Combining Classifiers for Multisensor Data Fusion

Devi Parikh, Min T. Kim, Joseph Oagaro, Shreekanth Mandayam, and Robi Polikar*
Department of Electrical and Computer Engineering,
Rowan University, Glassboro, NJ 08028, USA
{parikh55, kimm18, oagaro00}@students.rowan.edu, {shreek, polikar}@rowan.edu

Abstract - *Learn++ was recently introduced as an ensemble of classifiers based incremental learning algorithm, capable of retaining formerly acquired knowledge while learning novel information content from new datasets without requiring access to any of the previously seen data. In this contribution, we discuss the conceptual similarity between incremental learning and data fusion, the latter also requiring learning from new data, albeit composed of a different set of features. Following the technical description of the algorithm, we present our recent promising results on a real-world data fusion application of non-destructive evaluation for pipeline defect identification.*

Keywords: Data fusion, combining classifiers, ensemble systems, incremental learning, Learn++

1 Introduction

1.1 Incremental learning and data fusion

Classification algorithms usually require availability of an adequate and representative set of training data to generate an appropriate decision boundary and provide a satisfactory generalization performance. This is particularly true if an ensemble approach of classification is used and the classifiers are combined using trainable rules such as weighted majority voting, weighted sum rule and weighted product rules, as opposed to fixed rules such as the simple sum, product and majority voting rules [1]. However, acquisition of such data is expensive and time consuming, and consequently it is not uncommon for the entire data to become available gradually in small batches over a period of time. Furthermore, the datasets acquired in subsequent batches may introduce instances of new classes that were not present in previous datasets. In such settings, it is necessary for an existing classifier to be able to acquire the newly introduced knowledge without forgetting the previously learnt information. The ability of a classifier to learn in this fashion is usually referred to as *incremental learning*.

It is well known that data available from multiple sources underlying the same phenomenon may contain complementary information. For instance, in non-destructive evaluation of pipelines, defect information may be obtained from eddy current, magnetic flux leakage images, ultrasonic scans, thermal imaging, etc. Intuitively, if such information from multiple sources can be appropriately combined, the performance of a classification system can be improved. A classification system, capable of combining information from multiple sources or from multiple feature sets, is said to be capable of performing data fusion. Consequently, both incremental learning and data fusion involve learning from different sets of data. In incremental learning the datasets may introduce new classes, whereas in data fusion the datasets may contain different features, indicating a conceptual similarity between incremental learning and data fusion.

1.2 Ensemble approach incremental learning

A multiple classifier system (MCS) combines an ensemble of generally weak and/or diverse classifiers. The diversity in the classifiers allows different decision boundaries to be generated by using slightly different training parameters, such as different training datasets. The intuition is that each classifier will make a different error, and strategically combining these classifiers can reduce total error. Thus, MCS takes advantage of the so-called *instability* of the weak classifier and in turn generates a strong classifier [2-4]. Ensemble or MCS have attracted a great deal of attention over the last decade due to their reported superiority over single classifier systems on a variety of applications [5-7].

The ensemble approach has been widely used with a variety of algorithms to improve the generalization performance of a classification system. However, using this approach to solve the problem of incremental learning has been mostly unexplored. Recognizing the potential of this approach to solve the incremental learning problem, we have recently developed Learn++, where we have shown that Learn++ is indeed capable of incrementally learning from new data, without forgetting previously acquired knowledge and without requiring access to previous data, even when additional datasets introduce new classes [8]. The general approach in Learn++, much like those in other MCS algorithms, such as AdaBoost [9], is to create an ensemble of classifiers, where each classifier learns a subset of the dataset. The classifiers are then combined using weighted majority voting [10]. Learn++ differs from other techniques, however, in the way the data subsets are chosen to allow incremental learning of new data [8, 11].

* Corresponding author

0-7803-8566-7/04/$20.00 © 2004 IEEE.

