Hierarchical Distributed Data Classification in Wireless Sensor Networks

Xu Cheng Ji Xu Jian Pei Jiangchuan Liu
School of Computing Science
Simon Fraser University
British Columbia, Canada
Email: $\{$*xuc, jix, jpei, jcliu*$\}$*@cs.sfu.ca*

Abstract

Wireless sensor networks promise an unprecedented opportunity to monitor physical environments via inexpensive wireless embedded devices. Given the sheer amount of sensed data, efficient classification of them becomes a critical task in many sensor network applications. The large scale and the stringent energy constraints of such networks however challenge the conventional classification techniques that demand enormous storage space and centralized computation. In this paper, we propose a novel hierarchical distributed classification approach, in which local classifiers are built by individual sensors and merged along the routing path. The classifiers are iteratively enhanced by combining strategically generated pseudo data and new local data, eventually converging to a global classifier for the whole network. We demonstrate that our approach maintains high classification accuracy with very low storage and communication overhead. It also addresses a critical issue of heterogeneous data distribution among the sensors.

1. Introduction

The recent advances in transceiver and embedded hardware designs have made massive production of inexpensive wireless sensors possible. These devices can form a network with each node storing, processing and relaying the sensed data, usually to a base station for further computation. Such wireless sensor networks have been widely used in a broad spectrum of applications like wildlife monitoring, battlefield surveillance, and disaster relief [1], [2], [3]. Given the huge amount of the sensed data, classifying them becomes a critical task in many of these applications.

As an example, for wildlife monitoring, the sensor nodes continuously sense the physical phenomena such as temperature, humidity and sunlight, and meanwhile may also count the number of animals. The number reflects the suitability of the current environment for the animals. For example, if it is greater than a threshold, the environment is classified as suitable, and otherwise not. After learning the relation between the physical phenomena and the classes from such *training data*, we may later determine the suitability of the inquired environment from the external source. These inquiries are the *unseen data* which only have the physical phenomena but do not have the class label. The sensor data collection and object count have been extensively studied in the literature, e.g., the Great Duck Island Project [4], yet efficient classification for wireless sensor networks has not been well addressed.

Classification is typically done in two steps: first, a classifier is constructed to summarize a set of predetermined classes, by learning from a set of training data; then, the classifier is used to determine the classes of newly arrived data. Within this framework, there have been significant efforts in improving its speed and accuracy, most of which assume centralized storage and computation. The wireless sensor networks however pose a series of new challenges, particularly for the first step. First, the number of sensor nodes is huge, but each of them has only limited storage that can hardly accommodate all the training data of the whole network. Second, the sensor nodes are generally powered by non-rechargeable batteries, and energy efficiency is thus of paramount importance, which makes the straightforward solution of sending all the training data to the powerful base station quite inefficient [5], [6]. In short, conventional centralized solution is not directly applicable in this new type of network environment.

To address the above challenges, in this paper, we present a novel distributed solution for classification in energy-constrained sensor networks. Our approach hierarchically organizes the sensor nodes and performs classification in a localized and iterative manner, utilizing a decision tree method. Starting from each leaf node, a classifier is built based on its local training data. An upstream node, upon receiving the classifiers from its children, will use them with its own data to build a new classifier. These local classifiers will be iteratively enhanced from bottom to top and finally reach the base station, making a global classifier for all the data distributed across the sensor nodes. Since only the classifiers will be forwarded upstream, the energy consumption for transmission can be significantly reduced.

The key difficulty here however lies in training a new classifier from a mix of downstream classifiers and the local training dataset, which cannot be directly accomplished by the existing learning algorithms that work on dataset only. We address this problem by generating a *pseudo training*

978-1-4244-5113-5/09/$25.00 ©2009 IEEE

dataset from each downstream classifier. We develop a smart generation algorithm, which ensures that the pseudo data closely reflect the characteristics of the original local data. We also introduce a control parameter that adaptively balances the recovery quality and the amount of the data. Through extensive simulations, we demonstrate that our approach maintains high classification accuracy, with very low storage and communication overhead.

We also notice that, in practice, the data distribution across different sensor nodes is not necessarily homogeneous. For example, depending on the location, the data sensed by one node may always have low temperature and low humidity, and the data sensed by another node at a different location may always have high temperature and high humidity. We show strong evidence that such heterogeneity can easily lead to misclassification, and we propose an enhanced ID3 algorithm to mitigate its impact. To our knowledge, it is the first solution addressing this issue.

The remaining part of the paper is organized as follows. Section 2 gives a brief introduction of classification. Section 3 describes our approach. We evaluate our approach and present the results in Section 4. Section 5 reviews the related works in the literature. Finally, Section 6 concludes the paper.

2. Decision Tree Basics

Decision tree is one of the most important models for classification, and also serves as the foundation for our hierarchical distributed classification. A decision tree is a mapping from observations about an item to conclusions about its target value. A node in the tree is a test of some attribute, and a branch is a possible value of the attribute. To perform classification, we can start from the root, test the attribute, and move down to the tree branches. Figure 1 shows a decision tree example. For instance, it indicates that if the temperature is medium and the sunlight is weak, the environment is classified as suitable (Yes) for animals.

The most famous approach to build a decision tree is *ID3* (Iterative Dichotomiser 3), proposed by Quinlan [7]. The ID3 algorithm recursively constructs a tree in a top-down divide-and-conquer manner. To choose the best attribute at the current node, it calculates the information gain using entropy,

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i,$$

where S is the dataset and p_i is the proportion of each class. The information gain is then calculated as

$$Gain(S, A) = Entropy(S) - \sum_{v \in V(A)} \frac{|S_v|}{|S|} Entropy(S_v),$$

where $V(A)$ is the set of all possible values for attribute A, and S_v is the subset of S for which attribute A has value v.

