034	0.0379±0.0059	0.0616±0.0113	0.1110±0.0367
NB(30,10,1/3,0.9)	**0.0194±0.0038**	**0.0361±0.0067**	**0.0512±0.0115**	**0.0872±0.0216**

In Table 3 the comparison results of average error rate and standard deviation are shown on 100 tests. Network Boosting shows high resistance to noise than others. The significance test (t-test with significance level 0.05) shows that the results of NB(30,10,1/3,0.9) are significantly better than the results of AdaBoost when data are noisy (p=0.9, p=0.8, p=0.7). With the increasing β, NB algorithm shows higher ability on defeating noise for the weights of noisy data will increase slowly with bigger β. As every thing has two sides, with bigger β, the Network Boosting algorithm needs more training rounds to converge to the target hypothesis.

5 Conclusion

In this paper a technique for dynamic integration of classifiers was experimented. The algorithm is easily run in a distributed system. Under the game theory framework, we prove that the average expected loss suffered by row player (environment) is not much larger than that of the game value which means that as a dual problem the combined hypotheses is an approximate maxmin strategy. Through the cooperation between classifiers, the classifier network shows remarkable resistance to overfitting. For the additional computation requirement introduced by the communication between classifiers (nodes) is small and can be ignored, the efficiency of NB is equal to Bagging, AdaBoost and BB when the same number of base classifiers and sample parameter (when generating training data for each base classifier) are used.

The proposed dynamic integration technique Network Boosting was evaluated with C4.5, AdaBoost, bagging and BB on 16 data sets from the UCI machine learning repository. The results achieved are promising. In order to show the Network Boosting's ability on overfitting, we compared it with others on UnitSquare model.

In addition, under some conditions, Network Boosting can be reduced to AdaBoost and Bagging. If there is just one node in the network and weighted voting is used, then Network Boosting reduces to AdaBoost; if the training steps of Network Boosting is one, it just equates the bagging algorithm.

Through the experiments, random graph performs well and is a good choice for NB. How about the performance of Network Boosting if other topologies introduced? Such topologies include small-world network, scale-free network and grid network. Further researches are need in the future on explore the dynamic mechanism of Network Boosting on different network topologies.

References

1. Breiman, L.: Bagging predictors. Machine Learning, (1996) 24 (2):123-140
2. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. European Conference on Computational Learning Theory, (1995)
3. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In Proceedings of the Thirteenth International conference on Machine Learning, (1996)
4. Schapire, R.E.: A brief introduction to boosting. In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, (1999)
5. Dietterich, T.G.: An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine Learning, 40. (2000) 139-158
6. Krieger, A., Long, C., Wyner, A.: Boosting noisy data. In Proceedings of the Eighteenth International conference on Machine Learning, (2001)
7. Rätsch, G., Onoda, T., and Muller, K.R.: Soft margins for AdaBoost. Machine Learning, 42 (3). (2001) 287–320
8. Oza N.C.: AveBoost2: Boosting for Noisy Data. 5th International Workshop, Multiple Classifier Systems. (2004)

9. Fan, W., Stolfo, S.J., Zhang, J.: The application of AdaBoost for distributed, scalable and on-line learning. In SIGKDD, (1999)
10. Lazarevic, A., Obradovic, Z.: The distributed boosting algorithm. In SIGKDD, (2001)
11. Lazarevic, A., Obradovic, Z.: Boosting algorithm for parallel and distributed learning. Distributed and Parallel Databases, (2002) 11, 203-229
12. Merz, C.J., Murphy, P.M.: UCI repository of machine learning databases. http://www.ics.uci.edu/ ~mlearn/MLRepository.html. (1996)
13. Wang Shijun and Zhang Changshui: Weighted competition scale-free network. Phys. Rev. E **70**, 066127 (2004) .
14. Wang Shijun and Zhang Changshui: Microscopic model of financial markets based on belief propagation. Physica A **354C**, 496 (2005) .
15. Albert, R. and Barabási, A.: Statistical mechanics of complex networks. Reviews of Modern Physics, 74 (1). (2002) 47–97
16. Albert, R., Jeong, H. and Barabási, A.: Error and attack tolerance of complex networks. Nature, 406 (2000) 378-382
17. Bollobas, B.: Random Graphs. Academic, London, (1985)
18. Freund, Y., Schapire, R.E.: Game theory, on-line prediction and boosting. Proceedings of the Thirteenth International conference on Machine Learning, (1996)
19. Fudenberg, D., and Tirole, J.: Game Theory. MIT Press, (1991)
20. Freund, Y., and Schapire, R.E.: Adaptive game playing using multiplicative weights. Game and Economic Behavior, 29. (1999), 79-103
21. Breiman, L.: Prediction games and arcing algorithms. Neural Computation, 11. (1999) 1493-1517
22. Witten I.H. and Frank E.: *Data Mining: Practical machine learning tools with Java implementations* (Morgan Kaufmann, San Francisco, 2000)
23. Quinlan., J.R.: Bagging, boosting, and C4.5. In Proceedings of the Thirteenth NationalConference on Artificial Intelligence, AAAI Press/MIT Press. (1996) 725-730
24. Bauer, E., and Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning, 36 (1-2). (2000) 105-139
25. Melville P. and Mooney R.: Constructing Diverse Classifier Ensembles Using Artificial Training Examples. Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence. (2003) 505-510