Recognizing the above mentioned conceptual similarity between incremental learning and data fusion, we have evaluated Learn++ on benchmark and real world application requiring data fusion [12]. New ensembles of classifiers were generated from datasets comprised of different features, which were then combined using weighted majority voting. Although promising, the algorithm certainly had much room for improvement when initially used in the data fusion mode. Some of these gaps have now been filled, and in this paper, we describe how Learn++ can be used more efficiently as a general purpose approach for a variety of data fusion applications along with promising results on a real world application.

1.3 Ensemble approaches for data fusion

Several approaches have been developed for data fusion, for which ensemble approaches constitute a relatively new breed of algorithms. Traditional methods are generally based on probability theory, such as the Dempster-Schafer (DS) theory and its many variations. However, DS based algorithms require specific knowledge of the underlying probability distribution, which may not be readily available.

The majority of these algorithms have been developed in response to the needs of military applications, most notably target detection and tracking [13-15]. Ensemble approaches seek to provide a fresh and a more general solution for a broader spectrum of applications. Such approaches include simpler combination schemes such as majority vote, threshold voting, averaged Bayes classifier, maximum/minimum rules, and linear combinations of posterior probabilities [16,17]. More complex data fusion schemes are also widely used, including ensemble based variations of DS, neural and fuzzy systems, and stacked generalization [18 - 23].

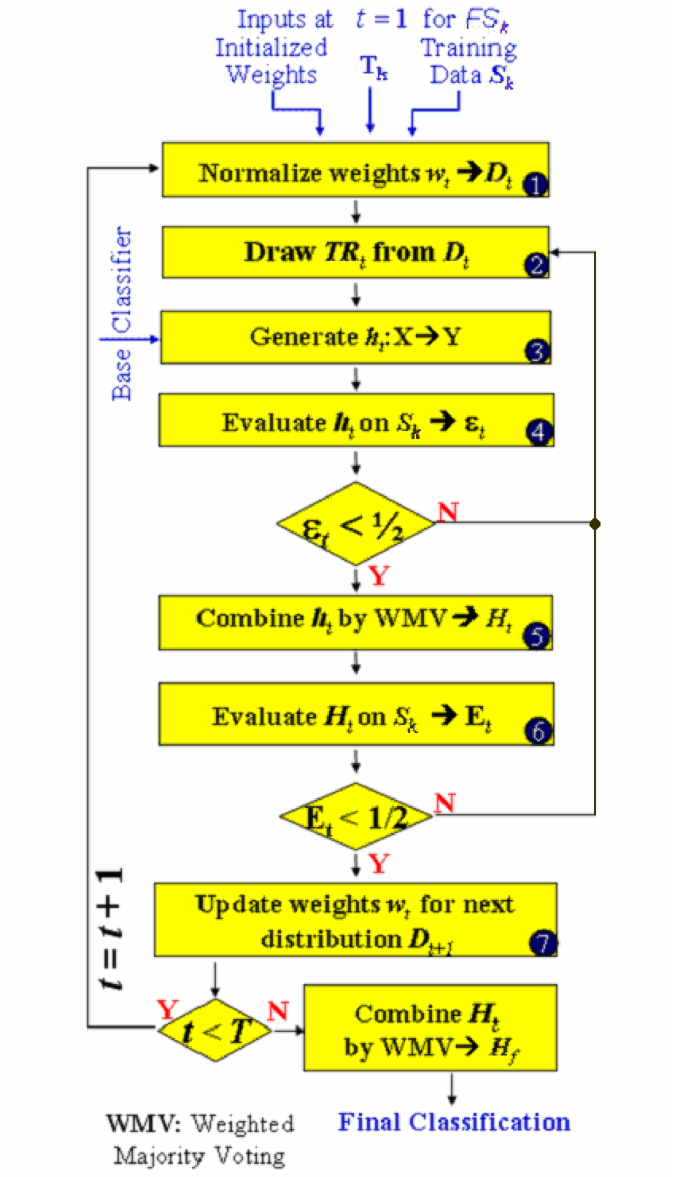
A related approach to data fusion and classifier combination schemes is input decimation: the use of different feature subsets in multiple classifiers [24, 25]. Input decimation can be useful in allowing different modalities, such as Fourier coefficients and pixel averages, to be naturally grouped together for independent classifiers [24]. Input decimation can also be used to lower the dimensionality of the input space by "weeding out features that do not carry strong discriminating information" [25].

