Machine Learning: Foundations **Fall semester, 2011**

Summary On MultiClass boosting

Lecturer: Yishay Mansour *Summary: Vera V.*

1.1 Introduction

Boosting refers to a general techniques of combining rules of thumb, or week classifiers, to form highly accurate combined classifiers. Variety of the learning algorithms(weak learner) can be employed to discover, making the algorithm widely applicable. The theory of boosting is well developed for the binary classification problem. The exact requirements on the weak learners are known: any algorithm that predicts better than random on any distribution over the training set is said to satisfy the weak learning assumption. Further, the boosting algorithms that minimize loss in the efficient manner have been designed(for example, AdaBoost). However, many classification problems involve more than two labels. Unfortunately, for such multiclass problems, a complete theoretical understanding of boosting is lacking. The extensions of the binary weak-learning condition to multiclass do not work. Requiring less error than random guessing turns out to be too weak for multiclass boosting. Requiring more than 50% accuracy even when the number of the labels is much larger than two is too stringent. The most common approach is to reduce multiclass problem to the binary classification. However, it is not clear whether the weak-learning conditions implicitly assumed by such reductions are the most appropriate. [1].
In the summary we review a set of previously developed techniques to solve multiclass problems, as well as AdaBoost.M1, AdaBoost.MH and AdaBoost.MR algorithms. We review a new developed framework and algorithms from [1] and compare their performance results.

1.2 Margin-based learning algorithms

A binary margin − based learning algorithm takes as input binary labeled training examples $(x_1, (y_1)), ..., (x_m, (y_m))$ where the instances $x_i \in \chi$ and the label $y_i \in \{-1, +1\}$. We use a learning algorithm to generate a real-valued function , also called *hypothesis* $f : \chi \to \mathbb{R}$ where f belongs to some *hypothesis space* H. The margin of an example $(x_i, (y_i))$ with respect to f is $yf(x)$, means that margin is positive when the *hypothesis* f classifies the input vector correctly. The training error is defined as follows:

$$\frac{1}{m}\Sigma_{i=1}^{m}[y_i f(x_i) \leq 0]$$

In our learning algorithms we tend to minimize some other nonnegative *loss function* of the margin, that is, to minimize

$$\frac{1}{m}\Sigma_{i=1}^{m}L(y_i f(x_i))$$

where $L : \mathbb{R} \to [0, \infty]$. Such kind of the loss functions help to prove the main theorem on the effectiveness of the output coding methods for multiclass problems.

1.3 Mulitclass, Multi-label Clasification Problems

Let Y be a finite set of of labels or classes, and $k = |Y|$. In the traditionaly classification settings, each eample $x \in X$ is assigned to a single class $y \in Y$. In the multi-label case, each $x \in X$ may belong to multiple lables in Y. The single case a clearly a special case of the multi-label clasification problem. One of the possibility is to seek a hypothesis $h\colon X \to Y$ which predicts just one of the labels of the given example. The goal then is to find such h which minimizes the probability that $h(x) \notin Y$ for a new example x. Formally, the error of the a hypothesis h with respect to the distribution D over examples (x, Y) is defined as follows:

$$one - err_D(h) = Pr_{(x,Y)\sim D}[h(x) \neq Y]$$

However, there exist other loss measures, namely Hamming loss and ranking loss that will be described later in this summary.

1.4 Previous work

1.4.1 AdaBoost

We will first show that the multiplicative weight-update Littlestone-Warmuth rule can be adapted to this model, yielding bounds that are slightly weaker in some cases, but applicable

to a con-siderably more general class of learning problems. An on-line allocation algorithm calles $Hedge(\beta)$ is a direct generalization of Littlestone and Warmuth's weighted majority algorithm. Given a set of expert predictions, it minimize mistakes over time.

Algorithm Hedge(β):

Parameters: $\beta \in [0,1]$

initial weight vector $w^1 \in [0,1]^N$ with $\Sigma_{i=1}^N w_i^1 = 1$

number of trials T

Do for t=1, 2, ..., T

1. Choose allocation $p^t = \frac{w^t}{\Sigma_{i=1}^N w_i^t = 1}$
2. Receive loss vector$l^t \in [0,1]^N$ from environment.
3. Suffer loss $p^t \cdot l^t$.
4. Set the new weights vector to be $w_i^{t+1} = w_i^t(\beta)^{l_i^t}$

Hedge Analysis:

Does not perform too much worse than best strategy:

$$L_{hedge(\beta)} \leq (-ln(w^1) - L_i ln(\beta)) \cdot Z$$
$$Z = 1/(1-\beta)$$

This online algorithm can be modified to boost the performance of weak learning algorithms. Boosting: We very briefly review the PAC learning model. If we have n classifiers, possibly looking at the problem from different perspectives, we would like to optimally combine them. For example: We have a collection of rules of thumb for predicting horse races, and we want to weight them. Given labeled data $\langle x, c(x)\rangle$, where c is the target concept, $c : x \to \{0,1\}$ $c \in$ C, the concept class.

For a new boosting algorithm,derived from the on-line allocation algorithm we described, the accuracy of the final hypothesis depends on the accuracy of all the hypotheses returned by WeakLearn, and so is able to more fully exploit the power of the weak learning algorithm (WeakLearn: For parameter ϵ , hypothesis has error $\epsilon \geq (0.5 - \gamma), \gamma > 0$).

AdaBoost Algorithm:

Input:

- Sequence of N labeled examples
- Distribution D over the N examples
- Weak learning algorithm (called WeakLearn)
- Number of iterations T

Initialize: $w^1 = D$

For t =1...T

1.Form probability distribution p from w

2.Call WeakLearn with distribution p

3.Calculate error $\epsilon_t = \sum_{i=1}^N p_i |h_t(x_i) y_i|$

4.Set $\beta_t = \epsilon_t/(1-\epsilon_t)$
5.Multiplicatively adjust weights (w) by $\beta_t^{1-|h_t(x_i)y_i|}$
Output:
• 1, if $\sum_{t=1}^{T} log(1/\beta_t)h_t(x) \geq 0.5 \sum_{t=1}^{T} log(1/\beta_t)$
• 0, otherwise
Computes a weighted average.
Analysis:
There is a dual relationship with Hedge
• Strategies$\longleftrightarrow$ Examples
• Trials $\longleftrightarrow$ Weak hypotheses
• Hedge increases weight for successful strategies, AdaBoost increases weight for difficult examples.
• AdaBoost has dynamic β.
The Bounds of AdaBoost:
•$\epsilon \leq 2^t \prod t = 1^T \sqrt{(\epsilon_t(1-epsilon_t))}$
• Previous bounds depended on maximum error of weakest hypothesis (weak link syndrome)
• AdaBoost takes advantage of gains from best hypotheses
Boosting for multi-class and regression problems:
Now we will present two extensions of the boosting algorithm to multi-class prediction problems in which each example belongs to one of several possible classes (rather than just two) $k \geq 3$ output labels, i.e. $Y = 1, ..., k$.
Error: Probability of incorrect prediction.
These Two algorithms are:
1. **AdaBoost.M1** More direct
2. **AdaBoost.M2** Somewhat complex constraints on weak learners

1.4.2 AdaBoost and multiclass reduction to the binary classification[2]

Output coding for multiclass problems

A lot of machine learning problems are faced with a multiclass learning problem in which each label y is chosen from a set Y of cardinality $k > 2$. We will describe here three approached of handling k-class problem using a binary margin-based algorithm as follows:

1.$One - against - all\ technique$ creates one binary problem for each of the k classes. That is, for each $r \in Y$, we apply the given margin-based learning algorithm to a binary problem in which all examples labeled $y = r$ are considered positive examples and all other examples are considered negative examples. We then end up with k hypotheses that somehow must

be combined.
2.*All − pairs technique* is to use the given binary learning algorithm to distinguish each pair of classes. Thus, for each distinct pair $r_1, r_2 \in Y$, we run the learning algorithm on a binary problem in which examples labeled $y = r_1$ are considered positive, and those labeled $y = r_2$ are considered negative. All other examples are ignored. Again, all $\binom{k}{2}$ hypotheses must be combined.
3.*Error correcting output codes* (*ECOC*) *technique* associates each class $r \in Y$ with a row of a "coding matrix" $M \in \{-1, +1\}^{kxl}$ for some l. The binary learning algorithm is then run once for each column of the matrix on the induced binary problem in which the label of each example labeled y is mapped to $M(y, s)$. Finally, we get l hypotheses f_s. Given an example x, we then predict the label y for which row y of matrix M is "closest" to $(f_1(x), ..., f_l(x))$.

In the paper the autors unified generalization of all three of these methods applicable to any margin-based learning algorithm. A given "coding matrix", $M \in \{-1, 0, +1\}^{kxl}$, is considered. Then, algorithm A is trained with labeled data of the form $(x_i, M(y_i, s))$ for all example i. Finally, A creates a hypothesis $f_s : \chi \to \mathbb{R}$, where $s \in \{1, ..., l\}$. And attempts to minimize the loss L on the induced binary problem. Thus, the *average binary loss* of the hypotheses f_s on the diven training set with respect to M and L as follows:

$$\frac{1}{ml}\Sigma_{i=1}^{m}\Sigma_{s=1}^{l}L(M(y_i, s)f_s(x_i)).$$

Hamming decoding

Let $M(r)$ denote row r of M and let $f(x)$ to be the vector of of predictions on an instance x:

$$f(x) = f_1(x), ..., f_l(x)$$

Hamming decoding technique predicts the label r that minimizes $d(M(r), f(x))$ for some distance d, where the distance defined as follows:

$$d_H(M(r), f(x)) = \Sigma_{s=1}^{l}(\frac{1-sign(M(y_i,s)f_s(x_i))}{2})$$

For an instance x and a matrix M, the predicted label $\hat{y} \in \{1, ...k\}$ is therefore

$$\hat{y} = argmin_r d_H(M(r), f(x))$$

Loss-based decoding

The idea is to choose the label r that is most consistent with the predictions $f_s(x)$ in the sense that, if example x were labeled r , the total loss on example (x, r) would be minimized over choices of $r \in Y$. Formally, this means that our distance measure is the total loss on a proposed example (x, r):

$$d_L(M(r), f(x)) = L(M(y_i, s)f_s(x_i))$$

The predicted label $\hat{y} \in \{1, ...k\}$ is therefore

$$\hat{y} = argmin_r d_L(M(r), f(x))$$

Conclusions

By comparing $Hamming$ and $Loss-based\ decoding$, we could say that the first one ignores entirely the magnitude of the predictions, which can often be an indication of a level of "confidence", as well as the relevant loss function L. The second method takes this potentially useful information into account. It is hardly clear that the weak-learning conditions implicitly assumed by such reductions are the most appropriate. Straightforward extensions of the binary weak-learning condition to multiclass do not work. Requiring less error than random guessing on every distribution, as in the binary case, turns out to be too weak for boosting to be possible when there are more than two labels. On the other hand, requiring more than 50% accuracy even when the number of labels is much larger than two is too stringent.

1.4.3 AdaBoost.M1 algorithm

This algorithm requires each classifier to have error less than 50 percent (stronger requirement than binary case). The algorithm is similar to regular AdaBoost algorithm except *three* things:
1. The Error is 1 if $h_t(x_i) \neq y_i$.
2. Algorithm outputs vector of length k with values between 0 and 1.
3. The algorithm is good just with error ≤ 0.5.

Algorithm:
Input : sequence of N examples $(x_1, y_1), ..., (x_N, y_N)$ with labels $y_1 \in Y = 1, ..., k$, distribution D over the examples weak lerning algorithm WeakLearn integer T specif. number of iterations.
Initialize the weight vector : $w_i^1 = d(i)$ for $i = 1, ..., N$
Do for t=1,...,T:
1.Set $p_t = \frac{w_t}{\sum_{i=1}^N w_i^t}$
2.Call WeakLearn, providing it with the distribution p^t; get back a hypothesis $h_t : X \to Y$
3.Calculate the error of $h_t : \epsilon_t = \sum_{i=1}^N p_i^t[h_t(x_i) \neq y_i]$ If $\epsilon_t > 0.5$, the set $T = t - 1$ and abort loop.
4.If $\beta_t = \epsilon_t/(1 - \epsilon_t)$
5.Set the new weights vector to be $w_i^{t+1} = w_i^t \beta_t^{1-[h_t(x_i)\neq y_i]}$
Output the following hypothesis:
$h_f(x) = argmax_{y \in Y} \sum_t (log\frac{1}{beta_t})[h_t(x) = y]$

Critics on Adaboost.M1:
Adaboost.M1 is adequate when the base learner is strong enough to achieve reasonably high accuracy, even on the hard distributions created by AdaBoost. However, this method fails if the base learner cannot achieve at least 50 percent accuracy when run on these hard distributions.

1.4.4 AdaBoost.M2 algorithm

This Algorith is more expressiveand more complex constraints on weak hypotheses.
AdaBoost.M2 requires the weak learner to output, rather than simply a label, an array of confidences associated with each possible labeling ofa n example. Additionally, the weak learner of A daBoost.M2 must have a training function which takes an array of penalties p, where pi corresponds to the penalty that the weak learner will suffer by misclassifying the given example as an i. (The greater the weak learners total penalties, the less it contributes to the final boosted learner.) Its benefit is that it Allows contributions from hypotheses with accuracy less than 0.5. The algorithm Defines idea of Pseudo-Loss:

$$ploss_q(h,i) = 0.5(1h(x_i,y_i) + \Sigma_{y\neq y_i} q(i,y)h(x_i,y)$$

It replaces straightforward loss of AdaBoost.M1 and makes use of information in entire hypothesis vector, not just prediction. The pseudo-loss of each weak hypothesis must be better than chance.
Algorithm:
$Input$: sequence of N examples$((x_1,y_1)...,(x_N,y_N))$ with labels $y_i \in Y = [1,...,k]$, distribution D over the N examples, weak learning algorithm WeakLearn, integer T specifying number of iterations.
$Extrainit$: $w_t^i = D(i)/(k-1)$
For each iteration $t = 1,...,T$:
1. $W_i^t = \Sigma_{y\neq y_i} w_{i,y}^t$

$q_t(i,y) = w_{i,y}^t / W_i^t$

$D_t(i) = W_i^t / \Sigma_{i=1}^N W_i^t$

2. WeakLearn gets D as well as q
3. Calculate ϵ_t as shown above
4. $\beta_t = \epsilon_t/(1-\epsilon_t)$
5. $w_{i,y}^{t+1} = w_{i,y}^t \cdot \beta_t^{(0.5)(1+h_t(x_i,y_i)h_t(x_i,y))}$
The error Bounds: $\epsilon \leq (k1)2^T \prod_{t=1}^T \sqrt{(\epsilon_t(1-\epsilon_t))}$, Where ϵ is traditional error and the ϵ_t are

pseudo-losses.
Conclusion:
AdaBoost.M1 and AdaBoost.M2 are equivalent for binary classification problems and differ only in their handling of problems with more than two classes.
For more than 2 classes, the main disadvantage of AdaBoost.M1 is that it is unable to handle weak hypotheses with error greater than 0.5 while AdaBoost. M2 can.
The advantage of the algorithm is that it primarily performs by directly selecting the discriminant features with the minimum training error. Since each weak learner only represents a partial data distribution, the final boosted classifier may not be restricted to the global data distribution.
The disadvantage is that this algorithm is very time consuming,because on each round it constructs classifiers according to the same number of classes.
Another disadvantage is that the Algorithm handles multiclass problems but not MultiLabel problems (it can't handle the case that a single example may belong to any number of classes).

