2005 7th International Conference on Information Fusion (FUSION)

A Multiple Classifier Approach for Multisensor Data Fusion

Devi Parikh
Electrical and Computer Engineering
Rowan University
Glassboro, NJ, USA.
parikh55@students.rowan.edu

Robi Polikar
Electrical and Computer Engineering
Rowan University
Glassboro, NJ, USA.
polikar@rowan.edu

Abstract - *In many applications of pattern recognition and automated identification, it is not uncommon for data obtained from different sensors monitoring a physical phenomenon to provide complimentary information. In such applications, data fusion - a suitable combination of the complimentary information - can offer more insight into the phenomenon than any of the individual data sources. We have previously introduced Learn^{++}, an ensemble based approach, as an effective automated classification algorithm that is capable of learning incrementally. Recognizing the conceptual similarity between data fusion and incremental learning, our approach is then to employ an ensemble of classifiers generated by using all of the data sources available, and strategically combine their outputs. We have observed that the prediction ability of such a system was significantly and consistently better than that of a decision based on a single data source across several benchmark and real world databases.*

Keywords: 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 or weighted product rule, 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 learned information. The ability of a classifier to learn in this fashion is referred to as *incremental learning*.

It many applications data available from multiple sources underlying the same physical 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; or diagnostic information may be obtained from several different medical tests. 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 based 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 random sampling of 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 systems 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 [5-8].

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^{++}, and shown that Learn^{++}

0-7803-9286-8/05/$20.00 ©2005 IEEE

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 [9]. The general approach in Learn^{++}, much like those in other MCS algorithms, such as AdaBoost [10], 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 [11]. Learn^{++} differs from other techniques, however, in the way the data subsets are chosen to allow incremental learning of new data [9, 12].

Recognizing the above mentioned conceptual similarity between incremental learning and data fusion, we have evaluated Learn^{++} on a benchmark application requiring data fusion [13]. New ensembles of classifiers were generated from datasets comprised of different features, which were then combined using weighted majority voting. The versatility of the algorithm to perform data fusion in varying scenarios, and its ability of fuse information from more than two sources has been demonstrated here. 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 two real world applications as well as a benchmark database.

1.3 Ensemble based approaches for data fusion

Several approaches have been developed for data fusion, for which ensemble based approaches constitute a relatively new breed of algorithms. Traditional methods are generally based on probability theory (Bayes theorem, Kalman filtering),or decision theory such as the Dempster-Schafer (DS) theory and its many variations.

The majority of these algorithms have been developed in response to the needs of military applications, most notably target detection and tracking [14-16]. Ensemble of classifiers based 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 [17,18]. More complex data fusion schemes are also widely used, including ensemble based variations of DS, template matching, neural and fuzzy systems, and stacked generalization [19 - 24]. Another related approach to data fusion using classifier combination schemes is input decimation: the use of different feature subsets in multiple classifiers [25, 26]. Input decimation can be useful in allowing different modalities, such as Fourier coefficients and pixel averages, to be naturally grouped together for independent classifiers [25]. 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" [26].

A useful addition to this list of techniques 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) incremental learning, and (iii) multisensor data fusion. In this paper we demonstrate the potential of Learn^{++} to offer such an alternative to existing 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 [27]. Specifically, the algorithm 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 or 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 illustrating the Learn^{++} algorithm, as applied to the data fusion problem, is provided in Figure 1, and is described in detail in the following paragraphs.

For each database, FS_k, $k=1,...,K$, 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 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. We note that for a two-class problem, 50% performance is equivalent to random guessing, and that is the least we can ask from a classifier. BaseClassifier can be any supervised classifier such as a multilayer perceptron, radial basis function, or a support vector machine, as their weaknesses can be controlled by adjusting their size and/or error goal with respect to the complexity of the problem. Sufficiently different decision boundaries can then be generated by these classifiers by training them with randomly selected training data subsets.

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. Using weak BaseClassifiers, therefore, not only saves computational time during training, but also helps prevent overfitting of the training data.