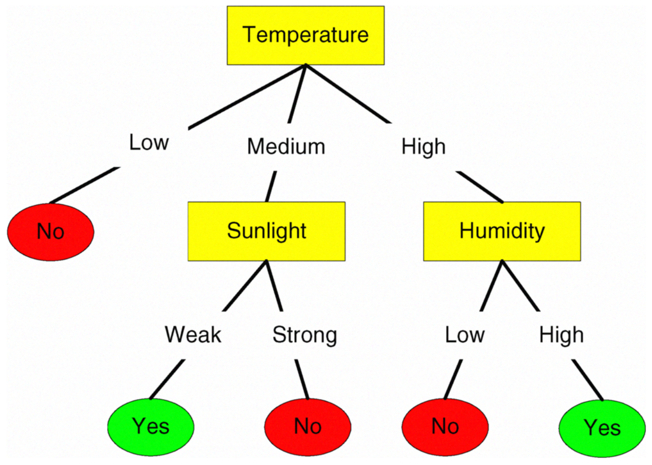

Figure 1. An example of a decision tree

The algorithm then creates a node for each possible attribute value, and partitions the training data into descendant nodes. There are three conditions to stop the recursion:

1) all samples at a given node belong to the same class;
2) no attribute remains for further partitioning;
3) there is no sample at the node.

The original ID3 algorithm, however, cannot be used to directly combine the local classifiers, nor does it preserve the data attribute distribution to support pseudo data recovery. In this paper, we will present a substantially enhanced version to address these challenges.

3. Hierarchical Distributed Classification

In this section, we first present the system overview, and then introduce our classification approach in detail, showing how to build the decision tree, how to generate the pseudo data, and how to build the classifier hierarchically. We discuss the accuracy and the energy consumption afterwards.

3.1. System Overview

We consider N sensor nodes $n_1, n_2, \ldots, n_N$ distributed in a field. Each node covers an area of the field and is responsible for collecting data within the area. The data reporting follows a spanning tree rooted at the base station n_0. The routing protocol design for forming the spanning tree is out of the scope of this paper, and there are indeed numerous solutions in the literature [5], [8].

Each sensor node n_i first collects its local training data D_i. If node n_i is a leaf node, it builds a classifier C_i by a learning algorithm $\aleph$, which we will illustrate in Section 3.2.

The node then sends C_i to its parent node, say n_j. We use a decision tree to represent the classifier[1], which, compared to the original data, is of a much smaller size.

An upstream node n_j, upon receiving the classifiers from its children, combines the children's classifiers with its local training data D_j to build an enhanced classifier C_j. These local classifiers will be iteratively enhanced from bottom to top and finally reach the base station, making a global classifier for all the data distributed across the sensors. Since only the classifiers will be forwarded upstream, the energy consumption for transmission can be significantly reduced. The sensor nodes may continuously sense new data and forward to upstream. Depending on the application requirement, the new data can be used either for updating the classifier or for classification based on the existing classifier.

The challenge here lies in training the enhanced classifier from a mix of downstream classifiers and the local dataset, which cannot be directly accomplished by the existing training algorithms that work on dataset only. To address this problem, a pseudo training dataset will be generated from each downstream classifier. For each child node n_i, node n_j will generate a set of pseudo training data D'_i from the classifier C_i, and then combine all these data with its own training data to build the enhanced classifier. Obviously, the pseudo data, recovered from classifiers, should closely reflect the characteristics of the original local data, e.g., the distribution of different classes and the attribute values. The amount of the pseudo data is also an important concern given that a sensor node generally has a limited memory. We will address these detailed issues in Section 3.3.

We list the major notations in Table 1. Also note that, in this paper, we do not consider issues like node failure or packet loss, which have been extensively addressed in the literature [5], [6], [9].

Notation	Explanation
N	the number of sensor nodes, excluding the base station
n_i	sensor node ($i = 1, 2, ..., N$)
n_0	base station
D_i	the training data collected by node n_i
D'_i	the pseudo data generated from classifier C_i
C_i	the local classifier built by node n_i
C_0	the global classifier built at the base station

Table 1. List of Notations

3.2. Building Decision Tree

In our system, the decision trees are built by the widely-used ID3 algorithm [7]. The basic ID3 algorithm, however, does not keep the information about the attribute distribution and the amount of the original training data, preventing pseudo data recovering from a classifier.

1. We will use "classifier" and "decision tree" interchangeably throughout this paper provided the context is clear.

To solve this problem, we let each leaf node record the count of each class (i.e., the number of positive and negative labels in this application scenario). Therefore, we have the knowledge about the amount of the samples for building each branch of the decision tree, and thus we can make the distribution of the generated pseudo data resemble the original ones.

Moreover, in the basic ID3 algorithm, if all the samples belong to the same class, the recursion stops (e.g., when temperature is low in Figure 1). Hence, the information of other attributes will be missing, which can cause problems with heterogeneous data distribution across different sensor nodes. For example, if all the training data sensed by a sensor node are below 10 degree in temperature and below 20% in humidity, and the class labels are all negative, using the basic ID3 algorithm, only one attribute, say temperature, will appear in the decision tree, and the information of humidity is completely missing. This will likely lead to a set of pseudo data generated with humidity uniformly distributed from 0% to 99%, which is clearly not the case for the original data.

Therefore, for the stop condition of the recursion, we eliminate the first one (referred to Section 2) in our enhanced ID3 algorithm. The new stop condition thus becomes when no attribute remains for further partitioning or when there is no sample.

