Confidence Estimation Using the Incremental Learning Algorithm, Learn++

Jeffrey Byorick and Robi Polikar

Electrical and Computer Engineering, Rowan University,
136 Rowan Hall, Glassboro, NJ 08028, USA.
byor4610@students.rowan.edu, polikar@rowan.edu

Abstract. Pattern recognition problems span a broad range of applications, where each application has its own tolerance on classification error. The varying levels of risk associated with many pattern recognition applications indicate the need for an algorithm with the ability to measure its own confidence. In this work, the supervised incremental learning algorithm Learn++ [1], which exploits the synergistic power of an ensemble of classifiers, is further developed to add the capability of assessing its own confidence using a weighted exponential majority voting technique.

1 Introduction

1.1 Incremental Learning

It is widely recognized that the recognition accuracy of a classifier is heavily incumbent on the availability of an adequate and representative training dataset. Acquiring such data is often tedious, time-consuming, and expensive. In practice, it is not uncommon for such data to be acquired in small batches over a period of time. A typical approach in such cases is combining new data with all previous data, and training a new classifier from scratch. This approach results in loss of all previously learned knowledge, a phenomenon known as catastrophic forgetting. Furthermore, the combination of old and new datasets is not even always a viable option if previous datasets are lost, discarded, corrupted, inaccessible, or otherwise unavailable.

Incremental learning is the solution to such scenarios, which can be defined as the process of extracting new information without losing prior knowledge from an additional dataset that later becomes available. Various definitions and interpretations of incremental learning can be found in literature, including online learning [2,3], relearning of previously misclassified instances [4,5], and growing and pruning of classifier architectures [6,7]. For the purposes of this work, an algorithm possesses incremental learning capabilities, if it meets the following criteria: (1) ability to acquire additional knowledge when new datasets are introduced; (2) ability to retain previously learned information; (3) ability to learn new classes if introduced by new data.

© Springer-Verlag Berlin Heidelberg 2003

1.2 Ensemble of Classifiers

Ensemble systems have attracted a great deal of attention over the last decade due to their empirical success over single classifier systems on a variety of applications. Such systems combine an ensemble of generally weak classifiers to take advantage of the so-called *instability* of the weak classifier, which causes the classifiers to construct sufficiently different decision boundaries for minor modifications in their training parameters, causing each classifier to make different errors on any given instance. A strategic combination of these classifiers, such as weighted majority voting [8], then eliminates the individual errors, generating a strong classifier. A rich collection of algorithms have been developed using multiple classifiers, such as AdaBoost [9], with the general goal of improving the generalization performance of the classification system. Using multiple classifiers for incremental learning, however, has been largely unexplored. Learn++, in part inspired by AdaBoost, was developed in response to recognizing the potential feasibility of ensemble of classifiers in solving the incremental learning problem. Learn++ was initially introduced in [1] as an incremental learning algorithm for MLP type networks. A more versatile form of the algorithm was presented in [10] for all supervised classifiers. We have recently recognized that inherent voting mechanism of the algorithm can also be used in effectively determining the confidence of the classification system in its own decision. In this work, we describe the algorithm Learn++, along with representative results on incremental learning and confidence estimation obtained on one real world and one benchmark database from the Univ. of California, Irvine (UCI) machine learning repository [11].

2 Learn++

The Learn++ algorithm, given in Fig. 1, exploits the synergistic power of an ensemble of classifiers to incrementally learn new information that may later become available. Learn++ generates multiple weak classifiers, each trained with different subsets of the data. For each database $\mathcal{D}_k$, $k=1,\dots,K$ that becomes available, the inputs to Learn++ are (i)$S_k = \{(x_i, y_i) \mid i = 1,\cdots,m_k\}$, a sequence of m_k training data instances x_i, along with their correct labels y_i, (ii) a weak classification algorithm **BaseClassifier** to generate weak classifiers, and (iii) an integer T_k specifying the number of classifiers (hypotheses) to be generated for that database. We require that **BaseClassifier** obtain at least 50% correct classification performance on its own training dataset, to ensure a meaningful classification performance for each classifier.

