Ensemble Confidence Estimates Posterior Probability

Michael Muhlbaier, Apostolos Topalis, and Robi Polikar

Rowan University, Electrical and Computer Engineering,
201 Mullica Hill Rd., Glassboro, NJ 08028, USA
{muhlba60, topali50}@students.rowan.edu
polikar@rowan.edu

Abstract. We have previously introduced the Learn^{++} algorithm that provides surprisingly promising performance for incremental learning as well as data fusion applications. In this contribution we show that the algorithm can also be used to estimate the posterior probability, or the confidence of its decision on each test instance. On three increasingly difficult tests that are specifically designed to compare posterior probability estimates of the algorithm to that of the optimal Bayes classifier, we have observed that estimated posterior probability approaches to that of the Bayes classifier as the number of classifiers in the ensemble increase. This satisfying and intuitively expected outcome shows that ensemble systems can also be used to estimate confidence of their output.

1 Introduction

Ensemble / multiple classifier systems have enjoyed increasing attention and popularity over the last decade due to their favorable performances and/or other advantages over single classifier based systems. In particular, ensemble based systems have been shown, among other things, to successfully generate strong classifiers from weak classifiers, resist over-fitting problems [1, 2], provide an intuitive structure for data fusion [2-4], as well as incremental learning problems [5]. One area that has received somewhat less of an attention, however, is the confidence estimation potential of such systems. Due to their very character of generating multiple classifiers for a given database, ensemble systems provide a natural setting for estimating the confidence of the classification system on its generalization performance.

In this contribution, we show how our previously introduced algorithm Learn^{++} [5], inspired by AdaBoost but specifically modified for incremental learning applications, can also be used to determine its own confidence on any given specific test data instance. We estimate the posterior probability of the class chosen by the ensemble using a weighted softmax approach, and use that estimate as the confidence measure. We empirically show on three increasingly difficult datasets that as additional classifiers are added to the ensemble, the posterior probability of the class chosen by the ensemble approaches to that of the optimal Bayes classifier. It is important to note that the method of ensemble confidence estimation being proposed is not specific to Learn^{++}, but can be applied to any ensemble based system.

N.C. Oza et al. (Eds.): MCS 2005, LNCS 3541, pp. 326–335, 2005.

2 Learn^{++}

In ensemble approaches using a voting mechanism to combine classifier outputs, the individual classifiers vote on the class they predict. The final classification is then determined as the class that receives the highest total vote from all classifiers. Learn^{++} uses weighted majority voting, a rather non-democratic voting scheme, where each classifier receives a voting weight based on its training performance. One novelty of the Learn^{++} algorithm is its ability to incrementally learn from newly introduced data. For brevity, this feature of the algorithm is not discussed here and interested readers are referred to [4,5]. Instead, we briefly explain the algorithm and discuss how it can be used to determine its confidence – as an estimate of the posterior probability – on classifying test data.

For each dataset ($\mathbb{D}_k$) that consecutively becomes available to Learn^{++}, the inputs to the algorithm are (i) a sequence of m training data instances $\boldsymbol{x}_{k,i}$ along with their correct labels y_i, (ii) a classification algorithm **BaseClassifier**, and (iii) an integer T_k specifying the maximum number of classifiers to be generated using that database. If the algorithm is seeing its first database ($k=1$), a data distribution (D_t) – from which training instances will be drawn - is initialized to be uniform, making the probability of any instance being selected equal. If $k>1$, then a distribution initialization sequence, initializes the data distribution. The algorithm then adds T_k classifiers to the ensemble starting at $t=eT_k+1$ where eT_k denotes the number of classifiers that currently exist in the ensemble. The pseudocode of the algorithm is given in Figure 1.

