Second International Symposium on Frontiers in Ambient and Mobile Systems (FAMS-2012)

Distribution Bottlenecks in Classification Algorithms

Ardjan Zwartjes¹, Paul J.M. Havinga¹, Gerard J.M. Smit¹, Johann L. Hurink¹

Abstract

The abundance of data available on Wireless Sensor Networks makes online processing necessary. In industrial applications for example, the correct operation of equipment can be the point of interest while raw sampled data is of minor importance. Classification algorithms can be used to make state classifications based on the available data.

The distributed nature of Wireless Sensor Networks is a complication that needs to be considered when implementing classification algorithms. In this work, we investigate the bottlenecks that limit the options for distributed execution of three widely used algorithms: Feed Forward Neural Networks, naive Bayes classifiers and decision trees.

By analyzing theoretical boundaries and using simulations of various network topologies, we show that the naive Bayes classifier is the most flexible algorithm for distribution. Decision trees can be distributed efficiently but are unpredictable. Feed Forward Neural Networks show severe limitations.

© 2012 Published by Elsevier Ltd.

1. Introduction

Online data processing is an important, but complex, task on Wireless Sensor Networks (WSNs) [1]. Even on small WSNs, large amounts of data can be sampled by the sensor nodes. Simple micro-controllers can already acquire samples at rates in the order of 10kHz. This is more than what can practically be transmitted using WSN radios. For most applications, this is not a problem since the raw data itself does not need to be retrieved. For domestic fire detection [3], for example, the amount of carbon-dioxide in the air is not the important information, the presence of a fire is. Another example is found in logistics: the state of the monitored products is of importance, while no human operator ever needs to see 10-bit temperature readings.

Online data processing comes in many forms, ranging from simple schemes to compress the data that is transmitted over the network, to complex event recognition algorithms that draw intelligent conclusions. This last group of algorithms can significantly reduce of the amount of data that needs to be transmitted. Drawing conclusions from the data in code running on the nodes removes the need to transmit the raw sensor readings. Considering that the energy needed to transmit a few bytes of data can also be used to perform a considerable amount of processing [8], it is clear that online intelligent processing is a promising area of research.

Email addresses: g.j.zwartjes@utwente.nl (Ardjan Zwartjes), p.j.m.havinga@utwente.nl (Paul J.M. Havinga), g.j.m.smit@utwente.nl (Gerard J.M. Smit), j.l.hurink@utwente.nl (Johann L. Hurink)

¹University of Twente, Postbus 217, 7500AE Enschede
1.1. Problem description

The mapping of complex intelligent algorithms on WSNs is a challenging task. A key aspect of the WSN vision is to obtain reliability through the application of multiple unreliable devices. Distribution of execution over multiple devices, however, is of no concern in most traditional research on intelligent algorithms [18]. This complication maybe an explanation for the limited success of practical realizations of intelligent algorithms on WSNs.

In this paper, we try to find an intelligent classification algorithm that is suitable for WSN applications. We look into the complications related to distribution, specifically regarding energy consumption for communication.

We limit our research to a comparison of three classification algorithms, covering a wide range of algorithm types. An important selection criteria is that the algorithms should be able to work within the constraints of WSN hardware. The analysis of these problems identifies the most favorable algorithm for WSN architectures. The scope of this research is limited to algorithms that do a periodic classification, without considering the evolution of data over time.

1.2. Related work

Online classification on WSNs has been an area of research for decades. Event detection [19, 20, 22], context recognition [14], outlier detection [6] and classification [13] are different techniques to automatically draw conclusions from sensor data.

Given the limited hardware capabilities of wireless sensor nodes, classification based on fixed threshold values is often applied [19, 20, 21, 3]. More sophisticated events, however, cannot be detected in this manner. For these events, machine learning techniques are applied.

Based on the scale of the network and the application requirements and constraints, classification algorithms can be executed on the base-station [13, 22], locally on the sensor nodes [4], or distributed over the network [6, 9, 12, 15, 17]. Distributed execution makes use of the full potential of WSN technology, providing robustness in the case of sensor failure, reduced communication towards the base-station and distribution of energy consumption. On the other hand, distributed execution provides the most challenges when considering implementation and communication.