A useful addition to this list of approaches would be a more general structure capable of using a variety of different classifier architectures and containing the ability to combine their outputs for (i) a stronger overall classifier, (ii) a classifier capable of incremental learning, and (iii) a classifier capable of data fusion. In this paper we introduce such an alternative to data fusion algorithms.

2 Learn++

The novelty of Learn++ is its incremental learning capability. It can learn new information as and when new data become available, without forgetting the previously acquired knowledge and without requiring access to the previous data, hence without suffering from catastrophic forgetting [26]. Specifically, Learn++ generates an ensemble of relatively weak classifiers for each new database that becomes available, where the outputs of each individual classifier of the ensemble are combined through weighted majority voting to obtain the final classification.

Weak / diverse classifiers are trained on a subset of the training data, randomly selected from a dynamically updated distribution over the training data instances. This distribution is biased towards those instances that have not been properly learned or seen by the previous ensemble(s). A block diagram demonstrating the Learn++ algorithm, as applied to the data fusion problem is provided in Figure 1, and is described in detail in the following paragraphs.



Inputs at $t=1$ for FS_k
Initialized Weights
T_k
Training Data S_k
Normalize weights $w_t \rightarrow D_t$ 1
Draw TR_t from D_t 2
Generate h_t:X→Y 3
Evaluate h_t on $S_k \rightarrow \varepsilon_t$ 4
$\varepsilon_t < ½$
N
Y
Combine h_t by WMV → H_t 5
Evaluate H_t on $S_k \rightarrow E_t$ 6
$E_t < 1/2$
N
Y
Update weights w_t for next distribution D_{t+1} 7
$t < T$
Y
N
Combine H_t by WMV→ H_f
Base Classifier
$t = t + 1$
WMV: Weighted Majority Voting
Final Classification

Figure 1. The Learn++ algorithm for data fusion

For each database, FS$_k$, $k=1,\dots,K$, comprised of a different set of features (obtained from the same particular application) that is submitted to Learn++, the inputs to the algorithm are (i) a sequence S_k of m_k training data instances $\boldsymbol{x_i}$ along with their correct labels y_i ; (ii) a supervised classification algorithm BaseClassifier, generating individual classifiers (henceforth, hypotheses); and (iii) an integer T_k, the number of classifiers to be generated for the k^{th} database.

The only requirement on the BaseClassifier algorithm is that it can obtain better than 50% correct classification performance on its own training dataset, so that a minimum reasonable performance can be expected from each classifier. Note that for a two-class problem, 50% performance is equivalent to random guessing. BaseClassifier can be any supervised classifier such as a multilayer perceptron, radial basis function, or a support vector machine. Their weakness can be controlled by adjusting their size and error goal with respect to the complexity of the problem. Sufficiently different decision boundaries can then be generated by these weak classifiers by training them with slightly different training datasets. It should be noted that most of the resources in generating a strong classifier are typically spent in fine-tuning the decision boundary. Since Learn++ requires only a rough estimate of the decision boundary from its weak classifiers, the expensive step of fine-tuning is avoided. This saves on computational time during training, and also helps prevent overfitting of the training data.

Each hypothesis h_t is trained on a different subset of the training data. This is achieved by initializing a set of weights for the training data, $\boldsymbol{w_t}$, and a distribution $\boldsymbol{D_t}$ obtained from $\boldsymbol{w_t}$. According to this distribution a training subset TR_t is drawn from the training data at the t^{th} iteration of the algorithm. The distribution $\boldsymbol{D_t}$ determines which instances of the training data are more likely to be selected into the training subset TR_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. At each subsequent iteration *t*, the weights previously adjusted at iteration *t-1* are normalized to ensure a legitimate distribution $\boldsymbol{D_t}$ (step 1).

