CSCE 970: Pattern Recognition	**Spring Semester, 1999**
Topic 10: System Evaluation (Performance Estimation)	
Lecturer: Stephen Scott	*Scribe: Iyad Abu-Jeib*

1 Introduction

After building the classifier, we need to asses its performance. This means that we want evaluate this classifier and see how well it will do on new data. In other words, we want to see how efficient is its classifying ability. In order to do that, we need to discuss how to estimate the error (using a finite set of data). One way to do such a thing is by using statistical hypothesis testing, which gives us confidence intervals for the error estimate. We will also use statistical hypothesis testing in order to contrast the performance of two classifiers. On our way to evaluate the classifier's performance, we face the question of how to select the training data and the testing data. We will handle this question from different aspects and discuss different methods of doing that. Another question we want to answer is: what about if the classifier's performance (measured by the error estimate) is not that good? If this is the case, then we may need to go back and review our previous steps. This means that we may need to select/generate new features, and/or build a new classifier (i.e, we need to redesign parts of or all the classification system). Finally, we will discuss and compare two new methods for building classifiers; boosting and bagging. Also, we will discuss several subjects related to the performance of classifiers.

2 Review of the Last Material

In order for us to build any classifier, we need to pass through several steps; feature generation, feature selection, classifier design, and system evaluation. All of those steps are necessary in order to guarantee a good and efficient classifier. Efficiency is, of course, measured in terms of speed and error. Feature selection involves choosing the features which discriminate well between the classes. This step is important for reducing the time complexity of training and/or classifying.

3 Estimating the Classification Error

Once we built the classifier, we need to evaluate it. This evaluation is mainly in terms of error and time complexity. Thus, estimating the error is very essential for system evaluation. Many methods, can be used in order to estimate such error. In the following, I am going to discuss briefly some of those methods.

(1) Estimating Bayesian Error.

If the classifier is Bayesian, then we may use integrals in order to estimate the classification error. The error estimate in this case is called *apparent* error probability. If there are two classes only; ω_1, ω_2, then this apparent error probability is given by

$$\hat{E} = 1 - \int max\{\hat{P}(\omega_1|x)\hat{p}(x), \hat{P}(\omega_2|x)\hat{p}(x)\}dx.$$

If the distributions are both normal with different means μ_1, μ_2, and with the same covariance matrix σ, then $\hat{E}$ is given by

$$\hat{E} = \frac{1}{\sqrt{2\pi}} \int_{\alpha}^{\infty} e^{-x^2/2} dx$$

where $\alpha = \sqrt{(\mu_1 - \mu_2)^t \sigma^{-1} (\mu_1 - \mu_2)}$ (i.e., it is the Mahalanobis distance between μ_1 and μ_2).

Notice that if μ_1, μ_2, and σ are all unknown, then they can be estimated exactly as we did in part (a) of question (1) of homework 1.

(2) Error Counting.

Of course, calculating the general integral might be very difficult, or even impossible. So, even if the classifier is Bayesian, we may need to look for other methods in order to estimate the error. One of those other methods is error-counting. Again, this method can not be used always. Another one is using statistical hypothesis testing. This method will be our next focus. But, before handling this method, let me give a brief introduction to the true error, estimated error, and goodness of error. This is included in what's called *error counting method.* So, let me take a look at this method.

First, assume that we have M classes; ω_i, $i = 1, ..., M$, and a test set S, with N_i feature vectors from each class ω_i, where $\sum_{i=1}^{M} N_i = N$. Assume also that the elements of S are independent and identically distributed (like the training set) from a distribution $\mathcal{D}$. Now let P_i be the probability of error on a feature vector from class ω_i, then the probability of mispredicting on k_i feature vectors from class ω_i, is $\binom{N_i}{k_i} P_i^{k_i} (1 - P_i)^{N_i - k_i}$.