Naive Bayes [4], Feed Forward Neural Networks (FFNNs) [4], decision trees [7] and support vector machines [5] have been proposed to detect events locally on individual sensor nodes. Distributed approaches, using data from multiple nodes, range from techniques based on distributed fuzzy engines [16], map based pattern matching [10], FFNNs, and naive Bayes classifiers [2].

Distributed computing has been an area of interest since the introduction of networked computers. Most of the research is focused on computing performance: Increasing the speed of computation by running code in parallel. The focus when distributing algorithms over WSNs is different. Computing performance is not the main concern, energy efficiency is. Because of this alternate focus, results from existing research on distributed systems are not always applicable[11].

1.3. Contributions

In this paper, we give the results of a comparison between three algorithms: FFNNs, naive Bayes classifiers and decision trees. We analyze the structure of the algorithms and the way this influences the options for distribution. This comparison demonstrates that the naive Bayes algorithm is suitable for distributed implementation on multiple network topologies. Furthermore, it demonstrates serious shortcomings of the FFNN algorithms with regard to distribution. Our method of analysis can be used in future research to assess the suitability of other algorithms for WSN applications.

2. Method

2.1. Selection of algorithms

The first step of this research is to select a suitable set of classification algorithms for comparison. WSNs as a target platform limit the size of the memory and the amount of computational power that can be used.
These constraints reduce the amount of algorithms that can be selected. A second aspect of importance is that we want our comparison cover a wide range of algorithms. Therefore, we want to investigate algorithms that work in fundamentally different ways. By choosing algorithms that represent different classes the conclusions can be seen in a broader view.

A selection used in previous work on the fault tolerance of three algorithms consisted of: The naive Bayes classifier, FFNNs and decision trees[23]. All three algorithms can work under the constraints provided by the WSN platform[3, 7, 12, 2]. Furthermore, these three algorithms are based on fundamentally different principles: naive Bayes is a statistical classifier, FFNNs are inspired by nature and decision trees use a sequential script to make classifications. This led to the selection of these three algorithms for comparison.

2.2. Distribution

To investigate how suitable the selected algorithms are for distributed execution, we start this research by analyzing the data-flow between separable parts of the algorithms and the consequences of these flows for distributed execution. For this, we first analyze a scenario where the entire algorithm is run in a central location, afterwards we look for ways to improve on the energy usage of this scenario by distributing parts of the algorithm over multiple nodes.

We identify separable parts of the algorithms by identifying the points in the algorithms where data from multiple sources is combined. These points are of interest because on these points data from those sources needs to be on the same place in the network.

Based on this analysis, we model various distribution schemes for which we estimate the energy consumption. These models are used to analyze both the total energy consumption over the network and the maximal energy consumption of an individual node. Using these models, we assess how suitable the three algorithms are for distribution and what the involved energy costs are.

Our models are based on a number of assumptions: The network consists of heterogeneous nodes, producing equal amounts of sensor data for each classification. The energy used to transmit the conclusion of a classification is negligible since this only is one bit of data. The network topologies are symmetrical in the sense that branches have an equal numbers of nodes. A final important assumption is that energy costs for processing are negligible in comparison to communication costs. This assumption can be supported as follows. The CC2420 radio, using a -10dBm transmit power, uses 134.4nJ to transmit one bit of data. Receiving one bit of data costs 236.4nJ [8]. The average power consumption of instructions on an ATmega128L micro-controller is roughly 5nJ. In the rest of this work we will not consider energy costs for computation.

We calculate the energy costs for distribution for three different topologies, namely: a single-hop network, a star topology and a binary tree topology. We give examples from simulations using 126 nodes for the single-hop and star topology, and 127 for the binary tree topology to keep symmetry. In the examples we use 5 branches for the star topology. For the tree topology, we use a binary tree of depth 7. In the simulations, we use the energy profile of a CC2420 radio transmitting on -10dBm [8].