For each iteration t, the instance weights, w_t, from the previous iteration are first normalized (step 1) to create a weight distribution D_t. A hypothesis, h_t, is generated using a subset of $\mathbb{D}_k$ drawn from D_t (step 2). The error, ε_t, of h_t is calculated: if $\varepsilon_t > ½$, the algorithm deems the current classifier h_t to be too weak, discards it, and returns to step 2; otherwise, calculates the normalized error β_t (step 3). The weighted majority voting algorithm is called to obtain the composite hypothesis, H_t, of the ensemble (step 4). H_t represents the ensemble decision of the first t hypotheses generated thus far. The error E_t of H_t is then computed and normalized (step 5). The instance weights w_t are finally updated according to the performance of H_t (step 6), such that the weights of instances correctly classified by H_t are reduced and those that are misclassified are effectively increased. This ensures that the ensemble focus on those regions of the feature space that are yet to be learned. We note that H_t allows Learn^{++} to make its distribution update based on the ensemble decision, as opposed to AdaBoost which makes its update based on the current hypothesis h_t.

3 Confidence as an Estimate of Posterior Probability

In applications where the data distribution is known, an optimal Bayes classifier can be used for which the posterior probability of the chosen class can be calculated; a quantity which can then be interpreted as a measure of confidence [6]. The posterior probability of class ω_j given instance $\boldsymbol{x}$ is classically defined using the Bayes rule as:

Input: For each dataset D_k $k=1,2,\ldots,K$

- Sequence of i=1,…,m_k instances $\boldsymbol{x}_{k,i}$ with labels $y_i \in Y_k = \{1,\ldots,c\}$
- Weak learning algorithm **BaseClassifier**.
- Integer T_k, specifying the number of iterations.

Do for $k=1,2,\ldots,K$

If k=1 **Initialize** $w_1 = D_1(i) = 1/m,\ \ eT_1 = 0$ for all i.

Else Go to Step 5 to evaluate the current ensemble on new dataset D_k, update weights, and recall current number of classifiers → $eT_k = \sum_{j=1}^{k-1} T_j$

Do for $t = eT_k + 1,\ eT_k + 2,\ldots,\ eT_k + T_k$:

1. Set $D_t = w_t \Big/ \sum_{i=1}^{m} w_t(i)$ so that D_t is a distribution.
2. **Call BaseClassifier** with a subset of D_k randomly chosen using D_t.
3. Obtain $h_t : X \rightarrow Y$, and calculate its error: $\varepsilon_t = \sum_{i:h_t(\boldsymbol{x}_i)\neq y_i} D_t(i)$

   If $\varepsilon_t > ½$, discard h_t and go to step 2. Otherwise, compute normalized error as $\beta_t = \varepsilon_t / (1-\varepsilon_t)$.

4. Call weighted majority voting to obtain the composite hypothesis

$$H_t = \arg\max_{y\in Y} \sum_{t:h_t(\boldsymbol{x}_i)=y_i} \log(1/\beta_t)$$

5. Compute the error of the composite hypothesis

$$E_t = \sum_{i:H_t(\boldsymbol{x}_i)\neq y_i} D_t(i)$$

6. Set $B_t=E_t/(1-E_t)$, $0<B_t<1$, and update the instance weights:

$$D_{t+1}(i) = D_t \times \begin{cases} B_t, & \text{if } H_t(\boldsymbol{x}_i) = y_i \\ 1, & \text{otherwise} \end{cases}$$

Call weighted majority voting to obtain the final hypothesis.

$$H_{final} = \arg\max_{y\in Y} \sum_{k=1}^{K} \sum_{t:h_t(\boldsymbol{x}_i)=y_i} \log(1/\beta_t)$$

Fig. 1. Learn^{++} Algorithm

$$P(\omega_j \mid \boldsymbol{x}) = \frac{P(\boldsymbol{x} \mid \omega_j) P(\omega_j)}{\sum_{k=1}^{N} P(\boldsymbol{x} \mid \omega_k) P(\omega_k)} \tag{1}$$