In Algorithm 1 and Algorithm 2 below, we describe the enhanced ID3 algorithm for building the decision tree. Note that a brief illustration of the key steps of the basic ID3 is in Section 2, and more details can be found in Quinlan's work [7].

Algorithm 1 LearningAlgorithm (D)

```
Require: training dataset $D$
  call EnhancedID3 ($root$, $D$, 0)
  return root
```

3.3. Generating Pseudo data

The pseudo data generation is one of the most important steps in our framework. A critical challenge here is to generate data that are as close to the original data as possible. In particular, the distribution of each attribute should closely resemble that of the original data.

Another issue is how many pseudo data should be generated. Intuitively, the fewer data we generate from the child nodes, the less weight they have. Since data from one node should not be considered less important than those from another, we need to generate the same amount of the pseudo data as the original data. Therefore, a sensor node close to the base station has to generate a huge amount of pseudo data, i.e., the same amount of the original data at all its descendants (not only the immediate children). This is often impossible given the limited memory of the embedded

```
Algorithm 2 EnhancedID3 ($p$, $D$, $d$)
Require: pointer to the decision tree $p$, training dataset $D$,
  depth of the decision tree node $d$
  if number of data $|D| = 0$ then
    return
  end if
  if $d$ = number of attributes of the training data then
    get the most common class label in $D \rightarrow$ attribute of $p$
    record the counts of the class labels for $p$
    return
  end if
  get the best attribute $\rightarrow target\_attribute$
  for all $target\_attribute$ value branches do
    add a child $p'$
    set the value range of the branch
    partition $D \rightarrow D'$ satisfying the value range
    call EnhancedID3 ($p'$, $D'$, $d+1$)
  end for
```

sensor nodes. To this end, we introduce a *preservation factor*, ranging from 0 to 1 (the base station always has a factor of 1), to control the amount of the generated pseudo data.

```
Algorithm 3 GeneratePseudoData ($C$, $a$)
Require: decision tree received from one child $C$, preservation factor $a$
  for all leaf nodes $node$ of decision tree $C$ do
    get rule $R$ of $node$
    get class label counts $\rightarrow c_1, c_2, \ldots, c_L$
    randomly generate $a \cdot \sum_{i=1}^{L} c_i$ data satisfying $R$
    assign class label $l_k (k = 1, \ldots, L)$ to the data
      with probability as the proportion of the class label
      $c_k / \sum_{i=1}^{L} c_i$
    add these data to pseudo data set $D'$
  end for
  return pseudo data set $D'$
```

Algorithm 3 summarizes our method to generate the pseudo data. For illustration, suppose the decision tree's leaf node represents a rule of when temperature is between 10 and 20, humidity is between 20 and 40, and sunlight is *normal*, there are 5 positive class labels and 45 negative ones. As such, the class label is negative. Assuming the preservation factor is set to 0.8, we then randomly generate $(5+45) \times 0.8 = 40$ data that satisfy the attribute requirement. It follows that each data has a probability $5/50 = 0.1$ to be assigned a class label as positive and 0.9 to be negative.

The original data are partitioned to each decision tree leaf node, and each set of generated data resembles part of the original data. Therefore, the combined pseudo data will largely reflect the characteristics the original training data.

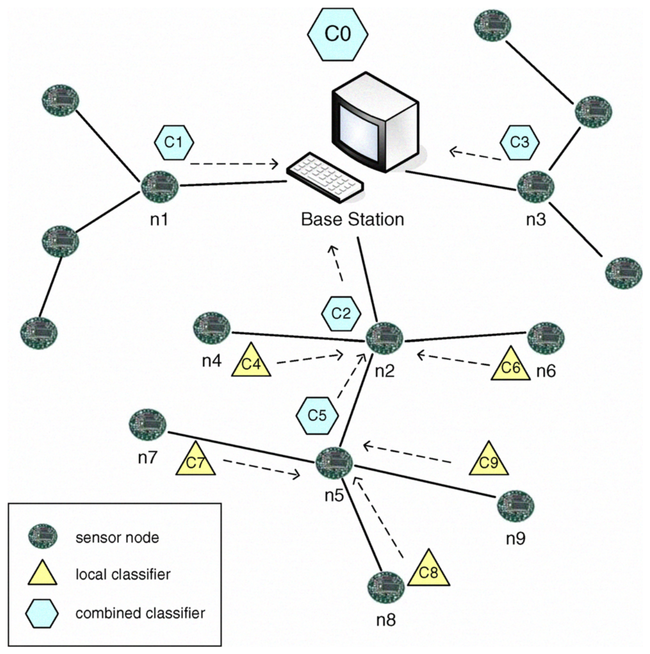

Figure 2. A hierarchical structure of the sensor network

We will closely examine the effectiveness of our method as well as the impact of the preservation factor in Section 4.

3.4. Hierarchical Classification

As mentioned above, we let the sensor nodes in the network be organized by a spanning tree, as illustrated in Figure 2. A leaf node builds the decision tree with the local sensed training data and sends the decision tree to the parent. The intermediate node periodically checks if there is any new classifier from children. If yes, it will generate a set of pseudo data for each new classifier, and combines them with its local data. There are two situations here. (1) If the node has never built any classifier, which indicates that it has never received any classifier from its children, it will combine the generated pseudo data with its local sensed data and performs the learning algorithm. (2) If the node has once built a classifier, the previous received classifiers may have already been discarded (due to the memory constraint). The node will then generate a set of pseudo data for its local classifier with the preservation factor being 1, and combines it with the other pseudo datasets to build the new decision tree.

