On the Effects of Input Unreliability on Classification Algorithms

Ardjan Zwartjes, Majid Bahrepour, Paul J.M. Havinga, Johann L. Hurink, and Gerard J.M. Smit

University of Twente
Drienerlolaan 5
7522NB Enschede
The Netherlands
{g.j.zwartjes,m.bahrepour,p.j.m.havinga,j.l.hurink,g.j.m.smit}@utwente.nl

Summary. 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. The raw sampled data is of minor importance. Classification algorithms can be used to make state classifications based on the available data for devices such as industrial refrigerators.
The reliability through redundancy approach used in Wireless Sensor Networks complicates practical realizations of classification algorithms. Individual inputs are susceptible to multiple disturbances like hardware failure, communication failure and battery depletion. In order to demonstrate the effects of input failure on classification algorithms, we have compared three widely used algorithms in multiple error scenarios. The compared algorithms are Feed Forward Neural Networks, naive Bayes classifiers and decision trees.
Using a new experimental data-set, we show that the performance under error scenarios degrades less for the naive Bayes classifier than for the two other algorithms.

1 Introduction

Online data processing is an important, but complex, task on Wireless Sensor Networks (WSNs) [3]. Even on small WSNs the amount of data that can be sampled by the sensor nodes is considerable. Simple micro-controllers can acquire samples of rates above 10kHz; far more than what can practically be transmitted using current WSN radios. For most applications, however, the raw data itself is not of interest. For example, in domestic fire detection [6] carbon-dioxide readings never need to reach a human operator. The presence of a fire however, is important information. In logistics the state of the monitored products is of importance, while 10-bit temperature readings are of limited value.

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 raw sensor readings. Considering that the energy needed to transmit a few bytes of data is significant[10], it is clear that online intelligent processing is a promising area of research.

1.1 Problem description

Implementing complex intelligent algorithms for WSNs, is a complicated matter. Unreliability of inputs is not taken into account in traditional research in this area [26]. For WSNs however, correctly handling input failure is a key requirement.

In this paper, we investigate the problems related to unreliable inputs for three widely used classification algorithms: Feed Forward Neural Networks (FFNN), decision trees and naive Bayes. The analysis of these problems identifies the algorithm that handles the unreliability of individual inputs on WSNs best.

The target application for this study is logistics. This application leads to some assumptions: First of all, we assume that the network consists of relatively small clusters of sensor nodes collaborating in the detection of local conditions; second, we assume that all nodes within such a cluster have a bidirectional communication path to all other nodes in the cluster.

Scenarios in which such a cluster can occur include sensors attached to machinery, for example a cooling cell for food storage, or sensors inside a container with products that need to be monitored.

The scope of this research is limited to algorithms that do periodic classifications, without considering the evolution of data over time. This aspect is left for further research.

1.2 Related work

Online classification on WSNs has been an area of research for considerable time now. Event detection [27, 28, 30], context recognition [20], outlier detection [8] and classification [19] 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 [27, 28, 29, 6]. More sophisticated events, however, cannot be detected in this manner. Pattern matching and machine learning techniques are better suited for these complicated tasks.

Based on the scale of the network and the application requirements and constraints, classification algorithms can be executed on the base-station [19, 30], locally on the sensor nodes [7], or distributed over the network [8, 16, 18, 21, 23]. Distributed execution uses the potential of WSN technology, by providing robustness for sensor failures, reduced communication and distribution of energy consumption. On the other hand, distributed execution provides the most challenges when considering implementation.

Naive Bayes [7], FFNNs [7] and decision trees [9] all have been used to detect events locally on individual sensor nodes. Distributed approaches, using collaboration between nodes, range from techniques based on distributed fuzzy engines [22], map based pattern matching [17], FFNNs, and naive Bayes classifiers [5].

1.3 Contributions

In this paper, we give the results of a comparison between three algorithms: FFNNs, naive Bayes classifiers and decision trees. By analyzing the way the algorithms handle failing inputs, we demonstrate that, compared to the other two algorithms, naive Bayes classifiers shows a superior performance.

A key difference of this comparison, compared to other research, is that we look into the effect of failing inputs on the classifiers without retraining. Retraining a distributed classifier every time an input fails, for example due to a drained battery or a lost connection, would put considerable demands on the available resources of a WSN.