Since class distributions are rarely known in practice, posterior probabilities must be estimated. While there are several techniques for density estimation [7], such techniques are difficult to apply for large dimensional problems. A method that can

estimate the Bayesian posterior probability would therefore prove to be a most valuable tool in evaluating classifier performance. Several methods have been proposed for this purpose [6-9]. One example is the softmax model [8], commonly used with classifiers whose outputs are binary encoded, as such outputs can be mapped into an estimate of the posterior class probability using

$$P(\omega_j \mid \boldsymbol{x}) \approx C_j(\boldsymbol{x}) = \frac{e^{A_j(\boldsymbol{x})}}{\sum_{k=1}^{N} e^{A_k(\boldsymbol{x})}} \tag{2}$$

where $A_j(\boldsymbol{x})$ represents the output for class j, and N is the number of classes. $C_j(\boldsymbol{x})$ is then the confidence of the classifier in predicting class ω_j for instance $\boldsymbol{x}$, which is an estimate of the posterior probability $P(\omega_j|\boldsymbol{x})$. The softmax function essentially takes the exponential of the output and normalizes it to [0 1] range by summing over the exponentials of all outputs. This model is generally believed to provide good estimates if the classifier is well trained using sufficiently dense training data.

In an effort to generate a measure of confidence for an ensemble of classifiers in general, and for Learn^{++} in particular, we expand the softmax concept by using the individual classifier weights in place of a single expert's output. The ensemble confidence, estimating the posterior probability, can therefore be calculated as:

$$P(\omega_j \mid \boldsymbol{x}) \approx C_j(\boldsymbol{x}) = \frac{e^{F_j(\boldsymbol{x})}}{\sum_{k=1}^{N} e^{F_k(\boldsymbol{x})}} \tag{3}$$

where

$$F_j(\boldsymbol{x}) = \sum_{t=1}^{N} \begin{pmatrix} \log\left(1/\beta_t\right) & h_t(\boldsymbol{x}) = \omega_j \\ 0 & otherwise \end{pmatrix} \tag{4}$$

The confidence, $C_j(\boldsymbol{x})$, associated with class ω_j for instance $\boldsymbol{x}$ is therefore the exponential of the sum of classifier weights that selected class ω_j, divided by the sum of the aforementioned exponentials corresponding to each class. The significance of this confidence estimation scheme is in its consideration of the diversity in the classifier decisions: in calculating the confidence of class ω_j, the confidence will increase if the classifiers that did not choose class ω_j have varying decisions as opposed to having a common decision, that is, if the evidence against class ω_j is not strong. On the other hand, the confidence will decrease if the classifiers that did not choose class ω_j have a common decision, that is, there is strong evidence against class ω_j.

4 Simulation Results

In order to find out if and how well the Learn^{++} ensemble confidence approximates the Bayesian posterior probability, the modified softmax approach was analyzed on three increasingly difficult problems. In order to calculate the theoretical Bayesian posterior probabilities, and hence compare the Learn^{++} confidences to those of Bayes-

ian probabilities, experimental data were generated from Gaussian distribution. For training, 100 random instances were selected from each class distribution, using which an ensemble of 30 MLP classifiers were generated with Learn^{++}. The data and classifier generation process was then repeated and averaged 20 times with randomly selected data to ensure generality. For each simulation, we also benchmark the results by calculating a mean square error between Learn^{++} and Bayes confidences over the entire grid of the feature space, with each added classifier to the ensemble.

4.1 Experiment 1

A two feature, three class problem, where each class has a known Gaussian distribution is seen in Fig. 2. In this experiment class 1, 2, and 3 have a variance of 0.5 and are centered at [-1, 0], [1, 1], and [1, -1], respectively. Since the distribution is known (and is Gaussian), the actual posterior probability can be calculated from Equation 1, given the known likelihood $P(\boldsymbol{x}|\omega_j)$ that can be calculated as

$$P(\boldsymbol{x} \mid \omega_j) = \frac{1}{(2\pi)^{d/2}\left|\boldsymbol{\Sigma}_j\right|^{1/2}} e^{-\frac{1}{2}\left[\left(\boldsymbol{x}-\boldsymbol{\mu}_j\right)^T \boldsymbol{\Sigma}_j^{-1}\left(\boldsymbol{x}-\boldsymbol{\mu}_j\right)\right]} \tag{5}$$

where d is the dimensionality, and $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ are the mean and the covariance matrix of the distribution from which j^{th} class data are generated. Each class was equally likely, hence $P(\omega_j)$=1/3. For each instance, over the entire grid of the feature space shown in Fig.2, we calculated the posterior probability of the class chosen by the Bayes classifier, and plotted them as a confidence surface, as shown in Fig.3a. Calculating the confidences of Learn^{++} decisions on the same feature space provided the plot in Fig 3b, indicating that the ensemble confidence surface closely approximates that of the Bayes classifier.

