Using Unsupervised Learning to Improve the Naive Bayes Classifier for Wireless Sensor Networks

Ardjan Zwartjes, Paul J.M. Havinga, Gerard J.M. Smit, Johann L. Hurink
PS, CAES, DMMP
University of Twente
Enschede, The Netherlands
g.j.zwartjes@utwente.nl, p.j.m.havinga@utwente.nl, g.j.m.smit@utwente.nl, j.l.hurink@utwente.nl

***Abstract*—Online processing is essential for many sensor network applications. Sensor nodes can sample far more data than what can practically be transmitted using state of the art sensor network radios. Online processing, however, is complicated due to limited resources of individual nodes. The naive Bayes classifier is an algorithm proven to be suitable for online classification on Wireless Sensor Networks. In this paper, we investigate a new technique to improve the naive Bayes classifier while maintaining sensor network compatibility. We propose the application of unsupervised learning techniques to enhance the probability density estimation needed for naive Bayes, thereby achieving the benefits of binning histogram probability density estimation without the related memory requirements. Using an offline experimental dataset, we demonstrate the possibility of matching the performance of the binning histogram approach within the constraints provided by Wireless Sensor Network hardware. We validate the feasibility of our approach using an implementation based on Arduino Nano hardware combined with NRF24L01+ radios.**

***Keywords*-Wireless sensor networks; Unsupervised learning; Classification algorithms.**

I. Introduction

Advancements in miniaturization and the declined cost of hardware have enabled the vision of Wireless Sensor Networks (WSN), where a network of tiny computers can monitor environments using sensors and wireless communication. The implementation of a practical WSN, however, is not a trivial task. Even on small WSNs, the amount of data that can be sampled by the sensor nodes is considerable. Simple micro-controllers can acquire samples at rates above 10kHz; this is more than what can practically be transmitted using current WSN radios.

For many applications the raw sensor data itself is not of interest. For example, in domestic fire detection [4] carbon-dioxide readings do not need to reach a human operator. The presence of a fire, however, is important information. In applications like this, online processing can be a valuable solution.

Online data processing comes in many forms, ranging from simple schemes to compress the data, to complex event recognition algorithms that draw intelligent conclusions. This last group of algorithms can result in considerable reductions in communication by removing the need to transmit the sensor readings. Considering that the energy needed to transmit a few bytes of data is significant [7], it is clear that online intelligent processing is a promising area of research.

A. Problem description.

The characteristics of WSN platforms limit the type of algorithms that can be used. Both memory and computational power are very limited [12], and the unreliability of individual nodes further complicates matters. The naive Bayes classifier is a classification algorithm that can be executed on simple hardware [11]. Its performance with regard to input unreliability and distributed execution make it an interesting algorithms for WSN applications [14], [15].

The naive Bayes classifier can be implemented in multiple ways. Some of which are unsuitable for WSN hardware, while others can show poor classification performance in certain circumstances. The goal of this research is to create a naive Bayes implementation that can run within the constraints provided by WSN hardware, provides excellent classification performance and has limited overhead caused by distribution.

B. Related work

An important part of the naive Bayes [13] algorithm is probability estimation. This part of the algorithm can be implemented in various ways. Perfect probability estimation requires complete knowledge of the data distribution of the measured data. For most scenarios, this is no feasible requirement. For practical purposes a sufficiently accurate probability estimation is needed, but the WSN platform limits the algorithms that can be chosen.

A straightforward choice for the probability estimation part of the naive Bayes classifier is the use of histograms [5]. This works by first dividing the input space for each input in a number of intervals. The second step is to determine for each interval how many samples belong to each class. These values give an estimate of the data distribution of the samples for each class over the intervals, which is needed for naive Bayes. Benefits of this approach for WSNs are: it does not require floating point operations, it is computationally inexpensive, no a-priori knowledge of the data distribution is required. These aspects make histogram based approaches suitable for WSN implementations.