Training subset TR_t is drawn according to $\boldsymbol{D_t}$ (step 2), and the weak classifier is trained on TR_t in step 3. A hypothesis h_t is generated by the t^{th} classifier, whose error ε_t, is computed on the entire (current) database S_k as the sum of the distribution weights of the misclassified instances (step 4)

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

As mentioned above, the error, as defined in Equation (1), is required to be less than 0.5 to ensure that a minimum reasonable performance can be expected from h_t. If this is the case, the hypothesis h_t is accepted and the error is normalized to obtain the normalized error.

$$\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}, 0 < \beta_t < 1 \tag{2}$$

If $\varepsilon_t \geq 0.5$ then the current hypothesis is discarded, and a new training subset is selected by returning to step 2. All *t* hypotheses generated thus far are then combined using a voting scheme to obtain a composite hypothesis H_t (step 5).

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

The voting scheme used by Learn++ is not quite democratic. Each hypothesis is assigned a weight as the logarithm of the reciprocal of its normalized error. Therefore, those hypotheses with smaller training error, indicating better performances, are awarded with a higher voting weight and thus have more say in the final classification decision. The error of the composite hypothesis H_t is then computed in a similar fashion as the sum of the distribution weights of the instances that are misclassified by H_t (step 6)

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

where, | · | evaluates to 1, if the predicate holds true and 0 otherwise. The normalized composite error B_t is obtained as

$$B_t = \frac{E_t}{1-E_t}, 0 < B_t < 1 \tag{5}$$

which is then used for updating the distribution weights assigned to individual instances

$$w_{t+1}(i) = w_t(i) \times \begin{cases} \mathrm{B_t}, H_t(x_i) = y_i \\ \\ 1, otherwise \end{cases} \tag{6}$$

Equation (6) indicates that the distribution weights of the instances correctly classified by the composite hypothesis H_t are reduced by a factor of B_t ($0<B_t<1$). Effectively, this increases the weights of the misclassified instances making them more likely to be selected to the training subset of the next iteration. We note that this weight update rule, based on the performance of the current ensemble, facilitates incremental learning. This is because, when a new dataset is introduced (particularly with new classes), the existing ensemble (H_t) is bound to misclassify the instances that have not yet been properly learned, and hence the weights of these instances are increased, forcing the algorithm to focus on learning novel information introduced by the new data.

At any point, a final hypothesis H_{final} can be obtained by combining all hypotheses that have been generated thus far.

Specifically for the data fusion applications, an ensemble of classifiers is generated as described above for each of the dataset (that uses a different set of features), but also an additional set of weights are introduced for each ensemble. These weights can be assigned based on former experience, if reliable prior information is available about the individual feature set (e.g., for the application of non destructive testing and evaluation of pipelines to identify defects in them, we may know that ultrasonic testing is usually more reliable then magnetic flux leakage data, and we may therefore choose to give a higher weight to the classifiers trained with ultrasound data), or they can be based on the performance of the ensemble trained on the particular feature set on its own training data. If such a weight assignment strategy is chosen, the weight of each classifier would be multiplied by the weight assigned to the ensemble to which it belongs. This adjusted weight of each classifier is then used during the weighted majority voting for the final hypothesis H_{final}

$$H_{final}(\mathbf{x}) = \arg\max_{y \in Y} \sum_{k=1}^{K} \sum_{t:h_t(\mathbf{x})=\mathbf{y}} \log\left(\frac{1}{\beta_t Er_k}\right) \quad (7)$$

where, Er_k is the optional weight assigned to the ensemble trained using a dataset from FS_k (and can assume of a value of 1 if this additional weight is not assigned). For the data fusion application discussed in this paper, Er_k was chosen to be the ratio of the number of instances misclassified by the composite hypothesis of the k^{th} ensemble to the total number of instances in its training dataset S_k. Learn++ used in the data fusion setting is illustrated in Figure 2.