The base station will build the global classifier. Initially, it waits until receiving all the classifiers from its children, and then generates pseudo data for each, and combines them to build the decision tree. Since the base station in general is a more powerful node, it could store all the classifiers from its immediate children. Therefore, when one of them updates the classifier, the base station discards the old one and re-performs the operations as above.

In Algorithm 4, we summarize the detailed procedure of the hierarchical classification.

Algorithm 4 HierarchicalClassification ()

```
if node is leaf then
    periodically collect data $D$
    $C \leftarrow$ LearningAlgorithm ($D$)
    send $C$ to its parent
else if node is not base station then
    periodically collect data $D$
    for all new classifiers $C_k$ from child $k$ do
        $D'_k \leftarrow$ GeneratePseudoData ($C_k, a$)
    end for
    $C \leftarrow$ LearningAlgorithm ($D \cup (\bigcup D'_k)$)
    send $C$ to its parent
else {//base station}
    if no classifier has been built then
        for all classifiers $C_j$ from child $j$ do
            $D'_k \leftarrow$ GeneratePseudoData ($C_k, 1$)
        end for
        $C \leftarrow$ LearningAlgorithm ($\bigcup D'_k$)
    else if receive new classifier $C_k$ from child $k$ then
        replace old classifier of child $k$ with $C_k$
        for all classifiers $C_k$ from child $k$ do
            $D'_k \leftarrow$ GeneratePseudoData ($C_k, 1$)
        end for
        $T \leftarrow$ LearningAlgorithm ($\bigcup D'_k$)
    end if
end if
```

3.5. Further Discussion

Our approach utilizes the ID3 as the basis for classification, so it inherits the effectiveness and efficiency of ID3 when building local classifiers [7]. To make it fit our application scenario better, we suggest that all the leaf nodes in the decision tree have the same depth as the number of the attributes, so that we can keep all the attribute information when abstracting the rule from each branch. Thanks to the modification, we can generate the pseudo data that are very similar to the original data. Apparently, this modification will increase the size of the decision tree, however, such increase is acceptable and it noticeably increases the classification accuracy, as will be validated in our performance evaluation.

In order to keep the high accuracy and achieve our goals of saving energy and storage space, we have introduced two parameters in our solution. One is the *class count*. It not only records the count but also indicates the distribution of all the class labels in the original dataset when we build the decision tree. In other words, we keep the "noise" information because the "noise" may be important in some cases, in particular, the heterogeneous data distribution. For example, suppose one decision tree branch has the negative label because the numbers of positive and negative training data satisfying the constraint are 1 and 9, while another decision tree branch has the positive label on the same attribute constraint because the counts are 99 and 1. Without recording the class label counts, we have no idea about which one is more accurate, and we will likely treat them equally. In fact, combining the two training dataset, we will obtain the positive label with the class counts 100 versus 10. The problem is particularly severe when the training dataset have the heterogeneous distribution, which critically demands the recording of the class count.

The other parameter is the preservation factor, which determines the amount of the pseudo data to generate. We introduce this parameter due to the limited memory of the sensor node. The smaller the preservation factor is, the more dominant the local training data are. For example, suppose a node has three children, each having 200 training data, and the node itself also has 200 training data. If the preservation factor is set to 1.0, the node will learn from 800 data, in which 25% are its local data. If the preservation factor is set to 0.1, the node will learn from 260 data, in which 77% are its local data. Intuitively, if the pseudo data can represent the original data very well, the greater the preservation factor is, the more accurate the classifier is, because every area should be treated equally. Otherwise, the greater the preservation factor is, the more noise it will make, thus decreasing the accurate. Therefore, the representativeness of the pseudo data of the original data is crucial in our solution. In the next section, we will closely examine its impact to the classification accuracy.

4. Performance Evaluation

4.1. Configuration and Dataset

We evaluate our framework with our customized simulator. We consider a square field consisting of $m \times m$ randomly deployed sensor nodes, with the base station being located in the center. We set m from 5 to 20, and for each m value, we build different topologies so that the spanning tree has height of 2, 3 and 4, respectively. Figure 3 shows an example of 25 nodes ($m = 5$) with height 2. Our results however have shown that the performance of our algorithm is mainly affected by the height of the spanning tree, while not the node population. This is intuitive because the algorithm is distributed and localized. Hence, in this section, we will focus on the average results (and the associated standard deviations) over the tested m values. We have also tested the topology with larger height, and the results show the

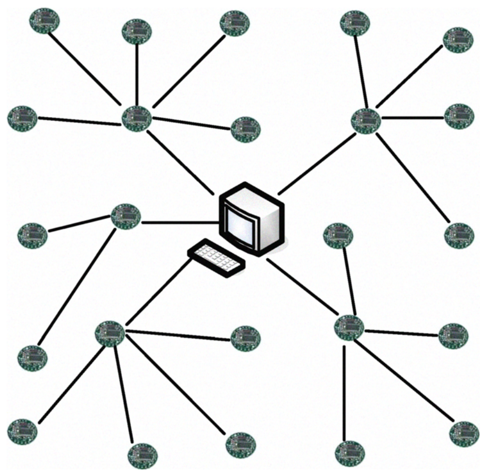

Figure 3. An example of topology of 25 nodes and height being 2

same trend as the smaller height, thus we only provide the results of height being 2, 3 and 4.