A part of this research was the gathering of a new experimental data-set [32]. This data-set consists of 11 streams of data from heterogeneous sensors. Three different states have been labeled for this data-set. These states differ on frequency and randomness of occurrence and can occur simultaneously, thereby causing interference in the sensor readings. This data-set can be a valuable resource for studies into various applications of WSNs and will be made publicly available.

1.4 Structure of the paper

The remainder of this paper is organized as follows: The approach used to create the results of this study is described in Section 2. Section 3 describes the actual results. Directions of future research are handled in Section 4. This paper is concluded by Section 5.

2 Method

This section describes the method used to create the results of this research. This section is divided in the method used to gather the experimental data-set (in Section 2.1), the selection of the algorithms (in Section 2.2) and the method used to compare the algorithms (in Section 2.3).

2.1 Data-set

The scope of this research is the application of WSNs in the field of logistics. Cold-chain management is an important branch of logistics where WSN technology can be applied [11, 12, 13]. Situations to be monitored in a cold chain include the correct functioning of machinery like cooling cells and the human

interaction with this machinery. Pieces of machinery in cold-chains are usually not perfectly isolated, which means that sensor readings can be influenced by other machines, humans, etc.

An application where similar conditions occur is the refrigerator in the social room of our research group. Suitable sensor interference is provided by the coffee machine positioned on top of this refrigerator. Because of these similarities, we have chosen this refrigerator as the environment from which we have created a new sensor data-set.

2.1.1 Conditions of interest In this research, we focus on the classification of three different binary conditions for the refrigerator and coffee machine:

1. Is the cooling system of the refrigerator running? This is a frequent and predictably occurring condition, without human interaction.
2. Is the coffee machine running? This is a less frequent condition, triggered by human interaction.
3. Is the refrigerator open? This is the rarest situation we look at, also triggered by human interaction.

We chose these conditions because of the variation in event types they provide. The frequency in which these conditions occur differs in orders of magnitude. Furthermore, there are human-triggered, and automatically initiated conditions.

The fact that the two machines are situated in the social corner of our research group, resulted in frequent and uncontrolled human interaction. Another aspect of these conditions, is the expected correlation between sensor readings. Opening the fridge door, for example, results in vibrations on the fridge door. These vibrations can also be seen on the coffee machine, but to a lesser extent. These factors together provide a realistic and challenging environment in which to do classification.

2.1.2 Set up of the experiment To detect the state of the fridge door, coffee machine and fridge we used Sun SPOT [1] sensor nodes. Sun SPOTs come equipped with a number of sensors [2]. We used the 3D accelerometer, the light sensor and the thermometer. Furthermore, we equipped one of the Sun SPOTs with additional sensors to measure the current through and voltage over the refrigerator and coffee machine. The current and voltage for the coffee machine and refrigerator were combined on one power outlet.

Table 1 show the placement of the sensor nodes and the sample rates used in our experiment. We chose this placement to give relevant and redundant information for the three conditions of interest.

The sample rates were chosen based on the type of sampled data. For example, light and temperature are mostly influenced by relatively low frequency phenomena, increasing the sampling rate above 10Hz does not add extra data about those phenomena. Furthermore, the sample rates are limited by hardware capabilities. For example, current and voltage samples could only be stored reliably at a rate of 160Hz.

Node location	Temperature	Light	3D Acceleration	Current	Voltage
Power outlet	-	-	-	160Hz	160Hz
Inside fridge door	10Hz	10Hz	100Hz	-	-
Outside fridge door	10Hz	10Hz	100Hz	-	-
Back coffee machine	10Hz	10Hz	100Hz	-	-

Table 1: Sample rates for various sensors

To get a complete data-set, without the risk of losing samples due to interrupted wireless communication, we connected all the Sun SPOTs to a central computer using USB cables. This data was stored on the computer using a MySQL database. Although this means that our experimental setup was no WSN in a strict sense, data collection was more reliable and resulted in a better data-set as base for our research. When needed in future work, the effects of wireless communication can be introduced in simulations.

2.1.3 Labeling of the data All human interactions with the coffee machine and refrigerator were recorded using two webcams and the open-source motion detection application "Motion" [25]. Using the captured videos and the data streams from the sensors, we manually labeled the three conditions of interest for a period of roughly two weeks.