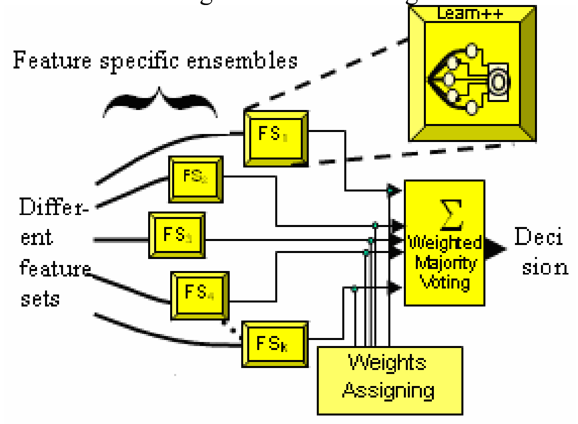


Figure 2. Pictorial representation of algorithm

To summarize, there are three sets of weights employed by the algorithm when used in the data fusion mode. The first two are common for incremental learning and data fusion, the last one is specific to a data fusion application and can be considered as optional. These weights are as follows:

- The weights assigned to the instances in the training data, used in determining which instances are more likely to be drawn into the next training subset for the next classifier.
- The weights assigned to each classifier based on its performance on its training data. These weights are used in weighted majority voting. The higher the training performance of the classifier, the higher voting weight and the more say it has in the final classification.
- The (optional) weight assigned to the entire ensemble of classifiers trained on data sourcing from a particular set of features. They also play a role in weighted majority voting in the final hypothesis. These weights may be assigned based on prior information (if available and reliable) or based on the performance of the ensemble on its training data (similar to weight assigned to the individual classifiers as explained above).

Simulation results of Learn++ on incremental learning using several datasets, as well as comparisons to the other methods of incremental learning such as Fuzzy ARTMAP can be found in [8] and references within. The simulation results of Learn++ on data fusion are presented below.

3 Results

While Learn++ was originally developed as an incremental learning algorithm, its ensemble structure allows it to be used in data fusion applications as well. This is because the algorithm can accept a new dataset even if it contains completely different features as compared to the data the algorithm has previously seen. When used in data fusion mode, Learn++ seeks to incrementally learn novel information content from databases that come from the same application but are composed of different features.

Implementing data fusion using Learn++ with the ensemble approach was tested on a real world application – identifying defects in pipelines using non-destructive techniques. Two datasets containing different features were fused. The first was a set of Magnetic Flux Leakage (MFL) images, and the second was a set of Ultrasonic Testing (UT) images. Both modalities can be used to detect and identify defects in pipes. Illustrations of these images along with the type of defect they represent are shown in Figure 3.

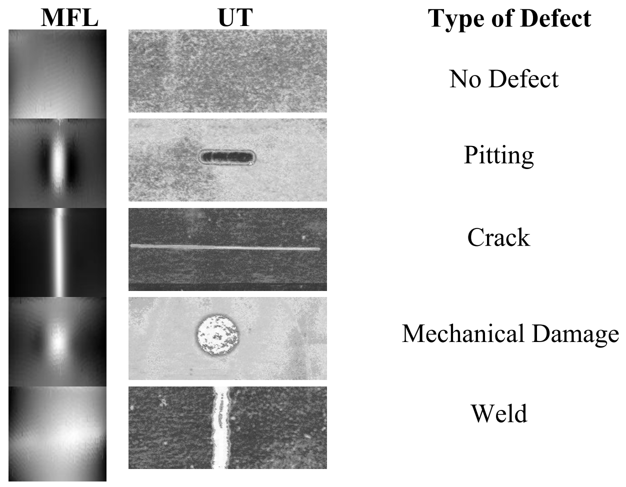


Figure 3. Sample MFL and UT images of defect types

The database consisted of 21 images from to a total of 5 classes: (i) No defect: 4 images; (ii) Pitting: 9 images; (iii) Crack: 4 images; (iv) Mechanical Damage: 4 images; (v) Weld: 4 images. Ten images (2 from each class) were used as the training data and the remaining 11 as the testing data. This distribution was kept constant for all dataset shuffles for different runs and to perform cross validation.