1.4.5 AdaBoost.MH and AdaBoost.MR algorithms

Hamming loss

Suppose our goal is to predict all and only all of the correct labels. Thus, we have to find such hypothesis $h\colon X \to 2^Y$ with respect to the distribution D, where the loss, called Hamming loss, is defined as follows:

$$hloss_D(h) = \frac{1}{k} E_{(x,Y)\sim D}[|h(x)\Delta Y|]$$

AdaBoost modification according to the Hamming loss measure

In order to moidify AdaBoost algorithm according to the Hamming loss measure, we can think of a set of labels Y for some example (x, Y) as specifying k binary labels. Similary, $h(x)$ can be viewed a k binary predictions. For $Y \subseteq \mathcal{Y}$, let us define $Y[l]$ for $l \in Y$ to be

$$Y_i[l] = +1 \text{ if } l \in Y_i, \text{ otherwise } Y_i[l] = -1$$

The main idea of the reduction is to replace each example (x_i, Y_i) by k examples $((x_i, l), Y_i[l])$ for $l \in Y_i$. As a result we have a new boosting algorithm, called AdaBoost.MH as follows:

Given: $(x_1, Y_1), ..., (x_m, Y_m)$, where $x_i \in X$ and $Y_i \in Y$
Initialize $D_1(i, l) = 1/mk$
For $t = 1, ..., T$:
1.Train weak learner using distribution D_t
2.Get weak hypothesis $h_t\colon X \times Y \to \mathbb{R}$

3.Choose $\alpha_t \in \mathbb{R}$
4.Update: $D_{t+1}(i,l) = \frac{D_t(i,l)exp(-\alpha_t Y_t[l]h_t(x_i,l))}{Z_t}$,
where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis: $H(x,l) = sign(\sum_{t=1}^{T} \alpha_t h_t(x,l))$
It was proved by authors (see Theorem 3[3]) that the following bound holds for the Hamming loss of the final hypothesis H on the training data: $hloss(H) \leq \Pi_{t=1}^{T} Z_t$, where $Z_t = \sum_{t,i} D_t(i,l)exp(-\alpha_t Y_i[l]h_t(x_i,l))$.
It was also proved(see Theorem 4[3]) that AdaBoost.MH can be applied to single-label multiclass classification problems and the resulting bound is at most $k\Pi_{t=1}^{T} Z_t$.
But, as we already said, this approach ignores the confidence with which each label was included or not included in $H(x)$.

Ranking loss

Here our goal is to find a hypothesis which *ranks* the labels with the hope that the correct labels will receive the highest ranks. The approach described here is closely related to one used by Freund et al. (1998) for using boosting for more general ranking problems.
To be formal, we now seek a hypothesis of the form $f: \mathcal{X}\times\mathcal{Y} \to \mathbb{R}$ with the interpretation that, for a given instance x, the labels in $\mathcal{Y}$ should be ordered according to $f(x,*)$.That is, a label l_1 is considered to be ranked higher than l_2 if $f(x,l_1) > f(x,l_2)$. We only considered the *crucial pairs* of labels (l_1,l_2) for which $l_1 \notin Y$ and $l_2 \in Y$. Such f misorders l_1, l_2 if $f(x,l_1) \leq f(x,l_2)$. And we seek for the such f that minimize the number of misorderings. Formally, our goal is to minimize the expected fraction of crucial pairs which are misordered, that is a follows:

$$E_{(x,Y)\sim D[\frac{|(l_1,l_2)\in(\mathcal{Y}-Y)\times Y:\ f(x,l_2)\leq f(x,l_1)|}{|Y||\mathcal{Y}-Y|}]}.$$

We denote this measure $rloss_D(f)$, where Y is not empty and never equal to $\mathcal{Y}$.
As a result we have a new boosting algorithm, called AdaBoost.MR as follows:

Given: $(x_1,Y_1),...,(x_m,Y_m)$, where $x_i \in \mathcal{X}$ and $Y_i \subset \mathcal{Y}$
Initialize $D_1(i,l_1,l_2) = \frac{1}{m|Y_i||\mathcal{Y}-Y_i|}$ if $l_1 \notin Y_i$ and $l_2 \in Y_i$, otherwise 0.
For $t = 1,...,T$:
1.Train weak learner using distribution D_t
2.Get weak hypothesis $h_t: X\times Y \to \mathbb{R}$
3.Choose $\alpha_t \in \mathbb{R}$
4.Update: $D_{t+1}(i,l_1,l_2) = \frac{D_t(i,l_1,l_2)exp(-0.5\alpha_t(h_t(x_i,l_1)-h_t(x_i,l_2)))}{Z_t}$,

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis: $f(x,l) = \sum_{t=1}^{T} \alpha_t h_t(x,l)$

Note, the update rule decreases the weight $D_t(i,l_1,l_2)$ if h_t gives a correct ranking, and increases this weight otherwise. Similary to the AdaBoost.MH algorithm, the authors proved(see Theorem 6[3]) that the the following bound holds for the ranking loss of on the training data:
$rloss(f) \leq \Pi_{t=1}^{T} Z_t$, where $Z_t = \sum_{t,l_1,l_2} D_t(i,l_1,l_2) exp(-0.5\alpha_t(h_t(x_i,l_1) - h_t(x_i,l_2)))$.
As in the previous section, we can use the ranking loss method for minimizing one-error, and therefore also for single-label problems. Indeed, Freund and Schapires (1997) "pseudoloss"-based algorithm AdaBoost.M2 is a special case of the use of ranking loss in which all data are single-labeled. And it was proved(see Theorem 7[3]) that with respect to any distribution D over observation (x,Y) we have that the resulting bound on one-error is at most $(k-1)rloss_D(f)$.

Conclusions on AdaBoost.MH and AdaBoost.MR

As we will see later in the work[1] done by Indraneel Mukherjee. Robert E. Schapire, the weak-learning condition implicitly used by AdaBoost.MH [3], which is based on a one-against-all reduction to binary, turns out to be strictly stronger than necessary for boostability, i.e. certain boostable spaces will fail to satisfy the conditions due to underfitting. On the other hand, the condition implicit to AdaBoost.MR(also called AdaBoost.M2) turns out to be exactly necessary and sufficient for boostability.