3. Analysis and results

In this section, we give the results of our theoretical analysis of the three algorithms. We use the symbols explained in (1).
The goal of distribution is to find sets \( p(n_1), \ldots, p(n_n) \mid (\bigcup p(n) \in N) = A \) that result in the smallest cost for the network.

Costs for WSNs can be measured in energy consumption and can be defined in several different ways. Minimizing the total energy consumption \( E \) on the entire network is a good option when maximizing the time until the last node runs out of energy is the goal. Minimizing the maximal energy usage of an individual node \( e_{\text{max}} \) is a good option when minimizing the time until the first node fails is the goal. The best option is application dependent. We will consider both options.

3.1. Baseline distribution performance

As a starting point for the analysis of the three algorithms, we use the most simple way to implement a classification algorithm on a WSN: sending all sampled data to a central node \( n_c \) and running the complete classifier there (\( p(n_c) = A \), \( p(n_i | n_i \neq n_c) = \emptyset \)). In a single-hop network, this would mean that all the nodes send their data, and the sink receives all data. This would lead to the energy consumption of \( E_{\text{base}} = (r+t)(|N|-1)d \) and the maximum energy usage for an individual node would be \( e_{\text{max}} = e(n_c) = r(|N|-1)d \). For line, star and tree topologies the energy consumed by \( n_c \) remains the same. Nodes on the path to \( n_c \), however, send and receive additional data. In the star topology, this would lead to \( e(n_i \in N | n_i \neq n_c) = td + (r+t)(\frac{|N|-1}{s} - H(n_i, n_c))d \). For a binary tree topology, this would lead to \( e(n_i \in N | n_i \neq n_c) = td + (r+t)(\sum_{j=1}^{l-1}H(n_i, n_j) 2^j)d \). In both cases \( e_{\text{max}} \) is still consumed by \( n_c \).

In a star topology, the total energy consumption would be \( E_{\text{base}} = \sum_{j=1}^{l} (r+t)jsd \). For a binary tree topology, the overall energy consumption would be \( E_{\text{base}} = \sum_{j=1}^{l} (r+t)2^jd \).

This approach would completely ignore all options for distributed computation and would send all data directly over the network. Clearly, this should not be the optimal solution. We use this approach as a baseline from where we look for improvements for the compared algorithms.

3.2. FFNN

In FFNNs, data is combined by neurons. Each neuron in the input layer combines the data of all the sensors into an output. Each neuron in the second layer combines each of these outputs into another output. This continues until the output layer combines its inputs in the classification result.

This structure of a simple FFNN presents an immediate problem for distribution: all the neurons are connected with all neurons in the adjacent layers. This means that, if the execution of one layer is distributed over multiple nodes, all data from the previous layer needs to be received by multiple nodes. \( n_c \) is the cheapest node on which to receive all data, since its average hop count to the other nodes is lowest. This implies that there is no scenario where one node receives all data that is more energy efficient in total energy usage than the baseline scenario. Therefore, the FFNN algorithm cannot use less energy than the baseline scenario.

For the baseline scenario, we have showed that \( e_{\text{max}} = r(|N|-1)d \). This value can also not be improved on, all nodes running parts of the input layer must receive all data from the other nodes, resulting in the
same energy consumption \( e_{\text{max}} \) as in the baseline scenario. In the case of a distributed FFNN, however, the nodes running part of the input layer also need to transmit their own data, thereby exceeding the \( e_{\text{max}} \) from the baseline scenario.

### 3.3. Naive Bayes

The structure of the naive Bayes algorithm provides better means for distribution than that of FFNNs. The naive Bayes algorithm starts with making a probability estimation of each input. A very straightforward step is to move these probability calculations to the sensor nodes providing the inputs. Assuming that the probability contains the same amount of bits as the sensor value, this does not provide any immediate communication benefits. The improvement can be found in the nature of the combining part of the algorithm. Using Bayes theorem, the equation for a basic classification \( \Pr(c=1|E) > \Pr(c=0|E) \) can be rewritten into Equation (2).

\[
\prod_i P(x(s_i,t)|c=1) / P(x(s_i,t)|c=0) > 1
\]

Equation (2) shows that the classification is the outcome of a product of the ratios between \( P(x(s_i,t)|c=1) \) and \( P(x(s_i,t)|c=0) \) for all sensor inputs and the ratio between the two classes. This product can be factored into ratios where the sensor data from all sensors on a node is combined. This would result in each node only having to send one local ratio, instead of ratios for all the sensors. Compared to the baseline scenario of Section 3.1, this would mean that each node only has to send one value per classification, instead of one for each sensor per classification.

