A Bayesian View of Boosting and Its Extension

Yifan Shi **Aaron Bobick** **Irfan Essa**
CPL / GVU Center, College of Computing
Georgia Institute of Technology
Atlanta, GA 30332
monsoon/afb/irfan@cc.gatech.edu

Abstract

In this paper, we provide a Bayesian perspective of boosting framework, which we refer to as Bayesian Integration. Through this perspective, we prove the standard ADABOOST *is a special case of the naive Bayesian tree with a mapped conditional probability table and a particular weighting schema. Based on this perspective, we introduce a new algorithm* ADABOOST.BAYES *by taking the dependency between the weak classifiers into account, which extends the boosting framework into non-linear combinations of weak classifiers. Compared with standard* ADABOOST, ADABOOST.BAYES *requires less training iterations but exhibits stronger tendency to overfit. To leverage on both* ADABOOST *and* ADABOOST.BAYES*, we introduce a simple switching schema* ADABOOST.SOFTBAYES *to integrate* ADABOOST *and* ADABOOST.BAYES*. Experiments on synthetic data and the UCI data set prove the validity of our framework.*

1 Introduction

The boosting framework has been widely used in machine learning community, with several compelling examples of this approach showing its strength and validity [11, 10, 9, 5]. There have been many mathematical explanations for its superior performance [1, 6, 4]. The ADABOOST algorithm works by integrating a set of weak classifiers, each of which is only moderately accurate, into the formation of a much better classifier. A widely acknowledged explanation is to view this process as an additive logistical regression [2]. This observation relies on the use of statistical principles in the boosting setup and reasons that boosting is an approximation to additive modeling on the logistic scale, using maximum Bernoulli likelihood. BNT/graphical model is another popular tool in machine learning [8, 7, 3] with solid theoretical support. Inference on bayesian framework is clear and intuitive to human expert.

In this paper, we put ADABOOST in the Bayesian framework and provide a new perspective for boosting, which we call Bayesian Integration. Through Bayesian Integration, we show that standard ADABOOST is a special case of a naive Bayesian tree with a mapped conditional probability table. The introduction of such a perspective allows us to link the two frameworks together and to leverage rich theoretical results on both sides. One immediate result is that ADABOOST can be used as an effective Bayesian network learning algorithm as it constructs a large naive Bayesian tree on top of BNT classifiers learnt from normal BNT algorithm.

Through Bayesian Integration, it is natural to consider the dependency between the weak classifiers. This is accomplished straight forwardly in the Bayesian framework by adding extra links between nodes. We introduce appropriate simplification, which leads to a new boosting algorithm: ADABOOST.BAYES. The key idea is that there is richer information in boosting framework that can't be expressed by an additive model. The ability to utilize the history of previous weak classifiers can be exploited not only by a refined integration of weak classifiers but also by a better fine grained training of weak classifiers. By introducing this observation to the model, we can get away with less training and faster detections. On the other hand, ADABOOST.BAYES tends to overfit to the model faster. To leverage the strength of both algorithms, we provide a switching schema (ADABOOST.SOFTBAYES) to integrate ADABOOST and ADABOOST.BAYES.

In section 2, we demonstrate the Bayesian perspective of boosting and its analytical exponential upper bound of training error. Then we show that ADABOOST is a special case of Bayesian integration with a certain weighing schema. In section 3, we discuss how to leverage dependency of weak classifiers history information by introducing ADABOOST.BAYES. In section 4, we provides a criterion to supervise the switching between ADABOOST and ADABOOST.BAYES, which leads to another algorithm ADABOOST.SOFTBAYES. Finally, in section 5, we provide experimental results on synthetic data as well as the UCI data sets.