Now if we estimated P_i, by $\hat{P}_i = k_i/Ni$, then $\hat{P}_i$ is unbiased estimator of P_i, because $E[k_i] = N_i P_i$, so

$$E[\hat{P}_i] = \frac{E[k_i]}{N_i} = \frac{N_i P_i}{N_i} = P_i.$$

Notice that the true error probability P_i is not known.

On the other hand, under all of the above assumptions the total error probability over S, is given by $\hat{P} = \sum_{i=1}^{M} \hat{P}(\omega_i) \hat{P}_i$. This is the same as the number of errors in S divided by the size of S. This total error estimate is also unbiased estimator of the total true error P, because

$$E(\hat{P}) = E(\sum_{i=1}^{M} \hat{P}(\omega_i) \hat{P}_i) = E(\sum_{i=1}^{M} \frac{N_i}{N} \frac{k_i}{N_i}) = E(\sum_{i=1}^{M} \frac{k_i}{N}) = \sum_{i=1}^{M} \frac{N_i P_i}{N} = \sum_{i=1}^{M} \frac{N_i}{N} P_i = P.$$

In the following, we want to show that the error estimate improves when the size of the sample increases. We look at this from two different aspects, which lead us to the desired conclusion.

The first one is the variance of the error estimate $\hat{P}$. This variance is given by

$$\sigma_{\hat{P}}^2 = \sum_{i=1}^{M} P^2(\omega_i) \frac{P_i(1 - P_i)}{N_i}.$$

It is easy to see that the above variance decreases as N_i, $i = 1, ..., M$, increases. Of course, smaller variance means better error estimate, and very small variance means that $\hat{P}$ is close to P.

The second one is the theory of *uniform convergence*, which says the following:

Let $\mathcal{F}$ be a set of functions (like a set of neural nets) and let A stands for the statement "the distance between the true error and the error estimate is greater than ϵ", then the probability of existence of a function $f \in \mathcal{F}$, for which A is achieved is less than β,

where

$$\beta = 4 \exp\ (V_F(1 + ln \frac{2N}{V_F}) - 2\epsilon - (N\epsilon^2 + 1/N)).$$

and $V_{\mathcal{F}}$ is the VC-dimension of $\mathcal{F}$, which measures the discriminatory ability of $\mathcal{F}$. This $V_{\mathcal{F}}$ indicates how many data is needed. In general, it is very difficult to compute, so instead we deal with its upper bound.

It is easy to see that the above probability decreases exponentially as N increases, which indicates that our error estimate becomes closer to the true one.

The last thing I want to mention here is that the error counting approach is the most popular one.

(3) Statistical Hypothesis Testing (based on error counting).

Statistical hypothesis testing when applies gives us a very good estimate of the error. In fact, this method enables us to know the confidence intervals of the error estimate. This method, which is based on the error counting approach, can be summarized as follows.

Let N be the total number of testing data, k the total number of mispredictions, P the error estimate, M the number of classes, and $\hat{P}$ the true error. Then k is binomially distributed, and the following formulas hold:

$$P = \sum_{i=1}^{M} P(w_i)P_i,\ \sigma_k^2 = NP(1-P),\ \hat{P} = \frac{k}{N},\ \sigma_{\hat{P}} = \frac{\sigma_k}{N} = \sqrt{\frac{P(1-P)}{N}}.$$

Now let the null hypothesis H_0 be $H_0 : \hat{P} = P$, and the alternative hypothesis H_1, be $H_1 : \hat{P} \neq P$.

Now we can use the z statistic which has a normal distribution approximated by $\mathcal{N}(0,1)$, and which is given by

$(*)$
$$z = \frac{\hat{P} - P}{\sqrt{\frac{P(1-P)}{N}}}$$

Our goal now is to find a nice relation relating P and $\hat{P}$. To do this, solve the above formula for P and get

$$P = \hat{P} - z\sqrt{\frac{P(1-P)}{N}}$$

Now if N is large enough ($N \geq 30$ and $N\hat{P}(1-\hat{P}) \geq 5$) , then $\hat{P}$ becomes close to P, so we can approximate P on the right hand side by $\hat{P}$ and get

$(**)$
$$P = \hat{P} - z\sqrt{\frac{\hat{P}(1-\hat{P})}{N}}$$

This means that z is the number of standard deviations $\hat{P}$ is far from P.