In the single-hop scenario, a further reduction in overall communication using the same sensor data is not possible. In order to be used in the final result, data from each node has to flow in the direction of a node where the final classification is made. This means that each node has to send data at least once, and that data has to be received at least once. This is exactly what happens if all nodes combine their data locally and send the resulting probability to \( n_c \). Therefore, further improvement without changing the algorithm is impossible.

If we call the number of bits in a local Bayes ratio \( d_b \), the total amount of energy used for the naive Bayes classifier in the single-hop scenario is \( E = (r+t)d_b(|N|-1) \). The central node receiving all the data still uses the highest amount of energy, namely \( r(|N|-1)d_b \).

In topologies other than single-hop, each node can combine its own local ratio with all received ratios and send this value in the direction of the central node. In this way, each node still only has to transmit one message, and each message only has to be received once, meaning that the overall energy consumption is the same as for the single-hop scenario. For the star-topology, the central node has to receive one message for each branch, leading to \( e_{\text{max}} = rsd \). For the binary tree topology, the node that uses the most energy is not the central node, but nodes that are neither root nor branch. These nodes have to receive two messages and send one, leading to \( e_{\text{max}} = (2r+t)d_b \).

### 3.4. Decision tree

The derivation of the energy consumption for the optimal distribution of a decision tree is less straightforward than that for the other algorithms. This is caused by several factors. First of all, the structure of a decision tree is dependent on the training phase, making a general solution harder to prove. Second, for each classification only a part of the tree is used. This means that whether specific sensor data is used or not is dependent on the readings from other sensors.

A straightforward way to run a decision tree on a WSN would be to run each decision node on the sensor node equipped with the corresponding sensor. In this way, only one bit of information has to be sent for each decision made. This would lead to the situation where the number of messages is limited by the depth of the decision tree and the hop distance between the used sensor nodes.

Further improvement is possible when another sensor of the sensor node running the current decision node is used deeper into the decision tree. The decision value based on this sensor could piggy-back on the
Single-hop

<table>
<thead>
<tr>
<th></th>
<th>$E$</th>
<th>$e_{max}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>$(r + t)(</td>
<td>N</td>
</tr>
<tr>
<td>FFNN</td>
<td>$(r + t)(</td>
<td>N</td>
</tr>
<tr>
<td>Bayes</td>
<td>$(r + t)(</td>
<td>N</td>
</tr>
<tr>
<td>Dec. Tree</td>
<td>$\geq 0$</td>
<td>$\geq 0nJ$</td>
</tr>
</tbody>
</table>

Table 1. Single-hop energy comparison

Star topology

<table>
<thead>
<tr>
<th></th>
<th>$E$</th>
<th>$e_{max}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>$E_{base} = \sum_{s=1}^{</td>
<td>N</td>
</tr>
<tr>
<td>FFNN</td>
<td>$E_{base} = \sum_{s=1}^{</td>
<td>N</td>
</tr>
<tr>
<td>Bayes</td>
<td>$(r + t)(</td>
<td>N</td>
</tr>
<tr>
<td>Dec. Tree</td>
<td>$\geq (r + t)H(n_{d}, n_{c})$</td>
<td>$\geq r + t$</td>
</tr>
</tbody>
</table>

Table 2. Star topology energy comparison

messages sent down the decision tree. This would save the sending and receiving of one message for each time this node is reached. On the other hand, it adds overhead to all intermediate messages.

Because of these mentioned factors, $E$ and $e_{max}$ for the decision tree algorithm cannot be calculated directly. The best case scenario is a decision tree of depth one, where only one node ($n_d$) needs to send one bit of data to the sink. According to our assumptions this amount of data is negligible meaning that $E$ and $e_{max}$ are zero. Decision trees with a higher depth can be far less efficient, in the worst case a sequence of decision nodes jumps back and forth between the far ends of the network. Overall, the distribution performance of the decision tree algorithm is rather unpredictable.