2 Bayesian View of Boosting

For a binary classification problem, it is assumed to be easy to construct a weak learner which for any training set will return a slightly better than random weak classifier. How to assemble such a set of weak classifiers into a strong classifier is the question boosting wants to solve. Assuming the terminology in [4], boosting iteratively calls the weak learner with an training set $\{< x_i, y_i >\}$ with weight distribution $\{D_{ti}\}$ to get a new weak classifier h_t, computes the linear coefficient α_t for h_t then forms the function $H(x)$ as a linear combination of $\{h_t\}$. A generalized version of ADABOOST can be found in [4]. As we will show in this section, the boosting process is equivalent to an growing naive bayesian tree with a specific weighting schema. We call this type of using a growing bayesian tree to integrate weak classifiers as *Bayesian Integration*.

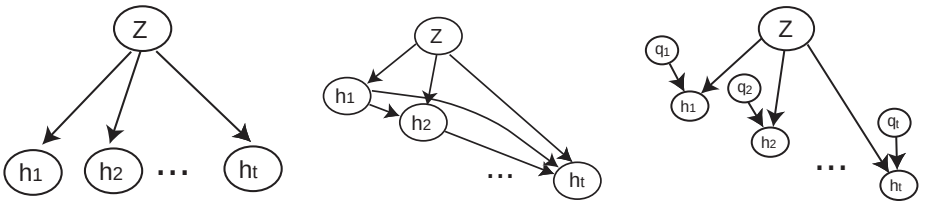
2.1 Bayesian Integration

A bayesian classification tree can be defined as following: define a random variable $z = \{1, -1\}$ as the root node of bayesian tree to represent the hidden label for observation x, define event Z as $z = 1$, $\overline{Z}$ as $z = -1$. For each new weak classifier h_t, construct an evidence node which has a single parent z. The iteration in boosting process is represented by attaching one new h_t at each iteration. The bayesian integration tree at time step t is illustrated in Fig.1.(a). This tree shall integrate the weak hypothesis set $\{h_t(x)\}$ into an overall classifier $F(x) = sign[P(Z|h_1(x), ..., h_t(x)) > 0.5]$

For terminology purpose, we will abbreviate $P(Z|h_1(x), ..., h_t(x))$ as P_t, $P(\overline{Z}|h_1(x), ..., h_t(x))$ as $\overline{P_t}$. For a specific point x_i, we abbreviate $h_t(x_i)$ as h_{ti}, $P(Z|h_1(x_i), ..., h_t(x_i))$ as P_{ti} and $P(\overline{Z}|h_1(x_i), ..., h_t(x_i))$ as $\overline{P_{ti}}$. The initial distribution is specified as $P(z) \equiv [P(Z), P(\overline{Z})] \equiv [P_0, \overline{P_0}]$.

To add an new node, we need to specify $P(h_t|z)$ for the new link. For now, we will just let them be parameters as

$$\begin{aligned} P(h_t|z) &\equiv [\, P(h_t = 1|Z), P(h_t = 1|\overline{Z}),\ P(h_t = -1|Z),\ P(h_t = -1|\overline{Z})] \\ &\equiv [P_t^1, P_t^2, P_t^3, P_t^4], \text{where } P_t^1 = 1 - P_t^3, P_t^2 = 1 - P_t^4 \end{aligned}$$



(a) (b) (c)

Figure 1: (a)Bayesian Integration Tree for Adaboost; (b)Bayesian tree with history dependency; (c)BNT for ADABOOST.BAYES

Given the tree, the posterior probability can be computed as following:

$$\begin{array}{ll} P_t & = \frac{P(z=1,h_1,..,h_t)}{P(h_1,..,h_t)} = \frac{P(z=1,h_1,..,h_{t-1})}{P(h_1,..,h_{t-1})} \cdot \frac{P(h_t|z=1)}{P(h_t|h_1,..,h_{t-1})} \\ & = P_{t-1} \cdot P(h_t|z=1)/K, \text{where } K = P(h_t|h_1,..,h_{t-1}); \\ \overline{P_t} & = \overline{P_{t-1}} \cdot P(h_t|z=-1)/K \end{array} \tag{1}$$

since $P_t + \overline{P_t} = 1$, we have

$$P_t = \frac{P_0 \prod P(h_t|Z)}{P_0 \prod P(h_t|Z) + \overline{P_0} \prod P(h_t|\overline{Z})} \tag{2}$$

This finishes bayesian integration. What remains is how to estimate P_0 and $P(h_t|z)$ from training set. In the rest of the section, we will show, by minimizing the upper bound B of training error $E \equiv E(sign(F(x) \neq y))$, we can obtain the optimal value.

