An Ensemble of Classifiers Approach for the Missing Feature Problem

Stefan Krause and Robi Polikar
Electrical and Computer Engineering, Rowan University,
136 Rowan Hall, Glassboro, NJ 08028, USA.

Abstract – A new learning algorithm is introduced that can accommodate data with missing features. The algorithm uses an ensemble of classifiers approach. The classifiers in the ensemble are trained with random subsets of the total number of available features. The approach takes advantage of the basic assumption that an unknown subset of the features is in fact adequate for the classification, or in other words, that are redundant, and possibly irrelevant features in the data. This assumption is in general true for most practical applications. We empirically show that if a certain number of networks produce a particular classification performance using all of the features, then the same classification performance can be reached even if some features are missing, as long as the same number of *useable* networks can be generated with the missing features. The proposed approach has its roots in the incremental learning algorithm, Learn^{++} which seeks to learn new information that is provided by additional datasets that may later become available, even when such data introduce new classes. We have modified the Learn^{++} algorithm for addressing the missing feature problem. The proposed algorithm showed surprisingly remarkable performance on three real-world applications, with up to 10% of the features missing in the validation / field data.

I. INTRODUCTION

A. The Missing Feature Problem

Most commonly used classification algorithms, including neural networks, require that the number and nature of the features be set before the training. Once the training is completed, the validation or the field data must contain the exact same features as the training data for the classification algorithm to determine the corresponding classes. If a particular instance is missing even a single feature, the classifier will not be able to produce a valid output for that input. It is not unusual for training, validation or field data to have missing features in some (or even all) of their instances, as bad sensors, failed pixels, malfunctioning equipment, unexpected noise causing signal saturation, data corruption, etc. are familiar scenarios in many practical applications. The pragmatic approach that is often followed in such cases is simply ignoring those instances with missing features. This rather brute-force approach is suboptimal, however, and may not even be feasible if all instances are missing one or more features. There are other theoretical approaches, many of which rely on Bayesian techniques for extracting class probabilities from partial data, by integrating over missing portions of the feature space [1,2,3,4]. Another approach is searching for the optimal subset of features so that fewer features are required; however, the problem still remains if one (or more) of these optimal features is missing, corrupted or unavailable otherwise.

The proposed algorithm, named Learn^{++}MF, follows an alternate strategy, combining an ensemble of classifiers approach with random feature selection. Basically, the idea is to train an ensemble of classifiers, each trained with a randomly selected subset of the features. When an instance with missing feature(s) needs to be classified, those classifiers trained with only those features that are presently available in the given instance are used to determine the correct classification. These classifiers are henceforth referred to as *usable classifiers*. This approach makes two basic assumptions: First, the feature set is redundant, and hence includes an unknown number of features that are actually not required, or even possibly irrelevant. Second, the features are assumed to be unrelated and/or independent (the value of any feature is independent of all others). While these assumptions are clearly not satisfied in all classification problems, they actually hold true in many practical applications. This is because, most practical applications, knowingly or otherwise, do use a redundant set of features that are in fact independent from each other. It is these applications for which the proposed approach is designed.

B. Ensemble of Classifiers

The predecessor of the proposed algorithm is the previously presented incremental learning algorithm, Learn^{++} [5,6,7]. Learn^{++} generates an ensemble of weak classifiers, each trained with a slightly different distribution of the training data, which are then combined using the weighted majority voting [8]. Learn^{++}, similar to other ensemble approaches, takes advantage of the *instability* of weak classifiers, which allows 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 then eliminates the individual errors, generating a strong classifier.

Using ensemble of classifiers has been well researched for improving classifier accuracy [9,10,11,12,13,14]; however, its potential for addressing the missing feature problem, as well as the incremental learning problem has been mostly unexplored. Learn^{++} was developed in response to recognizing the potential feasibility of ensemble of classifiers in solving the incremental learning problem, whereas Learn^{++}MF is designed to explore the feasibility of using

0-7803-7898-9/03/$17.00 ©2003 IEEE

the ensemble of classifiers approach for solving the missing feature problem.