1.5 Theory of Multiclass boosting[1]

As we already mentioned in our summary, the previous works were lack of understanding the correct weak-learning conditions for the multiclass settings. These considerations may significantly impact learning and generalization, because knowing the correct weak-learning conditions might allow the use of simpler weak classifiers, which in turn can help prevent overfitting. Furthermore, boosting algorithms that more efficiently and effectively minimize training error may prevent underfitting, which can also be important.
In the paper the authors created a broad and general framework for multiclass boosting, that formalizes the interaction between the boosting algorithm and the weak-learner. The correct weak-learning(minimal weak-learning) conditions that are neither too weak nor too strong were precisely defined. Boosting algorithm that uses minimal weak-learning condition were defined, which drives down the error efficiently. Experimental results that complement the theory were presented.

Framework for multiclass boosting

Notation:
1.Matrices will be denoted by capital letters like M.
2.Vectors will be denoted by small letters like v.
3.Entries of a matrix and vector will be denoted as M(i,j) or v(i).
4.ith row of a matrix will be denoted as M(i).
5.Inner product of two vectors u,v will be denoted $< u, v >$.
6.The Frobenius inner product of two matrices will be denoted as $M \bullet M^{'}$.
7.The indicator function is denoted by $I[\cdot]$.
8.The distribution over the set {1,...,k} will be denoted by Δ {1,...,k}.

Let us consider the boosting as a game between two players: Booster and Weak-Learner. Booster creates a distribution in each round $t = 1, ..., T$ and Weak-Learner returns a weak classifier achieving more that 50% accuracy on that distribution.
For the multiclass game we have:
At each round $t = 1, ..., T$
1.Booster creates a cost-matrix $C_t \in \mathbb{R}^{m \times k}$(and not a distribution) and passes to the Weak-Learner, where $C(i, l)$-the cost to classify example x_i as l.
2.Weak-Learner returns $h_t \colon X \to \{1, ..., k\}$ so that $C(i, h) = \sum_{i=1}^{m} C_t(i, h_t(x_i))$ is "small" enough, this is the cost of weak classifier.
3.Booster computes weight α_t based on the amount of the cost was incurred in the round t.

The final prediction will be as follows:

$$H(x) = argmax_l f_T(x, l), \text{ where } f_T(x, l) = \sum_{t=1}^{T} I[h_t(x) = l]\alpha_t.$$

Note, that for all i, $C(i, y_i) \leq C(i, \bar{y}_i)$, which means that labeling correctly should be less costly than labeling incorrectly. On the other hand, h requires less cost than predicting randomly(here we observe the binary settings): $\sum_i C(i, h(x_i)) \leq \sum_i (0.5 - 0.5\gamma)C(i, \bar{y}_i) + (0.5 + 0.5\gamma)C(i, y_i)$.
Thus, we can rewrite this condition in the following manner:

$$\forall C \in C^{bin}, \exists h \in H \colon C \bullet (1_h - U_{\gamma}^{bin}) \leq 0, \tag{1.1}$$

where all $C^{bin} \in \mathbb{R}^{m \times 2}$ satisfy $C(i, 1) \leq C(i, 2)$, and all $U_{\gamma}^{bin} \in \mathbb{R}^{m \times 2}$ contain rows of the form $(0.5 + 0.5\gamma, 0.5 - 0.5\gamma)$. Thus, under these assumptions, it could be shown that Weak-Learner searching space H satisfies the binary weak-learning condition. And, as we will see later, a vast variety of weak-learning conditions(used in AdaBoost.M1, AdaBoost.MH, AdaBoost.MR) could be captured using the following formulation:

$$\forall C \in \mathcal{C}, \exists h \in H \colon C \bullet (1_h - B) \leq 0, \tag{1.2}$$

where $\mathcal{C} \subseteq \mathbb{R}^{m \times k}$ is a misclassification cost matrix and $B \in \mathbb{R}^{m \times k}$ is a weight matrix for each misclassification, where $B(i,l)$ is a probability with which B predicts lalel l on i. The cost of the baseline B is denoted as $Cost(C,B) = \sum_i \sum_l C(i,l)B(i,l)$. The cost of the hypothesis h is denoted as $Cost(C,h) = \sum_i C(i,h(x_i))$. And (1.2) could be viewed as requirement that is $Cost(C,h) \leq Cost(C,B)$. If (1.2) holds, we say that a weak classifier space H satisfies the condition (C,B).

Necessary and sufficient weak-learning conditions

Let us denote a random player as a baseline predictor $B \in \mathbb{R}^{m \times k}$, where $B(i) \in \Delta\{1,...k\}$ and $i \in \{1,...,m\}$. The space of $edge-over-random$ baselines will be denoted by $B^{eor}_{\gamma} \subseteq \mathbb{R}^{m \times k}$, where for any $B \in B^{eor}_{\gamma}$ and for every example $i\colon \forall l \neq 1, B(i,1) \geq B(i,l) + \gamma$. Note, that there are a lot of $edge - over - random$ baselines for $k \geq 2$, since there is a lot of possibilities of random guessing. For binary settings problem we have a unique B. Let us denote a multiclass extension of the cost matrix C^{bin} by C^{eor}, where the rows of C^{eor} should come from some set $\{c \in \mathbb{R}^k\colon \forall l, c(1) \leq c(l))\}$. The, for any baseline $B \in B^{eor}_{\gamma}$, an $egde - over - random$ weak-learning codition (C^{bin}, B) will be induced. Note, that $C \bullet B$ is the expected cost of the $egde - over - random$ baseline B on C.
Let us define a collection H of weak-classifiers to be *boostable*, if there exists a distribution λ that linearly separates the data.
Recall the Minimax Theorem:

Theorem 1.1 *Let C, D be non-empty closed convex subsets of $\mathbb{R}^m, \mathbb{R}^n$ respectively, and let K be a continuous finite concave-convex function on C D. If either C or D is bounded, one has*

$$min_{v \in D} max_{u \in C} K(u,v) = max_{u \in C} min_{v \in D} K(u,v).$$

Theorem 1.2 *If a weak classifier space H satisfies a weak-learning condition (C^{eor}, B), for some $B \in B^{eor}_{\gamma}$, then H is boostable.*

Proof: Applying the minimax theorem yields

$$0 \geq max_{C \in C^{eor}} min_{h \in H} C \bullet (1_h - B) = min_{\lambda \in \Delta(H)} min_{C \in C^{eor}} C \bullet (H_{\lambda} - B), \tag{1.3}$$

where $H_{\lambda} \triangleq \sum_{h \in H} \lambda(h) 1_h$, and the first inequality is due to (1.2). Let λ^* be a minimizer of the min-max expression. Unless the first entry of each-row of $(H_{\lambda^*} - B)$ is the largest, the right hand side of the min-max expression can be made arbitrarily large by choosing $C \in C^{eor}$ appropriately. For example, if in some row i, the $j_0 th$ element is strictly larger than the first element, by choosing

$$C(i,j) = \begin{cases} 1 & \text{if } j = 1, \\ -1 & \text{if } j = j_0, \\ 0 & \text{otherwise.} \end{cases}$$

we get a matrix in C^{eor} which causes for each i, $H_\lambda(i,1) - B(i,1) - H_\lambda(i,j_0) + B(i,j_0) > 0$, which is contradiction to (1.3) Hence, the convex combination of the weak classifiers, obtained by choosing each weak classifier with weight given by λ^*, perfectly classifies the training data, in fact with a margin γ.