Notice that we can solve (*exactly*) for P, as follows

First, square the two sides of $(*)$, and get

$$\frac{z^2 P(1-P)}{N} = (\hat{P} - P)^2.$$

Thus,

$$z^2(P - P^2) = N(\hat{P}^2 - 2P\hat{P} + P^2).$$

$\Longrightarrow z^2P - z^2P^2 + 2P\hat{P}N - NP^2 = N\hat{P}^2$

$\Longrightarrow -(z^2+N)P^2 + (2\hat{P}N + z^2)P = N\hat{P}^2$

$\Longrightarrow (z^2+N)P^2 - (2\hat{P}N + z^2)P = -N\hat{P}^2$

$\Longrightarrow P^2 - \frac{(2\hat{P}N+z^2)}{(z^2+N)}P = -\frac{N\hat{P}^2}{(z^2+N)}$

$\Longrightarrow P^2 - \frac{(2\hat{P}N+z^2)}{(z^2+N)}P + (\frac{(2\hat{P}N+z^2)}{2(z^2+N)})^2 = -\frac{N\hat{P}^2}{(z^2+N)} + (\frac{(2\hat{P}N+z^2)}{2(z^2+N)})^2.$

Now if we let $\alpha = -\frac{N\hat{P}^2}{(z^2+N)} + (\frac{(2\hat{P}N+z^2)}{2(z^2+N)})^2$, then the above formula becomes:

$$(P - \frac{2\hat{P}N+z^2}{2(z^2+N)})^2 = \alpha.$$

Therefore, P is the positive one of then following two

$$-\sqrt{\alpha} + \frac{2\hat{P}N+z^2}{2(z^2+N)},\ \sqrt{\alpha} + \frac{2\hat{P}N+z^2}{2(z^2+N)}$$

Thus, when we know $\hat{P}$, and when we decide about the desired confidence level, $1-\rho$, we can determine how close $\hat{P}$ from P, by using $(**)$. Of course, we need to get the z_ρ-value from the $\mathcal{N}(0,1)$ table:

ρ	0.2	0.15	0.1	0.05	0.02	0.01	0.002
$1-\rho$	0.8	0.85	0.9	0.95	0.98	0.99	0.998
z_ρ	1.282	1.440	1.645	1.967	2.326	2.576	3.090

Notice that $1-\rho$ is the area under e^{-x^2} between $-z_\rho$ and z_ρ, and ρ is the remaining area. The idea will become more clear after the following example.

Example 1 *Let $N = 70$, $k = 7$, and $\rho = 0.02$. Then $\hat{P} = k/N = 0.1$ (notice that $N > 30$ and $N\hat{P}(1-\hat{P}) = 70.(0.1) * (0.9) = 6.3 > 5$), and $z_\rho = 2.326$.*

Therefore, $0.1 - (2.326).\sqrt{\frac{(0.1)(0.9)}{70}} < P < 0.1 + (2.326).\sqrt{\frac{(0.1)(0.9)}{70}}$.

I.e., we are 98% confident that $P \in (\hat{P} - 0.003, \hat{P} + 0.003)$. Or equivalently, $0.097 < P < 0.103$.

Thus, with a confidence of 98%, H_0 holds.

Notice that, in the above example, our calculation of the confidence error interval itself has errors which come from different sources. One of those is approximating P by $\hat{P}$ in (**). Another one is approximating the integral under the curve of e^{-x^2}.

There are many other error estimation methods. Some of those are: Hoeffding's inequality, error estimation without testing data, the rotation estimate, the boostrap, the posterior error estimate, the error-reject, and smoothing the error count. But again, and as I have previously mentioned, the most popular error estimation method is error counting.