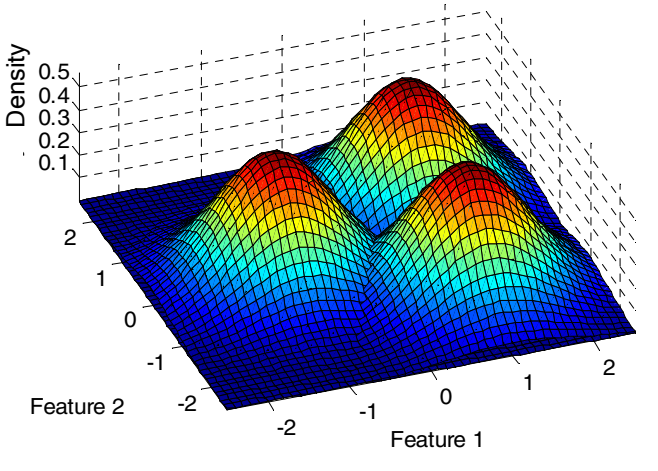

Fig. 2. Data distributions used in Experiment 1

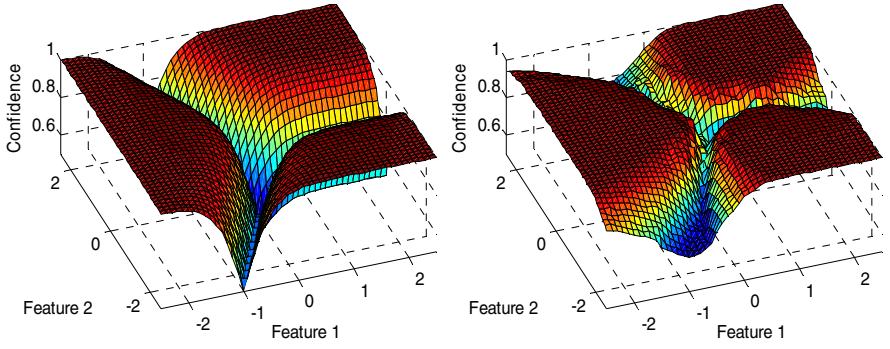

Fig. 3. (**a**) Bayesian and (**b**) Learn^{++} confidence surface for Experiment 1

It is interesting to note that the confidences in both cases plummet around the decision boundaries and approach 1 away from the decision boundary, an outcome that makes intuitive sense. To quantitatively determine how closely the Learn^{++} confidence approximates that of Bayes classifier, and how this approximation changes with each additional classifier, the mean squared error (MSE) was calculated between the ideal Bayesian confidence surface and the Learn^{++} confidence – over the entire grid of the feature space - for each additional classifier added to the ensemble. As seen in Fig.4, MSE between the two decreases as new classifiers are added to the ensemble, an expected, but nevertheless immensely satisfying outcome. Furthermore, the decrease in the error is exponential and rather monotonic, and does not appear to indicate any over-fitting, at least for as many as 30 classifiers added to the ensemble.

The ensemble confidence was then compared to that of a single MLP classifier, where the confidence was calculated using the MLP's raw output values. The mean squared error was calculated between the resulting confidence and the Bayesian confidence and has been plotted as a dotted line in Fig. 4 in comparison to the Learn^{++} confidence. The single MLP differs from classifiers generated using the Learn^{++} algorithm on two accounts. First, the single MLP is trained using all of the training data where each classifier in the Learn^{++} ensemble is trained on 2/3 of the training data. Also, Learn^{++} confidence is based on the discrete decision of each classifier. If there were only one classifier in the ensemble, "all" classifiers would "agree" resulting in a confidence of 1. Therefore, confidence of a single MLP can only be calculated based on the (softmax normalized) actual output values unlike Learn^{++} which uses a weighted vote of the discrete output labels.