■

Theorem 1.3 *For every boostable weak classifier space H, there exists a $\gamma > 0$ and $B \in B_\gamma^{eor}$ such that H satisfies the weak-learning condition (C^{eor}, B).*

Proof: H is boostable implies there exists some distribution λ^* such that

$$\forall j \neq 1, i \colon H_{\lambda^*}(i,1) > H_{\lambda^*}(i,j)$$

Let γ to be the minimum of the above expression over all possible (i,j), and let $B = H_{\lambda^*}$. Then, $B \in B_\gamma^{eor}$, and

$$max_{C \in C^{eor}} min_{h \in H} C \bullet (1_h - B) \leq min_{\lambda \in \Delta(H)} min_{C \in C^{eor}} C \bullet (H_\lambda - B) \leq max_{C \in C^{eor}} C \bullet (H_{\lambda^*} - B) = 0, . \tag{1.4}$$

where the equality follows since by definition $B - H_{\lambda^*} = 0$ ■

But, still we have to choose the right weak-learning condition. However, there are theoretical examples showing each condition in the family is too strong, which means that there are examples that contains boostable spaces, but the condition (1.2) does not hold. The weakening of the condition could be done in the following way:

$$\forall C \in \mathcal{C}, \exists h \in H \colon C \bullet 1_h \leq max_{B \in B_\gamma^{eor}} C \bullet B. \tag{1.5}$$

Now we want to prove that the weak-learning conditions used in AdaBoost.MH and AdaBoost.MR match the condition family defined in (1.2).

Theorem 1.4 *The weak-learning conditions used in AdaBoost.MH is equivalent to (C^{MH}, B^{MH}).*

Proof: Recall that for any matrix with non-negative entries $d(i,l)$ the weak-hypothesis should achieve $0.5 + \gamma$ accuracy

$$\sum_i (I[h(x_i) \neq y_i] d(i,y_i) + \sum_{l \neq y_i} I[h(x_i) = l] d(i,l)) \leq (0.5 - \gamma) \sum_i \sum_l d(i,l) \tag{1.6}$$

This is because our weak-learner has to perform better than random guessing.This can be rewritten as

$$\sum_i (-I[h(x_i) = y_i] d(i,y_i) + \sum_{l \neq y_i} I[h(x_i) = l] d(i,l)) \leq \sum_i ((0.5 - \gamma) \sum_{l \leq y_i} d(i,l) - (0.5 + \gamma) d(i,y_i)) \tag{1.7}$$

Now our goal is to show that Adaboost.MH learning condition can be rewritten in (1.2) form. By mapping

$$C(i,l) = \begin{cases} d(i,l) & \text{if } l \neq y_i, \\ -d(i,l) & \text{if } l = y_i, \end{cases}$$

the above can be rewritten in the following way

$$\forall C \in \mathbb{R}^{m \times k} s.t. \{C(i,y_i) \leq 0, C(i,l) \geq 0, l \neq y_i\},$$

$$\exists h \in H\colon \sum_i C(i,h(x_i)) \leq \sum_i ((0.5+\gamma)C(i,y_i) + (0.5-\gamma)\sum_{l \neq y_i} C(i,l)) = \sum_i < C(i), B_\gamma^{MH}(i) > .$$

By assuming, WLOG, $\forall i\colon y_i = 1$ the above condition is the same as

$$\forall C \in C^{MH}, \exists h \in H\colon C \bullet (1_h - B_\gamma^{MH}) \leq 0,$$

i.e. the (C^{MH}, B_γ^{MH}) weak-learning condition. ■

Theorem 1.5 *The weak-learning conditions used in AdaBoost.MR is equivalent to* (C^{MR}, B^{MR}).

Proof: Recall that for any matrix with non-negative cost-vectors $\{d(i,l)\}, l \neq y_i$ the weak-hypothesis should s.t. the following

$$\sum_i \sum_{l \neq y_i} (I[h(x_i) = l] - I[h(x_i) = y_i]) d(i,l) \leq -2\gamma \sum_i \sum_{l \neq y_i} d(i,l) \tag{1.8}$$

i.e.

$$\sum_i (-I[h(x_i) = y_i] \sum_{l \neq y_i} d(i,l) + \sum_{l \neq y_i} I[h(x_i) = l]) d(i,l)) \leq -2\gamma \sum_i \sum_{l \neq y_i} d(i,l) \tag{1.9}$$

By mapping

$$C(i,l) = \begin{cases} d(i,l) & \text{if } l \neq y_i, \\ -\sum_{l \neq y_i} d(i,l) & \text{if } l = y_i, \end{cases}$$

the above can be rewritten in the following way

$$\forall C \in \mathbb{R}^{m \times k} s.t. \{C(i,l) \geq 0 for l \neq y_i \, C(i,y_i) = -\sum_{l \neq y_i} d(i,l)\},$$

$$\exists h \in H\colon \sum_i C(i,h(x_i)) \leq -\gamma \sum_i (2 \sum_{l \neq y_i} d(i,l)) = -\gamma \sum_i (-C(i,y_i) + \sum_{l \neq y_i} C(i,l)) = \sum_i < C(i), B_\gamma^{MR}(i) >.$$

By assuming, WLOG, $\forall i\colon y_i = 1$ the above condition is the same as

$$\forall C \in C^{MR}, \exists h \in H\colon C \bullet (1_h - B_\gamma^{MR}) \leq 0,$$

i.e. the (C^{MR}, B_γ^{MR}) weak-learning condition. ■
Next, we show that weak-learning condition from (1.5) is equivalent to the one used by AdaBoost.MR. Recall, AdaBoost.MR weak-learning condition is given by (C^{MR}, B_γ^{MR}), where the rows of C^{MR} sum up to zero and B_γ^{MR} is the matrix that has γ on the first column and $-\gamma$ on all other columns.

Theorem 1.6 *A weak classifier space H s.t. AdaBoost.MR's weak-learning condition* (C^{MR}, B_γ^{MR}) *iff s.t. (1.5).Moreover, this condition is equivalent to being boostable.*

Proof:
=> see Theorem 1.5.
<=Assume that a weak classifier space H s.t. (1.5). We want to prove that such H s.t. AdaBoost.MR's weak-learning condition (C^{MR}, B_γ^{MR}). Suppose $B \in B_{2\gamma}^{eor}$. Note, that $C^{MR} \subset C^{eor}$. We will show the following holds

$$\forall C \in C^{MR}, B \in B_{2\gamma}^{eor}\colon C \bullet 1_h \leq max_{B \in B_{2\gamma}^{eor}} C \bullet B \leq C \bullet B_\gamma^{MR},$$

Since $B \in B_{2\gamma}^{eor}$, this implies $B - B_\gamma^{MR}$ has the largest value in its first column. Then, for any $C \in C^{MR}$,

$$C \bullet (B - B_\gamma^{eor}) = \sum_i \sum_j C(i,j)(B - B_\gamma^{MR})(i,j) = \sum_i [\sum_{j=2}^k C(i,j)(B - B_\gamma^{MR})(i,j) + C(i,1)(B - B_\gamma^{MR})(i,1)] = \sum_i [\sum_{j=2}^k C(i,j)(B - B_\gamma^{MR})(i,j) - \sum_{j=2}^k C(i,j)(B - B_\gamma^{MR})(i,1)] = \sum_i \sum_{j=2}^k C(i,j)[(B - B_\gamma^{MR})(i,j) - (B - B_\gamma^{MR})(i,1)].$$