Learn++ starts by initializing a set of weights for the training data, $\boldsymbol{w}$, and a distribution $\boldsymbol{D}$ obtained from $\boldsymbol{w}$, according to which a training subset TR_t and a test subset TE_t are drawn at the t^{th} iteration of the algorithm, $t=1,\dots,T_k$, where $S_k = TR_t \bigcup TE_t$. Unless *a priori* information indicates otherwise, this distribution is initially set to be uniform, giving equal probability to each instance to be selected into the first training subset. The variation of instances within the training data subsets is achieved by iteratively updating the distribution of weights $\boldsymbol{D}$. At each iteration t, the weights adjusted at iteration t-1 are normalized to ensure that a legitimate distribution, $\boldsymbol{D}_t$, is obtained. TR_t and TE_t are then drawn according to $\boldsymbol{D}_t$ and **BaseClassifier** is trained with the training subset. A hypothesis h_t is obtained as the t^{th} classifier, whose error ε_t is computed on the entire (current) database S_k simply by adding the distribution weights of the misclassified instances

$$\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i) \tag{1}$$

If $\varepsilon_t > ½$, h_t is discarded and a new TR_t and TE_t are selected. If the error is less then half, then the error is normalized and computed as

$$\beta_t = \varepsilon_t / (1-\varepsilon_t), \qquad 0 \le \beta_t \le 1 \tag{2}$$

Hypotheses generated in all previous iterations are then combined using weighted majority voting to form a *composite hypothesis* H_t using

$$H_t = \arg\max_{y\in Y} \sum_{t:h_t(x)=y} \log\frac{1}{\beta_t} \tag{3}$$

where the sum of weights associated with each classifier is computed for every class present in the classification task. A higher weight is given to classifiers that perform better on their specific training sets. The composite hypothesis H_t is obtained by assigning the class label to an instance x_i that receives the largest total vote. The composite error made by H_t is then computed as

$$E_t = \sum_{i:H_t(x_i)\neq y_i} D_t(i) = \sum_{i=1}^{m} D_t(i)[| H_t(x_i) \neq y_i |] \tag{4}$$

where $[| \cdot |]$ evaluates to 1, if the predicate holds true. Similar to the calculation of β_t, a normalized composite error B_t is computed as

$$B_t = E_t / (1-E_t), \qquad 0 \le B_t \le 1 \tag{5}$$

The weights $w_t(i)$ are then updated to obtain D_{t+1}, which is used for the selection of the next training and testing subsets, TR_{t+1} and TE_{t+1}, respectively. The distribution update rule which comprises the heart of the algorithm is given by

$$w_{t+1}(i) = w_t(i) \times \begin{cases} B_t, & \text{if } H_t(x_i) = y_i \\ 1, & \text{otherwise} \end{cases} = w_t(i) \times B_t^{1-[|H_t(x_i)\neq y_i|]}. \tag{6}$$

This rule reduces the weights of those instances that are correctly classified by the composite hypothesis H_t, so that their probability of being selected into the next training subset is reduced. When normalized during iteration *t+1*, the weights of misclassified instances are increased relative to the rest of the dataset. We emphasize that unlike AdaBoost and its variations, the weight update rule in Learn++ looks at the classification output of the composite hypothesis, not to that of a specific hypothesis. This weight update procedure forces the algorithm to focus more on instances that have not been properly learned by the ensemble. When Learn++ is learning incrementally, the instances introduced by the new database are precisely those not learned by the ensemble. After T_k hypotheses are generated for each database $\mathcal{D}_k$, the final hypothesis is obtained by the weighted majority voting of all composite hypotheses:

$$H_{final} = \arg\max_{y \in Y} \sum_{k=1}^{K} \sum_{t:H_t(x)=y} \log \frac{1}{B_t} \tag{7}$$

```
Input: For each dataset drawn from $\mathcal{D}_k$ $k=1,2,\ldots,K$
  • Sequence of $m_k$ examples $S_k = \{(x_i, y_i) \mid i = 1,\cdots,m_k\}$
  • Weak learning algorithm BaseClassifier
  • Integer $T_k$, specifying the number of iterations
Initialize $w_1(i) = D_1(i) = 1/m_k$, $\forall i$, $i=1,2,\ldots,m_k$
Do for each $k=1,2,\ldots,K$:
  Do for $t= 1,2,\ldots,T_k$:
    1. Set $D_t = \mathbf{w_t} \Big/ \sum_{i=1}^{m} w_t(i)$ so that $D_t$ is a distribution
    2. Draw training $TR_t$ and testing $TE_t$ subsets from $D_t$.
    3. Call BaseClassifier to be trained with $TR_t$.
    4. Obtain a hypothesis $h_t$: $X \rightarrow Y$, and calculate the error of $h_t$: $\varepsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$
       on $TR_t$+$TE_t$. If $\varepsilon_t$> ½ , discard $h_t$ and go to step 2. Otherwise, compute normalized error as $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$.
    5. Call weighed majority voting and obtain the composite hypothesis
       $H_t = \arg\max_{y \in Y} \sum_{t:h_t(x)=y} \log(1/\beta_t)$
    6. Compute the error of the composite hypothesis
       $E_t = \sum_{i:H_t(x_i) \neq y_i} D_t(i) = \sum_{i=1}^{m} D_t(i) [|H_t(x_i) \neq y_i|]$
    7. Set $B_t = E_t / (1 - E_t)$, and update the weights: $w_{t+1}(i) = w_t(i) \times B_t^{1-[|H_t(x_i) \neq y_i|]}$
Call Weighted majority voting and Output the final hypothesis:
       $H_{final}(x) = \arg\max_{y \in Y} \sum_{k=1}^{K} \sum_{t:h_t(x)=y} \log(1/\beta_t)$
```

Fig. 1. Learn++ Algorithm

3 Confidence Estimation

An intimately relevant issue is the confidence of the classifier in its decision, with particular interest in whether the confidence of the algorithm improves as new data becomes available. The voting mechanism inherent in Learn++ hints to a practical approach for estimating confidence: decisions made with a vast majority of votes have better confidence then those made by a slight majority. We have implemented McIver and Friedl's weighted exponential voting based confidence metric [12] with Learn++ as

$$C_j(x) = P(y = j \mid x) = \frac{e^{F_j(x)}}{\sum_{k=1}^{N} e^{F_k(x)}}, \quad 0 \le C_j(x) \le 1 \tag{8}$$

where $C_j(x)$ is the confidence assigned to instance x when classified as class j, $F_j(x)$ is the total vote associated with the j^{th} class for the instance x, and N is the total number of classes. The total vote $F_j(x)$ class j receives for any given instance is computed as

$$F_j(x) = \sum_{t=1}^{N} \begin{pmatrix} \log\frac{1}{\beta_t} & h_t(x) = j \\ 0 & otherwise \end{pmatrix} \tag{9}$$

The confidence of winning class is then considered as the confidence of the algorithm in making the decision with respect to the winning class. Since $C_j(x)$ is between 0 and 1, the confidences can be translated into linguistic indicators, such as those shown in Table 1. These indicators are adopted and used in tabulating the results.

Table 1. Confidence percentages represented by linguistic indicators

Confidence Percentage Range	**Confidence Level**
$90 \le C \le 100$	Very High (VH)
$80 \le C < 90$	High (H)
$70 \le C < 80$	Medium (M)
$60 \le C < 70$	Low (L)
$C < 60$	Very Low (VL)

Equations (8) and (9) allow Learn++ to determine its own confidence in any classification it makes. The desired outcome of the confidence analysis is to observe a high confidence on correctly classified instances, and a low confidence on misclassified instances, so that the low confidence can be used to flag those instances that are being misclassified by the algorithm. A second desired outcome is to observe improved confidences on correctly classified instances and reduced confidence on misclassified instances, as new data becomes available, so that the incremental learning ability of the algorithm can be further confirmed.