2.1.4 Data features Since we are targeting WSN architectures, there is a limit to the number of classifications that can be made per unit of time. Therefore we preprocessed the data to reduce the sample rates. For this research, we chose a 1Hz frequency.

The risk of reducing the sample frequency is the loss of information. In order to keep valuable information in our data-set, we split the sensor streams into multiple feature streams. An example of a feature is the RMS value of a signal over the last second. For voltage and current, this provides a meaningful indication of the power consumed. A good selection of features can make classification more straightforward.

In this research, we have used the following features for the various sensor streams.

3D Acceleration: Peak magnitude without gravity of the last second and magnitude of the resampled signals.

Light: The first derivative of the low pass filtered signal (at a cut-off frequency of 1Hz) and the resampled value of the low pass filtered signal (at a cut-off frequency of 1Hz)

Temperature: The first derivative of the low pass filtered signal (at a cut-off frequency of 1Hz) and the resampled value of the low pass filtered signal (at a cut-off frequency of 1Hz)
Voltage: RMS value of the last second.
Current: Peak value of the last second and the RMS value of the last second.

Using these features for the streams shown in Table 1, resulted in 21 new streams. These 21 streams were used as input for the classification algorithms.

2.2 Selection of algorithms

A comparison of all available classification algorithms is outside the scope of this research. Therefore, we opted to make a selection of three different algorithms. For successful WSN implementation the used algorithms need to be computationally cheap, but able to perform efficient classifications. Based on related work on WSNs [6, 9, 18, 5] we selected the FFNN algorithm, the naive Bayes algorithm and the decision tree algorithm. These three algorithms are commonly used and have been implemented on WSNs before.

2.2.1 Feed Forward Neural Networks An algorithm that is frequently used for recognition tasks is the FFNNs algorithm[15]. FFNNs can be seen as directed acyclic graphs where the nodes without predecessors are used to feed information into the network (the input layer). Nodes without successors give the resulting output of the FFNN (the output layer). Each node in the graph is a processing element (neuron) that combines its inputs and generates an output.

Parameters that change the neuron's output include the weights assigned to the inputs, the transfer function and the bias. Learning algorithms are used to automatically adapt these parameters to generate a desired output for a given input.

The simple structure of FFNNs makes them easy to implement and leaves some options to see how the model works after the learning phase.

2.2.2 Naive Bayes Classifiers Naive Bayes classifiers use Bayesian statistics and Bayes' theorem to find the probability $P(C|E)$ that a given input $E = (x(s_1,t), x(s_2,t), ..., x(s_n,t))$ belongs to a class C. This probability $P(C|E)$ is estimated using Equation (1) [31]. In order to make a binary classification, the naive Bayes algorithm calculates both $P(c = 1|E)$ and $P(c = 0|E)$. The class with the highest probability is the final classification.

$$P(C|E) = \frac{P(E|C)P(C)}{P(E)} \tag{1}$$

In this case, class C can, for example, be the class of samples where the fridge is open, $x(s_i,t)$ is the output of sensor s_i on time t. The algorithm is called naive because of the assumption that all the inputs $x(s_i,t)$ have an independent contribution to $P(C|E)$.

The most time and resource consuming part of the naive Bayes classifier is the computation of $P(E|C)$. Accurately estimating this probability is important for

the classification result. In current literature of pattern recognition and machine learning, it is proposed that this probability can be estimated using standard data distributions, such as the Gaussian or Poisson distribution [4].

Another approach is the use of histograms. In this approach, the input space is partitioned into several intervals. The number of occurrences of each condition is counted for each interval. This number is used to determine the probability that an input will fall in a certain interval given classification C.

2.2.3 Decision trees Decision trees are classification algorithms that use tree like graphs to model and evaluate discrete functions [28, 27]. The input of a decision tree can contain either continuous or discrete values. The output, however, contains only discrete values.

Construction of a decision tree for classification can be done using a training algorithms like ID3 and C4.5[24]. Training algorithms use a data-set to find a decision tree of minimal depth that performs the classification. The number of nodes or depth of the decision tree should be minimized to reduce time and memory complexities. The training algorithms are usually local search greedy algorithms that result in a locally optimal decision tree.

2.3 Metrics

Rating the performance of classification algorithms is not always a straightforward task. Calculating the classification accuracy alone does not necessarily give a good indication for how well a classifier discriminates between classes. When classifying rare states, for example, a high accuracy can be reached by never detecting the state. A more complete insight in classification performance can be given by calculating the sensitivity, specificity, positive predictive value and negative predictive value for a classifier. With this approach however, it is difficult to say which metric is the most important.