Since $C(i,j) > 0$ for $j > 1$, and $(B - B_\gamma^{MR})(i,j) - (B - B_\gamma^{MR})(i,1) \leq 0$ we have the our result. ■
Now, it remains to show that if $\exists \gamma > 0\colon H$ s.t. (C^{MR}, B_γ^{MR}) then H is boostable.
Proof: We can use minimax theorem, as we already see previously,

$$0 \geq max_{C \in C^{MR}} min_{h \in H} C \bullet (1_h - B_\gamma^{MR}) = min_{\lambda \in \Delta(H)} max_{C \in C^{MR}} C \bullet (H_\lambda - B_\gamma^{MR})$$

The left side of the expression due to H s.t. (C^{MR}, B_γ^{MR}).
For any i_0 and $l_0 \neq 1$, the following cost-matrix C s.t. $C \in C^{MR}$,

$$C(i,l) = \begin{cases} 0 & \text{if } i \neq i_0 \text{ and } l \notin \{1, l_0\} \\ 1 & \text{if } i = i_0 \text{ and } l = l_0 \\ -1 & \text{if } i = i_0 \text{ and } l = 1 \end{cases}$$

Let λ^* to minimize the right side of the expression. Then $C \bullet (H_{\lambda^*} - B_\gamma^{MR}) \leq 0$ implies $H_{\lambda^*}(i_0, 1) - H_{\lambda^*}(i_0, l_0) \geq 2\gamma > 0$. Since this is true for all i_0 and $l_0 \neq 1$, H is boostable. ■

Unfortunately, AdaBoost.MH has too strong demands on the weak classifier. Let us look at the following example:
a space H predicts correctly $(1/k + \gamma)m$ of m examples and $\gamma > 0$. We can calculate that using U_γ and cost matrix that contains -1 in the first column, otherwise 0, gives us $C \bullet (1_h - U_\gamma) = -m(1/k + \gamma) + m((1 - \gamma)/k + \gamma) = -m\gamma/k \leq 0$. On the other hand, based on Theorem 1.4, AdaBoost.MH weak learning condition is (C^{MH}, B_γ^{MH}). And for B_γ^{MH} contains $(1/2+\gamma, 1/2-\gamma, ..., 1/2-\gamma)$, we have $C \bullet (1_h - B_\gamma) = -m(1/k+\gamma)+m(1/2+\gamma) = m(1/2-1/k) > 0$ for $k > 2$. Hence, H fails to satisfy AdaBoost.MH weak learning condition. Weak-learning conditions of AdaBoost.MH and AdaBoost.M1 are same in our framework[see 7-S.6]

Algorithms

In this section the OS algorithm will be described by analyzing the boosting games ,which employs edge-over-random weak-learning conditions.

Drifting games and optimal strategy employed to drive down training error[6]
There are two players in the game called the *shepherd* and the *adversary*. The game is played in T rounds using m chips. On each round, the shepherd specifies a *weight vector* $w_i^t \in \mathbb{R}^n$ for each *chip* i. The direction of this vector, $v_i^t = w_i^t/||w_i^t||_p$, specifies a desired of drift, while the length of the vector $||w_i^t||_p$ specifies the relative importance of moving the chip in the desired direction. In response, the adversary chooses a drift vector z_i^t for each chip i. The adversary is constrained to choose each z_i^t from a fixed set $B \subseteq \mathbb{R}^n$. Moreover, the z_i^t's must satisfy

$$\sum_i < w_i^t, z_i^t > \geq \delta \sum_i ||w_i^t||_p \tag{1.10}$$

or equivalently

$$\frac{\sum_i ||w_i^t||_p v_i^t z_i^t}{\sum_i ||w_i^t||_p} \geq \delta, \tag{1.11}$$

where $\delta > 0$ is a fixed parameter of the game. The position of chip i at time t, denoted by s_i^t , is simply the sum of the drifts of that chip up to that point of time. Thus, $s_i^1 = 0$, $s_i^{t+1} = s_i^t + z_i^t$. And our goal is to minimize

$$\frac{1}{m} \sum_i L(s_i^{T+1}), \tag{1.12}$$

after T rounds, where L is the $lossfunction$.
Drifting games and their relation to boosting[6]

We can recast boosting as just described as a special-case drifting game as follows:
1.The chips $i = 1, ..., m$ are identified with examples $\{(x_1, y_1), .., (x_m, y_m)\}$.
2.The game is one-dimensional so that $n = 1$.
3.The drift of a chip $z_i^t \in B$ is +1 if example i is correctly classified by h_t and -1 otherwise, where $B = \{+1, -1\}$.
4.By assuming $w_i^t \geq 0$, the distribution $D_i^t = \frac{w_i^t}{\sum_i w_i^t}$.
Then, for $\gamma > 0$ the condition $Pr_{i\sim D_t}[y_i \neq h_t(x_i)] \leq 0.5 - \gamma$ equalent to

$$\sum_i [\frac{w_i^t}{\sum_i w_i^t} \frac{1 - z_i^t}{2}] \leq 0.5 - \gamma$$

or

$$\sum_i < w_i^t, z_i^t > \geq 2\gamma \sum_i w_i^t \tag{1.13}$$

By letting $\delta = 2\gamma$, we can see that (1.13) is tha same as (1.10). Finally, if we define L as follows:

$$L(s) = \begin{cases} 1 & \text{if } s \leq 0 \\ 0 & \text{if } s > 0, \end{cases}$$

we have that the error of the final hypothesis is equal to (1.12). The following algorithms are derived from the very general drifting games described above. And optimal payoffs can be very well approximated by certain potential function as was done in [6].
The OS algorithm:
1.Fix the number of rounds T.
2.Fix edge-over-random weak-learning condition (C, B).
3.Fix α_t to be 1, for simplicity.
4.Let $f_T(x, l)$ to be $\sum_{t=1}^{T} I[h_t(x) = l]\alpha_t$.
5.For $\{h_1, ..., h_T\} \in H^{all}$ the optimum Booster payoff can be written as

$$min_{C_1 \in C} max_{C_1 \bullet (1_{h_1} - B) \leq 0} ... min_{C_T \in C} max_{C_T \bullet (1_{h_T} - B) \leq 0} (1/m) \sum_i L(f_T(x_i, 1), ..., f_T(x_i, k)).$$

Here $L \colon \mathbb{R}^k \to \mathbb{R}$ is a loss, k is the the amount of different labels, and we select L to be a *proper* function,which means increasing in the weights on the correct label, otherwise decreasing.
In order to analyze the optimal payoff let us define for any $b \in \mathbb{R}$ the *potential function* $\phi_t^b \colon \mathbb{R}^k \to \mathbb{R}$ as follows:

$$\phi_0^b = L; \phi_t^b(s) = min_{c \in \mathbb{R}^k; \forall l: c(1) \leq c(l)} max_{p \in \Delta\{1,..,k\}} \{E_{l\sim p}[\phi_{t-1}^b(s + e_l)] \colon E_{l\sim p}[c(l)] \leq < b, c > \}, \tag{1.14}$$

where $e_l \in \mathbb{R}^k$ is the unit vector whose lth coordinate is 1; otherwise zero. $\phi_t^b(s_t)$ estimates of whether an example x will be misclassified, where $s_t(l) = \sum_{t'=1}^{t-1} I[h_{t'}(x) = l]$. Using these

function, Schapire[6] proposed a Booster strategy, which in each round t, constructs C - a matrix s.t.(1.14) and b=B(i). The the results from [6] were extended and as follows

Theorem 1.7 *Suppose the weak-learning condition is given by (C, B), if Booster employs the OS algorithm, then the average potential of the states $(1/m)\sum_i \phi_t^{B(i)}(s(i))$ never increases in any round.*

ϕ_t^b for distribution $b \in \{1, ..., k\}$ can be computed using homogeneous random walk $R_b^t(s)$ technique. Recall random walk $R_b^t(s)$ - the random position of a particle after t time steps, that starts at location $x \in \mathbb{R}^k$, and in each step moves in direction e_l with probability $b(l)$.