Using the ensemble of weak classifiers approach has additional benefits. First, the training time is often less for generating multiple weak classifiers compared to training one strong classifier. This is because, strong classifiers spend a majority of their training time in fine tuning the desired decision boundary, whereas weak classifiers completely skip the fine-tuning stage as they only generate a rough approximation of the decision boundary. Intimately related to fast training, weak classifiers are also less likely to suffer from overfitting problems, since they avoid learning outliers, or quite possibly a noisy decision boundary.

Since the ensemble of classifiers is combined through a majority voting process, each classifier can be trained using a different set of features. By creating an ensemble with various subsets of the features used to train each classifier, a subset of the classifiers can still be used when a particular instance is missing certain features. This is because an instance missing certain features can be classified by those classifiers that did not use the missing features for training.

II. THE LEARN^{++}MF ALGORITHM

As mentioned above, the Learn^{++}MF algorithm uses an ensemble of classifiers, each of which is trained with a random subset of the entire feature set. While not essential to the algorithm, we initially assume that the training data has no missing features, and/or there is sufficient training data with all features intact. The algorithm focuses on the more commonly seen, and potentially more annoying case of, field data containing missing features. The algorithm generates a number of classifiers each of which requires only a subset of all features. Hence, when an instance with missing features needs to be identified, only those classifiers that did not use the missing features in their own training data are used for classification. All other classifiers that were trained with one or more of the missing features of the given instance are simply disregarded for the purpose of classifying the current instance.

In order to keep track of which classifiers are used in classifying any given instance, we define the *universal set* and *usable set* of classifiers. The universal set of classifiers includes all classifiers that have been generated by the algorithm thus far. The usable set of classifiers is the instance specific set of actual classifiers that can be used in identifying the given instance. Similarly, we can also define the set of *unusable classifiers*, which for any given instance, are those classifiers that require the features missing in the given instance. We show empirically that if U classifiers yield a certain classification performance on data with no missing features, then a similar performance can be achieved simply by generating additional weak classifiers to obtain a total of U *usable classifiers* on data that have missing features. The assumptions for this property to hold are the availability of a dataset with sufficient redundancy, and a set of weak classifiers each of which is trained with slightly different parameters.

The pseudocode and the block diagram of the Learn^{++}MF algorithm are provided in Fig.1 and Fig. 2, respectively. The inputs to the algorithm are (1) the training data set D; (2) the percentage of features *pof* to be used for training individual classifiers; (3) the number of classifiers to be created T; and (4) the sentinel value *sen* used in the data to designate a missing feature. The data set D contains m number of instances, each with a total of f number of features. The algorithm is set to iteratively run T times, generating one additional classifier (hypothesis) at each iteration. A discrete probability distribution $\boldsymbol{P}_t$ is created, essentially giving a weight to each feature. At each iteration t, a subset of features, $\boldsymbol{F}_{selection}(t)$, is drawn according to this distribution such that those features with higher weights are more likely to be selected into $\boldsymbol{F}_{selection}(t)$ to be used in training current iteration's classifier, h_t. Before the first iteration, $\boldsymbol{P}_1$ is initialized to be uniform, unless there is reason to choose otherwise, so that each feature has equal likelihood of being selected into $\boldsymbol{F}_{selection}(1)$.

Training

Input:

- Sentinel value *sen*.
- Integer T, specifying the number of iterations.
- Training data set $D = \{(\mathbf{x}_i, \mathbf{y}_i) \mid i = 1, \cdots, m\}$ with m instances and f features.
- Percentage of features used to train each classifier *pof*.