2.2 Upper bound

The most common training objective is to minimize the expected training error $E = E(sign(F(x) \neq y)) = \sum sign(F(x_i) \neq y_i)$. Since binary function $sign$ is discontinuous at decision boundary, it is inconvenient to analyze E directly. Instead, following common practice in the boosting literature, we turn to provide a continuous upper bound B_t.

$$\begin{array}{ll} E_t \leq B_t & \equiv \sum (P_{ti}/\overline{P}_{ti})^{-y_i} = \sum (\frac{P_{t-1,i} \cdot P(h_{ti}|z_i=1)}{\overline{P}_{t-1,i} \cdot P(h_{ti}|z_i=-1)})^{-y_i} \\ & = \frac{P_t^2}{P_t^1} \sum_{h_{ti}=1, y_i=1} \frac{\overline{P}_{t-i,i}}{P_{t-1,i}} + \frac{P_t^4}{P_t^3} \sum_{h_{ti}=-1, y_i=1} \frac{\overline{P}_{t-i,i}}{P_{t-1,i}} \\ & + \frac{P_t^1}{P_t^2} \sum_{h_{ti}=1, y_i=-1} \frac{P_{t-1,i}}{\overline{P}_{t-i,i}} + \frac{P_t^3}{P_t^4} \sum_{h_{ti}=-1, y_i=-1} \frac{P_{t-1,i}}{\overline{P}_{t-i,i}} \end{array}$$

Define

$$\begin{array}{ll} O^1 = \sum_{h_{ti}=1, y_i=1} \frac{\overline{P}_{t-1,i}}{P_{t-1,i}}, & O^2 = \sum_{h_{ti}=1, y_i=-1} \frac{P_{t-1,i}}{\overline{P}_{t-1,i}}, \\ O^3 = \sum_{h_{ti}=-1, y_i=1} \frac{\overline{P}_{t-1,i}}{P_{t-1,i}}, & O^4 = \sum_{h_{ti}=-1, y_i=-1} \frac{P_{t-1,i}}{\overline{P}_{t-1,i}} \end{array}$$

and define $\mu = \frac{P_t^1}{P_t^2}, \nu = \frac{P_t^3}{P_t^4}$. Substituting into B_t, we have

$$B_t = O^1/\mu + O^3/\nu + \mu O^2 + \nu O^4 \tag{3}$$

Solving for optimal value, we have

$$\mu = \sqrt{\frac{O^1}{O^2}},\ \nu = \sqrt{\frac{O^3}{O^4}} \tag{4}$$

Given:$(x_i, y_i), ..., (x_m, y_m)$; $x_i \in \chi, y_i \in \{-1, +1\}$
Initialize $P_{0i} = P_0 = \frac{\sqrt{M^+}}{\sqrt{M^+}+\sqrt{M^-}}$, $\overline{P}_{0i} = 1 - P_{0i}$
For t=1,...,T
- Train weak hypothesis $h_t : \chi \to R$ recommend to use $D_{ti} = (P_{ti}/\overline{P}_{ti})^{-y_i}$ as training set weight, which is identical to Adaboost
- Compute $P(h_t|z)$ according to Eq.5
- Compute
$P_{ti} = \frac{P_{t-1,i}P(h_{ti}|Z)}{P_{t-1,i}P(h_{ti}|Z)+\overline{P}_{t-1,i}P(h_{ti}|\overline{Z})}$
$\overline{P}_{ti} = 1 - P_{ti}$

