A Distributed Approach for Prediction in Sensor Networks

Sabine M. McConnell and David B. Skillicorn
School of Computing
Queen's University
{mcconnell,skill}@cs.queensu.ca

Abstract

Sensor networks in which the sensors are capable of local computation create the possibility of training and using predictors in a distributed way. We have previously shown that global predictors based on voting the predictions of local predictors each trained on one (or a few) attributes can be as accurate as a centralized predictor. We extend this approach to sensor networks. We also show that, when the target function drifts over time, sensors are able to make local decisions about when they need to relearn to capture the changing class boundaries.

Keywords: sensor networks, prediction accuracy, classification task, distributed data mining, anomaly detection, local models

1 Introduction

As sensors increasingly become active devices, with their own processing power, rather than simply passive input devices, new possibilities for implementing distributed algorithms in a network of sensors become possible. One important class of such algorithms are predictors, which use the sensor inputs to predict some output function of interest – one topical example is the use of ocean overpressure sensors and seismic detectors to predict a tsunami.

Sensor networks are of two broad kinds: peer-to-peer or hubbed. In a peer-to-peer (or *ad hoc*) network, each sensor has access to some neighbors and the overall network resembles a random graph. In a hubbed network, the network structure is a tree, where leaves are sensors, the root is some more substantial computational device, and the internal nodes may either be ordinary sensors, resemble the root node, or something in between. We will consider only hubbed sensor networks.

The trivial solution to learning a predictor in such a sensor network is of course simply to transmit the data from all of the sensors to the root, and carry out all of the predictor training and prediction at the central site. This has several disadvantages: moving the raw data to the root increases power consumption, uses bandwidth and introduces latency. In addition, the sensors remain passive devices.

We show that prediction in a hubbed sensor network can be distributed in the following way: each sensor learns a local predictive model for the global target classes, using only its local input data. Only the predicted target class for each reading is then transmitted to the root, which determines the appropriate prediction by voting the target classes of the local predictors. This approach has a number of desired properties: it is no less (and potentially even more) accurate than a centralized predictive model; it requires only the target class prediction to be transmitted from the sensor, rather than the entire raw data; and it allows each sensor to locally determine its behavior and respond to its environment, for example deciding when to relearn its local model in response to changes in the input data.

The contributions of this paper are:

- A framework for building and deploying predictors in sensor networks that pushes most of the work out to the sensors themselves. This builds on our earlier work on distributed data mining. We show that prediction performance is not negatively impacted by using the framework instead of a centralized learning approach.

- We show how the use of local predictive models enables sensors to respond to changes in data by relearning when their local predictive accuracy changes. This creates new possibilities, such as allowing sensors to predict only some target classes, for example those representing anomalies, therefore further reducing the required bandwidth.

Because sensors only transmit model predictions, and not raw data, our approach can also be used in settings where privacy is a concern, for example when sensors belong to different organizations or governments.

2 Related Work

The deployment of data mining techniques in sensor networks is an open research problem. For a general discussion of the characteristics and requirements of sensor networks see the survey by Akyildiz *et al.* [1]. One application is the distributed stream mining system Vedas [9] by Kargupta *et al.*, which monitors data streams obtained in moving vehicles before reporting them to a base. More recently, Bandyopadhyay *et al.* [2] introduced a distributed clustering algorithm based on the k-means clustering algorithm in a peer-to-peer environment.

The current limitation of data mining applications in sensor networks is that existing distributed data mining techniques impose heavy demands on computation and/or communication. In addition to the trivial approach of sending all collected data from the individual sensors to the root, where any standard data-mining technique can then be applied, two techniques for distributed data mining are known: the Collective Data Mining (CDM) framework introduced by Kargupta *et al.* [8], and metalearning [5].

The CDM framework encompasses a variety of data mining techniques, while metalearning is algorithm independent. The CDM framework is built upon the fact that any function, i.e. the function to be learned by the data mining algorithm, can be expressed as a (possible infinite) sum of basis functions and their corresponding coefficients. It is those coefficients that are learned from the local datasets and then combined into a global model. This is computationally expensive, and the sum is only approximated, which puts an upper boundary on the accuracy that can be achieved. In metalearning, outputs from classifiers built at separate sites and from separate datasets are combined by sending them as input to a second level of classifiers, which then produce the final result. Because metalearning requires additional resources, both for the transmission of the models as well as the computation of the models themselves, it is not well suited to applications in sensor networks.