4 Simulation Results on Learn++

Learn++ has been tested on a diverse set of benchmark databases acquired from the UCI Machine Learning Repository, as well as a few real-world applications, both for incremental learning – where new datasets included new classes – and for estimating the confidence of Learn++ in its own decisions. The incremental learning results with new classes are presented in [1]. In this paper, we present the results on confidence estimation, and we use two databases, one benchmark database from UCI, and one real world on gas sensing, as representative simulations.

4.1 Volatile Organic Compound (VOC) Database

The VOC database is a real world database for the odorant identification problem. The instances are responses of six quartz crystal microbalances to five volatile organic compounds, including ethanol (ET), octane (OC), toluene (TL), trichloroethelene (TCE), and xylene (XL), constituting a five class, six feature database.

Three datasets S_1, S_2 and S_3, where each dataset included approximately one third of the entire training data, were provided to Learn++ in three training sessions for incremental learning. The data distribution and the percent classification performance are given in Table 2. The performances listed are on the validation data, *TEST*, following each training session. Table 3 provides an actual breakdown of correctly classified and misclassified instances falling into each confidence range after each training session. The trends of the confidence estimates after subsequent training sessions are given in Table 4. The desired outcome on the actual confidences is high to very high confidences on correctly classified instances, and low to very low confidences on misclassified instances. The desired outcome on confidence trends is increasing or steady confidences on correctly classified instances, and decreasing confidences on misclassified instances, as new data is introduced.

Table 2. Data distribution and performance on VOC database

	Ethanol	**Octane**	**Toluene**	**TCE**	**Xylene**	**Test Perf. (%)**
S_1	13	11	12	9	15	84.3
S_2	9	10	14	11	16	85.8
S_3	8	9	24	10	9	87.7
TEST	34	34	62	34	40	-------

Table 3. Confidence results on VOC database

		VH	**H**	**M**	**L**	**VL**
Correctly Classified	**S_1**	149	9	3	8	3
	S_2	163	6	2	4	0
	S_3	172	0	2	0	5
Misclassified	**S_1**	16	4	3	6	3
	S_2	25	2	0	0	2
	S_3	23	0	1	0	1

Table 4. Confidence trends for the VOC database.

	Increasing/Steady	**Decreasing**
Correctly Classified	172	7
Misclassified	9	16

The performance figures in Table 2 indicate that the algorithm is improving its generalization performance as new data becomes available. The improvement is modest, however, as majority of the new information is already learned in the first training session. Other experiments, where new data introduced new classes, showed remarkable performance increase as reported in [1]. Table 3 indicates that the vast majority of correctly classified instances tend to have very high confidences, with continually improved confidences at consecutive training sessions. While a considerable portion of misclassified instances also had high confidence for this database, the general desired trends of increased confidence on correctly classified instances and decreasing confidence on misclassified ones were notable and dominant, as shown in Table 4.

4.2 Glass Database

The glass database, retrieved from UCI repository, is a 10-feature, 6-class database with samples of glass from buildings, vehicles, containers, tableware, and headlamps. The buildings include two types of glass, which are float processed and non-float processed. This database was also provided incrementally in three training sessions, with each session using one of the datasets, $S_1 \sim S_3$. The distribution of data, as well as the performance on the validation dataset, is shown in Table 5. The confidence results are shown in Table 6, while the confidence trends are provided in Table 7.