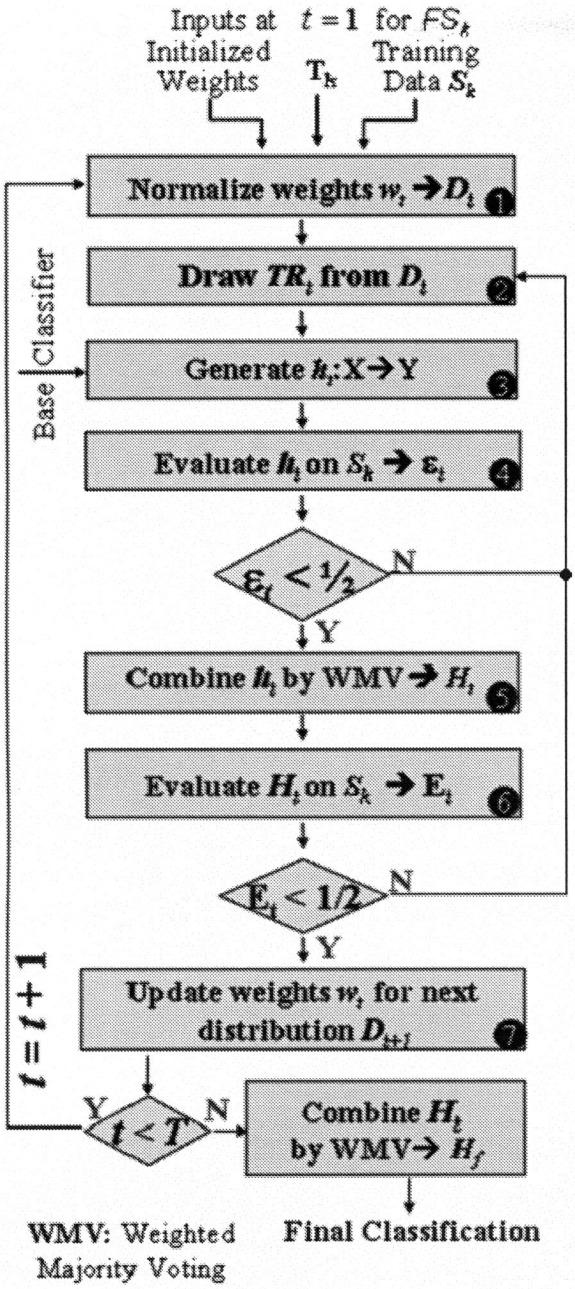

Fig. 1. The Learn^{++} algorithm for data fusion

Each hypothesis h_t, generated during the t^{th} iteration of the algorithm, is trained on a different subset of the training data. This is achieved by initializing a set of weights for the training data, w_t, and a distribution D_t obtained from w_t. According to this distribution a training subset TR_t is drawn from the training data. The distribution 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 D_t (step 1).

Training subset TR_t is drawn according to 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 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 $\varepsilon_t \geq$ ½, 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 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)

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

where, $|\cdot|$ evaluates to 1, if the predicate holds true and 0 otherwise. The normalized composite error B_t is

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

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

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

Eq. (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 ($0<B_t<1$). When the distribution is re-normalized during step 1 of the next iteration, the weights of the misclassified instances are effectively increased, 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 or features), 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 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}

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

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 shown in Fig. 2.

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 using the algorithm in the incremental learning or data fusion mode, whereas the last one is specific to using the algorithm in the data fusion mode only. 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 training subset for the next classifier.

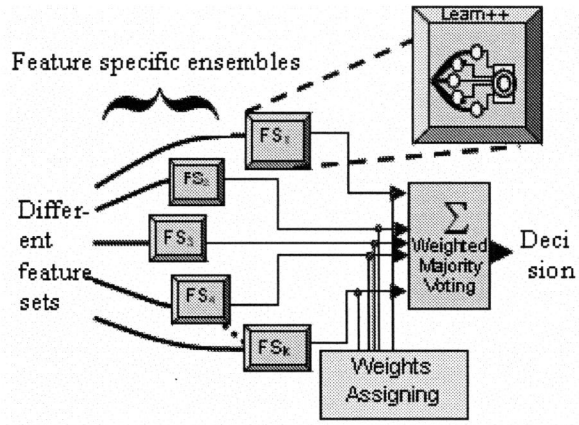

Fig. 2. Schematic representation of algorithm

- 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 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 [9] 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 datasets that come from the same application but are composed of different features.

Implementing data fusion using Learn^{++} with the ensemble approach was tested on three databases, two real world applications and a benchmark database from the UCI Repository of Machine Learning.