Theorem 1.8 *If L is proper, and $b \in \Delta\{1, ..., k\}$ s.t. $\forall l$: $b(1) \geq b(l)$, then $\phi_t^b(s) = E[L(R_b^t(s))]$. And the vector achieving the minimum in the right hand side of (1.14) is given by $c(l) = \phi_{t-1}^b(s + e_l)$.*

Lemma 1.9 *If L as exponential loss, namely, $L(s) = e^{\eta_2(s_2-s_1)}+...+e^{\eta_k(s_k-s_1)}$, where $\eta_l > 0$ then the solution in (1.8) will be $\phi_t^b(s) = \sum_{l=2}^k (a_l)^t e^{\eta_l(s_l-s_1)}$, where $a_l = 1-(b_1+b_l)+e^{\eta_l}b_l + e^{-\eta_1}b_1$.*

Proof: The proof will be by induction.
1.**Basis**: for $t = 0$, $\phi_0^b(s) = L(s)$
2.**Induction hypothesis**: we assume that for $t-1$, $\phi_{t-1}^b(s) = \sum_{l=2}^k (a_l)^{t-1} e^{\eta_l(s_l-s_1)}$
3.**Induction step**: $\phi_t^b(s) = E_{l\sim b}[\phi_{t-1}^b(s + e_l)] =$
$b_1\phi_{t-1}^b(s + e_l)+, ..., +b_k\phi_{t-1}^b(s + e_l) =$ {by induction hypothesis}=
$= b_1 \sum_{l=2}^k (a_l)^{t-1} e^{\eta_l(s_l-s_1-1)} +$
$+ (b_2+, ..., +b_k) \sum_{l=2}^k (a_l)^{t-1} e^{\eta_l(s_l-s_1+1)} +$
$+ (b_2+, ..., +b_k) \sum_{l=2}^k (a_l)^{t-1} e^{\eta_l(s_l-s_1)} -$
$- \sum_{l=2}^k b_l (a_l)^{t-1} e^{\eta_l(s_l-s_1)} =$
$= b_1 \sum_{l=2}^k (a_l)^{t-1} e^{\eta_l(s_l-s_1-1)} +$
$+ (b_2+, ..., +b_k) \sum_{l=2}^k (a_l)^{t-1} e^{\eta_l(s_l-s_1+1)} +$
$+ (1-b_1) \sum_{l=2}^k (a_l)^{t-1} e^{\eta_l(s_l-s_1)} -$
$- \sum_{l=2}^k b_l (a_l)^{t-1} e^{\eta_l(s_l-s_1)} =$
$= \sum_{l=2}^k (a_l)^{t-1}(1 - b_1 - b_l + e^{\eta_l}b_l + e^{-\eta_l}b_1) e^{\eta_l(s_l-s_1)} = \sum_{l=2}^k (a_l)^t e^{\eta_l(s_l-s_1)}$

■

When the condition is (C^{eor}, U_γ), the relevant potential will be $\phi_t(s) = k(\gamma, \eta)^t \sum_{l=2}^k e^{\eta(s_l-s_1)}$, where $k(\gamma, \eta) = 1 + \frac{1-\gamma}{k}(e^\eta + e^{-\eta} - 2) - (1 - e^{-\eta})\gamma$. The cost matrix will be updated as follows

$$C(i,j) = \begin{cases} (e^\eta - 1)e^{\eta(s_l-s_1)} & \text{if } l > 1, \\ (e^{-\eta} - 1)\sum_{j=2}^k e^{\eta(s_j-s_1)} & \text{if } l = 1, \end{cases} \tag{1.15}$$

Thus, from Theorem 1.7 we have the average loss $(1/m)\sum_i L(s_t(i))$ changes by factor $k(\gamma,\eta)$ at every t. Hence, the final loss will be at most $(k-1)k(\gamma,\eta)^T$.
The problem is that $\gamma_1,...,\gamma_T$ could not be fixed, since if their small OS algorithm could not be able to make enough progress in the initial rounds, otherwise Weak-Learner fails to meat the weak-learning condition in latter rounds. There are two approaches to fix this problem:
1.Fixed sequence of edges
In this case we a prescribed sequence of edges $\gamma_1,...,\gamma_T$ the weak-learning condition $(C^{eor}, U^{\gamma t})$ in each round t is different. And the cost matrix C is updated as follows:

$$C(i,j) = \begin{cases} (e^{\alpha}-1)e^{f_{t-1}(i,j)-f_{t-1}(i,1)} & \text{if } l>1, \\ (e^{-\alpha}-1)\sum_{j=2}^{k} e^{f_{t-1}(i,j)-f_{t-1}(i,1)} & \text{if } l=1, \end{cases} \tag{1.16}$$

In this case the final loss will be at most $(k-1)\Pi_t k(\gamma_t,\alpha_t)$.
2.Adaptive sequence of edges
Adaptive algorithm using non adversarial Weak-Learner works as follows:
at each round $t=1,...,T$
1.Create cost matrix C_t
2.Receive weak classifier h_t with edge δ_t
3.Compute weight α_t and update $f_t = f_{t-1} + \alpha_t h_t$
where $\alpha_t = 0.5ln(\frac{1+\delta_t}{1-\delta_t})$ and cost matrix

$$C_{t+1}(i,l) = \begin{cases} e^{f_{t-1}(i,j)-f_{t-1}(i,1)} & \text{if } l>1, \\ -\sum_{j=2}^{k} e^{f_{t-1}(i,j)-f_{t-1}(i,1)} & \text{if } l=1, \end{cases} \tag{1.17}$$

After T rounds the loss, and hence the error of the algorithm will be at most $(k-1)\Pi_t\sqrt{1-\delta_t^2} \leq (k-1)exp{-}0.5\sum_t \delta_t^2$.

1.6 Experiments

Adaptive algorithm using minimal weak-learning condition was run and compared with AdaBoost.M1 and AdaBoost.MH. Weak classifiers - bounded size decision trees were taken. In the first experiment the test error were measured vs. maximum tree size allowed as weak classifiers for 500 rounds running. It could be seen from the Figure 1(a) [1] that the test error of the new algorithm was lower that the error of AdaBoost.M1 and AdaBoost.MH. On the other hand, the authors investigated how fast the test error drops with rounds, when the maximum tree size was 5. It could be seen from the Figure 1(b)[1] that AdaBoost.M1 and AdaBoost.MH drive down the error for a few rounds. But since boosting keeps creating harder cost-matrices, the small-tree learning algorithms are no longer able to meet excessive requirements of AdaBoost.M1 and AdaBoost.MH. However, the new algorithm easily meet by weak learner.