Table 5. Data distribution and generalization performance on glass database

	Float	**Non-Float**	**Vehicle**	**Container**	**Table**	**Lamp**	**Test Perf. (%)**
S_1	14	22	3	1	2	6	84.0
S_2	14	17	4	3	2	9	85.5
S_3	17	13	6	3	2	7	92.8
TEST	25	24	4	6	3	7	------

Table 6. % Confidence Results on Glass Database

		VH	**H**	**M**	**L**	**VL**
Correctly Classified	**S_1**	0	0	17	6	35
	S_2	47	1	5	2	5
	S_3	57	4	2	2	3
Misclassified	**S_1**	0	0	0	0	11
	S_2	0	0	0	1	9
	S_3	1	0	0	2	2

Table 7. Confidence trends for glass database

	Increasing/Steady	**Decreasing**
Correctly classified	63	1
Misclassified	3	2

For the glass database, the above-mentioned desirable traits are even more remarkable. The majority of correctly classified instances fell into a very high confidence range, while the misclassified instances fell into the very low confidence range. Positive attributes were also seen in the confidence trends where the majority of correctly classified instances had an increasing or steady confidence through consecutive training sessions. Furthermore, the incremental learning ability of the algorithm is also demonstrated through the improved generalization performance (from 83% to 93%) on the TEST dataset with availability of additional data.

5 Discussion and Conclusions

Apart from the incremental learning ability of Learn++, it was found that the algorithm can also assess the confidence of its own decisions. In general, majority of correctly classified instances had very high confidence estimates while lower confidence values were associated with misclassified instances. Therefore, classifications with low confidences can be used as a flag to further evaluate those instances. Furthermore, the algorithm also showed increasing confidences in correctly classified instances and decreasing confidences in misclassified instances after subsequent training sessions. This is a very comforting outcome, which further indicates that algorithm can incrementally acquire new and novel information from additional data. Work is in progress to further test the algorithm's capabilities on a more diverse set of real world and benchmark databases.

Acknowledgement. This material is based upon work supported by the National Science Foundation under Grant No. 0239090.

References

[1] R. Polikar, L. Udpa, S. Udpa, V. Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Tran. on Systems, Man, and Cybernetics(C).*, Vol. 31, no. 4, November 2001.

[2] P. Winston, "Learning structural descriptions from examples," In *The Psychology of Computer Vision* P. Winston (ed.), pp. 157–209, McGraw-Hill: New York, NY, 1975.

[3] N. Littlestone, "Learning quickly when irrelevant attributes abound," *Machine Learning*, vol. 2, pp. 285–318, 1988.

[4] P. Jantke, "Types of incremental learning," *Training Issues in Incremental Learning*, A. Cornuejos (ed.) pp.26-32, The AAAI Press: Menlo Park, CA 1993.

[5] M.A. Maloof and R.S. Michalski," Selecting examples for partial memory learning," *Machine Learning*, vol. 41, pp. 27–52, 2000.

[6] F.S. Osario and B. Amy," INSS: A hybrid system for constructive machine learning," *Neurocomputing*, vol. 28, pp. 191–205, 1999.

[7] J. Ghosh and A,c, Nag," Knowledge enhancement and reuse with radial basis function networks," *Proceeding of International Joint Conference on Neural Networks*, vol. 9, no. 2, pp. 1322–1327, 2002.

[8] N. Littlestone and M. Warmuth, "Weighted Majority Algorithm," *Information and Computation*, vol. 108, pp. 212-261, 1994.

[9] Y. Freund and R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, 1997.

[10] R. Polikar, J. Byorick, S. Krause, A. Marino, and M. Moreton, "Learn++: A classifier independent incremental learning algorithm," *Proceedings of International Joint Conference on Neural Networks*, May 2002.

[11] C. L. Blake and C. J. Merz. (1998) UCI repository of machine learning databases. Dept. Inform. and Comput. Sci., University of California, Irvine. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html.

[12] D. McIver and M. Friedl, "Estimating Pixel-Scale Land Cover Classification Confidence Using Nonparametric Machine Learning Methods," *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 39, No. 9, September 2001.