3.1 Nondestructive evaluation (NDE) of pipelines database

Nondestructive evaluation is primarily concerned with detection and identification of flaws in various types of materials. Data fusion methods that specifically target NDE data have been developed, which usually that take advan

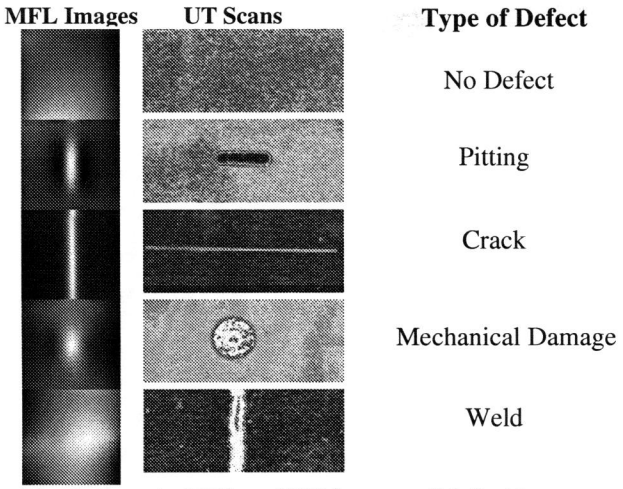

Fig. 3. Sample MFL and UT images of defect types

tage of one or more of the methods mentioned above [28, 29]. 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 Fig. 3.

Two-dimensional discrete Fourier transform based features were extracted from each imaging modality with 15 features for MFL and 72 features for the UT. 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 randomly selected for training and the remaining 11 were used for validation. This distribution was kept constant for all datasets while the data were shuffled for different random selections of the training set for cross validation purposes.

The algorithm was evaluated using a single hidden layer MLP as the base classifier with every possible combination of the following parameters: error goal ranging from 0.05 to 0.08 in steps of 0.01, number of hidden layer nodes ranging from 5 to 45 in steps of 5 and the number of classifiers ranging from 10 to 60 in steps of 10. The simulations results obtained by using these parameters are shown 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 216 experiments. For example, in 72.22% to 88.89% of the simulations (156 to 192 out of 216 simulations) - varying depending on the number of classifiers – the data fusion performance was better than either of the individual MFL or UT performance. Similarly, 5.56% to 27.78% of the times (12 to 60 out of 216 simulations), the data fusion performance was the same as the higher of the MFL or UT performances, and so on. We note that the undesirable cases (where the data fusion performance is the lower of MFL and UT, i.e. row 5, or lower than both,

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)	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

i.e., row 6) never occur. The results indicate that the algorithm is indeed able to extract additional information when the two databases are fused.

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 validation data. This was repeated 40 times to obtain 40 different partitions of data to train and validate the algorithm. The cross validation results are summarized in Table 2. These numbers suggest that the data fusion performance is significantly better than either of the individual MFL and UT performances, even when the classifier parameters are optimized for each feature set.

Table 2: Generalization performances - 95% CI

Dataset	Average generalization performance
MFL	81.60 ± 3.62 %
UT	79.87 ± 2.69 %
Fused	95.02 ± 2.00 %

3.2 Multiple features database

The multiple features database was obtained form the UCI Machine Learning Repository. It consisted of features of handwritten numerals ('0'-'9') extracted from a collection of Dutch utility maps. It had 200 instances from each of the 10 classes. There were 6 feature sets as listed below with a total of 649 features:

Feature Set 1: Profile correlations – 216 features
Feature Set 2: Fourier coefficients – 76 features
Feature Set 3: Karhunen Loeve coef. – 64 features
Feature Set 4: Pixel averages – 240 features
Feature Set 5: Zernike moments – 47 features
Feature Set 6: Morphological features – 6 features

Three hundred out of the available 2000 instances were used for training, and the rest were used for validation.

The algorithm was evaluated on this database for every possible combination of the following range of parameters for each of the feature sets over 2 different partitions of the available data: error goal ranging from 0.01 to 0.06 in steps of 0.01, number of hidden layer nodes ranging from 10 to 25 in steps of 5 and number of classifiers ranging from 1 to 35 in steps of 1.

Statistical analysis was performed over these various combinations of performance values (a total of 1680 such combinations) and an optimum set of parameters was picked for each feature set, indicated in Table 3.

Table 3: Optimum parameters picked for each feature set

Feature Set	Error goal	No. hidden layer nodes	No. classifiers
1	0.01	20	10
2	0.01	10	10
3	0.01	15	10
4	0.01	10	10
5	0.01	25	10
6	0.01	25	10

The Learn^{++} algorithm was evaluated using these parameters over the corresponding feature sets for 50 different partitions of the data and data fusion was performed. All 6 feature sets were fused. The results obtained are summarized in Table 4.