3 Learning Predictors in Sensor Networks

Provided that each of the sensors provides an attribute value (reading) at the same time, the global view of the collected data is a matrix. Each row of the matrix corresponds to a (global) reading and each column to an attribute. Each sensor may collect only a single attribute, so that it is responsible for only one column of the dataset, or it may collect multiple attributes. A target class is associated with each global reading; for *training* data, there is a further column containing these target classes. After deployment, the goal is to predict the target class for each global reading.

In our approach, each sensor builds a local predictor that predicts the target class based only on the attribute(s) available to it locally. At first, it might seem as if such predictors would be too poor to be useful, especially given the amount of research directed towards more heavyweight distributed data-mining techniques as described above. However, we have shown [10] that, even when a model is built from each attribute individually, predictive performance is not necessarily worse than that of a centralized predictor. This follows partly from the same reason that Bayesian classifiers are often useful, even when the independence assumption on which they depend is violated. Domingos and Pazzani [6] attribute the success of the naive Bayes classifier to the fact that even though the class probabilities are estimated wrongly, their ranking is preserved, i.e. the class that should be assigned to with the highest probability is still the highest under the naive Bayes assumption. It should also be noted that many predictive model

building algorithms do not account for attribute correlation well, even when they purport to.

Our model of a sensor network is as follows. The network is a tree, with a powerful computational device at the root, and sensors, with limited power, processing, and memory capabilities at the leaves. Bandwidth in the network is a scarce resource, in part because transmission by the sensors consumes power. In addition, available resources vary across different sensor networks. Sensors gather data synchronously (logically if not physically), and the overall goal is to predict some function of all data inputs.

During the learning phase, each sensor builds a predictor for the global target class based on values of its local input data. The accuracy of each local predictor can be determined using test data for which the correct target class predictions are known. During deployment, each sensor receives data from the outside world, and sends its target class predictions for each of these readings (directly or through other sensors) to the root. The root uses voting to choose a global target class from these individual predictions. Note that the root can tell whether each individual sensor's prediction was 'right' or 'wrong' by whether it agreed with the overall vote, and this can act as a surrogate for prediction accuracy for each local predictive model.

There are several variations in this overall approach, depending on the capabilities of the sensors and the particular predictive technique being used. Some predictive techniques require access to all of the training data at once (for example, decision trees), while others require only one reading at a time (for example, neural networks). In both cases, sensors must have enough memory to store the model itself and perhaps some statistics about it; but in the former case, a sensor must also have enough memory to hold the entire training data for its input attribute(s).

When only one reading is stored at a time, the training regime is given in Algorithm 1. For model building techniques that must see the data multiple times (for example, the epochs required in backpropagation training of neural networks), the data must be transmitted multiple times from the root. Hence small memory at the sensors is bought at the cost of increased communication.

Algorithm 1 Learning with extremely limited resources

for all objects in the training data **do**
  Root sends values for attribute i to sensor i
end for
Sensor i uses the attribute value to build its local model

The test regime is outlined in Algorithm 2. During deployment, each sensor classifies each new input it collects according to the local model and then sends its prediction to the base. Voting can either be unweighted, so that each sensor's prediction makes the same contribution to the global prediction, or weighted by factors such as the test set accuracy of each sensor's local predictor or the confidence with which local models assign input values to target classes. The number of target classes is typically much smaller than the range of values in the attribute domain, and so much less bandwidth is required to transmit predictions than raw data. It should be noted at this point that even a model that requires the entire training data for its construction can sometimes be modified to learn incrementally. A number of incremental algorithms are known, for example those proposed by Thomas [11] and Utgoff [12]. This implies that adaptations of these techniques could be deployed even in sensor networks with extremely limited resources.