The data has three attributes and a class label. The three attributes are temperature, humidity and sunlight. The first two are numerical attributes, ranging from 0 to 49 and from 0 to 99, and the last one is a categorical attribute, having values $weak$, $normal$ and $strong$. For simplicity, we consider the two numerical attributes as categorical attributes, as the temperature has values $[0, 10)$, $[10, 20)$, $[20, 30)$, $[30, 40)$ and $[40, 50)$, and the humidity values $[0, 20)$, $[20, 40)$, $[40, 60)$, $[60, 80)$ and $[80, 100)$. The class label is either $positive$ or $negative$, say, indicating whether or not the environment is suitable for the animal. We randomly generate data having the temperature between 0 and 49, humidity between 0 and 99, and sunlight being 0, 1 or 2. We manually define some rules to assign each data a class label. We also consider noise and thus add a factor ε, which indicates the data has the probability of ε to be the other class label determined by the rules.

We generate 10 training datasets, each having $200 \cdot (m^2 - 1)$ data. Among the 10 datasets, five of them have $\varepsilon = 1\%$ noise, and five of them have $\varepsilon = 10\%$ noise. For each dataset, we make it into two versions, one is called *heterogeneous data* and the other is called *homogeneous data*. In the heterogeneous data set, the data distribution depends on the location of the sensor nodes, while the homogeneous data is independent on the location, and is randomly and uniformly distributed across sensor nodes.

For homogeneous data, we just randomly divide and assign all the training data to the $(m^2 - 1)$ sensor nodes as the local training data, thus each sensor node has 200 training data. For heterogeneous data, we assume that the temperature increases from left to right, and the humidity increases from bottom to top in the sensor field; the attribute of sunlight is uniformly distributed. For example, a node in the bottom right corner is supposed to have the data with temperature between 40 and 49, humidity between 0 and 19, and a node in the top left has the data with temperature between 0 and 9, humidity between 80 and 99. For each data in one dataset, we first calculate its coordinates according to the above, and then assign the data to the sensor node of that location if the training set of the node is not full (200 data). If that node is full, we then assign this data to another random node. For each training dataset, we perform different simulations that have different data distributions. For example, one is described as above, and another one is that temperature increases from bottom to top, humidity increases from right to left, and so forth.

In our experiments, we generate 10 test datasets, each having 1000 data. Among the 10 datasets, five of them have $\varepsilon = 1\%$ noise, and five of them have $\varepsilon = 10\%$ noise. If we use the training data with $\varepsilon = 1\%$ noise to learn, we use the test data with $\varepsilon = 1\%$ noise to test (same as the data with $\varepsilon = 10\%$ noise).

4.2. Baseline for Comparison

We have also implemented an *ensemble method* [10] for the baseline comparison. The ensemble method is to construct a set of base classifiers and take a majority voting on predictions in classification. The method can significantly improve the accuracy of prediction, because if the base classifiers are independent, then the ensemble makes a wrong prediction only if more than half of the base classifiers are wrong. For example, suppose there are two classes and each base classifier has an error rate of 35%. With 25 base classifiers, the error rate will be $\sum_{i=13}^{25} \binom{25}{i} 0.35^i 0.65^{25-i} = 0.06$ only.

The ensemble method has been widely adopted in distributed classification [11]. In our evaluation, we customize the ensemble method to our application scenario. Specifically, all the nodes learn from their local training data to build the classifiers, and send the classifiers to the base station through multi-hop routing. The base station then conducts the second step of the classification, in particular, the base station tests the unseen data with all the classifiers and takes a majority voting to get the final decision. Obviously, sending all the local classifiers to the base station consumes more energy than only sending the local classifier to the parents. Moreover, the ensemble method does not accommodate to heterogeneous data, thus the accuracy of classification can be low, as will be shown later.

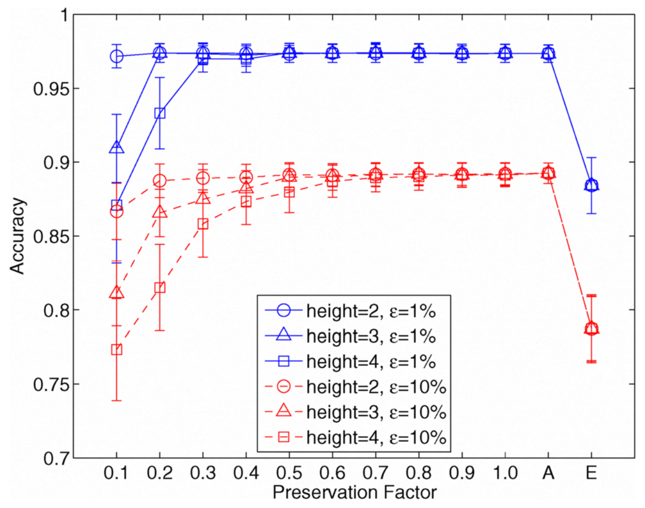

Figure 4. Comparison of accuracy for different preservation factor, noise and height with heterogeneous data

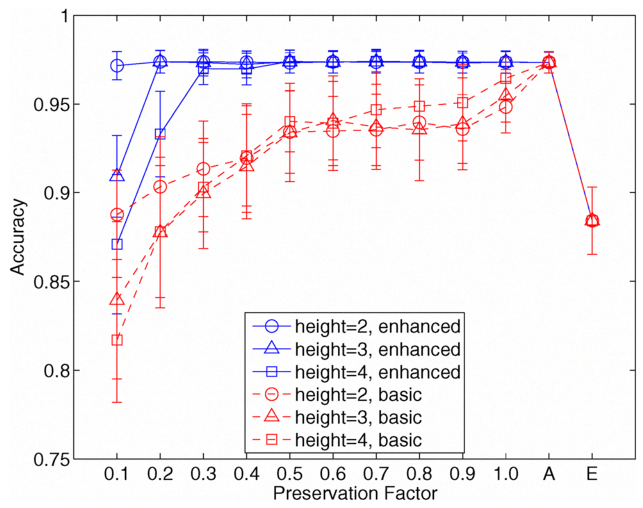

Figure 5. Comparison of enhanced and basic ID3 algorithms for heterogeneous data with $\varepsilon = 1\%$

4.3. Impact of Preservation Factor, Noise and Height for Heterogeneous Data

We first examine our algorithm with different preservation factors, noises and heights for the heterogeneous data. We plot the results in Figure 4. The x axis is the preservation factor ranging from 0.1 to 1.0, and we also evaluate the best-possible accuracy of learning from the entire dataset (assuming one sensor node collects all the data and builds the classifier), referred to as "A", and accuracy of the ensemble method, referred to as "E".