4.2 Experiment 2

To further characterize the behavior of this confidence estimation scheme, Experiment 1 was repeated by increasing the variances of the class distributions from 0.5 to 0.75, resulting in a more overlapping distribution (Fig. 5) and a tougher classification problem.

Learn^{++} was trained with data generated from this distribution, its confidence calculated over the entire grid of the feature space and plotted in comparison to that of Bayes classifier in Fig. 6. We note that low confidence valleys around the decision boundaries are wider in this case, an expected outcome of the increased variance.

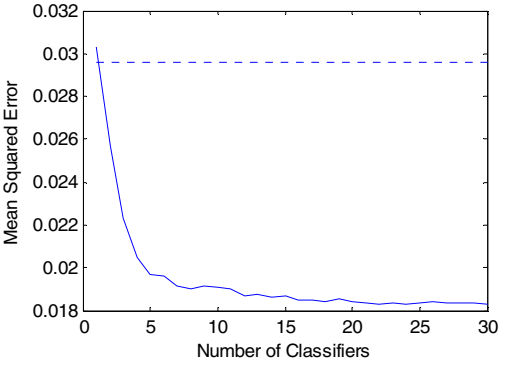

Fig. 4. Mean square error as a function of number of classifiers - Experiment 1

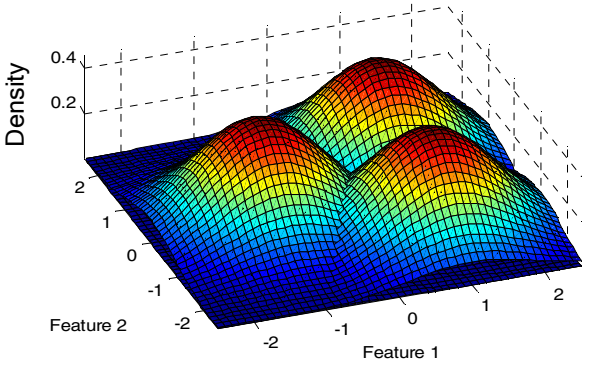

Fig. 5. Data distributions used in Experiment 2

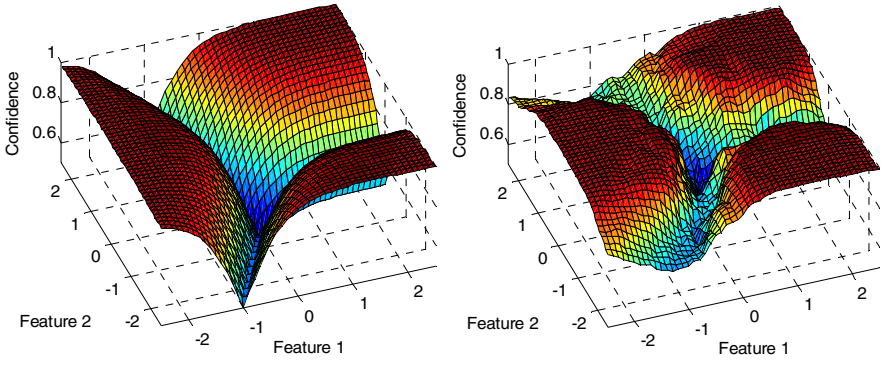

Fig. 6. (**a**) Bayesian and **(b)** Learn^{++} confidence surface for Experiment 2