Initialize $P_1(j) = 1/f, \forall j, j = 1, \cdots, f$

Do for $t = 1,2,...,T$:

1. Set $P_t = P_t \Big/ \sum_{j=1}^{f} P_t(j)$ so that P_t is a distribution.
2. Draw *pof%* of features for $\boldsymbol{F}_{selection}(t)$ from $\boldsymbol{P}_t$.
3. Generate a weak classifier using only those features in $F_{selection}(t)$ for each instance in training.
4. Obtain a hypothesis h_t : X → Y, and calculate the classification performance $Perf_t$ for h_t. If $Perf_t$ < 50 %, discard h_t and go to step 2.
5. Set $P_t(F_{selection}(t)) = P_t(F_{selection}(t)) \cdot \frac{1}{f}$

(Validation / Testing)

Do for $i = 1,2,...,m$:

6. $\boldsymbol{M}_{feat}(i) = \arg(\mathbf{x}_i(j) == sen), \ \forall j, j = 1, \cdots, f$.
7. $H_t = \arg\max_{y \in Y} \sum_{t=h_t(x)=y} [\,| M_{feat}(i) \notin F_{selection}(t) |\,]$.

end loop

end loop

$H_{final} = H_t$

Fig. 1 Learn^{++}MF algorithm.

This distribution, which is updated later in the algorithm, is normalized in step 1 of the iterative loop, so that its sum is equal to 1, and that a legitimate distribution $\boldsymbol{P}_t$ is obtained

$$P_t = P_t \Big/ \sum_{j=1}^{f} P_t(j) \qquad (1)$$

where j is an index on features. Next, *pof* % of the features are randomly drawn from $\boldsymbol{P}_t$ in step 2. The features selected constitute the set $\boldsymbol{F}_{selection}(t)$. We note that the *pof* value should be selected carefully. A high *pof* value will yield better individual classifiers because each classifier will be trained with more features. However, fewer classifiers will then be available to classify instances with missing features. Conversely, a very low *pof* value will cause the classifiers to be too weak to achieve a meaningful classification performance. Experimental trials have shown that a typical *pof* value should be between 15-40% (of f, the total number of features), depending on the redundancy of the features and/or complexity of the classification problem. Once $\boldsymbol{F}_{selection}(t)$ is obtained, the t^{th} hypothesis h_t is generated in step 3 using the features in $\boldsymbol{F}_{selection}(t)$. The trained classifier h_t is then tested on the training data in step 4. We expect that h_t achieve a minimum of 50% correct classification performance on its training data to ensure that it has a meaningful classification capacity. If this requirement is not satisfied by h_t, then a new $\boldsymbol{F}_{selection}(t)$ is drawn and a new classifier is generated.

Next, the distribution $\boldsymbol{P}_t$ is updated in step 5 according to

$$P_t(F_{selection}(t)) = P_t(F_{selection}(t)) \cdot \frac{1}{f} \qquad (2)$$

such that the weights of those features that appear in $\boldsymbol{F}_{selection}(t)$ are reduced by a factor of f. Those features that were not in the current selection effectively have their weights increased when $\boldsymbol{P}_t$ is normalized again in step 1 of iteration t+1. This feature weight update rule ensures that every feature has an equal probability of being selected, by reducing the weights of those that have been previously selected.

During the validation phase, the algorithm searches for sentinels in each instance to be classified. Therefore, the values for the missing features must have been replaced with the sentinel, *sen,* before validation. The sentinel value should be chosen so that it is greatly out of the range of values that could conceivably occur in the data. This will ensure that actual values are not mistaken for the sentinel value. Therefore all features j, $j=1,\ldots,f$ with a sentinel value in the given instance are flagged and placed into the set of missing features $\boldsymbol{M}_{feat}(i)$ for that instance $\mathbf{x}_i$. Finally, all classifiers h_t whose feature selection list $\boldsymbol{F}_{selection}(t)$ did not include those in $\boldsymbol{M}_{feat}(i)$ (that is, all classifiers that did not use any of the features in $\boldsymbol{M}_{feat}(i)$) are combined through majority voting to determine the classification of instance $\mathbf{x}_i$. This constitutes the composite hypothesis $H_t(i)$ for that instance:

$$H_t(i) = \arg\max_{y \in Y} \sum_{t=h_t(x)=y} [\,|\, M_{feat}(i) \notin F_{selection}(t) \,|\,] \qquad (3)$$