In order to make a direct comparison between classifiers, we have used a technique coming from Receiver Operator Characteristics (ROC) analysis [14]. In ROC analysis, a plot is made of the sensitivity on the Y-axis versus the False Positive Rate (FPR) on the X-axis. The FPR is defined as $FPR = 1 - Specificity$. In this plot (Figure 1), the line $y = x$ is called the no discrimination line, classifiers on this line are not able to discriminate between the classifications in any way. Each classifier can be assigned a point in this graph based on the FPR and the sensitivity. The distance from this point to the no discrimination line is an indication of how well the classifier can discriminate between the classes.

In this research we compare the impact of failing inputs on various classification algorithms. To make this comparison the absolute value of this distance is of limited importance. In order to demonstrate the impact of failing inputs we have normalized over the performance using all inputs. This allows us to directly compare the relative performance penalty for failing inputs for all the algorithms.

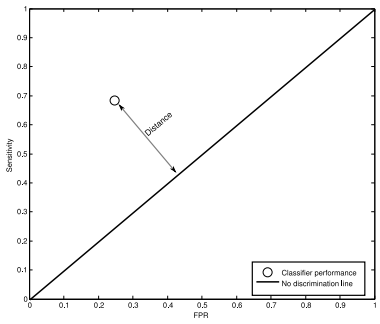


Fig. 1: The distance to the no discrimination line is an indicator for classifier performance

2.4 Classifier training

In this research, we have trained classifiers using each of the three algorithms, for each of the three conditions, resulting in a set of nine trained classifiers. The classifiers were trained over a training set and their performance was assessed using a verification set.

The training sets were selected at random from the complete data-set. The only non random aspect of the training set selection is the bias between samples with positive and negative labels. For FFNNs and naive Bayes, this bias was changed to ensure a sufficient representation of both classification options.

The size of the training-set was 10000 samples for all the training runs, except for classification of rare conditions. In our case, we had to reduce the number of training samples for the "fridge open" classifications, because of the limited number of positive samples. The size of the verification set was roughly one million samples and was formed by all data, except for the training-set.

For each algorithm, the training was repeated 20 times and the classifier performing the best on the verification set was selected. The nine resulting classifiers, one of each type for each condition, are used for the rest of this paper.

2.5 Robustness

In order to be suitable for WSNs, the used algorithms should keep working without intervention when one or multiple sensors fail. In case of sensor failure, some loss of performance is unavoidable, but the impact should be minimal for suitable algorithms.

We have investigated the impact of sensor failure on the three algorithms. To compare the three algorithms, we analyzed the selected trained classifiers for each algorithm and created a ranking of the features in order of influence on the classification result. The analysis of the feature importance for the FFNN algorithms was done by calculating the weight of each input throughout the FFNN starting from the output. For naive Bayes, we looked at how much the

one vs. zero ratios of the intervals deviated from the global one vs. zero ratio. The feature ranking for the decision tree algorithm was made based on the purity of the split made by each decision node, weighted over the probability of the node being used in a classification.

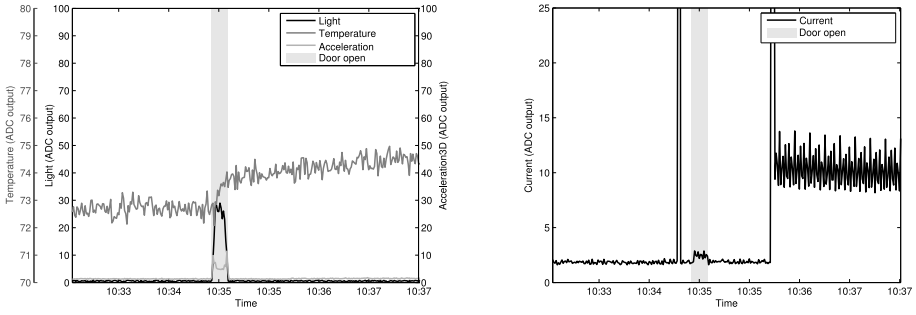
Using this ranking, we ran the classifiers in scenarios with an increasing number of failing inputs. We started by running the algorithms with all features as input and continued by dropping the most important feature until just the least useful feature was left.