Algorithm 2 Testing with extremely limited resources

for all objects in the test data **do**
  Root sends values for attribute i to sensor i
  Sensor i uses its local predictor to predict the target class
  Sensor i sends its prediction to the root
  The root votes using the predictions of each sensor to get a global prediction
end for
The root computes the overall prediction accuracy using the target class labels of the test data
The root computes the accuracy of each local predictor and sends this to each sensor

When each sensor has a significant amount of

memory available, a less rigid training and testing regime is possible as outlined in Algorithms 3 and 4, respectively. After testing, the root can send each sensor both its per-reading accuracy and its global accuracy. The sensors are deployed exactly as above. The amount of storage required at each sensor has to be sufficient to store the larger of the training and test set data, in addition to the target class in each case.

Algorithm 3 Learning with larger resources

for all sensors **do**
  Root sends the columns corresponding to each sensor's attributes and a copy of the target attribute to each sensor
  Each sensor builds a local predictor from this data
end for

Algorithm 4 Testing with larger resources

for all sensors **do**
  Root sends the columns corresponding to each sensor's attributes and a copy of the target attribute to each sensor
  Each sensor uses its local predictor to generate a list of predicted target classes and sends them to the root
end for
Root votes using the target class predictions from each sensor to get a global prediction for each test-set reading

These strategies can be used with any weak learner. The local deployed error rate is then defined as the error rate of the locally built model as compared to the results achieved by the global model. The local deployed error rate is measured against the global error rate, which implies that relative changes rather than absolute values are of interest.

4 Trend detection

In some sensor networks, the target function may change over time, and the predictive model may need to be relearned. This is difficult to handle in a centralized way since a drop in the global predictive accuracy could signal either a change in the target function or a problem with the predictor. Assuming that not all local predictors will fail at the same time due to random data fluctuations, and because the global predictive model depends on many local predictors, a change in a local predictor's accuracy can then be used to trigger *local* relearning to improve that predictor's accuracy in response to changes in input data.

After deployment, the root should send the correct class label for each prediction back to each sensor. A sensor can use this to track its accuracy on observed inputs (which might be expected to match its training accuracy, or at least to remain stable over time) and/or to label each of the recent observed readings with its correct class label. Through the knowledge of class labels, a sensor can then discover that its predictions are starting to be less effective, which might trigger new local behavior (relearning its predictive model), or new global behavior (relearning all of the predictors or, alternatively, removing this sensor from predictions). If weighted voting is used, a sensor whose predictions start to become less accurate is automatically downweighted at the root. The availability of global target classes also means that recent data can be used as training data for relearning local models.

5 Experiments

5.1 Basic distributed data mining The feasibility of the distributed voting approach was demonstrated using the following experiment that simulates distributed execution on an artificial dataset. A more detailed evaluation of the effectiveness of this approach, utilizing a variety of real life datasets containing a large range of numbers of classes, samples and attributes of varying type can be found in [10].

Datasets were generated by drawing from two normal distributions in 10-dimensional space, choosing various different separations for the means in each dimension, and different magnitudes for the variances. The target class is the distribution from which each row was drawn. 500 samples were drawn from each distribution, giving a dataset with 1000 rows and 10 columns.

Each sensor receives a single column of this dataset, together with the corresponding class labels. This is the worst case scenario, since a sensor might be gathering more than one signal at a time. These data were further separated into training and test sets using the out-of-bag estimator procedure suggested by Breiman [3]: samples are drawn with replacement until 1000 samples have been selected. The remaining rows, typically about one-third of the original dataset, are used as the test set. A confidence measure for each sensor's predictive model is then obtained by using this test set. Such a confidence measure is expected to be as accurate as if the test set were of the same size as the training set, so that confidence intervals will be small for this data. Global prediction accuracies are computed in two ways: simple voting, and voting weighted by the probability with which a sensor assigns a particular reading to a class.

The predictive model used is the J48 decision tree implemented in WEKA (`www.cs.waikato.ac.nz/ml/weka/`), although the approach will work for any weak learner. The achieved accuracies for both voting schemes on this dataset are shown in Table 1. The accuracies achieved by building a single decision tree on the whole dataset (the centralized solution) are included for comparison.