3.5. Simulation and summary

Tables 1, 2 and 3 show both the derived formulas for the energy consumption and the results of a simple Matlab simulation on the given topologies. For this simulation, we used the energy profile of a CC2420 radio set to a -10dBm transmission power. We have not taken overhead in messages in account, the given energy figures are purely the energy used to transmit the bits of sensor data and intermediate results.

The results for the decision tree algorithm are the best case scenario where the entire classification can be done on a single node. In real life scenario’s, such a classifier will only be useful for trivial classification problems and might not be desirable given the dependence on a single node.

These results show that the naive Bayes classifier can be predictably mapped on different types of network topologies, resulting in data reductions of up to 98% in our simulations. This improvement makes the naive Bayes classifier a better choice for WSN implementations than the FFNN algorithm. This, combined with the fact that the naive Bayes classifier works well under error scenarios [23], leads us to the conclusion that the naive Bayes classifier is an algorithm that is exceptionally suitable for WSN platforms.

4. Conclusion

When looking into the options for distributing classification algorithms, FFNNNs are limited by the high number of data flows between parts of the algorithm. Regarding communication, the most efficient approach is to run the complete algorithm on a central node. In many circumstances decision trees require the least amount of communication, making them the most energy efficient. Decision trees, however, are not without drawbacks. If there is a bad match between the network topology and the structure of the decision tree a
lot of data can be sent back and forward on the network, resulting in higher energy costs. Furthermore, the sequential nature of decision trees make the algorithm vulnerable to node failures.

The order in which parts of the Naive Bayes classifier are executed is very flexible and can be perfectly adapted to the network topology. This makes the behavior of the naive Bayes classifier very predictable, even without a-priori knowledge.

Given the fact that most WSNs are dynamic systems where individual nodes can fail or can be replaced, we believe that the naive Bayes classifier, of the three compared algorithms, is the most suitable to use on WSNs.

With respect to the problem of classification on WSNs in general, we believe that careful consideration of the compatibility of algorithms with WSN architectures can prevent complications during implementation. Some existing algorithms like the FFNN have inherent problems when combined with WSN architectures.

5. Future work

This paper highlights some important aspects of the various algorithms with respect to WSNs. There are, however, many obstacles before a viable implementation is ready. This section describes some areas with room for further research.

Experimental verification

In both this work and in [23], we have shown that the naive Bayes classifier has some useful properties with regard to WSN architectures. The given results, however, are mostly theoretical. In WSN research, real-life performance often differs from the expected results. Therefore, we are working on an experimental verification of the gathered results using a real deployed WSN.

Network maintenance and installation

While theoretical analysis of algorithms and even practical experiments can provide valuable insight in the performance of algorithms on WSNs, real life deployments are far more dynamic. Deployment of the WSN and the training of a classification algorithm on such a network in an unaccessible environment combined with the complexities of replacing defective sensor nodes with new nodes in a trained classification network are matters that need to be investigated.

We are working on research where we investigate the entire life cycle of a classification network and assess the complexities to use various algorithms during all the phases of this cycle.

Algorithm optimization

In this paper, we have looked into the performance of three algorithms in their most basic implementation. Optimizations in the structure of the algorithms can cut communication costs or improve reliability in case of failing inputs. Examples include the use of two stage neural networks, combining the benefits of multiple classifiers, etc. The effects of these optimizations on the classification performance need to be investigated. For example, how well would a WSN work if the nodes first fuse their local sensor data with
a local FFNN and use a second stage FFNN to fuse these results on a central node? Although this type of classification has been done before, the exact impact on the performance of optimizing a classifier for distribution is an area of interest.

**Time aspects**

Evolution of conditions over time is an important consideration in the training of classifiers. In this research, we have looked into classifications from moment to moment. Feedback from previous classifications, however, could provide valuable information to improve classification performance. This feedback would change the structure of the algorithms. The effects of this change on the options for distribution is a direction of future research.

**References**