A Machine Learning Approach for Energy Efficiency in Buildings

Flavio Ambrosi and Enrico Scapin
{vr352229, vr353597}@studenti.univr.it

1 Introduction

We present a machine learning approach to model a wireless sensor network used to gather data in the coffe room of our university. Basically, we have extracted data from a network composed of sensor. We have modeled a Bayesian Network to model the dependencies among sensor.

In this way, it is possible to learn the likelihood of specific sensor readings, given those related to the other sensors.

Our aim is provide a method to build a fault detection system similar in [3]. Although our network doesn't model time, we are able to provide a simple method to detect possible faults of appliances that impact on the energy consumption.

2 Preliminaries

A Bayesian Network is a graphical model composed of a set of nodes that represent random variables, and arcs between nodes represent conditional dependence assumptions between random variables. Hence they provide a compact representation of joint probability distributions.

In more details a Bayesian network is a directed graph in which each node corresponds to a random variable, which may be discrete or continuous. A set of directed links or arrows connects pairs of nodes: if there is an arrow from node X to node Y, X is said to be a parent of Y. The graph has no directed cycles i.e., it is a directed acyclic graph. Each node X_i has also a conditional probability distribution (CPD) $\mathbf{P}(X_i|Parents(X_i))$ that quantifies the effect of the parents on the node.

We note that if the variables are discrete, the CPDs can be represented as a Conditional Probability Tables (CPTs), which lists the probability that the child node takes on each of its different values for each combination of values of its parents.

2.1 Learning

The topology (structure) and the parameters of each CPD can be both learned from data. However, since learning structure is much harder than learning parameters, we have designed the topology of the network according to our intuition and the method given in Chapter 14.2 of Russell and Norvig [2].

On the contrary, we used Maximum-Likelihood Estimation to learn the CPDs of our nodes. It is worth noting that maximum-parameter learning on discrete models simply becomes a ratio: the probability that an event of a random variable occurs, conditioned to the particular set of events for the random variables of its parents, is the ratio between the number of times this event occur and the number of times the events of its parents occur.

$$\mathbf{P}(X = x|Y_1 = y_1, \ldots, Y_n = y_n) = \frac{N(X = x, Y_1 = y_1, \ldots, Y_n = y_n)}{N(Y_1 = y_1, \ldots, Y_n = y_n)}$$

where $N(Z_1 = z_1, \ldots, Z_k = z_k)$ is the number of times the random variables $Z_1, \ldots, Z_k$ assume the values $z_1, \ldots, z_n$.

Furthermore, we can also model hidden nodes i.e., random variables whose values are not known, but their CPTs must be inferred from the other nodes in the network. In that case, we have to use a different method to calculate the parameter estimation ı.e., the EM (Expectation-Maximization) algorithm. The explanation of this algorithm is beyond the scope of this paper, we refer the reader to Chapter 20.3 of Russell and Norvig [2].

2.2 Inference

Bayesian Networks offer an efficient formalism to make probabilistic inference in many practical situations, such as computing the *full joint distribution* of the network or the *posterior probability distribution* of some variables.

The full joint distribution states how much the model is consistent with the data and it is defined as the product of the local conditional distributions:

$$\mathbf{P}(x_1, \ldots, x_n) = \prod_{i=1}^{n} \mathbf{P}\left(x_1|parents(X_i)\right) \tag{1}$$

where $X_1, \ldots, X_n$ are the random variables involved in the network, $x_1, \ldots, x_n$ the respective possible events, and $parents(X_i)$ the values of $Parents(X_i)$.

On the other hand, we can also employ the Bayesian Network to compute the probability of a random variable, given some observed events.

$$\mathbf{P}(X|\mathbf{e}_1, \ldots, \mathbf{e}_n) = \alpha\mathbf{P}(X, \mathbf{e}_1, \ldots, \mathbf{e}_n) = \alpha \sum_{\mathbf{y}} \mathbf{P}(X, \mathbf{e}_1, \ldots, \mathbf{e}_n, \mathbf{y}) \tag{2}$$

where X denotes the query variable; $\mathbf{E}$ the set of evidence variables $E_1, \ldots, E_m$ and $\mathbf{e}_1, \ldots, \mathbf{e}_n$ the observed events; finally $\mathbf{Y}$ denotes the hidden variables $Y_1, \ldots, Y_l$.

3 Description of the system

3.1 Environment and Bayesian Network

The environment which models the Bayesian network is the coffee room of our university.