Table 4: Generalization performances - 95% CI

Feature Set	Average generalization performance
1	93.23 ± 0.25 %
2	76.74 ± 0.30 %
3	93.46 ± 0.20 %
4	93.06 ± 0.29 %
5	79.40 ± 0.29 %
6	70.93 ± 0.33 %
Fusion	96.57 ± 0.17 %

Consistent with the non-destructive evaluation of pipelines database, the classification performance obtained by fusing the feature sets is better than each individual feature set, with statistical significance at a 95% confidence level – even when the classifier performances are optimized for each feature set. We note that such optimization is important, because when the parameters are optimized for each feature set, there is less to be gained from the fusion – unless the data fusion algorithm is indeed extracting and merging complimentary information.

3.3 Volatile organic compounds database

This database was produced from responses of twelve quartz crystal microbalances (12 features) to twelve volatile organic compounds (VOC). The twelve VOCs (classes) included acetone (AC), acetonitrile (ACN), toluene (TL), xylene (XL), hexane (HX), octane (OC), methanol (ME), ethanol (ET), methyethylketone (MEK), tricholoroethylene (TCE), tricholoroethane (TCA), and dicholoroethane (DCA). There were seven responses for each VOC, which represent seven different concentration levels, namely 70, 140, 210, 280, 350, 500, and 700 parts per million (ppm). There were in all 84 instances (7 from each class). Four instances were randomly picked from each class and were used for training while the remaining 36 instances were used for testing. The available 12 features were randomly divided into three feature sets (with four features each) to simulate a data fusion scenario, as shown below:

- Feature Set 1: Features 1, 9, 2, 8
- Feature Set 2: Features 10, 12, 7, 4
- Feature Set 3: Features 11, 5, 3, 6

The range of parameters used for each feature set was as follows. Error goals: 0.0005, 0.001, 0.003, 0.005, 0.007, 0.009, 0.01, 0.03, 0.036, 0.043, 0.05; number of hidden layer nodes ranging from 15 to 45 in steps of 10, and the number of classifiers ranging from 1 to 15 in increments of 1. Every possible combination of the above parameters was simulated over three different partitions of the database. Statistical analysis was performed over these 1980 combinations of performance values (for each feature set) and a set of optimum parameters was determined for each feature set. These are indicated in Table 5.

Table 5: Optimum parameters picked for each feature set

Feature Set	Error goal	No. hidden layer nodes	No. Classifiers
1	0.0005	15	5
2	0.0005	15	5
3	0.0005	45	5

Cross validation was performed over 100 partitions of the data to obtain a best estimate of the true fusion performance. All three feature sets were fused. The results are summarized in Table 6.

Table 6: Generalization performances - 95% CI

Feature Set	Average generalization performance
1	84.56 ± 1.04 %
2	86.06 ± 1.32 %
3	86.14 ± 1.10 %
Fusion	91.28 ± 0.86 %

Consistent with the previous two experiments, the data fusion performance was better than each of the individual feature set performance, with statistical significance at a 95% confidence level. It should also be noted, that in all three databases, the width of the confidence interval for the fusion performance was narrower, indicating a more consistent performance as compared to the individual feature sets.

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 two real world data fusion applications: identifying defect types from UT and MFL images and identifying the type of a volatile organic compound present in the environment based on gas sensor responses. We have also evaluated the algorithm on a benchmark database specifically designed for data fusion simulations: the multiple features database obtained from the UCI Repository of Machine Learning. The results indicate that the Learn^{++} algorithm, when used to combine information contained from two or more datasets, consistently performed significantly better then each of the testing modalities individually. Also, the confidence intervals of the fusion performance were consistently narrower compared to those obtained with individual feature sets, indicating a more robust performance. 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, with an added advantage of improved classification performance. 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 databases is currently underway. Experiments are being conducted to track the classification performance as feature sets are fused one at a time. Also, the effects on fusion performance of combining the feature sets in different orders (ascending or descending order of their performance) are being observed. Tests are also being conducted to observe the sensitivity of the data fusion performance to varying parameters (for instance, using randomly selected moderate parameters for the individual feature sets, as opposed to optimized parameters). It is expected that fusing feature sets that have been optimized yields a smaller margin of increase in performance using data fusion and can be used as a fine tuning step. On the contrary, fusing features that are not optimally obtained would provide a larger margin of improvement on the fusion performance, and thus can be used as an alternative for the expensive optimization process.

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] 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] L.I. Kuncheva, *Combining Pattern Classifiers – Methods and Algorithms*, Hoboken, NJ: Wiley Interscience, 2004.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

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

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

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

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

[28] X. E. Gros, *NDT Data Fusion*, Arnold Publishers, 1997.

[29] T. Baun and E. Blasch, "Multisensor fusion of Acoustic Wave and Eddy Current Techniques for Part Inspection," *Proceedings of the 22th Review of Progress in Quantitative Nondestructive Evaluation*, Seattle, 2002.