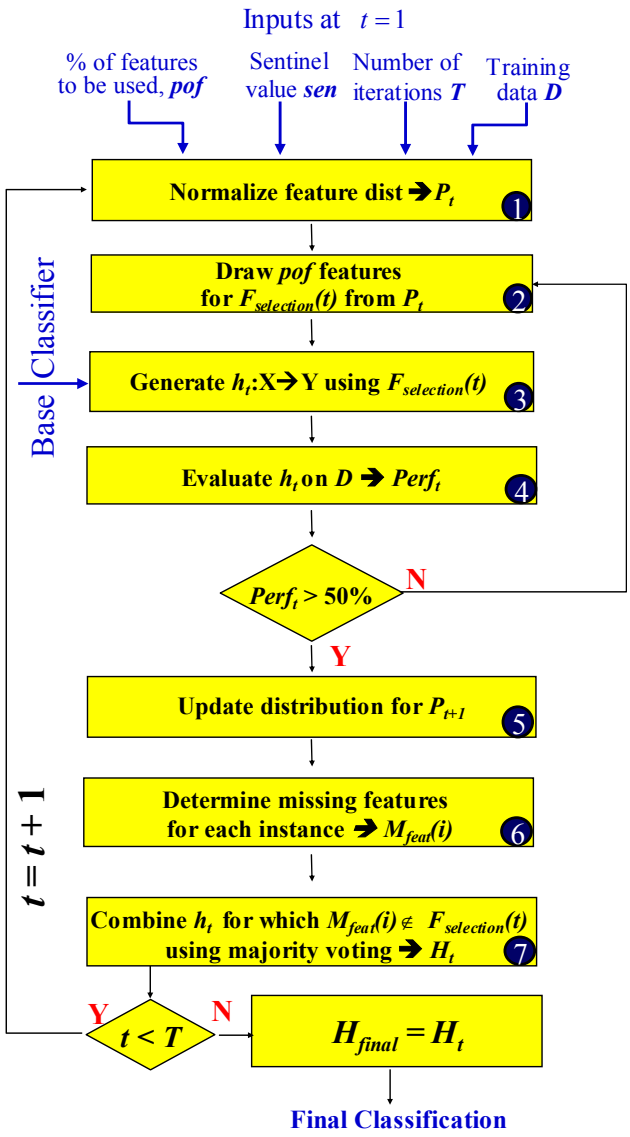

Fig. 2. Block diagram of the Learn^{++}MF algorithm

where [| • |] evaluates to 1 if the predicate holds true. Equation (3) essentially picks the class that receives the highest vote among all classifiers.

After T classifiers are generated, the current composite hypothesis becomes the final hypothesis. T should be chosen large enough to create adequate number of usable classifiers for the problem at hand.

III. Simulation Results

The algorithm was tested on a variety of databases. In each case different percentages of features (the *pof* parameter) were used for training. The databases used included a gas identification database, an optical character recognition database, and a database classifying radar returns through the ionosphere. In all cases, multi-layer perceptron (MLP) networks were used as the base classifier. Since MLPs in

the ensemble were trained as weak classifiers, training parameters, such as the error goal or number of hidden layer nodes did not need fine tuning.

A. Gas Identification (GI) Database

The gas identification database used in this study consisted of responses of six quartz crystal microbalances to five volatile organic compounds, including ethanol (ET), xylene (XL), octane (OC), toluene (TL), and trichloroethelene (TCE). The database included 384 six-dimensional signals. 180 were used for training and 204 were used for testing. The data distribution is shown in Table 1. Each network in the ensemble was trained with 2 out of 6, (*pof* = 33.3%), of the available attributes. We found that 50 usable networks would yield a classification performance of 81.4% when no features were missing. Additional networks were created to ensure that there would still be 50 usable networks with as many as 10% of the features missing in the test data. The ensemble was tested with five sets of test data. Each had a different percentage of features randomly replaced with sentinels to simulate missing features. We note that the *total number of features* in the dataset is defined as the number of features per instance times the number of instances. The percentages of features removed represent the fraction of total number of features that were removed. Five datasets were created for this database with 0.0%, 2.5%, 5.0%, 7.5%, and 10.0% of the features removed. The classification performances are given in Table 2.