Appendix. Proof of Theorem 1

Proof: For $t = 1, ..., T$, we have that

$$\begin{aligned}
&\sum_{i=1}^{l} w_{k,t+1}(i) = \sum_{i=1}^{l} w_{k,t}(i)\,\beta^{M(i,Q_{k,t})+\sum\limits_n M(i,Q_{n,t})} \\
&\le \sum_{i=1}^{l} w_{k,t}(i)\left(1-(1-\beta)\,M(i,Q_{k,t})\right)\beta^{\sum\limits_n M(i,Q_{n,t})} \\
&\le \sum_{i=1}^{l} w_{k,t}(i)\left(1-(1-\beta)\,M(i,Q_{k,t})\right)\beta^{\min\limits_i \sum\limits_n M(i,Q_{n,t})} \\
&= \left(\sum_{i=1}^{l} w_{k,t}(i)\right)\left(1-(1-\beta)\,M(P_{k,t},Q_{k,t})\right)\beta^{\min\limits_i \sum\limits_n M(i,Q_{n,t})}
\end{aligned} \tag{9}$$

The first line uses the definition of $w_{k,t+1}(i)$. The second line comes from the fact that $\beta^x \le 1-(1-\beta)x$ for $\beta>0$ and $x\in[0,1]$. The last line uses the definition of $P_{k,t}$. So we can get the following inequation if we unwrap the Eq.(9)

$$\begin{aligned}\sum_{i=1}^{l} w_{k,t+1}(i) &\le l\prod_{t=1}^{T}\left(1-(1-\beta)M(P_{k,t},Q_{k,t})\right)\times\prod_{t=1}^{T}\beta^{\min\limits_i \sum\limits_n M(i,Q_{k,t})}\\ &= l\prod_{t=1}^{T}\left(1-(1-\beta)M(P_{k,t},Q_{k,t})\right)\times\beta^{\sum\limits_{t=1}^{T}\min\limits_i \sum\limits_n M(i,Q_{k,t})}\end{aligned}\qquad(10)$$

Next, note that, for any j ,

$$\sum_{i=1}^{l} w_{k,T+1}(i) \ge w_{k,T+1}(j) = \beta^{\sum\limits_{t=1}^{T} M(j,Q_{k,t})+\sum\limits_n M(j,Q_{n,t})}$$

Combining with Eq.(10) and taking logs gives

$$\begin{aligned}&(\ln\beta)\sum_{t=1}^{T}\left\{M(j,Q_{k,t})+\sum_n M(j,Q_{n,t})\right\}\\ &\le \ln l+\sum_{t=1}^{T}\ln\left(1-(1-\beta)M(P_{k,t},Q_{k,t})\right)+\ln\beta\sum_{t=1}^{T}\min_i\sum_n M(i,Q_{n,t})\\ &\le \ln l-(1-\beta)\sum_{t=1}^{T}M(P_{k,t},Q_{k,t})+\ln\beta\sum_{t=1}^{T}\min_i\sum_n M(i,Q_{n,t})\end{aligned}\qquad(11)$$

Since $\ln(1-x)\le -x$ for $x<1$. Rearranging terms, and noting that this expression holds for any given j

$$\begin{aligned}&\sum_{t=1}^{T}M(P_{k,t},Q_{k,t})\\ &\le \tfrac{\ln l}{1-\beta}+\tfrac{\ln(1/\beta)}{1-\beta}\min_j\sum_{t=1}^{T}\left\{M(j,Q_{k,t})+\sum_n M(j,Q_{n,t})\right\}+\tfrac{\ln\beta}{1-\beta}\sum_{t=1}^{T}\min_i\sum_n M(i,Q_{n,t})\\ &= \tfrac{\ln l}{1-\beta}+\tfrac{\ln(1/\beta)}{1-\beta}\min_P\sum_{t=1}^{T}\{M(P,Q_{k,t})\}+\tfrac{\ln(1/\beta)}{1-\beta}\times\\ &\left(\min_j\sum_{t=1}^{T}\left\{\sum_n M(j,Q_{n,t})\right\}-\sum_{t=1}^{T}\min_i\sum_n M(i,Q_{n,t})\right).\end{aligned}$$