The method in which the histogram borders are defined, however, is of great importance. Different methods can give very different results [10]. The most basic approach divides the input space in intervals of equal width. Given an uniform data distribution this works very well, with other data distributions like Gaussian distributions, however, sparsely populated intervals can lead to a decline in classification robustness [10]. Small random variations in training data can have a significant influence on the classification output.

A method that improves on this problem is the so called binning histogram approach [10]. In this approach, the input space is divided in equally populated intervals, thereby ensuring that each interval contains a relevant amount of samples. A drawback of this approach is that in order to obtain equally populated intervals, the training data needs to be stored and sorted. This is not a task that can be executed within the memory constraints provided by WSN hardware.

In this work, we propose the use of unsupervised learning techniques to create a suitable partitioning of the input space. Our hypothesis is that unsupervised learning techniques can obtain results similar to the binning histogram approach, without exceeding the limits of WSN hardware.

II. Method

The first step in our work was an investigation using an offline dataset. We looked into the performance of binning and fixed width histograms and compared these with a number of trained classifiers using unsupervised learning. We investigated two different unsupervised learning algorithms: Self Organizing Maps (SOM) [8] and K-Means [9]. These algorithms were chosen because of their suitability for WSN implementations. We do not claim that these two algorithms are the best choice, but they are well known and suitable to proof the concept of our approach.

To compare the different classifiers in multiple situations, we used the distance to the Receiver Operator Characteristic (ROC) center line for each classifier [6]. This distance provides a metric for a classifier's capability to discriminate between multiple classes, regardless of the bias between these classes [14]. We determined the mean ROC distance and the standard deviation for all classifiers over ten training runs.

In each training run we trained the classifiers using 5000 samples from each class, or if one of the classes was rather rare then we used the maximum amount of samples we could take without using more than half of the total samples of a class for training. We used this limit to ensure that there was enough data, not used during training, to validate the performance of the classifier.

For the K-Means algorithm we used the implementation provided by Matlab, for the SOM we created a custom implementation. We chose to implement the SOM algorithm ourselves in order to gain experience for the experimental validation described later in this section.

The dataset used in this investigation was made for previous research [14], [15]. It is a dataset from multiple sensors situated around a refrigerator and coffee machine in the social corner of our research group. Three different states were manually labeled for this dataset, namely: the state of the the cooler of the refrigerator, the state of the coffee machine and the state of the fridge door.

We trained classifiers for each of the different states, using all the different approaches for interval determination.

A. *Experimental verification*

After the experiments with the offline dataset showed promising results, we verified these results using an experimental implementation. The platform for this experiment provided all the complications found on WSNs. More specifically, we wanted to validate our approach on a platform with a cheap 8-bit microcontroller, a low power radio and a network topology where nodes could have a hop-distance of at least three.

On this platform, we demonstrated the feasibility of implementing an unsupervised learning algorithm to determine histogram intervals within the given constraints. Furthermore, we demonstrated that these intervals can be used as a base for a naive Bayes classifier running on such a network. We used the distribution scheme proposed in [15] to make multiple sensor nodes collaborate in a distributed naive Bayes classifier.

We chose a classification task for which we could easily provide automatic training information: the presence of people in an office. The capability to detect this kind of information is useful for many home-automation and safety applications.

The exact details of our implementation are described in Section III-B.

III. Results

This section describes the results gathered in this research.

A. *Offline results*

Figures 1 to 3 show the results from our tests on the offline dataset. In addition to the fixed width histogram approach and the binning histogram approach we tested the K-Means algorithm and SOMs.

Figure 1 shows the result for the fridge running event. This event was the most common in the used datasets, these results were all obtained using 10000 samples.

Figure 2 shows the result for the coffee machine running event. This event occurred less frequent then the fridge running event. The amount of positive samples limited our training set size to 5703 samples.