TABLE 1.DATA DISTRIBUTION FOR THE GI DATABASE

	Training Data	Test Data
ET	30	34
OC	30	34
TL	50	62
TCE	30	34
XL	40	40
Total	180	204

TABLE 2.PERFORMANCE ON THE GI DATABASE

% Features Missing	Total # of Classifiers	# of Usable Classifiers	Test Perform.
0.0%	50	50	81.4%
2.5%	53	50	81.4%
5.0%	56	50	81.4%
7.5%	59	50	81.4%
10.0%	62	50	81.4%

Intuitively, the best generalization performance was expected from the set with no features missing, with decreasing performances on the sets that had increasing number of missing features. However, we found out that when 50 usable networks were available for each set, the generalization performances were identical. As expected, a larger (62) number of networks was required when 10% of features were missing, to obtain 50 usable networks, as compared to other sets.

B. Optical Character Recognition (OCR) Database

This benchmark database, obtained from the UCI machine learning repository [15], consisted of 1200 training instances and 1797 test instances of digitized hand written characters. The data distribution is shown in Table 3. The characters were digits, 0 through 9, digitized on an 8x8 grid, creating 64 features for 10 classes (two pixels had constant zero values throughout the entire data, and hence were removed, leaving effectively 62 features). Twelve out of 62, (*pof* = 19.3%), features were used for training individual classifiers. For this database, 59 networks achieved a performance of 94.5% on the test data with no missing features. The previously described process of randomly removing features was also repeated for this database. The classification performance values are given in Table 4.

TABLE 3. DATA DISTRIBUTION FOR THE OCR DATABASE

	Training Data	Test Data
0	100	178
1	150	182
2	100	177
3	150	183
4	100	181
5	150	182
6	100	181
7	150	179
8	100	174
9	100	180
Total	1200	1797

TABLE 4. PERFORMANCE ON THE OCR DATABASE

% Features Missing	Total # of Classifiers	# of Usable Classifiers	Test Perform.
0.0%	59	59	94.5%
2.5%	80	59	94.5%
5.0%	110	59	95.0%
7.5%	149	59	92.2%
10.0%	210	59	93.7%

This case did not yield the exact same performance values when features were missing, unlike the GI database. It did, however, have very consistent performances all within 2% of each other. Interestingly, having no features missing does not always guarantee the highest performance. In this case the dataset with 10.0% of the features removed did better than the dataset with 7.5% of the features removed. In fact the dataset with 5.0% features removed performed even better than the dataset with all features present. This can be explained by the presence of irrelevant features whose mere existence may hinder the classification performance [16]. In such cases, a large number of networks are needed to provide the equivalent number of usable networks, e.g., when 10.0% of features were missing, three times as many (210) networks were required to achieve 59 usable networks.

The performance relationship between the total number of classifiers versus usable classifiers illustrates an interesting relationship, as shown in Figs. 3 and 4. The y-axis in each figure is classification performance. The x-axis in Fig. 3 is the total number of classifiers and the x-axis in Fig. 4 is the number of usable networks. In both cases the bold line is the no-missing-features case, plotted as a solid line with X's through it. We note the obvious outcome that it takes longer for those classifiers trained with larger percentage of

features missing to achieve a desired performance figure. However, plotting performance versus number of usable networks, we notice that all plots move right on top of each other (Fig.4). This illustrates that, at least for databases with sufficient redundancy, a large number of features may be missing without affecting the classification performance, when the described ensemble approach is used.