From these results, it can be seen that the overall classification accuracy for both voting approaches is comparable to or better than that of the centralized approach datasets in all cases. In addition, the weighted voting approach outperforms the simpler voting scheme. For an explanation of the observations and a more general experimental evaluation, see [10].

5.2 Effect of trends during deployment We now consider the effect of a target function, and hence a class boundary, that changes with time. In the following experiments, we generated data similar to that described above, but moved the means of the distributions incrementally.

A dataset of a 1000 samples was drawn from two normal distributions whose means were centered at the origin and (1,1,...,1) with variance one. Ten further datasets, each of size 1000, were generated from two normal distributions with the same variance; the means of these two distributions were moved in lockstep in axis-parallel steps according to the schedule shown in Table 2 for a total distance of 20 in each dimension. For example, attribute 1 changed by 10% of its total change at each iteration, while attribute 3 made the total change only during the last iteration. The extension to the previous deployment is that each sensor tracks its local deployed error rate (which is derived from the global prediction, not from the 'true' prediction) When this accuracy changes sufficiently, the sensor relearns its local model, using recent data and the target class labels reported to it by the root. Algorithm 5 outlines the approach.

Algorithm 5 Relearning during deployment

```
Create a local model in each sensor
Classify each new reading according to the local model
if local accuracy deviates then
   Relearn the local model
end if
Send the resulting classifications to the base
Combine the predictions from the sensors in the base using a voting scheme
```

The deviation that triggers relearning was taken to be 5 percentage points of prediction accuracy for each attribute. Figures 1 and 2 show how the accuracy changes as the target function changes, how relearning is triggered, and how the prediction accuracy subsequently improves.

Figure 3 shows the effect of the data change and relearning on the overall accuracy achieved at the base after combining the classifications sent from the sensors using both of the voting schemes.

This strategy assumes the change in class boundaries will not require relearning in all attributes simultaneously, for then the global class labels would not be appropriate surrogates for the correct prediction. This assumption is reasonably robust, since a simultaneous relearning would indicate a drastic change in the target function, which is unlikely in most realistic settings.

We make the following observations.

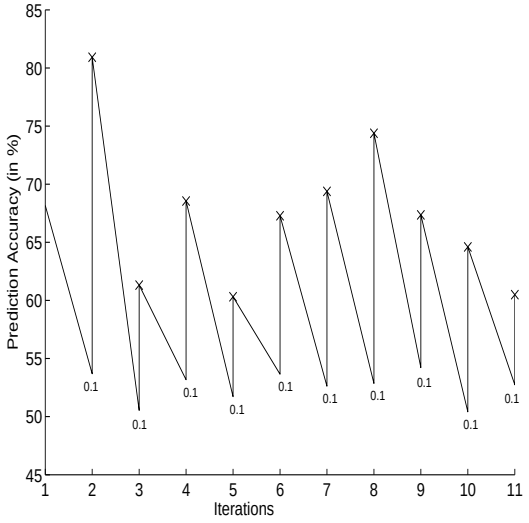
- *Relearning corresponds to changes in target*

Distance between means in each dimension	**Variance**	**Number of dimensions =attributes**	**Simple voting**	**Weighted voting**	**Centralized prediction**
1	0.5	10	98.10 (0.88)	99.28 (0.30)	94.31 (1.43)
1	0.5	5	96.61 (0.65)	96.61 (0.65)	94.96 (1.07)
1	1	10	88.05 (2.43)	89.03 (1.61)	82.75 (2.12)
1	1	5	81.16 (3.08)	81.16 (3.08)	77.70 (2.02)
2	0.5	10	100 (0)	100 (0)	99.11 (0.56)
2	1	10	98.81 (0.48)	99.15 (0.31)	95.10 (1.21)
2	0.5	5	99.97 (0.09)	99.97 (0.09)	99.79 (0.25)
2	1	5	96.98 (0.65)	96.98 (0.65)	94.38 (1.26)