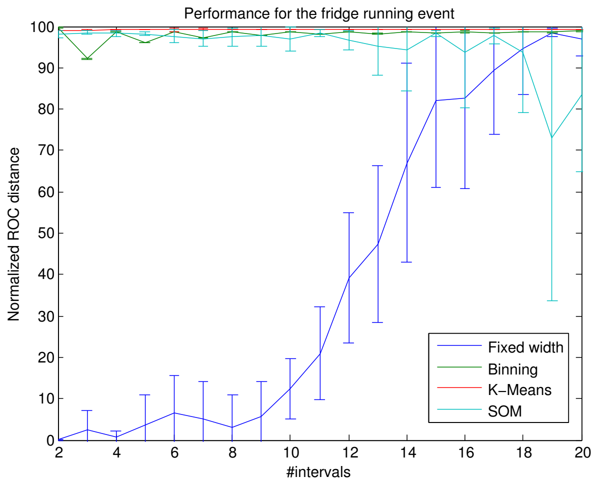

Figure 1. Fridge running performance

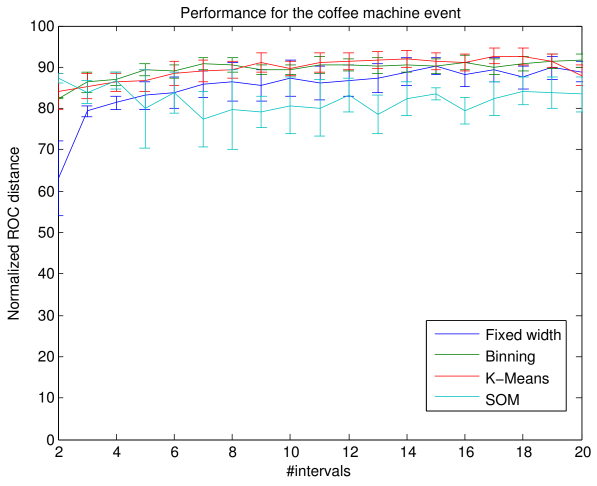

Figure 2. Coffee machine performance

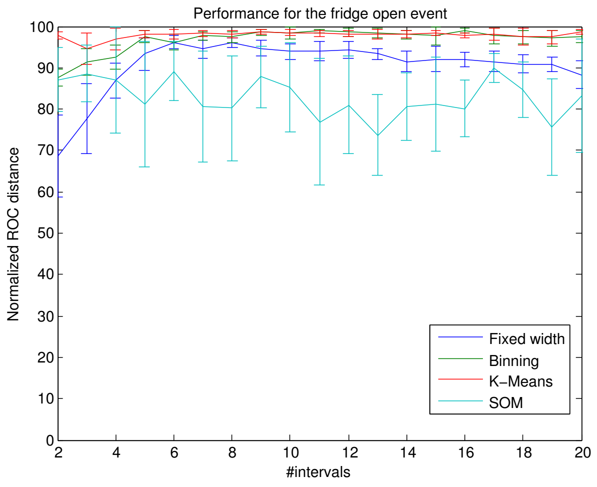

Figure 3. Fridge open performance

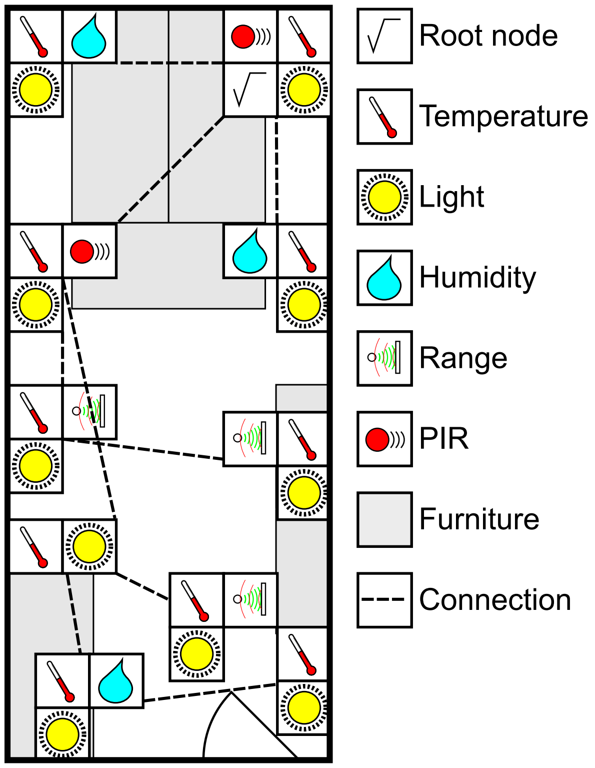

Figure 4. Network topology in office

Figure 3 shows the results for the fridge open event. This event was the rarest in our dataset. The amount of positive samples limited the training set size to 435 samples.

B. Implementation

As a platform for our implementation, we have chosen the Arduino Nano [1] experiment board. Arduino is a cheap platform for experimentation with electronic circuits. The Arduino Nano is equipped with an ATmega328 micro-controller, which has a limited set of 8-bit instructions and 2KB of SRAM (see Table I). We consider these specifications a realistic representation of WSN hardware.

For the radio, we have attached a NRF24L01+ [2] to the Arduino Nano (see Table II) . The NRF24L01+ is a low power 2.4Ghz radio that is well supported on the Arduino platform . We used the RF24Network library [3] to create a tree topology with a maximum hop distance of four. The topology of our network is shown in Figure 4. Also, the types of sensors with which each node is equipped are shown in Figure 4. The root node in this network is where all data is combined in a final classification. Each node transmits a combination of its local estimations and that of its children to its parent. Also, the types of sensors with which each node is equipped are shown in Figure 4.

We deployed ten sensor nodes around an office, using magnets to attach the nodes to white-boards and beams in the ceiling. This allowed for quick maintenance and

Type	ATmega328
Clock speed	16Mhz
Program memory	32KB
SRAM	2KB
EEPROM	1KB

Table I
MICRO-CONTROLLER

Type	NRF24L01+
Band	2.4Ghz
Data rate	2Mbps
Voltage	1.9-3.6V
Current	<13.5mA

Table II
RADIO

deployment. The ten sensor nodes were equipped with a number of different sensors. All nodes were equipped with LM35 temperature sensors and photo transistors to act as light sensors. Three nodes were furthermore equipped with ultrasonic range finders, three other nodes with humidity sensors. Finally two of the nodes were equipped with Passive Infra-Red (PIR) sensors to provide training information about the presence of movement in the office.