4 Comparing the Error of Different Classifiers

When we have more than one classifier, it is important to compare them and decide which one has less error. Thus, we need to know how to compare the error probability of any two different classifiers. Of course, when we have more than two, all we need to do is to compare every one of them with the rest. For example, if we have K classifiers, say $c_1, c_2, ..., c_K$, then we need to make $(K-1)!$ comparisons. Suppose after making all of such comparisons, we reached the following inequality: $P(c_{i_1}) \leq P(c_{i_2}) \leq ... \leq P(c_{i_K})$, where $\{i_j\}_{j=1}^{K}$ is an arrangement for the sequence $\{1, 2, ..., K\}$, and $P(h)$ is the probability of error for classifier h. Then, we can say that classifier c_{i_1} has the least error and so it is the best one to use.

Again we will use statistical hypothesis testing for such comparisons. So, assume that we have two classifiers f and g and we want to determine, with a certain confidence, which one of the inequalities: $P(f) > P(g)$, $P(g) > P(f)$ holds. If it turned out that the first one holds with a high confidence, then we will conclude that classifier g has less error and so it is better to use. Now let $d = P(f) - P(g)$ and $\hat{d} = \hat{P}(f) - \hat{P}(g)$, and define the hypothesis, H_0 and H_1, as follows: $H_0 : d = 0$, $H_1 : d \neq 0$.

Now since $\hat{P}(f)$ and $\hat{P}(g)$ are both normally distributed, then so is their difference, $\hat{d}$. Moreover,

$$\sigma_{\hat{d}}^2 = \sigma_{\hat{P}(f)}^2 + \sigma_{\hat{P}(g)}^2 \approx \frac{\hat{P}(f)(1-\hat{P}(f)+\hat{P}(g)(1-\hat{P}(g)}{N}.$$

Therefore, the z statistic, is

$$z = \frac{\hat{d}-d}{\sigma_{\hat{d}}} = \frac{\hat{d}-d}{\sqrt{\frac{\hat{P}(f)(1-\hat{P}(f)+\hat{P}(g)(1-\hat{P}(g)}{N}}}.$$

As a summary, we can say that if the confidence level is $1-\rho$, then
(1) $\hat{d} - z_\rho \sigma_{\hat{d}} < d < \hat{d} + z_\rho \sigma_{\hat{d}}$. This gives us the range in which d lies with respect to $\hat{d}$.
(2) $d < \hat{d} + z_{2\rho} \sigma_{\hat{d}}$. This gives us an upper bound.
(3) $d > \hat{d} - z_{2\rho} \sigma_{\hat{d}}$. This gives us a lower bound.

Example 2 *If $\hat{P}(f) = 0.3$, $\hat{P}(g) = 0.2$, $N = 100$, and the confidence level is $1-\rho$, then $\hat{d} = \hat{P}(f) - \hat{P}(g) = 0.1$ and $\sigma_{\hat{d}} = \sqrt{\frac{(0.3)(0.7)+(0.2)(0.8)}{100}} = 0.060828$. Now using the lower bound, we can say that with a confidence level of $1-\rho$, $d > 0.1 - 0.060828 z_{2\rho}$. Now we can determine when $P(f) > P(g)$ as follows:*

$P(f) > P(g)$ if $d > 0$. But, $d > 0$, when $0.1 - 0.060828 z_{2\rho} > 0$; i.e., when $z_{2\rho} < 0.1/0.060828 \approx 1.644$. Notice that the corresponding ρ for $z = 1.645$, is 0.1. Thus, $2\rho = 0.1$, and so $\rho = 0.05$. Thus, we are 95% confident that $P(f) > P(g)$.

Notice that with the same level of confidence (95%), we have
$d \in (0.1 - 0.060828\ z_{0.05}, 0.1 + 0.060828\ z_{0.05}) = (-0.0196, 0.2197)$.