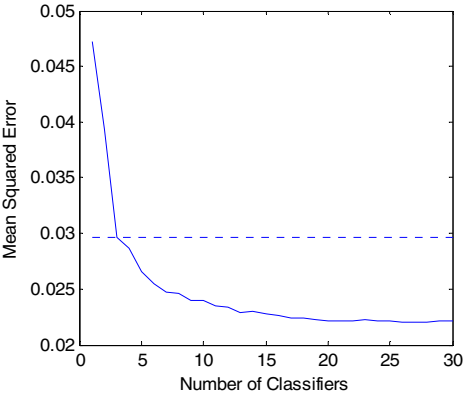

Fig. 7. Mean square error as a function of number of classifiers - Experiment 2

Fig.7 shows that the MSE between the Bayes and Learn^{++} confidences is once again decreasing as new classifiers are added to the ensemble. Fig. 7 also compares Learn^{++} performance to a single MLP, shown as the dotted line, as described above.

4.3 Experiment 3

Finally, an additional class was added to the distribution from Experiment 1 with a variance of 0.25 and mean at [0 0] (Fig. 8), making it an even more challenging classification problem due to additional overlap between classes.

Similar to the previous two experiments, an ensemble of 30 classifiers was generated by Learn^{++}, and trained with data drawn from the above distribution. The confidence of the ensemble over the entire feature space was calculated and plotted in comparison with the posterior probability based confidence of the Bayes classifier over the same feature space. Fig. 9 shows these confidence plots, where the Learn^{++} based ensemble confidence (Fig. 9b) closely approximates that of Bayes (Fig. 9a).

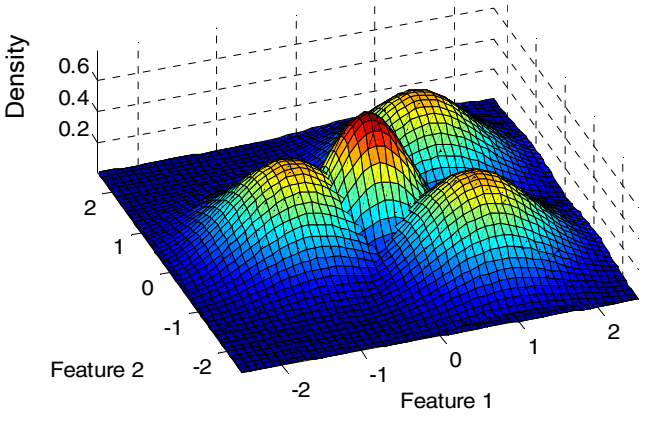

Fig. 8. Data distributions used in Experiment 3

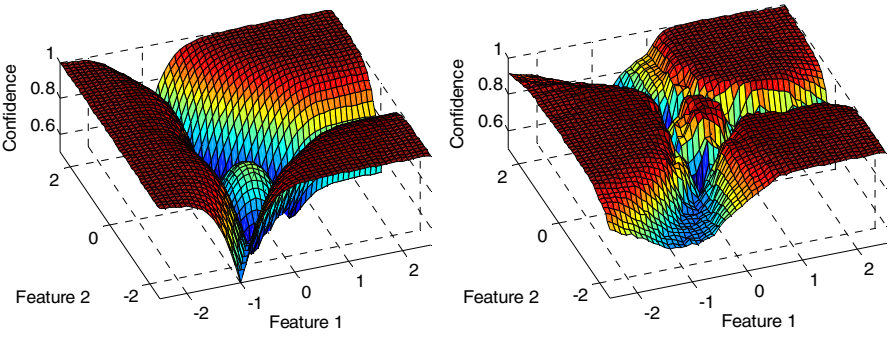

Fig. 9. (**a**) Bayesian and **(b)** Learn^{++} confidence surface for Experiment 3

Fig. 9 indicates that Learn^{++} assigns a larger peak confidence to the middle class than the Bayes classifier. Since the Learn^{++} confidence is based on the discrete decision of each classifier, when a test instance is presented from this portion of the space, most classifiers agree on the middle class resulting in a high confidence. However, the Bayesian confidence is based on the distribution of the particular class and the distribution overlap of the surrounding classes, thus lowering the confidence.