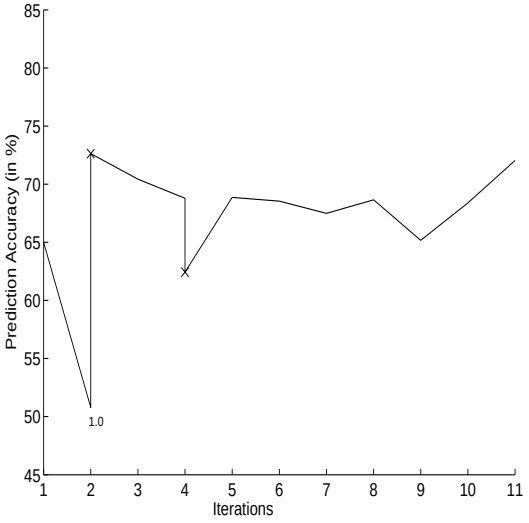
Table 1: Mean global prediction accuracies over 10 trials, using simple and weighted voting, on datasets with different class mean separation and variances. The values in parentheses indicate standard deviations.

Iteration	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Attribute 1	0.0	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Attribute 2	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Attribute 3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
Attribute 4	0.0	0.5	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.0	0.0
Attribute 5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.4	0.2
Attribute 6	0.0	0.4	0.0	0.4	0.0	0.0	0.1	0.0	0.1	0.0	0.0
Attribute 7	0.0	0.2	0.0	0.2	0.0	0.2	0.0	0.2	0.0	0.2	0.0
Attribute 8	0.0	0.2	0.2	0.2	0.2	0.2	0.0	0.0	0.0	0.0	0.0
Attribute 9	0.0	0.5	0.2	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.1
Attribute 10	0.0	0.3	0.0	0.0	0.3	0.0	0.0	0.0	0.3	0.0	0.1

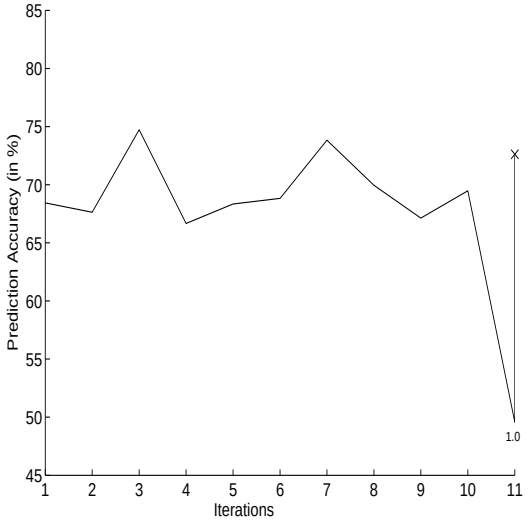
Table 2: Change of dimension means in each iteration (as % of the total change)