Thus, the probability for d to be positive is more than 11 times higher than the probability for d being negative. This is, of course, expected because $\hat{P}(f)$ is greater than $\hat{P}(g)$. Now d is positive means that $P(f) > P(g)$. While, d is zero, means $P(f) = P(g)$. The question now is what does it mean for d to be negative? If d is negative, then $P(f) < P(g)$. So, we see that it is still possible for $P(g)$ to be greater than $P(f)$, although $\hat{P}(f)$ is greater than $\hat{P}(g)$. But, of course, this is not highly probable, since we are (from the first part of this example) 95% confident that $P(f) > P(g)$. Thus, it is only 5% possible that $P(f) \leq P(g)$.

5 Training and Testing with Finite Data Sets

Of course, when we were discussing the classification error probability and all such related issues, we have assumed that we have the training and testing data available for us. Also, we have assumed that such data is finite. The matter of partitioning the data set into training and testing is not a simple one. There are many different approaches for doing that. Some of them are not reliable in terms of their error. So, the big question which arises now is: how to decide about the training data and the testing data? Can we use the same set of data for training and testing? If not, then what are the other alternatives and how to partition the data set? This is the subject of this section. So, let me discuss some of the methods for doing that.

(1) *The Resubstitution Method:* Here the same data set is used for both training and testing. As one might expect, this method is not reliable, because it yields a very optimistic error estimate. In fact, in order for this method to give a reasonable good estimate, both the size of the data set (N), and N/l must be large. Where l is the dimension of the feature vectors. But, of course, this is not always possible, or it might be difficult to be achieved. For this reason, it is not reliable to use the same data set for training and testing.

(2) *Holdout Method:* Here we divide the data set into two sets; one of them for training and the other for testing. This method also has at least two problems. The first one is reducing the size of the training data set and the testing data set, and the second one is how to partition the data set. For this reason, we need other methods which overcome those two problems.

(3) *Leave-One-Out (or Deleted) Method:* Here we partition the data set into training and testing N times (N is the size of the data set). Each time, we train on $N-1$ of the data and test on the remaining one. The excluded one is different for each time. For example, if the excluded one in the first time is x_j, for some $1 \leq j \leq N$, then we never exclude x_j again; i.e, we never use x_j for testing. Thus, this method (which leaves us with unbiased error estimate) overcomes the problems of the resubstitution method and the holdout method. But, still it has its own problems. One of those problems is the *high computational complexity.*

Now the question is: what relationships do we have between the error of the above three methods? Answer: for the last two, the estimates are very similar provided that the sizes of the training and testing sets are comparable. If the classifier is Bayesian, then the error estimate of the second (the holdout method)is an upper bound for the true error, and the error estimate of the first (the resubstitution method) is a lower bound for the true error.

The relationship between the error estimates of the three methods can be described in the following plots. Notice that P is the error probability, and P_e is the average error as $N \longrightarrow \infty$. Notice also that the error estimate improves as N increases.

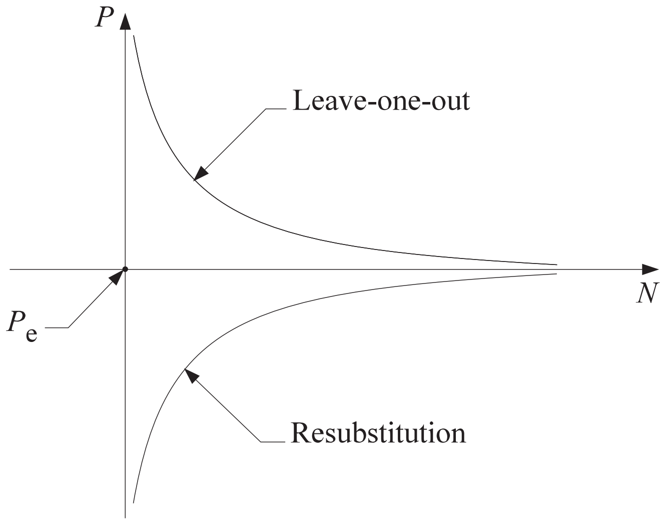

Figure 1: The relationship between the Bayesian error estimates for the holdout/leave-one-out method and the resubstitution method