Finally, the MSE between the Learn^{++} confidence and the Bayesian confidence, plotted in Fig. 10, as a function of ensemble population, shows the now-familiar characteristic of decreasing error with each new classifier added to the ensemble. For comparison, a single MLP was also trained on the same data, and its mean squared error with respect to the Bayesian confidence is shown by a dotted line.

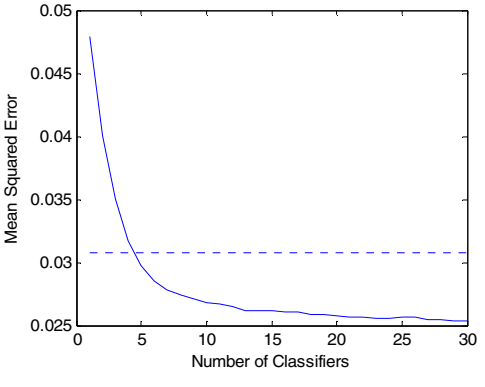

Fig. 10. Mean square error as a function of number of classifiers - Experiment 3

5 Conclusions and Discussions

In this contribution we have shown that the confidence of an ensemble based classification algorithm in its own decision can easily be calculated as an exponentially nor-

malized ratio of the weights. Furthermore, we have shown - on three experiments of increasingly difficult Gaussian distribution - that the confidence calculated in this way approximates the posterior probability of the class chosen by the optimal Bayes classifier. In each case, we have observed that the confidences calculated by Learn^{++} approximated the Bayes posterior probabilities rather well. However, in order to quantitatively assess exactly how close the approximation was, we have also computed the mean square error between the two over the entire grid of the feature space on which the two classifiers were evaluated. We have plotted this error as a function of the number of classifiers in the ensemble, and noticed that the error decreased exponentially and monotonically as the number of classifiers increased; an intuitive, yet quite satisfying outcome. No over-fitting effects were observed after as many as 30 classifiers, and the final confidences estimated by Learn^{++} was typically within 2% of the posterior probabilities calculated for the Bayes classifier. While these results were obtained by using Learn^{++} as the ensemble algorithm, they should generalize well to other ensemble and/or boosting based algorithms.

Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant No. ECS-0239090, "CAREER: An Ensemble of Classifiers Approach for Incremental Learning."

References

1. Kuncheva L.I. *Combining Pattern Classifiers, Methods and Algorithms*, Hoboken, NJ: Wiley Interscience, 2004.
2. Y. Freund and R. Schapire, "A decision theoretic generalization of on-line learning and an application to boosting," *Computer and System Sci.,* vol. 57, no. 1, pp. 119-139, 1997.
3. Kuncheva L.I., "A Theoretical Study on Six Classifier Fusion Strategies," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp. 281–286, 2002.
4. Lewitt M. and Polikar R., "An ensemble approach for data fusion with Learn^{++}," *Proc. 4th Int. Work. on Multiple Classifier Systems*, (Windeatt T .and Roli F., eds.) LNCS vol. 2709, pp. 176-186, Berlin: Springer, 2003.
5. Polikar R., Udpa L., Udpa S., and Honavar V., "Learn^{++}: An incremental learning algorithm for supervised neural networks," *IEEE Trans. on System, Man and Cybernetics (C)*, vol. 31, no. 4, pp. 497-508, 2001.
6. Duin R.P., Tax M., "Classifier conditional posterior probabilities," *Lecture Notes in Computer Science*, LNCS vol. 1451, pp. 611-619, Berlin: Springer, 1998.
7. Duda R., Hart P., Stork D., In *Pattern Classification 2/e*, Chap. 3 &4, pp. 80-214, New York, NY: Wiley Interscience, 2001.
8. Alpaydin E. and Jordan M. "Local linear perceptrons for classification." *IEEE Transactions on Neural Networks* vol. 7, no. 3, pp. 788-792, 1996.
9. Wilson D., Martinez T., "Combining cross-validation and confidence to measure fitness," *IEEE Joint Conf. on Neural Networks*, vol. 2, pp. 1409-1414, 1999.