The Learn++ algorithm was run several times in data fusion mode with different combinations of the parameters such as the error goal and number of hidden layer nodes in the MLP networks. Classifiers were added to the ensemble until classification performance leveled off beyond a certain number of classifiers. The algorithm was run using a single hidden layer MLP as the base classifier with the following parameters (observed to be the optimum range based on prior experience): error goal: 0.05 ~ 0.08 in steps of 0.01, and number of hidden layer nodes: 5 ~ 45 in steps of 5. Every possible combination of the above parameters was used. Also, the experiment was duplicated using a different partition of the data (a different selection of the instances used for training and testing). Therefore, there were 36 generalization performance values for each partition of data, yielding a total of 72 generalization performances. The number of classifiers in an ensemble trained on each of the two feature sets could vary from 1 to 50 classifiers. The results obtained are summarized in Table 1. The second column of Table 1 indicates the percentages of different comparisons between the data fusion performance and the individual MFL and UT performances, out of the 72 experiments. For example, in 31.94% (23 out of the 72) of the simulations, the data fusion performance was better than either of the individual MFL or UT performance. Similarly, 40.28% of the times (29 out of 72 simulations), the data fusion performance was the same as the higher of the MFL or UT performances, and so on. Adding the numbers in rows 5 and 6, it can be seen that the proportions of undesirable cases (when the data fusion performance is the lower of MFL and UT or worse than both) is about 11.11% of the times data fusion was performed (8 out of 72). The most desirable cases (when the data fusion performance is the higher of MFL and UT or better than both) are about 72.22% of the times (52 out of 72) data fusion was performed. These results were promising but not optimal.

Table 1: Comparing the data fusion performance to the individual performance of MFL (MFLp) and UT (UTp)

Data fusion performance combining two feature sets is	Proportions %
Greater than max (MFLP, UTP)	31.94
Equal to max (MFLP, UTP)	40.28
Equal to both MFLP and UTP	9.72
In between min (MFLP, UTP) and max (MFLP, UTP)	6.94
Equal to min (MFLP, UTP)	8.33
Less than min (MFLP, UTP)	2.78
Total	100

We have noticed that the strategy used to determine the number of classifiers in each feature set, results in the number of classifiers in different ensembles to vary substantially (e.g., far more classifiers were generated with MFL signals than with UT signals). In a data fusion application, carrying out weighted majority voting among ensembles having different number of classifiers results in a bias in the decision. The hypothesis is biased towards the ensemble with more classifiers, and the results obtained, as summarized in Table 1, were thus not optimal. One of the modifications made to the algorithm (to fit the data fusion scenario) was to use a fixed number of classifiers in all ensembles trained on the different feature sets being fused. This introduced a third parameter to be set – number of classifiers, in addition to error goal and number of hidden layer nodes.

The Learn++ algorithm was run again for every possible combination of the following set of parameters: Error goal: 0.05 ~ 0.08 steps of 0.01; Number of hidden layer nodes: 5 ~ 45 steps of 5; Number of classifiers: 10 ~ 60 steps of 10.

The results obtained for these values are shown in Table 2. Similar to Table 1, the second column indicates the percentages (out of 216 simulation experiments) of various comparisons of the data fusion performance with respect to individual MFL or UT performances. We note that the undesirable cases (where the data fusion performance is the lower of MFL and UT, or lower than both) have been completely eliminated, when equal number of classifiers are used. Furthermore, in 72% to 89% of all simulations (varying with respect to the number of classifiers used, 10~60), the data fusion generalization performance was better than either of the individual MFL or UT generalization performances, indicating that the algorithm is indeed able to extract additional information when the two databases are fused.