1.7 A new Framework definition for multiclass multilabel settings

Indraneel Mukherjee. Robert E. Schapire in their work defined the framework for multiclass single label problem settings, by assuming that the correct label $y_i = 1$. Let us look at the following example:
We have a machine learning problem, where examples in the traning set describe newspaper properties, such words, as "*shares*","*interest*","*Obama*", etc. Clearly, these newspapers belong to more than two classes simultaneously. It could be two different classes, such as "*politics*" and "*economics*".
We will try to define a new framework for such kinds of machine learning problems base on the existing framework for single label setting as follows:
1.Let us define a collection of $H_{ML}(ML$ means multiclass) of weak classifiers to be eligible for boosting, or simply boostable, if there exists a distribution λ on this space that separates the data set(with multiple lable choice) , $(x_i, Y_i), ..., (x_m, Y_m)$, where $\forall i = 1, ..., m \colon x_i \in X$, and $Y_i \in Y$ as follows:

$$\forall i, \forall l \in Y_i \colon argmax_{l \in \{1,...,k\}} \sum_{h \in H_{ML}} \lambda(h) I[h(x_i) = l] = 1, \tag{1.18}$$

2.Let us define a set of cost marices $C^{eor}_{ML} \subseteq \mathbb{R}^{m \times k}$ as follows:
the rows of the cost matrix $C \in C^{eor}_{ML}$ come from set $\{c \in \mathbb{R}^k \colon \forall l \in Y_i, \forall d \in Y - Y_i, c(l) \leq c(d)\}$, where $\forall l, m \in Y_i$, such that $l \neq m, c(l) = c(m)$, which means that we give the same cost for the correct labels.
3.Let us define a set of baselines $B^{eor}_{\gamma,ML} \subseteq \mathbb{R}^{m \times k}$, and call it $edge - over - random$ baseline set for ML settings. Any baseline matrix $B_{ML} \in B^{eor}_{\gamma,ML}$, whose rows are distribution over the labels, $B_{ML}(i) \in \Delta\{1, ..., k\}$, is more likely to predict the correct label than an incorrect one, while the distribution over correct labels, come from $Y_i \in Y$ will be uniformly distributed. Formally, $\forall i, \forall d \in Y - Y_i, \forall l \in Y_i, B(i, l) \geq B(i, d) + \gamma$ and $\forall i, \forall l, m \in Y_i$, such that $l \neq m, B(i, l) = B(i, m)$.
Thus, for every baseline we introduce the condition (C^{eor}_{ML}, B_{ML}), which we call an $edge - over - random$ multilabel weak-learning condition (MLWLC).
We say that a weak classifier space H_{ML} s.t. the (C^{eor}_{ML}, B_{ML}) if

$$\forall C \in C^{eor}_{ML}, \exists h \in H_{ML}, C \bullet (1_h - B_{ML}) \leq 0, means, \forall i, Cost(C(i), h_i) \leq Cost(C(i), B(i)) \tag{1.19}$$

Further, WLOG, we assumed that in each row first $|Y_i|$ elements are labeled correcly, means get value 1; otherwise 0.

Theorem 1.10 *If a weak classifier space H_{ML} s.t. a MLWLC (C^{eor}_{ML}, B_{ML}) for some $B_{ML} \in$*

$B^{eor}_{\gamma,ML}$, *then* H_{ML} *is boostable.*

Proof: Assuming, by contradiction, H_{ML} is boostable, but does not s.t. a MLWLC (C^{eor}_{ML}, B_{ML}). =>from (1.19) we have

$$\forall h \in H_{ML}, \forall C \in C^{eor}_{ML}, C \bullet (1_h - B_{ML}) > 0. \tag{1.20}$$

From assumption, H_{ML} is boostable. Then, exists some $\lambda^* \in \Delta(H_{ML})$ such that

$$\forall j = 1, ..., |Y_i|, \forall d = |Y_i| + 1, ..., k, H(i, j) > H(i, d) \tag{1.21}$$

As previously done, we employed the minimax theorem. Let λ^* be the minimum of the above expresstion over all possible (i, j). And let some $B_{ML} = H_{\lambda^*}$. Then, $B_{ML} \in B^{eor}_{\gamma,ML}$. Hence,

$0 < max_{C \in C^{eor}_{ML}} min_{h \in H_{ML}} C \bullet (1_h - B_{ML}) = min_{\lambda \in \Delta H_{ML}} max_{C \in C^{eor}_{ML}} C \bullet (H_\lambda - B_{ML}) = max_{C \in C^{eor}_{ML}} C \bullet (H_{\lambda^*} - B_{ML}) = 0$,where $H_\lambda \triangleq \sum_{h \in H_{ML}} \lambda(h) 1_h$ and equality follows since by definition $H_{\lambda^*} - B_{ML} = 0$. Clearly we got contradiction to (1.20).

■

Theorem 1.11 *For every boostable multiclass weak classifier space* H_{ML}*, there exists a* $\gamma > 0$ *and* $B \in B^{eor}_{\gamma,ML}$ *such that* H_{ML} *satisfies the weak-learning condition* (C^{eor}_{ML}, B)*.*

Proof: H_{ML} is boostable implies there exists some distribution λ^* such that

$$\forall j = 1, ..., |Y_i|, \forall d = |Y_i| + 1, ..., k, H(i, j) > H(i, d) \tag{1.22}$$

The rest of the proof is similar to the proof on the Theorem 1.3.

■

1.8 Summary

1.We saw that AdaBoost.MH is a popular multiclass boosting algorithm has too strong demands on weak classifier. And weak-learning conditions are same in the mentioned frameworks.
2.However, the weak-learning condition implicit to AdaBoost.MR[3], turns out to be exactly necessary and sufficient for boostability. 3.The main advantages to use a completely adversarial Weak-Learner in the OS algorithm permitted to chose weak classifiers without restrictions. Hence, the error guarantees enjoyed in this situation will be universally applicable.

4.We saw the drifting games relation to the boosting, and efficient error and cost matrix calculation based on the defined potential function.
5.Requiring a fixed edge by the OS algorithm is unduly pessimistic or overly optimistic.
6.There are to types of the boosting algorithms were considered
• Optimal efficient algorithm for an fixed EOR, which is Boost-by-majority[Freund' 95], Non-adaptive algorithm, that requires the prior knowledge of γ
• Adaptive algorithm, which assume minimal weak-learning condition, not optimal, but provably very efficient.
7.The theory of multiclass relates to the single-label prediction for the multiclass setting problems, but multi-label settings were not observed yet.
8.We tried to use the same idea as in [1] and extend the framawork for the multiclass multilable settings and prove that if a weak classifier space H_{ML} s.t. a multiclass multilabel weak-learning conditions(MLWLC) (C^{eor}_{ML}, B_{ML}) for some $B_{ML} \in B^{eor}_{\gamma,ML}$, then H_{ML} is boostable, and vice versa.

1.9 References

[1] A Theory of Multiclass Boosting, Indraneel Mukherjee. Robert E. Schapire. Princeton, 2010.

[2] Reducing multiclass to binary a unifying approach for margin classifiers, Erin L. Allwein, Robert E. Schapire,Yoram Singer,The Journal of Machine Learning Research

[3]Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. Machine Learning, 37(3):297336, December 1999.

[4]Robert E. Schapire. The boosting approach to machine learning: An overview. In MSRI Workshop on Nonlinear Estimation and Classification, 2002.

[5] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1):119139, August 1997.

[6]Robert E. Schapire, Drifting Games, Machine Learning, 43(3):265-291,June 2001

[7]Supplement: A Theory of Multiclass Boosting