(4) *Leave-k-Out (or Rotation) Method:* This method is similar to the third one. The only difference is that instead of excluding one feature vector each time and testing on it, we exclude k feature vectors. Testing is done on those k and training on the remaining $N - k$. This method can be considered as a compromise between the holdout and the leave-one-out methods (provided that $k \neq 1$ and $k \neq N/2$). Notice that k must be an integer between 1 and N. Notice also that when $k = 1$, this method becomes the leave-one-out method, and when $k = N/2$, it becomes the holdout method. The question which comes to mind now is: how we choose the k testing feature vectors? There are three approaches which we can adopt. The first one is to select the $k-$testing data sets in advance, such that each one is disjoint from the rest; e.g. test first on the first k feature vectors and train on the rest, then test on the next k feature vectors and train on the rest, and so on. Of course, k here must be a divisor of N, and the process is repeated N/k times. Thus, in this approach, testing will not be done on the same feature vector twice. The second approach is to select the $k-$testing data sets randomly. Thus, testing may be performed on the same feature vector more than once. The third approach is to test on the worst k feature vectors, where worst is measured according to the *least squares cost function.* Now it becomes trivial to ask what difference, in terms of error estimate, is there between this method and the third one (leave-one-out)? The answer is the computational complexity of this method is lower. Notice also that the error estimate can be taken to be the average of the error on all the test sets. Notice also that when $k \geq 30$, we can use statistical hypothesis testing to compare classifiers, using the t statistic and with $k-1$ degrees of freedom.

Of course, the error estimate for the last four methods is based on the error counting approach.

(5) *The Bootstrap Method:* Here we form (by a random sampling with replacement) a multiset S^* from the available data set S. Training is done on S and testing on $S^* - S$. The main advantage of this method is that it overcomes the limited number of available data.

6 Weighted Combinations of Classifiers

Sometimes weighted combinations of classifiers may improve significantly the error estimate, a property which the single classifiers may not have. In this section, we will discuss one of the methods based on such combinations. This method is called *boosting*. Another alternative method, which gives almost the same accuracy is called *bagging*. Of course, the disadvantage of such combinations over the single classifiers is the higher computational complexity. The difference between boosting and bagging is mainly in the way the training data is selected. Bagging uses sampling with replacement, while boosting uses weighted sampling. On the other hand, they have many things in common. For example, both of them require that the learning system is unstable; i.e. small changes in the training set result in different classifiers. Also, the extra computations they involve is known in advance. However, it is not clear which one is better: bagging or boosting. There are some claims that boosting is better in some cases. On the other hand, there are some experiments done on two different data sets, where both boosting and bagging were experimented on them under the same conditions (e.g., number of trials). On one of them, boosting did significantly better than bagging, while the converse happened with the other. The last thing worth mentioning about bagging is that the error of this algorithm (I mean bagging), for $T = 50$, was found in the range of $57\% - 94\%$ of the single classifier's error.

In the following, I will give a brief description of the boosting algorithm.

6.1 Boosting

As, I have previously mentioned, the goal of the boosting algorithm is to find a weighted combination of classifiers which performs reasonably better than each single classifier. This algorithm can be described as follows:

Let the training set be S and assume that the number of classes is 2; ω_1, ω_2. First decide about the number of trials (T) for the algorithm, and then we repeat the following steps: select another training set S_t according to a distribution $\mathcal{D}_t$, where t is the trial number. This distribution $\mathcal{D}_t$ assigns a weight for each feature vectors, which leaves the chance for selecting a feature vector more than once or not selecting it at all. Thus, the new training set probably will contain some repetition. For $t = 1$, the weight for all feature vectors is the same. Then, run the learning algorithm, which is called a weak learner, on S_t. The result is a classifier (weak hypothesis) h_t. This classifier maps each feature vector to 1 or -1. After that we run this classifier h_t on a testing data drawn according to $\mathcal{D}_t$, and based on the results, we generate the next distribution $\mathcal{D}_{t+1}$, which gives more weight to the misclassified feature vectors. Then, we run the weak learner again, but this time on S_{t+1} created from S_t according to $\mathcal{D}_{t+1}$.