Table 2: Comparing the data fusion performance to the individual performance of MFL (MFLp) and UT (UTp)

Data fusion performance combining two feature sets is	Proportions %*
Greater than max (MFLP, UTP)	72.22 to 88.89
Equal to max (MFLP, UTP)	5.56 to 27.78
Equal to both MFLP and UTP	0 to 2.78
In between min (MFLP, UTP) and max (MFLP, UTP)	0 to 5.56
Equal to min (MFLP, UTP)	0
Less than min (MFLP, UTP)	0

* vary with number of classifiers

We have also picked the optimal combination of parameters and performed cross validation with respect to different partitioning of the available data into training and test data. The optimal values, as determined by statistical analysis were found to be 0.05 error goal, 30 hidden layer nodes and 30 classifiers trained with each dataset.

For cross validation, we randomly picked 2 instances from each of the 5 classes. The 10 instances thus picked were used as the training data, and the remaining 11 instances were used as the testing data. This was repeated over 40 times to obtain over 40 different partitions of data to train and test the algorithm. The cross validation results are summarized in Table 3. These numbers suggest that the data fusion performance is significantly better than either of the individual MFL and UT performances.

Table 3: Generalization performances - 95% CI

Dataset	Average generalization performance
MFL	81.60 $\pm$ 3.6207 %
UT	79.87 $\pm$ 2.6938 %
Fused	95.02 $\pm$ 1.9985 %

4 Conclusions

Recognizing the conceptual similarities between incremental learning and data fusion, the Learn++ algorithm – originally developed for incremental learning – has been evaluated in a data fusion setting. The algorithm incrementally and sequentially learns data comprised of different sets of features by generating an ensemble of classifiers for each dataset, and then combining them through a modified weighted majority voting scheme. We have evaluated the algorithm on a real world data fusion application for defect identification, where the two different datasets consisted of UT and MFL signals of the same pipeline specimens. The results indicate that the Learn++ algorithm, when used to combine information contained in two datasets, performed significantly better then each of the testing modalities individually. Our simulation results on other benchmark datasets (not presented here for space considerations) indicate that the algorithm can fuse not only two datasets, but also additional datasets. Therefore, the advantage of Learn++ is that data from different measurement modalities or feature groups can be sequentially added without having to retrain the entire system. The ability of the algorithm to learn incrementally as well as to fuse different datasets to extract additional information not available in either dataset makes Learn++ a versatile algorithm. Further testing of the algorithm on additional real world and benchmark data is currently underway.

Acknowledgements

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

References

[1] S. Prabhakar, A. K. Jain, "Decision level fusion in fingerprint verification," *Pattern Recognition*, vol. 35, pp. 861-874, 2001.

[2] L.K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993-1001, 1990.

[3] T.G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization," *Machine Learning*, vol. 40, no. 2, pp. 1-19, 2000

[4] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123-140, 1996.

[5] T.G. Dietterich, "Ensemble methods in machine learning," *Proc. 1st Int. Workshop on Multiple Classifier Systems (MCS 2000)*, LNCS vol. 1857, pp. 1 – 15, Springer: New York, NY, 2000.

[6] T. Windeatt and F. Roli (eds), *Proc. 3rd Int. Workshop on Multiple Classifier Systems (MCS 2002),* LNCS vol. 2364, p. 1-15, Springer: New York, NY, 2002

[7] T. Windeatt and F. Roli (eds), *Proc. 4th Int. Workshop on Multiple Classifier Systems (MCS2003)*, LNCS, vol. 2709, Springer: New York, NY, 2003.

[8] R. Polikar, L. Udpa, S. Udpa, V. Honavar, "Learn++: an incremental learning algorithm for supervised neural networks," *IEEE Trans Systems, Man and Cybernetics*, vol.31, no.4, pp.497-508, 2001.