From the figure, we find that the preservation factor does affect the accuracy, especially when it is small, the height is big, and the noise is large. When the noise is very small (1%), the preservation factor does not affect the accuracy when the factor is larger than 0.4, regardless of the height. When the noise becomes greater (10%), the preservation factor should be at least 0.8, so as not to decrease the accuracy. When the preservation factor is very small, the larger the height is, the less accuracy it achieves. We also find that when the preservation factor is large enough, the accuracy is almost the same as that of learning from the entire training data, i.e., the practically optimal accuracy.

The figure indicates that our mechanism to generate pseudo data works quite well, in particular, when the factor is large enough, the pseudo data can well represent the original data. When the preservation factor is small, the accuracy becomes relatively lower. The reason is that, from the whole system's view, the areas that are close to the base station generally become dominative, but as mentioned before, we should consider different areas equally. Moreover, when the height is higher, the noise will be accumulated, hence the accuracy will be reduced.

Comparing with the ensemble method, when the preservation factor is large enough, our approach achieves much higher accuracy, and when the noise is greater, the difference is even larger. This is because in the ensemble method, for heterogeneous data, each sensor node only learns a part of the data (e.g., low temperature and low humidity). In other words, in the ensemble method, only a few classifiers are responsible for a certain test data. If a given unlabeled data has the attribute value that is much different from its training data, it probably needs to randomly guess a class label, which will greatly decrease the accuracy.

4.4. Comparison of Enhanced and Basic ID3 Algorithm

We next compare our enhanced ID3 algorithm with the basic ID3 algorithm. We clarify that we modify the basic ID3 because it does not suitable for generating pseudo data in our approach for the heterogeneous data distribution. We again use the heterogeneous data with different noises, and plot the results in Figure 5 and Figure 6, respectively.

We find that the enhanced ID3 algorithm is not noticeably affected by the preservation factor (when the factor is large enough), nor the height. On the other hand, the accuracy of basic ID3 algorithm is much lower than ours, and is particularly lower when the preservation factor is smaller and the height is larger.

The difference is because when all the samples belong to the same class label, the basic algorithm stops the recursion, while our enhanced version continues. As mentioned before, in heterogeneous data distribution, it is probably that all the data from one node are below 10 in temperature and below 20 in humidity, and all the labels are negative; if we utilize

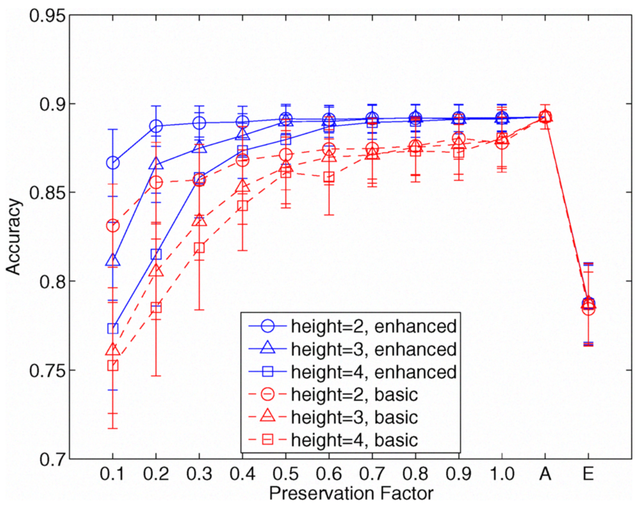

Figure 6. Comparison of enhanced and basic ID3 algorithms for heterogeneous data with $\varepsilon = 10\%$

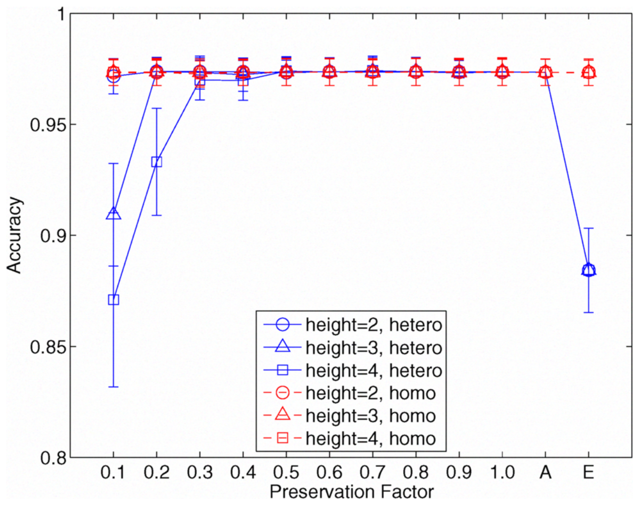

Figure 7. Comparison of heterogeneous and homogeneous data distribution with $\varepsilon = 1\%$

the basic ID3, the decision tree is likely to contain only one attribute, say temperature, with the information of humidity being completely missing. As such, when generating the pseudo data, the humidity has to be uniformly distributed from 0 to 99, adding remarkable noises.

To validate this, we also examine their performance with the homogeneous data, and the results show that the two algorithms perform almost the same.