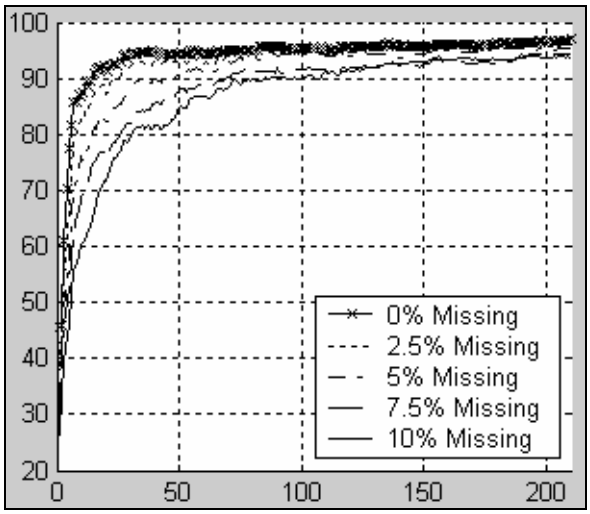

Fig. 3 Performance vs. actual networks on the OCR database

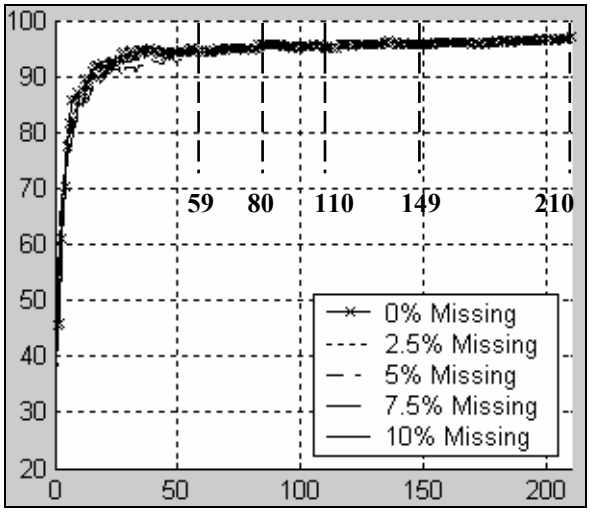

Fig. 4 Performance vs. usable networks on the OCR database

The vertical lines in Fig.4 indicate the end of the plot (# of classifiers generated) for each dataset.

C. Ionosphere Radar Return (ION) Database

This benchmark database, also obtained from the UCI machine learning repository [15], consisted of 220 training instances and 131 test instances of radar returns through the ionosphere. This system consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kW. The targets were free electrons in the ionosphere. "Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not; whose signals pass through the ionosphere. The data distribution is shown in Table 5. Nine out of 34, (*pof* = 26.5%), of the available features were used for training individual classifiers. For this database, 53 usable networks achieved a performance of 94.7% on the test data when no features were missing. The same process of randomly removing features that was performed on the previous two databases for testing was also repeated here. The classification performance values are given in Table 6.

TABLE 5. DATA DISTRIBUTION FOR THE ION DATABASE

	Training Data	Test Data
Good	140	85
Bad	80	46
Total	220	131

TABLE 6. PERFORMANCE ON THE ION DATABASE

% Features Missing	Total # of Classifiers	# of Usable Classifiers	Test Perform.
0.0%	53	53	94.7%
2.5%	67	53	94.7%
5.0%	85	53	94.7%
7.5%	106	53	93.9%
10.0%	142	53	95.4%

This database also produced very consistent classification performances. There was a very slight drop off in performance for the dataset with 7.5% of the features missing. However, there was actually about a one percent increase in performance for the dataset with 10.0% of the features missing as compared to the other three datasets.

One might also wonder, what generalization performance could be achieved, if the entire set of features were made available to the algorithm (as opposed to *pof*%). This would constitute the gold standard to compare the classifier performances with missing features.

TABLE 7. PERFORMANCE ON THE ION DATABASE WITH ALL FEATURES USED

% Features Missing	Total # of Classifiers	# of Usable Classifiers	Test Perform.
0.0%	34	34	94.7 %

As indicated in Table 7, when all 34 features were used during training, the classification performance was virtually identical to the case where only a random 9 out of the 34 features were used, though only 34 (weak) classifiers were required to achieve this performance as opposed to 53. We also note that the equivalence of the number of classifiers used by the algorithm and the number of features (34) is purely coincidental.