[9] Y. Freund and R. Schapire, "A decision theoretic generalization of online learning and an application to boosting," *Computer and System Sciences*, vol. 57, no. 1, pp. 119-139, 1997.

[10] N. Littlestone and M. Warmuth, "Weighted majority algorithm," *Information and Computation*, vol. 108, pp. 212-261, 1994.

[11] R. Polikar, J. Byorick, S. Krause, A. Marino, M. Moreton, "Learn++: A classifier independent incremental learning algorithm for supervised neural networks," *Proc. of Int. Joint Conf. on Neural Networks (IJCNN 2002)*, vol.2, pp. 1742-1747, Honolulu, HI, May 2002.

[12] M. Lewitt and R. Polikar, "An ensemble approach for data fusion with Learn++," *Proc. 4th Int. Workshop on Multiple Classifier Systems (MCS 2003)*, LNCS vol. 2709 , pp. 176-185, Springer: New York, NY,2003.

[13] D. Hall and J. Llinas, "An introduction to multisensor data fusion," *IEEE Proceedings*, vol. 85, no. 1, 1997.

[14] D. Hall and J. Llinas (editors), *Handbook of multisensor data fusion*, CRC Press: Boca Raton, FL, 2001.

[15] L. A. Klein, Sensor *and Data Fusion Concepts and Applications*, SPIE Press, vol. TT35: Belingham, WA, 1999.

[16] J. Grim, J. Kittler, P. Pudil, and P. Somol, "Information analysis of multiple classifier fusion," *Proc. 2nd Intl Workshop on Multiple Classifier Systems*, LNCS vol. 2096, pp. 168-177, Springer: New York, NY, 2001.

[17] J. Kittler, M. Hatef, R.P. Duin, J. Matas, "On combining classifiers," *IEEE Trans on Pattern Analysis and Machine Intelligence,* vol. 20, no.3, pp. 226-239, 1998.

[18] L.O. Jimenez, A.M. Morales, A. Creus, "Classification of hyperdimensional data based on feature and decision fusion approaches using projection pursuit, majority voting and neural networks," *IEEE Trans Geoscience and Remote Sensors*, vol. 37, no. 3, pp 1360-1366, 1999.

[19] G.J. Briem, J.A. Benediktsson, and J.R. Sveinsson, "Use of multiple classifiers in classification of data from multiple data sources," *Proc. of IEEE Geoscience and Remote Sensor Symposium*, vol. 2, pp. 882-884, Sydney, Australia, 2001.

[20] F.M. Alkoot, J. Kittler. "Multiple expert system design by combined feature selection and probability level fusion," *Proc of the 3rd Intl Conf on FUSION 2000,* vol. 2, pp. 9-16, 2000

[21] D. Wolpert, "Stacked generalization," *Neural Networks*, vol. 2, pp 241-259, 1992.

[22] L.I. Kuncheva, "Switching between selection and fusion in combining classifiers: an experiment," *IEEE Trans. on Sys., Man and Cyber.*, vol. 32(B), no. 2, pp. 146-156, 2002.

[23] L.I. Kuncheva, "A theoretical study on six classifier fusion strategies, " *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp. 281-286, 2002.

[24] L. Kuncheva and C. Whitaker, "Feature subsets for classifier combination: an enumerative experiment," *Proc. 2nd Intl Workshop on Multiple Classifier Systems*, LNCS vol. 2096, pp. 228-237, Springer: New York, NY, 2001.

[25] N. Oza and K. Tumer, "Input decimation ensembles: decorrelation through dimensionality reduction," *2nd Intl Workshop on Multiple Classifier Systems*, LNCS vol. 2096, pp. 238-247, Springer: New York, NY, 2001.

[26] R. French, "Catastrophic forgetting in connectionist networks," *Trends in Cognitive Sciences*, vol. 3, no.4, 1999.