Output the final hypothesis:
$$F(x_i) = sign(P_{ti} > \overline{P}_{ti})$$

Figure 2: Bayesian Integration Algorithm

Given:$(x_i, y_i), ..., (x_m, y_m)$; $x_i \in \chi, y_i \in \{-1, +1\}$
Initialize $P_{0i} = \frac{\sqrt{M^+}}{\sqrt{M^+}+\sqrt{M^-}}$, $\overline{P}_{0i} = 1 - P_{0i}$
For t=1,...,T
- Compute $D_{ti} = (P_{t-1,i}/\overline{P}_{t-1,i})^{-y_i}$
- Split $\{(x_i, y_i, D_{ti})\}$ into 2 sets by q_{t-1} use each set to train weak classifiers h_t^1 and h_t^2 which jointly form h_t
- Compute CPD $P(h_t|q_t, z)$ according to Eq.9
- Compute
$P_{ti} = \frac{P_{t-1,i}P(h_{ti}|q_{ti},Z)}{P_{t-1,i}P(h_{ti}|q_{ti},Z)+\overline{P}_{t-1,i}P(h_{ti}|q_{ti},\overline{Z})}$
$\overline{P}_{ti} = 1 - P_{ti}$

Output the final hypothesis:
$$F_t(x_i) = sign(P_{ti} > \overline{P}_{ti})$$

Figure 3: ADABOOST.BAYES Algorithm

Since $P_t^1 + P_t^3 = 1$ and $P_t^2 + P_t^4 = 1$, we have $\frac{P_t^1}{1-P_t^4} = \sqrt{\frac{O^1}{O^2}}$, $\frac{1-P_t^1}{P_t^4} = \sqrt{\frac{O^3}{O^4}}$. Solve it, we get the optimal position to minimize B_t,

$$P_t^1 = \frac{\sqrt{O^1O^3}-\sqrt{O^1O^4}}{\sqrt{O^2O^3}-\sqrt{O^1O^4}},\ P_t^4 = \frac{\sqrt{O^2O^4}-\sqrt{O^1O^4}}{\sqrt{O^2O^3}-\sqrt{O^1O^4}} \tag{5}$$

Similar process will provide $P_0 = \frac{\sqrt{M^+}}{\sqrt{M^+}+\sqrt{M^-}}$ where M^- is the number of negative training examples, M^+ is the number of positive examples.

Just like selecting appropriate α in boosting, this optimal position takes a greedy step to decrease the upper bound B_t of training error in bayesian integration. Through the following theorem, we will see, B_t will decrease at exponential speed.

Theorem 1: Assuming the above notation in the paper, if P_t^1, P_t^4 are assigned as Eq.5, the upper bound of error shall decrease exponentially:

$$B_t/B_{t-1} < \sqrt{1-r^2}, r \equiv min(\frac{|O^1 - O^2|}{O^1 + O^2}, \frac{|O^3 - O^4|}{O^3 + O^4})$$

Proof:From Eq.3, notice $B_{t-1} = O^1 + O^2 + O^3 + O^4$, we have $\frac{B_t}{B_{t-1}} = \frac{2\sqrt{O^1O^2}+2\sqrt{O^3O^4}}{O^1+O^2+O^3+O^4}$. It is easy to verify

$$\frac{2\sqrt{O^1O^2}}{O^1+O^2} \le \sqrt{1-r^2},\ \frac{2\sqrt{O^3O^4}}{O^3+O^4} \le \sqrt{1-r^2} \tag{6}$$

Therefore,$B_t \le B_{t-1}\sqrt{1-r^2}$. (End of Proof)

This concludes the optimal parameter for Bayesian Integration. As stated in Theorem 1, there is no explicit constraint on how to update the training set weight D_{ti} to obtain the new h_{ti}. But to have the exponential decrease, we need $r > 0$. That is equivalent to have $O_t^1 \neq O_t^2$ and $O_t^3 \neq O_t^4$. To guarantee the above condition, it is recommended to train the weak classifier with $D_{ti} = (P_{ti}/\overline{P}_{ti})^{-y_i}$. The algorithm is summarized in Fig.2.

2.3 Representing ADABOOST

Bayesian integration defines a set of algorithms by permitting different $P_0 and P(h_t|z)$. As shown in following theorem, ADABOOST is just one of them.