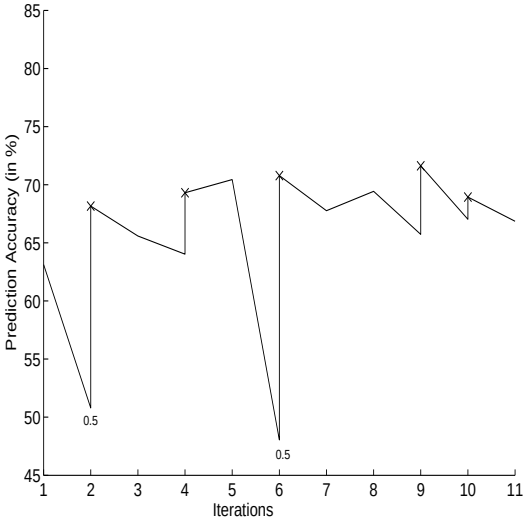
(a) Attribute 1



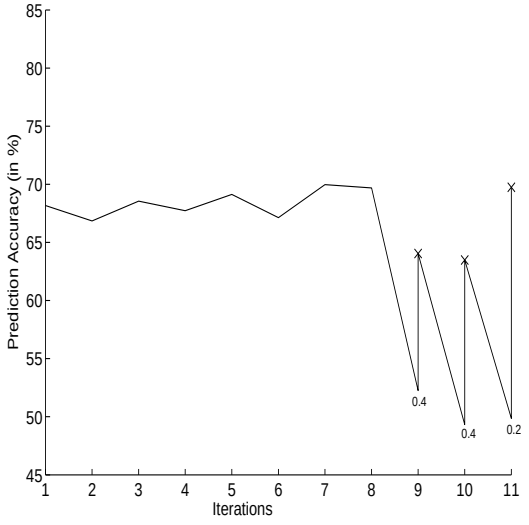
(b) Attribute 2



(c) Attribute 3

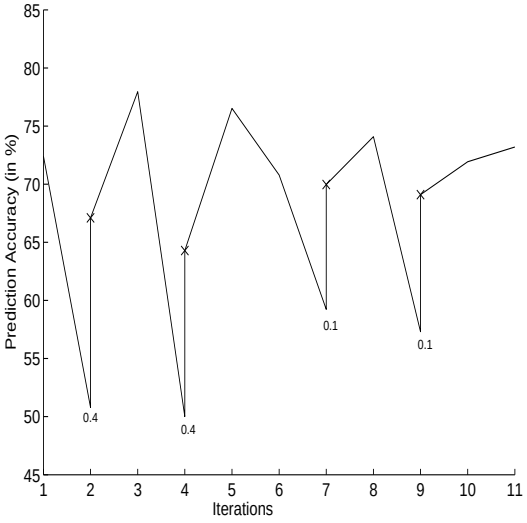


(d) Attribute 4

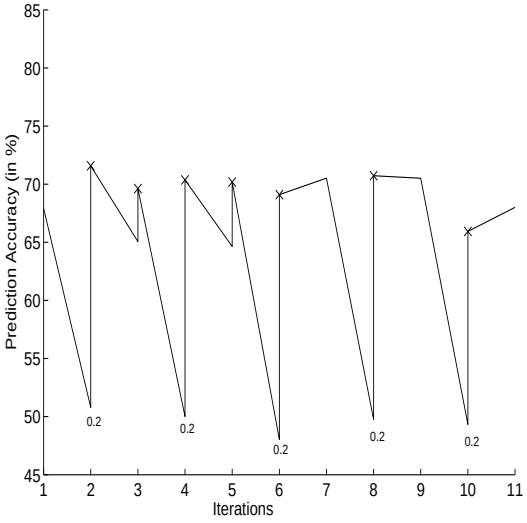


(e) Attribute 5

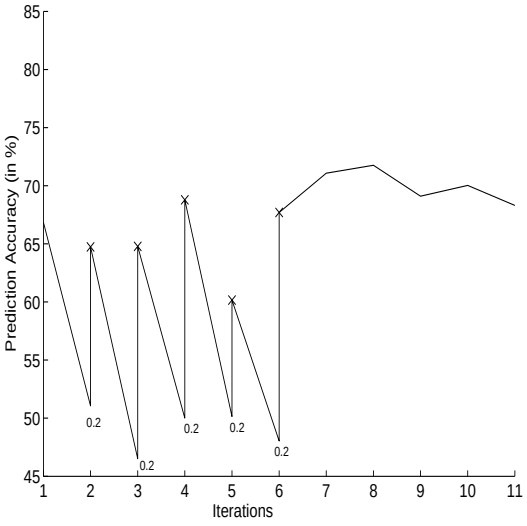
Figure 1: Accuracies for Attributes 1 through 5 over time. Values marked with an x indicate the accuracies after relearning was triggered. Numerical values indicate the percentage change of the target class centers in that dimension (and so for that attribute).