Each node was equipped with at most three sensors that were used by the Bayes classifier, the PIR sensors were just used as training feedback. We sampled each sensor at an arbitrary rate of 5Hz, and distilled three features from each sample stream, namely, the average value over the last second, the peak value over the last second and the slope over the last second.

We chose Self Organizing Maps (SOM) as the unsupervised algorithm to implement because these tend to divide the input space in equally populated partitions, making the expected results similar to the binning algorithm. Each node trained a SOM for each feature derived from each sensor, meaning there were up to nine SOMs per node. We chose four neurons as the size of the SOM, this number was a result of the memory constraints of the Arduino Nano.

For the distribution scheme, we used the method proposed in [15], meaning that each node had to send a single message to its parent in the network topology for each classification. We let the network make one classification per second.

Our implementation uses ten bytes of memory per sensor to buffer the sample data, 16 bytes per SOM, and 32 bytes to store each Bayes histogram. Given a typical node in our setup has three sensors with three features per sensor, our algorithm uses 462 bytes of memory. Each node uses the available EEPROM to periodically store the trained classifier.

We let our experimental setup run for a couple of days, where each node used its local sensors to create a SOM of its input space and used the feedback from the two PIR sensors to train its naive Bayes classifiers. We did not gather exact result of the accuracy of our platform, but we did periodically check if the networks classification output matched the real presence of people in the office. We noted a gradual improvement in classification performance as the training proceeded, a clear sign that our approach works as intended.