Theorem 2 Assuming the above terminology, if

$$\begin{cases} P_0 = 1/M \\ P_t^4 = P_t^1 = \frac{\sqrt{O^1+O^4}}{\sqrt{O^1+O^4}+\sqrt{O^2+O^3}} \end{cases} \tag{7}$$

then, for the same set of weak hypothesis h_{ti}, the decision function for ADABOOST and Bayesian Integration are identical, i.e., $F(x) = H(x)$. Then by assuming the same set of weight $D_{ti} = (P_{ti}/\overline{P}_{ti})^{-y_i}$, Bayesian Integration is identical to Adaboost.

If we take Theorem 2 as a bridge between ADABOOST and bayesian network, it is easy to see ADABOOST as well as any Bayesian Integration algorithm constructs a naive bayesian tree. On the other hand, Theorem 2 enables us to use BNT method to improve boosting. That is what we propose in the next section.

3 Dependency between classifiers: ADABOOST.BAYES

ADABOOST is a linear combination of weak classifiers. When the weak classifier is too simple, such linear combination can't produce a correct boundary. For example, Xor pattern, e.g. Fig.4(a), can't be recognized by a decision stump learner and boosting such weak learner won't improve much. To solve the problem, it needs 2nd order information. Through Theorem 2 we can easily map the problem into the BNT domain. There, it is obvious that the inability of ADABOOST is caused by the false assumption that, given the root nodes, all the evidence nodes are independent of each other.

The solution in the BNT domain is natural and straight forward. For each new evidence node, add both the hidden root node and the history evidence node as parents, illustrated in Fig.1.(b). To construct such a BNT, we need to specify 2^t parameters as conditional probability table for evidence node h_t. It inevitably leads to overfitting. Instead, we use only one node to summarize the dependency information. For every new classifier, we can generate one new evidence node h_t and a new parent node q_t, then link the root node z and q_t to h_t, as illustrated in Fig.1.(c).

To summarize the history dependency on h_{t+1}, there is a very convenient variable that already exists, $P(z|h_1, ..., h_t)$, which represents the most recent belief. We can simply let $q_{t+1} \equiv P(z|h_1, ..., h_t)$. But, to keep the model as simple as possible, we further constrain q_{t+1} to be a binary evidence node:$q_{t+1} \equiv sign(P(z|h_1, .., h_t) > 0.5) \equiv F_t(x)$. The new conditional probability table for h_t now becomes $P(h_t|q_t, z) \equiv [P_t^j, j = 1...8]$, where $P_t^j \equiv P(h_t = -sign((j-1)\&4)|q_t = -sign((j-1)\&2), z = -sign((q-1)\&1))$.

$F_t(x)$ is the last step decision on x. There is no cost to generate such an evidence node. And because it is an evidence node, there is no need to specify the prior $P(q_t)$. All we have to specify is $P(h_t|z, q_t)$. Once the model is prescribed, the iterative updating rule can be obtained from standard BNT inference. Similar to Eq.2, we have

$$P_t = \frac{P_0 \prod P(h_t|Z,q_t)}{P_0 \prod P(h_t|Z,q_t) + \overline{P}_0 \prod P(h_t|\overline{Z},q_t)} \tag{8}$$

Following the same optimization process as the previous section, we can obtain the optimal value for $P(h_t|z, q_t)$ by minimizing the upper bound B_t, which gives us the following results:

$$\begin{cases} P_t^1 = \frac{\sqrt{O^1O^3} - \sqrt{O^1O^4}}{\sqrt{O^2O^3} - \sqrt{O^1O^4}}, & P_t^4 = \frac{\sqrt{O^2O^4} - \sqrt{O^1O^4}}{\sqrt{O^2O^3} - \sqrt{O^1O^4}} \\ P_t^5 = \frac{\sqrt{O^5O^7} - \sqrt{O^5O^8}}{\sqrt{O^6O^7} - \sqrt{O^5O^8}}, & P_t^8 = \frac{\sqrt{O^6O^8} - \sqrt{O^5O^8}}{\sqrt{O^6O^7} - \sqrt{O^5O^8}} \end{cases} \tag{9}$$

$$O^j \equiv \sum_{\substack{q_i = -sign((j-1)\&4) \\ h_{ti} = -sign((j-1)\&2) \\ y_i = -sign((j-1)\&1)}} (\frac{P_{t-1,i}}{\overline{P}_{t-1,i}})^{-y_i}, \text{where } j = 1, ..., 8$$