IV. DISCUSSIONS & CONCLUSIONS

We presented the Learn^{++}MF algorithm, employing an ensemble of classifiers, as an alternate and practical approach to the missing feature problem. The algorithm generates multiple classifiers, each trained with a subset of the features available. When an instance of unknown label that has missing features is presented to the algorithm, the algorithm simply sorts through all classifiers, and picks those that did not use the missing features in its training. These classifiers are then combined through majority voting.

Three databases, drawn from practical real life applications, were used to evaluate the proposed algorithm. The initial results have been very promising, indicating the feasibility of the approach for those applications that satisfy the two basic assumptions made by the algorithm. First, we assume that the feature set is redundant, with an unknown number of features that are in fact not required or not relevant to the classification problem. Second, we also assume that these features are independent of each other. These assumptions are not overly restrictive, as many practical applications do in fact use a redundant number of independent features. An ideal group of applications would be those using sensors, where the response of each sensor is not affected by those of others. Each of the three applications described above fall into this category.

The algorithm is particularly useful, when one or more of the sensors malfunction, or when some of the data become corrupted. It may be expensive, difficult, impractical or even impossible to recollect such data, making it essential to be able to classify data with missing features. The proposed algorithm was able to accommodate data with up to 10% of the features missing with virtually no performance degradation compared to the case with no missing features. The missing features can be 10% of the sensors not functioning, or 10% of varying number of sensor outputs not being available, or any intermediate scenario.

It was particularly promising to observe that a similar performance level could be achieved when classifying data with missing features as compared to classifying data with no features missing. In fact, in certain cases, the performance with missing features was better than that of no missing features, indicating that some features were indeed redundant.

Future work on this algorithm would investigate the possibility of training the classifiers with a varying amount of features. We will also investigate whether the algorithm can accommodate a larger percentage (than 10%) of missing features. An automated procedure for determining the optimum *pof* value is also being explored.

ACKNOWLEDGEMENTS

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] V. Tresp, R. Neuneier, S. Ahmad, "Efficient methods for dealing with missing data in supervised learning," G. Tesauro, D. S. Touretzky, and Leen T. K., editors, *Advances in Neural Information Processing Systems 7*. MIT Press, 1995.

[2] V. Tresp, S. Ahmad, R. Neuneier, "Training neural networks with deficient data," J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*. Morgan Kaufmann, 1994.

[3] S. Ahmad and V. Tresp, "Some solutions to the missing feature problem in vision," C. L. Giles, Hanson S. J., and Cowan J. D., editors, *Advances in Neural Information Processing Systems 5*. Morgan Kaufman, 1993.

[4] A. Morris, M. Cooke, P, Green, "Some Solutions to the Missing Feature Problem in Data Classification, with Application to Noise Robust ASR," *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, (ICASSP98),* vol. 2, pp: 737 - 740 , 1993.

[5] R. Polikar, L. Udpa, S. Udpa, V. Honavar, "Learn++: An incremental learning algorithm for multilayer neural networks," *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Proc.*, vol. 6, pp. 3414-3417, 2000.

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

[7] R. Polikar, J. Byorick, S. Krause, A. Marino and M. Moreton, "Learn++: A Classifier Independent Incremental Learning Algorithm for Supervised Neural Networks," *Proc. Int. Joint Conf. Neural Networks (IJCNN2002)*, vol. 2, pp. 1742-1747, Honolulu, HI, 2002.

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

[9] 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.

[10] M.I. Jordan and R.A. Jacobs, " Hierarchical mixtures of experts and the EM algorithm," *Proc. of Int. Joint Conf. on Neural Networks (IJCNN 1993),* pp. 1339-1344, 1993.

[11] 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.

[12] 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.

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

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

[15] C.L. Blake and C.J. Merz, UCI Repository of machine learning databases at http://www.ics.uci.edu/~mlearn/ MLRepository.html. Irvine, CA: University of California, Dept. of Information and Computer Science, 1998.

[16] R. Polikar, R. Shinar, L. Udpa, M. Porter, "Artificial intelligence Methods for Selection of an Optimized Sensor Array for Identification of Volatile Organic Compounds," *Sensors and Actuators B: Chemical*, vol. 80, Issue 3, pp 243-254, December 2001.