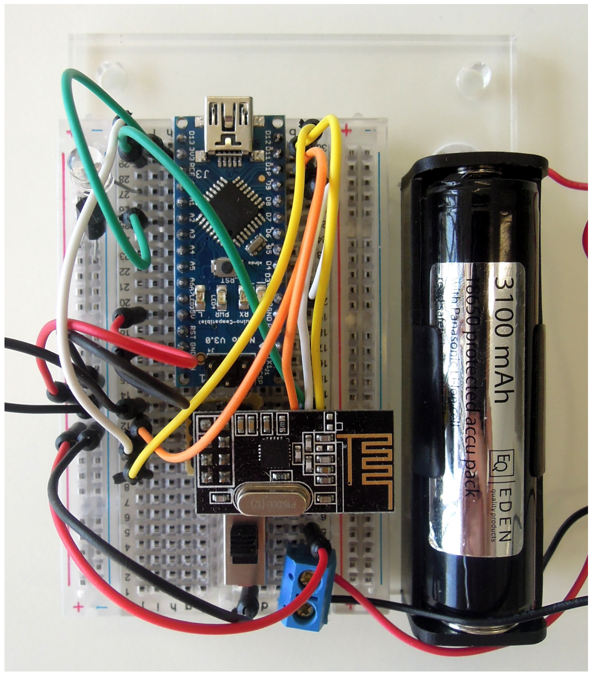

Figure 5. Sensor node

IV. ANALYSIS

In this section, we analyze the results provided in the previous section.

A. Offline results

As shown in Figures 1 to 3, binning histograms clearly give a better performance than the fixed width histograms. Only with a large amount of intervals, the performance of the fixed width histograms starts to improve. This is a clear indication that careful selection of the interval borders can result in improved classification performance.

For the fridge running event, both unsupervised learning algorithms show performance similar to the binning histogram approach and far superior to the fixed width histogram approach, especially with a low number of intervals. For the two rarer events, K-Means shows results similar to the binning approach, our SOM implementation however shows some less favorable results with a larger number of intervals.

This result can be explained by the way the Matlab version of K-Means is implemented. In Matlab K-Means works with an iterative process repeating the training until a nearly optimal result is achieved. Our SOM implementation, however, uses each sample only once, which is more realistic when considering WSN implementations. Given a large enough training set both solutions will converge on a correct partitioning, which explains the results for the fridge running event. For smaller training sets, however, the SOM has not yet converged to a stable solution which explains the decreased performance for these two events. The lower performance that can be seen for higher number of intervals can be explained in a similar way: the higher the number of intervals, the lower the number of samples that is used to train each interval. We expect that given enough samples, binning histograms, K-Means and SOMs will show the same behavior.

For practical implementations, this behavior is not problematic. Each sensor node only uses its local data to create the SOM, therefore each node can gather the information needed to create its SOM without using its radio. This means that the energy needed to create the SOM is minimal and all that is needed to determine the correct interval borders is patience.

A result that is clear for all events is that when using a low number of intervals, unsupervised learning algorithms can help create a much better naive Bayes classifier. This reduction in the number of intervals means that this approach can be implemented using even less memory than the fixed width interval approach.

B. Experimental verification

Our experimental implementation shows that our proposed solution is possible within the memory and computational constraints of WSN hardware. Our implementation runs on a simple micro controller with 2KB of RAM and an 8-bit instruction set. The used SOM algorithm automatically creates a suitable partitioning of the input space for each sensor. Given proper training data, we expect that the performance showed in the offline tests can be achieved in real life.

Our current implementation does not gather accuracy results, we could only visually check if the network was performing the desired task. As a proof that the algorithm was working as desired, this was sufficient. Especially considering that we did not investigate the optimal learning method for the SOM.

We consider the use of Arduino hardware for WSN experiments very successful. Arduinos are a versatile platform to which a variety of sensors can be attached. The addition of a radio allowed us to create a real sensor network. One limitation of the Arduino platform is energy usage. It was not designed with energy efficiency in mind and, for example, contains some LEDs that cannot be turned off.