4.5. Impact of Training Data Distribution

To further understand the impact of the distribution of the data set, we perform comparative experiments with both the heterogeneous data and the homogeneous data. Figure 7 and Figure 8 plot their respective accuracy results with different noises.

We find that for the heterogeneous data, when the preservation factor is small, the accuracy is low. On the other hand, for the homogeneous data, the factor does not affect the accuracy. For data with small noise, when the preservation factor is greater than 0.4, the two achieve the same accuracy. If the noise is larger, the preservation factor has to go beyond 0.8 to achieve the same accuracy. This is because in homogeneous data distribution, all the nodes have similar training data, thus the classifiers built by the nodes are almost the same, which leads to the high similarity between the generated pseudo data and the local training data. Therefore, the accuracy is independent on the preservation factor for homogeneous data.

In the ensemble method, the accuracy for heterogeneous data is much lower than that of homogeneous data and our hierarchical approach. This has been explained in the

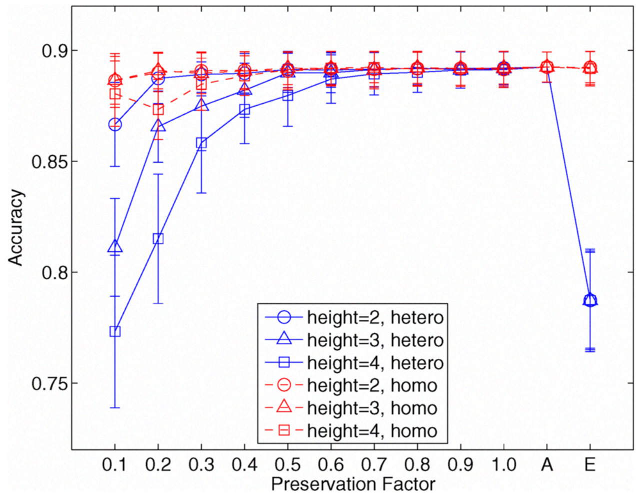

Figure 8. Comparison of heterogeneous and homogeneous data distribution with $\varepsilon = 10\%$

first experiment. For the homogeneous data, the ensemble method has the same accuracy as our approach.

4.6. Comparison of Energy Consumption

Finally, we investigate the energy consumption, which is one of the most important concerns in wireless sensor networks. We compare our hierarchical classification with the ensemble method that transmits all the classifiers to the base station. The energy consumption consists of the computation energy consumption and the transmission energy consumption, which is significantly larger than the prior one [6], thus we neglect the computation energy consumption.

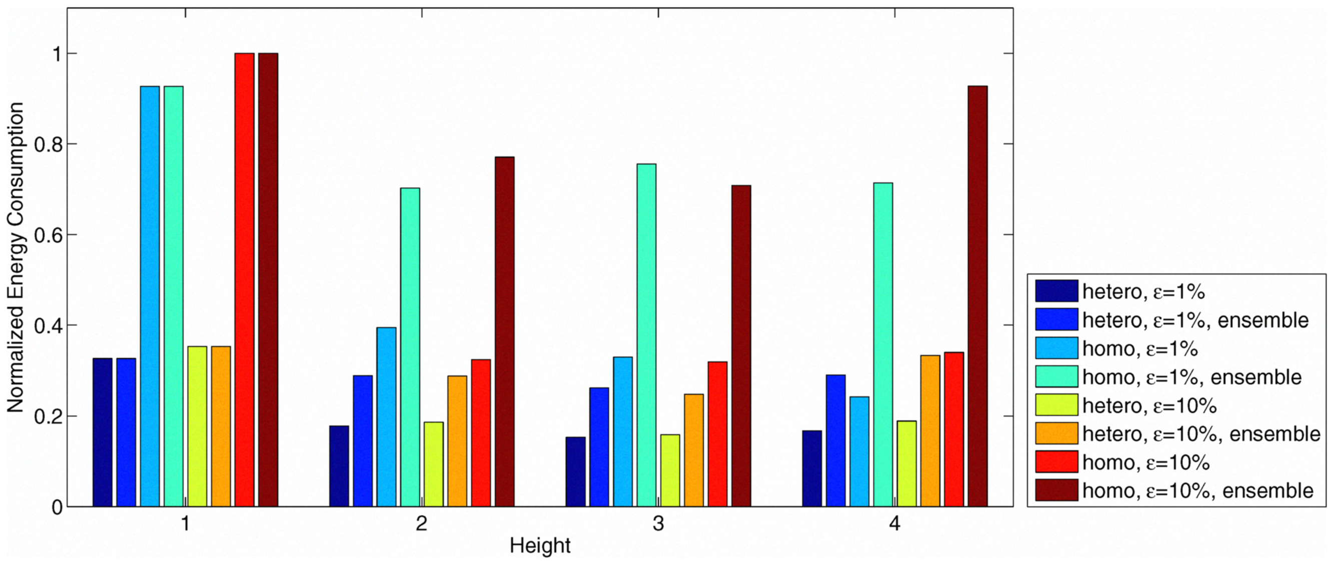

Figure 9. Comparison of energy consumption

The transmission energy consumption depends on the size of the transmitted data and the distance between the two nodes. Generally, it is proportional to the data size and square of the distance [6]. We normalize the result and plot in Figure 9.

From the figure, we find that the energy consumption of our approach is much lower than the ensemble method in all the situations when the height is greater than 1. On average, our method saves nearly half of the energy spent in the ensemble method in all situations. The greater the height is, the more energy we save, simply because a classifier is only forwarded to the parent in our solution, while it is forwarded all the way to the base station in the ensemble method.