3 Results

This section describes the results gathered during the course of this study.

3.1 Data-set

Data was gathered using the experimental setup over a period of two weeks. Using the video results and Matlab scripts, the three conditions of interest were labeled and the feature streams were produced. Figure 2 shows an example where the fridge is opened, as observed by multiple sensors.



(a) Inside fridge door (b) Power sensor

Fig. 2: Door open event seen by 2 different nodes

As expected, the frequency of occurrence of the three different conditions are in different orders of magnitude. The "fridge running" condition occurs 23.7% of the time, the coffee machine is on 0.65% of the time and the fridge door is open 0.05% of the time.

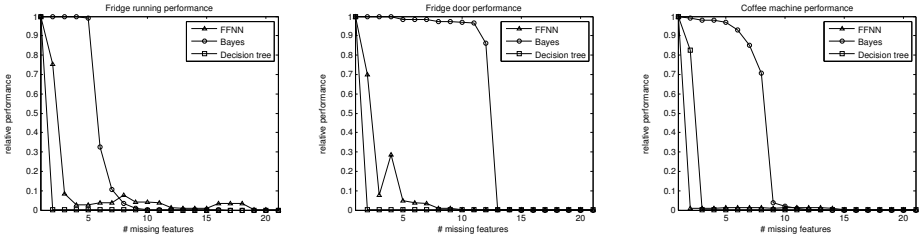
3.2 Robustness

Figure 3 shows the performance of the algorithms in various scenarios. When using a FFNN classifier, it can be seen that more then one failing feature has a severe impact on the performance.

The performance of the decision tree algorithm already shows a large drop when a single feature is dropped. This effect can be explained by the way this algorithm works. Decision trees, unlike FFNNs and naive Bayes classifiers, do not combine multiple results into an answer where all the inputs have a small influence. Only a subset of the inputs are used for sequential binary decisions. If one of those inputs is left out, entire parts of the decision tree become unreachable.

The naive Bayes classifier performs quite well, up to a certain point, when features are dropped. Depending on the condition that is being classified, multiple important features can be dropped without large effects on the classification performance.

Altogether, the naive Bayes classifier handles the error scenarios best.



(a) Fridge running (b) Fridge open (c) Coffee Machine

Fig. 3: Performance for various classifications

4 Future work

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

A key aspect of WSNs is their distributed nature. Since radio communication costs a significant amount of memory, radio usage should be kept to a minimum. Therefore, the amount of communication involved in the distribution of classification algorithms is an important consideration. Over the years numerous distributed classifiers have been developed, but a fundamental investigation of the properties that allow or limit the distribution of algorithms remains of interest.

Another direction of research is the maintenance required to keep a classifier working. With respect to classification algorithms, maintenance costs provide some interesting areas of research. The complexity of adding new nodes to a network running a classifier, for example, could increase the Total Cost of Ownership (TCO) for dynamic applications. Another example is the reprogramming of the network to do an additional classification. If this is a very time consuming process, the involved man-hours could cost a significant amount of money.

Investigating the effects of these two aspects on the TCO, for various algorithms is a promising direction for further research.

5 Conclusion

When looking into the effects of unreliable inputs, FFNN and decision tree classifiers show clear drawbacks. The high dependence on individual inputs and the inherently discrete nature of decision trees make this algorithm sensitive to input failure. Although naive Bayes is also influenced by sensor failure, we firmly believe that this algorithm overall shows the best compatibility with WSN architectures.

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.

A final note is on the data-set created for this research. The fact that this data-set was created in an uncontrolled environment and was labeled for events visible on multiple sensors makes it an excellent resource for simulations based on real data. Therefore, we believe that it can be valuable for other researchers. We will make it publicly available [32].