V. Future work

Although this work gives some valuable insights in the application of clustering algorithms on naive Bayes classifiers, we have by no means created a system that can be directly applied on real life problems. This section describes some areas of research that need work before real life applications can be made.

Network maintenance and installation: While theoretical analysis of algorithms and 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: Although this paper shows promising results, we have not looked into the optimal settings for all the parameters. Different learning functions for the unsupervised algorithms, for example, can probably improve the speed with which the classifier learns, or the accuracy. Our offline results, for example, showed that the K-Means algorithm had a better performance than the SOM algorithm, especially for smaller training sets. Although this is probably caused by the fact that we used the standard Matlab toolbox for K-Means which uses the training data in multiple iterations, it is clear that it had better results with the same data. These are aspects that need to be investigated.

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 another direction of future research.

VI. Conclusion

In this work, we have demonstrated the merits of using unsupervised learning algorithms to determine histogram borders for naive Bayes. This approach can result in a significant improvement of the classifier over the traditional fixed width histogram approach, both on performance and on robustness. When given enough training data, this approach matches the performance of the binning histogram approach, without excessive memory or computational requirements. It should be noted that it is important to use enough data to train the unsupervised learning algorithms, if the unsupervised algorithm has not converged to a stable state undesirable results are possible.

We have demonstrated the validity of our offline results with an implementation on a realistic WSN platform. Next to the validation of our approach, our implementation demonstrates that the Arduino platform is an accessible platform for WSN experiments. The fact that our implementation was based on cheap and readily available components makes us recommend this approach for other research on WSNs.

REFERENCES

[1] Arduino. http://www.arduino.cc [retrieved: June, 2012].
[2] Nrf24l01+. http://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01P [retrieved: June, 2012].
[3] Rf24network library. https://github.com/maniacbug/RF24Network [retrieved: April, 2012].
[4] M. Bahrepour, N. Meratnia, and P. J. Havinga. Automatic fire detection: A survey from wireless sensor network perspective. Technical report, Centre for Telematics and Information Technology, 2007.
[5] M. Bahrepour, N. Meratnia, and P. J. Havinga. Fast and accurate residential fire detection using wireless sensor networks. *Environmental Engineering and Management Journal*, 9(2):pp. 215–221, 2010.
[6] A. P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):pp. 1145–1159, 1997.
[7] N. Chohan. Hardware assisted compression in wireless sensor networks. 2007.
[8] S. Haykin. *Neural Networks: a comprehensive foundation*. Prentice Hall, 2 edition, 1999.
[9] A. Jain, M. Murty, and P. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):pp. 264–323, 1999.
[10] S. Kotsiantis and D. Kanellopoulos. Discretization techniques: A recent survey. *International Transactions on Computer Science and Engineering*, 32(1):pp. 47–58, 2006.
[11] E. Tapia, S. Intille, and K. Larson. Activity recognition in the home using simple and ubiquitous sensors. *Pervasive Computing*, 3001:pp. 158–175, 2004.
[12] M. Vieira, J. Coelho, C.M., J. da Silva, D.C., and J. da Mata. Survey on wireless sensor network devices. *Emerging Technologies and Factory Automation*, 1:pp. 537–544, 2003.
[13] H. Zhang. The optimality of naive bayes. *17th Florida Artificial Intelligence Research Society Conference*, 2004.
[14] A. Zwartjes, M. Bahrepour, P. J. Havinga, J. L. Hurink, and G. J. Smit. On the effects of input unreliability on classification algorithms. *8th International ICST Conference on Mobile and Ubiquitous Systems*, 2011.
[15] A. Zwartjes, P. J. Havinga, G. J. Smit, and J. L. Hurink. Distribution bottlenecks in classification algorithms. *The 2nd International Symposium on Frontiers in Ambient and Mobile Systems (FAMS)*, August 2012.