We also find that the energy consumption for the heterogeneous data is lower than that for homogeneous data. This is because in the enhanced ID3 algorithm, the recursion stops when there is no sample at the node, and thus for the heterogeneous data, it is likely that the data from a node all exist in one branch. Therefore the size of the decision tree is smaller than that in the homogeneous data scenario.

According to the figures, the noise does not affect the energy consumption much. We believe that it is because the noise does not change the transmission distance, and it also does not noticeably change the size of the decision trees.

As we modify the basic ID3 algorithm, the constructed decision tree is thus larger in our enhanced ID3 algorithm. We also examine the size of the decision tree built in the two algorithms, and find that the size in enhanced ID3 is 45% larger than that in the basic ID3 algorithm. Considering that the basic ID3 algorithm cannot even work due to its low accuracy in our application, this increased size is reasonably acceptable.

5. Related Work

There are many classic methods for centralized classification, such as bagging [12], boosting [13] and AdaBoost [14]. Random forest [15] is another advanced tool, which combines tree predictors, such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. Random inputs and random features produce good results in the classification.

For distributed data classification using the ensemble method, several techniques have been proposed [16], [17]. Distributed classification in peer-to-peer networks is also studied [11]. The authors proposed an ensemble approach, in which each peer builds its local classifiers on the local data and then combines all the classifiers by plurality voting. These distributed solutions however are not customized for sensor networks. In particular, they do not consider the limit of memory sizes, nor the energy consumption, which are two critical concerns with battery-powered sensor nodes. Directly applying these distributed classifications into our application scenario will thus result in poor performance, as demonstrated through our simulations.

There have been numerous works on data gathering in wireless sensor networks. For example, LEACH [6] and PEGASIS [18] attempt to make the data collection task energy efficient. There are also a few related works about classification in this context [19], [20], [21], but they have focused on detecting or tracking objects, while not giving detailed design of classifiers. To the best of our knowledge, our work is the first addressing energy-efficient distributed classification for wireless sensor networks, particularly with heterogeneous data distribution across the sensor nodes.

6. Conclusion

In this paper, we have proposed a novel hierarchical distributed classification approach in wireless sensor networks. In particular, we consider a practical scenario in which the data distribution is heterogeneous, which is seldom studied before. In our solution, local classifiers are built by individual sensor nodes and merged along the routing path. The classifiers are iteratively enhanced by combining strategically generated pseudo data with new local data, eventually converging to a global classifier for the whole network. We demonstrate that our approach maintains high classification accuracy with very low storage and communication overhead.

Acknowledgment

This research is supported by a Canadian NSERC Discovery Grant, an NSERC Discovery Accelerator Grant and an NSERC Strategic Project Grant.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[2] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the World with Wireless Sensor Networks," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2001.

[3] S. Kumar, F. Zhao, and D. Shepherd, "Collaborative Signal and Information Processing in Microsensor Networks," *IEEE Signal Processing Magazine*, March 2002.

[4] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," in *Proceedings of ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002.

[5] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks," in *Proceedings of ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 1999.

[6] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, 2002.

[7] J. R. Quinlan, "Induction of Decision Trees," in *Machine Learning*. Kluwer Academic Publishers, 1986, ch. 1, pp. 81–106.

[8] J. Wieselthier, G. D. Nguyen, and A. Ephremides, "On the Construction of Energy-Efficient Broadcast and Multicast-trees in Wireless Networks," in *Proceedings of IEEE INFOCOM*, 2000.

[9] F. Bouhafs, M. Merabti, and H. Mokhtar, "A Node Recovery Scheme for Data Dissemination in Wireless Sensor Networks," in *Proceedings of IEEE International Conference on Communications (ICC)*, 2007.

[10] T. G. Dietterich, "Ensemble Methods in Machine Learning," in *Proceedings of the First International Workshop on Multiple Classifier Systems (MCS)*, 2000.

[11] P. Luo, H. Xiong, K. Lu, and Z. Shi, "Distributed Classification in Peer-to-Peer Networks," in *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2007.

[12] L. Breiman, "Bagging Predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 2002.

[13] Y. Freund, "Boosting a Weak Learning Algorithm by Majority," in *Proceedings of Workshop on Computational Learning Theory*, 1990.

[14] Y. Freund and R. E. Schapire, "A Decision-theoretic Generalization of On-line Learning and An Application to Boosting," in *Proceedings of European Conference on Computational Learning Theory (EuroCOLT)*, 1995.

[15] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[16] A. Lazarevic and Z. Obradovic, "The Distributed Boosting Algorithm," in *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2001.

[17] N. V. Chawla, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, "Learning Ensembles from Bites: A Scalable and Accurate Approach," *Machine Learning Research*, vol. 5, pp. 421–451, 2004.

[18] S. Lindsey, C. Raghavendra, and K. M. Sivalingam, "Data Gathering Algorithms in Sensor Networks Using Energy Metrics," *Transactions on Parallel and Distributed Systems*, vol. 13, no. 9, pp. 924–935, 2002.

[19] R. R. Brooks, P. Ramanathan, and A. M. Sayeed, "Distributed Target Classification and Tracking in Sensor Networks," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1163–1171, 2003.

[20] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita, "A Line in the Sand: A Wireless Sensor Network for Target Detection, Classification, and Tracking," *Computer Network*, vol. 46, no. 5, pp. 605–634, 2004.

[21] L. Gu, D. Jia, P. Vicaire, T. Yan, L. Luo, A. Tirumala, Q. Cao, T. He, J. A. Stankovic, T. Abdelzaher, and B. H. Krogh, "Lightweight Detection and Classification for Wireless Sensor Networks in Realistic Environments," in *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.