ADABOOST.BAYES can also be explained in the boosting framework. Conceptually, we cut the training set into four history dependent sections by $q_t \in \{-1, 1\}$ and $h_t \in \{-1, 1\}$. Each section has different distribution of positive and negative examples. We retrieve one α for each section. With history information q_t and four α, we generate a non-linear model.

To leverage the ability of describing the details in four sections, we can further modify boosting iteration by training two classifiers for each of the two sections separated by q_{t-1}. Then combine the classifiers into h_t and resume the iteration. That constitutes ADABOOST.BAYES, as illustrated in Fig.3. By splitting the training set, each weak classifier are facing a local therefore simpler hidden pattern.

4 Avoid overfit: ADABOOST.SOFTBAYES

Preliminary experiments show ADABOOST.BAYES has a stronger ability to combine the weak classifiers, but it is easier to become overfit. One dominant reason is, ADABOOST.BAYES emphasizes the local features as it splits the training set to find simpler and easier pattern to learn. When there are not enough samples in the section, ADABOOST.BAYES tends to lock onto sampled pattern that exists only in training set.

Comparatively, ADABOOST is more robust to avoid overfit but weaker to deal with complex boundary while ADABOOST.BAYES is just the opposite. As we have shown, algorithms of boosting family can be expressed in either ADABOOST-like formality or BNT-like formality. Their iterations are both independent and interchangeable. Therefore, we suggest to switch between the two algorithms and hope to have virtues on both sides.

Therefore, when there are enough positive and negative samples within each of the sections generated by q_t, we assume ADABOOST.BAYES, otherwise we fall back to ADABOOST. The ratio of positive/negative samples in each section to total training set serves as the threshold to switch back and fore. We empirically set it as 5%. We call this algorithm ADABOOST.SOFTBAYES.

5 Experiments

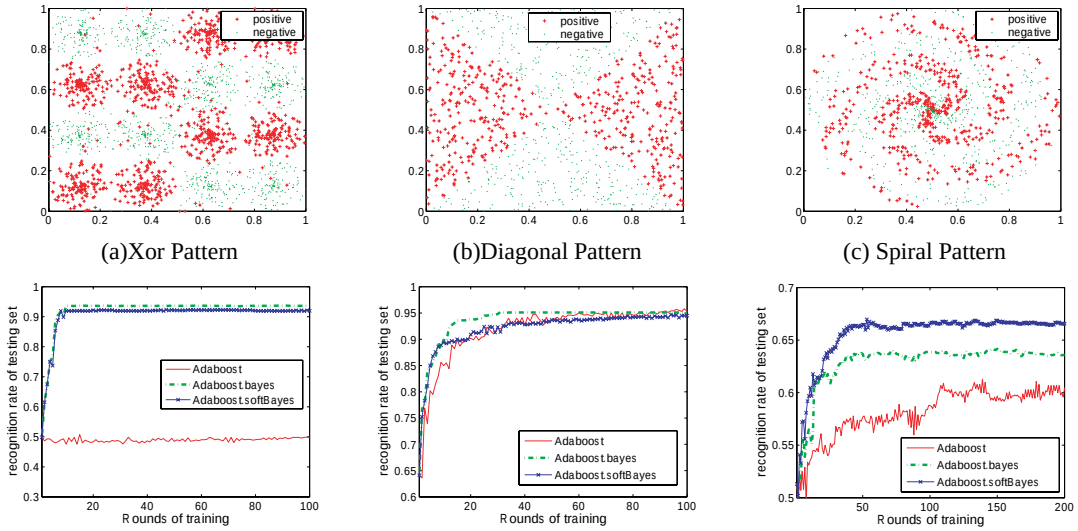
We perform several experiments on synthetic data as well as UCI data set to compare the ability of ADABOOST, ADABOOST.BAYES and ADABOOST.SOFTBAYES. To emphasize the boosting ability, we only use the decision stump as the simple learner in all experiments.