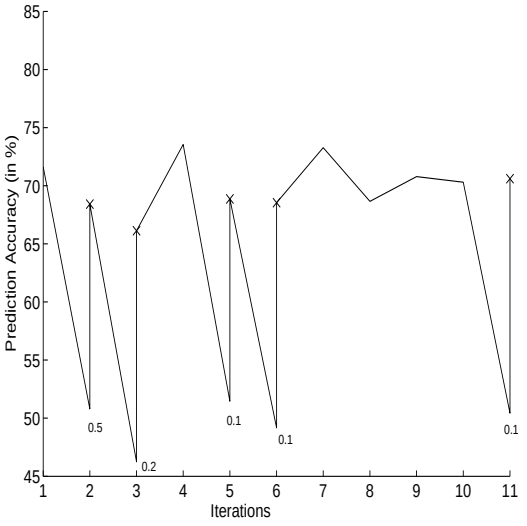
(a) Attribute 6



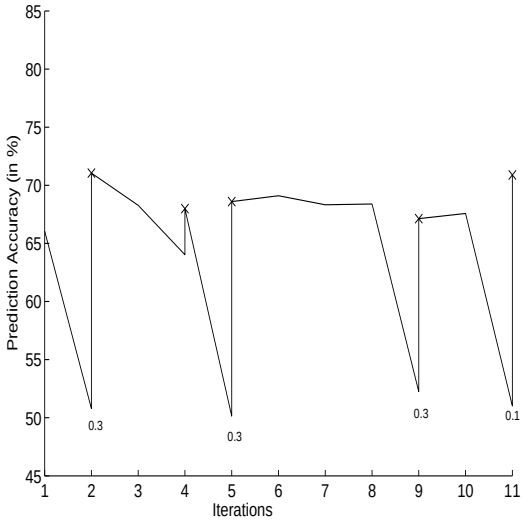
(b) Attribute 7



(c) Attribute 8



(d) Attribute 9



(e) Attribute 10

Figure 2: Accuracies for Attributes 6 through 10 over time. Values marked with an x indicate the accuracies after relearning was triggered. Numerical values indicate the percentage change of the target class centers in that dimension (and so for that attribute).

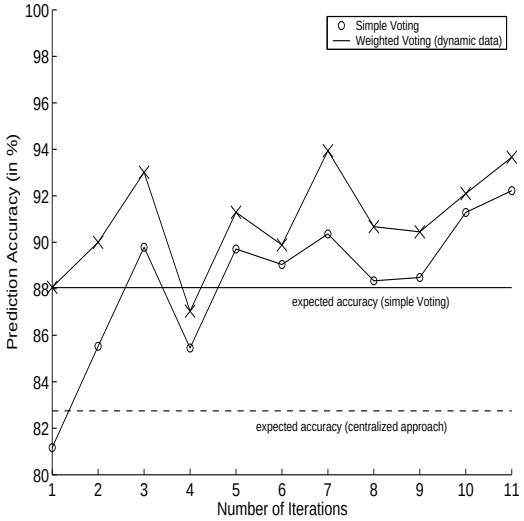


Figure 3: Overall prediction accuracy with moving class boundaries.

function. For each change in the target class centers, and independent of the amount of change, relearning is triggered for the local models. This is true for all attribute values and across all iterations. For example, the model describing attribute 1 is relearned at each iteration after the initial one, corresponding to the change of the distribution mean in that dimension.

- *Relearning occurs occasionally even without changes in the data.* This is seen for example in the plot for attribute 10, where relearning occurred in iteration 4. It was observed that over different runs of the experiment, relearning without corresponding target class changes varied in both frequency and the timing of its occurrence. We therefore suspect that this is in response to occasional fluctuations in the datasets due to the randomness of initial construction and selection of the training and test sets.

- *The overall classification accuracy for both the simple and the weighted voting scheme is as good or better than that of the centralized data mining approach.* We know from the first experiment that the overall accuracies achieved for a dataset with a separation between distribution means of 1 in each dimension and a variance of 1 are 88.05% and 89.03% for the simple and weighted voting scheme, respectively. In addition, the estimated value for the accuracy for a centralized approach is 82.75%. From the results depicted in Figure 3, we see that the overall accuracies achieved through the majority voting schemes with relearning is equivalent or better to those achieved by a centralized data mining approach.

- *The global accuracy at the root is better than the individual accuracies obtained from the local models.* This is due in part to the ensemble effect, which allows the combined accuracy to be superior to each of the predictors contained in the ensemble, provided that the local predictors are both accurate and make different errors on different data [7].