References

1. Sun spot world. `http://www.sunspotworld.com/`.
2. Sun spot edemo technical datasheet. Technical datasheet, Sun Microsystems Inc., 16 Network Circle, Santa Clara, CA94025 USA, 2010.
3. I. F. Akyildiz, W. Su, Y. Sankarasubramamiam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, 2002.
4. E. Alpaydin. *Introduction to machine learning.* MIT Press, 2004.
5. M. Bahrepour, N. Meratnia, and P. Havinga. Sensor fusion-based event detection in wireless sensor networks. *Sensor Fusion*, pages 1–8, 2009.
6. 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.
7. M. Bahrepour, N. Meratnia, and P. J. Havinga. Use of ai techniques for residential fire detection in wireless sensor networks. *AIAI 2009 Workshop Proceedings*, 2009.
8. M. Bahrepour, Y. Zhang, N. Meratnia, and P. J. Havinga. Use of event detection approaches for outlier detection in wireless sensor networks. *ISSNIP 2009*, 2009.
9. E. Cayirci, H. Tezcan, Y. Dogan, and C. Vedat. Wireless sensor networks for underwater survelliance systems. *Ad Hoc Networks*, 4:431–446, 2006.
10. N. Chohan. Hardware assisted compression in wireless sensor networks. 2007.
11. L. Evers, M. Bijl, M. Marin-Perianu, R. Marin-Perianu, and P. Havinga. Wireless sensor networks and beyond: A case study on transport and logistics. 2005.
12. L. Evers, P. Havinga, and J. Kuper. Flexible sensor network reprogramming for logistics. *IEEE International Conference on Mobile Adhoc and Sensor Systems*, pages 1–4, 2007.

13. L. Evers, P. J. Havinga, J. Kuper, M. E. Lijding, and N. Meratnia. Sensorscheme: Supply chain management automation using wireless sensor networks. *IEEE Conference on Emerging Technologies and Factory Automation*, pages 448–455, 2007.
14. T. Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27:861–874, 2006.
15. S. Haykin. *Neural Networks: a comprehensive foundation*. Prentice Hall, 2 edition, 1999.
16. G. Jin and S. Nittel. Ned: An efficient noise-tolerant event and event boundary detection algorithm in wireless sensor networks. *7th International Conference on Mobile Data Management*, 2006.
17. A. Khelil, F. K. Shaikh, B. Ayari, and N. Suri. Mwm: A map-based world model for wireless sensor networks. *Autonomics*, 2008.
18. B. Krishnamachari and S. Iyengar. Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor networks. *IEEE Transactions on Computers*, 53(3), 2004.
19. D. Li, K. D. Wong, Y. H. Hu, and A. M. Sayeed. Detection, classification and tracking of targets. *IEEE Signal Processing Magazine*, 19:17–29, 2002.
20. C. Lombriser, D. Roggen, M. Stäger, and G. Tröster. Titam: A tiny task network for dynamically reconfigurable heterogeneous sensor networks. *Kommunikation in verteilten systemen*, 3:127–138, 2006.
21. X. Luo, M. Dong, and Y. Huang. On distributed fault-tolerant detection in wireless sensor networks. *IEEE Transactions on Computers*, 55(1):58–70, 2006.
22. M. Marin-Perianu and P. J. Havinga. D-fler - a distributed fuzzy logic engine for rule-based wireless sensor networks. *Lecture Notes in Computer Science: Ubiquitous Computing Systems*, 4836/2007:86–101, 2007.
23. F. Martincic and L. Schwiebert. Distributed event detection in sensor networks. *Proceedings of the International Conference on Systems and Networks Communications*, page 43, 2006.
24. S. K. Murthy. Automatic construction of decision trees from data: A multi-disiplinary survey. *Data Mining and Knnowledge Discovery*, 2:345–389, 1998.
25. R. D. Pack. Motion - web home. `http://www.lavrsen.dk/foswiki/bin/view/Motion/WebHome`.
26. S. Russel and P. Norvig. *Artificial Intelligence A Modern Approach*. Prentice Hall Series in Artificial Intelligence. Prentice Hall, 1995.
27. M. L. Segal, F. P. Antonio, S. Elam, J. Erlenbach, K. R. de Paolo, and S. Beach. Method and apparatus for automatic event detection in a wireless communication system. US. Patent, 2000.
28. C. T. Vu, R. A. Beyah, and Y. Li. Composite event detection in wireless sensor networks. *Performance, Computing and Communications*, pages 264–271, 2007.
29. G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, pages 18–25, 2006.
30. W. Xue, Q. Luo, L. Chen, and Y. Liu. Contour map matching for event detection in sensor networks. *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, 2006.
31. H. Zhang. The optimality of naive bayes. *17th Florida Artificial Intelligence Research Society Conference*, 2004.
32. A. Zwartjes. Fridge dataset. `http://ps.ewi.utwente.nl/Datasets.php`.