First, we run all 3 algorithms on the Xor data Fig.4(a). We randomly sample 1000 positive points + 1000 negative points as training set and another 500 positive+500 negative points as testing set (All synthetic data sets are constructed in the same manner with randomly sampled equal positive and negative data points). As shown in Fig.4(d), ADABOOST falls far away from perfect as it lacks the ability to construct complex decision boundary from simple boundary of stump classifier. ADABOOST.BAYES and ADABOOST.SOFTBAYES both perform well and approximately the same.

Second, we construct a data set that is easy to become overfit for stump learner: a diagonal pattern Fig.4(b). As shown in Fig.4(e), ADABOOST performs the best when the decision boundary can be approximated by the weak classifiers. ADABOOST.SOFTBAYES has approximately the same performance as ADABOOST.

Third, we construct a spiral pattern Fig.4(c) that needs more than one step of history information for stump classifier, which means if we take ADABOOST.BAYES as automatically assemble 2 weak classifiers as a cascade, then the boundary of this pattern is even more complex, just like Xor is beyond ADABOOST. As shown in Fig.4(f), ADABOOST.BAYES is still superior to ADABOOST even though the actual boundary is beyond its ability. Meanwhile, ADABOOST.SOFTBAYES still sticks to the highest possible performance.

Finally, we run all 3 algorithms on real data sets from UCI repository. The performance is compared by the highest possible recognition rate on the testing set of the cross-validation tests which are shown in Table.1. As we can see, ADABOOST.SOFTBAYES not only has a superior recognition rate but also tends to require fewer training circles.



(a)Xor Pattern (b)Diagonal Pattern (c) Spiral Pattern

(d)Recognition Rate of Xor Pattern (e)Recognition Rate of Diagonal Pattern (f)Recognition Rate of Spiral Pattern

Figure 4: Synthesized data and corresponding recognition rate.

	Recognition Rate			Iteration level for best performance		
UCI data set	**ADABOOST**	**ADABOOST .bayes**	**ADABOOST .softBayes**	**ADABOOST**	**ADABOOST .bayes**	**ADABOOST .softBayes**
Pima	0.759	0.768	0.768	8	3	3
Wdbc	0.953	0.954	0.957	132	9	32
Sonar	0.857	0.848	0.862	32	4	75
Ionosphere	0.918	0.920	0.944	3	2	12
German	0.778	0.776	0.758	59	33	5
Adult	0.854	0.852	0.853	93	99	83

Table 1: testing result on UCI data sets

6 Conclusion

Through Bayesian Integration, two major machine learning frameworks, ADABOOST and BNT, are connected together. We hope this linkage will enable a better understanding of the rich literature in both societies. As an example of leveraging the convenience of thinking in both sides, we derive a new algorithm ADABOOST.BAYES as an improvement of ADABOOST to take dependency between classifiers into account. Compared with ADABOOST, ADABOOST.BAYES requires lower number of training iterations, but overfit the model faster. To avoid the overfit problem, we go further to construct a switching schema ADABOOST.SOFTBAYES which integrates ADABOOST and ADABOOST.BAYES together. Experiments show ADABOOST.SOFTBAYES possesses a stronger classification ability.

Acknowledgements

The authors thank James Rehg, Jianxin Wu, Pei Yin and Howard Zhou for valuable discussion.