It should be noted here that the transmission of a probability along with the class prediction from the sensors to the base requires additional communication and might not be feasible if resources are limited. However, the accuracy achieved by the simple voting scheme is equivalent to the accuracy achieved by a centralized approach and therefore sufficient.

We have assumed that sensor reading, sensor transmission of local prediction, and voting at the root are synchronous. This is a moderately strong

assumption, since it would require a common clock. In fact, this requirement can be relaxed in several ways, which will be discussed in more detail in a subsequent paper. For example, the root may recalculate the vote whenever a new prediction is reported from a sensor, freeing sensors to report their predictions asynchronously. A major application of sensors networks is as anomaly detectors for complex anomalies that can only be detected by concerted changes in the data at several sensors. This can be modelled as a two-class problem (Safe vs. Alarm). The amount of communication required is greatly reduced if sensors only report predictions for the Alarm class; the root can then predict an Alarm in response to some number of local predictions of Alarm by the sensors.

6 Conclusions

In this paper, we have presented a framework for building and deploying predictors in sensor networks in a distributed way, by building local models at the sensors and transmitting target class predictions rather than raw data to the root. At the root, local predictions are combined using weighted or unweighted voting. This framework is appropriate for the limited resources found in sensor networks, due to power, bandwidth and computational limits. We have also showed how the use of local predictive models enables sensors to respond to changes in targets by relearning local models when their local predictive accuracy drops below a threshold. This enables effective distributed data mining in the presence of moving class boundaries, and also creates new possibilities, for example the use of sensors to detect anomalies, even when the criteria for an anomaly changes over time. Finally, because only model predictions rather than data are transmitted, the framework is also suitable for settings where data confidentiality is a concern.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, *A Survey on Sensor Networks*, IEEE Communications Magazine, August 2002, pp. 102–114.

[2] S. Bandyopadhyay, C. Gianella, U. Maulik, H. Kargupta, K. Liu, and S. Datta, *Clustering Distributed Data Streams in Peer-to-Peer Environments*, 2004, Accepted for publication in the Information Science Journal, in press.

[3] L. Breiman, *Out-of-bag Estimation*, 1996, Technical Report, Statistics Department, University of California, Berkeley, Ca.

[4] L. Breiman, *Random Forests*, Machine Learning, 45(1), 2001, pp. 5–32.

[5] P. Chan and S. Stolfo, *Experiments in Multistrategy Learning by Meta-Learning*, Proceedings of the Second International Conference on Information and Knowledge Management (Washington, DC), 1993, pp. 314-323.

[6] P. Domingos and M. Pazzani, *Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier*. In Proceedings of the Thirteenth International Conference on Machine Learning (ICML), 1996

[7] L. Hansen, and P. Salamon, *Neural Network Ensembles* , IEEE Transactions on Pattern Analysis and Machine Intelligence,(12) 1990, 993–1001.

[8] H. Kargupta, B. Park, D. Hershberger, and E. Johnson, *Collective Data Mining: A New Perspecive Towards Distributed Data Mining*, Advances in Distributed Data Mining, Eds: H. Kargupta and P. Chan, AAAI/MIT Press, 1999.

[9] H. Kargupta, R. Bhargava, K. Liu, M. Powers, P. Blair, S. Bushra, J. Dull, K. Sarkar, M. Klein, M. Vasa, and D. Handy, *VEDAS: A Mobile and Distributed Data Stream Mining System for Real-Time Vehicle Monitoring*, Proceedings of the SIAM International Data Mining Conference, Orlando, 2004.

[10] S. McConnell, and D. Skillicorn, *Building Predictors from Vertically Distributed Data*, Proceedings of the 14th Annual IBM Centers for Advanced Studies Conference, Markham, Canada, 2004, pp.150–162.

[11] S. Thomas, S. Bodagala, K. Alsabti, and S. Ranka, *An Efficient Algorithm for the Incremental Updation of Association Rules in Large Databases*, Knowledge Discovery and Data Mining, 1997, pp. 263–266.

[12] P.Utgoff, *An Improved Algorithm for Incremental Induction of Decision Trees*, Proceedings of the International Conference on Machine Learning, 1994, pp. 318–325.