When the algorithm is terminated, we take a weighted combination of all the classifiers we got; $\{h_t\}_{t=1}^{T}$, to get the desired classifier. Of course, the weight of each single classifier h_t, depends on its accuracy. So, we need to keep track of the error ϵ_t for each h_t. Notice that the above algorithm keeps running until the trials are finished. Another alternative is to terminate the algorithm if either: the error ϵ_t is greater than 0.5, for some t, or if $\epsilon_t = 0$, for some t.

Notice also that the weak learner does better than random guessing (which has a 50% probability of error), and that the strong learner gives an upper bound generalization, provided there is enough data available.

One version of the boosting algorithm, is the following

Input: A set of vectors $\{(x_i, y_i)\}_{i=1}^{N}$, where $\{x_i\}_{i=1}^{N}$ are feature vectors, and $\{y_i\}_{i=1}^{N}$ is the corresponding class of each x_i, $i = 1, ..., N$. Notice that y_i is either -1 or 1 (i.e., there are two classes; ω_1, ω_2).

Initialize $\mathcal{D}_1(i) = 1/N$, $\forall i = 1, ..., N$. Notice that $\mathcal{D}_t(i)$ is the weight of (x_i, y_i) under the distribution $\mathcal{D}_t$.

For $t = 1$ to T, do

∗ Create the training set S_t using $\mathcal{D}_t$.

∗ Run the weak learner on S_t, to get the weak hypothesis h_t.

∗ Evaluate h_t, and compute its error $\epsilon_t = \sum_{i \in G_t} \mathcal{D}_t(i)$. Where $G_t = \{i : h_t(x_i) \neq y_i\}$.

∗ Compute the weighting factor: $\alpha_t = \frac{1}{2}\ \ln(\frac{1-\epsilon_t}{\epsilon_t})$.

∗ Define $\mathcal{D}_{t+1}$, by $\mathcal{D}_{t+1}(i) = \frac{e^{-\alpha_t y_i h_t(x_i)}}{Z_t}\ \mathcal{D}_t(i)$.

End for.

ouput: The boosting classifier: $H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$.

Comments on the Algorithm:

• α_t is large and negative when ϵ_t is large, and it is large and positive when ϵ_t is small.

• If $\epsilon_t < 0.5$, then the weight of the misclassified feature vector is increased.

• $\epsilon_t < 0.5 \Longleftrightarrow \alpha_t > 0$.

• If h_t correctly classifies x_i, then $\mathcal{D}_t(i)$ exponentially decreases, and the converse is true.

• The purpose of the *normalization factor* Z_t is to make $\mathcal{D}_{t+1}$ a distribution.

• When ϵ_t (the error) is large, then the prediction of h_t (acting as a member of H) is flipped strongly. This is the purpose of multiplying each h_t by its weighting factor α_t.

• The weighting factor α_t measures the importance of h_t.

• The error estimate of H decreases exponentially to zero, when ϵ_t is always less than 0.5.

• If $\epsilon_t = 0.5 - \gamma_t$, $\forall t = 1, ..., T$, then the training error of the final hypothesis is bounded above by $\exp(-2\sum_{t=1}^{T} \gamma_t^2)$. This means that if each weak hypothesis is better than random guessing, then $\omega_t > 0$, $\forall t$. Therefore, the training error goes to zero exponentially.

• There exists also an upper bound for the classification error, which involves: T, N, the training error, and the VC-dimension.

• Some experimental cases showed that the testing error continues to improve even after the training error drops to zero. This contradicts what one might expect.

• Can use binary labels or real labels. Also, it deals with the multiclass case.

• The boosting algorithm is connected to game theory and it is very successful on neural networks and decision tree learners.

• There is more than one version of the boosting algorithm.

References

Watanable, Satosi (1972). Frontiers of Pattern Recognition". New York: Academic Press.

Devijver, Pirre A. & Kittler Josef (1982). Pattern Recognition: A statistical approach. London: Prentice-Hall International, INC..