References

[1] Yoav Freund, Robert E. Schapire, "A decision-theoretic generalization of on-line learning and application to boosting", Journal of Computer and System Sciences, 55(1), pp119-139,1997

[2] J. Friedman and T. Hastie and R. Tibshirani, "Additive logistic regression: a statistical view of boosting", The Annals of Statistics, vol.28, No.2, pp337-386,2000

[3] Jonathan S. Yedidia, William T. Freeman and Yair Weiss , "Bethe free energy, Kikuchi approximations and belief propagation algorithms", NIPS'2000

[4] Robert E.Shapire, Yoram Singer, "Improved Boosting Algorithms Using Confidence-rated Predictions", Machine Learning, vol.37, No.3, pp297-336, 1999.

[5] Antonio Torralba, Kevin Murphy and William Freeman, "Sharing features: efficient boosting procedures for multiclass object detection", CVPR'04, vol'2, pp762-769

[6] Robert E.Schapire, Yoav Freund, Peter Barlett, "Boosting the margin: A new explanation for the effectiveness of voting methods", Annal of Statistics., 26(5), pp1651-1986, 1998.

[7] Nir Friedman, Daphne Koller, "Being Bayesian about Network Structure: A Bayesian Approach to Structure Discovery in Bayesian Networks", Machine Learning, 50, pp95-126, 2003.

[8] David Heckerman, "A tutorial on learning with Bayesian networks", In "Learning in graphical models", pp301-354,1999

[9] Stan Z. Li, Zhenqiu Zhang, Heung-Yeung Shum, Hongjiang Zhang, "FloatBoost learning for classification", NIPS 2003

[10] Ce Liu and Heung-Yeung Shum, "Kullback-leibler boosting", CVPR'2003, Vol.I, pp587-594

[11] Paul Viola and Michael Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features", CVPR'2001, Vol.I, pp511-518

Appendix

Theorem 2 Assuming the terminology in the paper, if

$$\begin{cases} P_0 = 1/M \\ P_t^4 = P_t^1 = \frac{\sqrt{O^1+O^4}}{\sqrt{O^1+O^4}+\sqrt{O^2+O^3}} \end{cases} \tag{10}$$

then, for the same set of weak hypothesis h_{ti}, the decision function for ADABOOST and Bayesian Integration are identical: $F(x) = H(x)$

Proof: Define

$$U_t = log(\tfrac{P_t}{\overline{P}_t}) \quad V_t = \textstyle\sum(\alpha_t h_t)$$

then we have:

$$\begin{array}{lll} F(x) & = sign(P_t > \overline{P}_t) & = sign(log(\frac{P_t}{\overline{P}_t})) \quad = sign(U_t) \\ H(x) & = sign(\sum(\alpha_t h_{ti})) & = sign(V_t) \end{array}$$

To prove the theorem, all we have to do is to show $U_t = V_t$. Following the proof of recursion induction, we have $U_0 = V_0 = 0$.

Assume $U_{t-1} = V_{t-i}$. From Eq.1,

$$U_t \quad = U_{t-1} + log\tfrac{P(h_t|z=1)}{P(h_t|z=-1)} \quad = V_{t-1} + log\tfrac{P(h_t|z=1)}{P(h_t|z=-1)} \quad = V_{t-1} + \tfrac{h_t}{2} log\tfrac{O^1+O^4}{O^2+O^3} \tag{11}$$

By unravelling the update rule ADABOOST, we have

$$D_{t+1,i} \quad = \frac{exp(-\sum_t \alpha_t y_i h_{ti})}{m\prod_t Z_t} \quad = \frac{exp(-y_i V_{ti})}{Z}$$

$$\begin{array}{lll} \alpha & = \frac{1}{2} log(\sum_{y_i=h_{ti}} w_{ti} / \sum_{y_i \neq h_{ti}} w_{ti}) & = \frac{1}{2} log(\sum_{y_i=h_{ti}} exp(-y_i V_{ti}) / \sum_{y_i \neq h_{ti}} exp(-y_i V_{ti})) \\ & = \frac{1}{2} log((O^1+O^4)/(O^2+O^3)) & \end{array}$$

Substitute α into Eq.11, we have $U_t = V_{t-1} + \alpha h_t = V_t$.

(End of Proof)