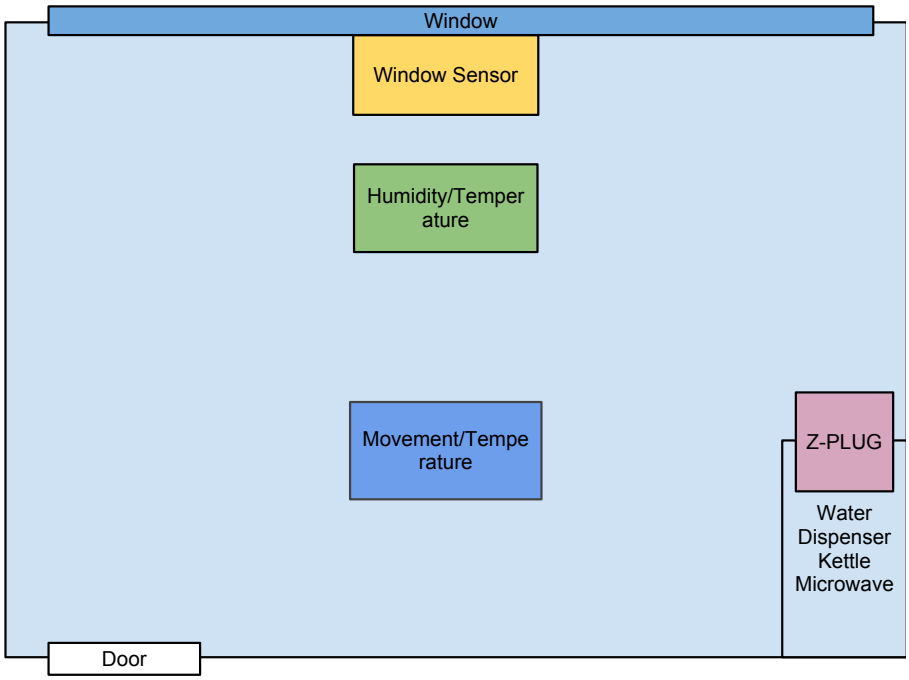


Figure 1: Environment and Network Sensors

As Picture 3.1 shows there is one door to access the room and, on the opposite side, a window; in the right corner there are three electrical appliances: a microwave oven, a kettle and a water dispenser.

We used data acquired from 6 sensors:

- opening sensor placed on the windows,
- temperature and humidity sensor near the window,
- temperature and movement sensor near the door
- energy consumption sensor, called Z-Plug, to which all the electrical appliances are plugged.

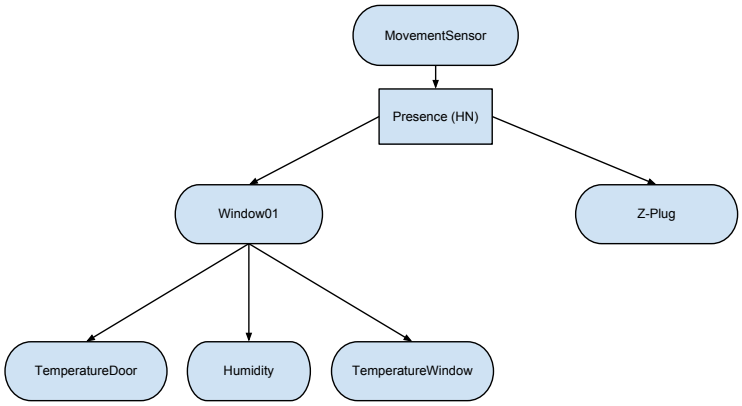
These sensors communicate with a central unit through ZigBee protocol (wireless) and the data collected are saved in a SQLite database[1].

The aim of our Bayesian Network is to model the environment by the data gathered from the whole sensors. Therefore there are six observable nodes that model the output of each sensor and one hidden node that models the possible presence of a person inside the room.

[1] `http://www.sqlite.org/`

Number	Name	Modelled data
1	MovementSensor	Motion detection
2	Presence (Hidden)	Presence inside the room
3	Window	Opening and closing of the window
4	Z-Plug	Energy consumption [$Watt$]
5	TemperatureDoor	Temperature near the door
6	Humidity	Room humidity
7	TemperatureWindow	Temperature near the window

The picture below shows the structure of our Bayesian Network



3.2 Data acquisition and Learning in BNT ToolBox

The tool we used to work with the Bayesian networks is the Bayesian Network Toolbox[2] (BNT) based on the Matlab environment. It has been developed by Kevin Murphy and it allows to work with both static and dynamic Bayesian network, performing learning and inference.

3.2.1 Data array generation for learning function

In order to work on a Bayesian network it is necessary to define the network and perform learning on data we had.

The learning function of BNT Toolbox needs a matrix with the samples of collected data. Each row of the matrix represents a node, whereas each column a sample gathered from every sensor in a given time instant.

The sensors in the room provied data in various format and with different temporal cadence hence to collect data in a coherent way we performed the following steps:

1. We have defined a *timeline*, based on the rate sampling of the temperature sensor close to the door: 3 minutes. The choice of sensor is arbitrary: it is important to fix the *timeline* and to adapt other sensors' samples to it.

[2] `http://code.google.com/p/bnt/`

2. Through SQL queries, for each temperature value we have extracted the reading of both the humidity and other temperature sensors that have *timestamp* closest to values of *timeline*. Namely, given the *timeline* value, $referenceValue$, and the tuples *timestamp* T, the closest reading can be computed as

$$\forall x \in T \quad min(abs(x - referenceValue))$$

3. Regarding the window and movement sensors, we use a binary domain indicating, whether the sensor was activated during the last 3 minutes time interval.

4. We have finally queried the database in order to extract the Z-Plug energy consumption values.

Once all values have been extracted in the correct way we have aggregated them in the matrix `data`, which input for the learning function.

We note that we have treated temperature, humidity and power data as discrete random variables, even if they describe continuous physical quantities. In that way the "movement" and "window" variables' domain is binary, while the cardinality of the Z-plug, temperature and humidity domain is the range between the minimum and the maximum detection of the correspondent discretized values. In order to avoid huge discrete domains we have chosen to approximate values to the first decimal number.

Finally, we have left empty the matrix row relative to the hidden node as required by the learning function.

3.2.2 Bayesian Network Generation and Learning

Using the data in the data matrix [#`nodes` × #`samples`], we have created the Bayesian Network and generated the CPT. In particular, the network is represented by a `struct` created by the function `mk_bnet`, mainly containing an adjacency-matrix, the domain cardinality of each node and the index of observable nodes (the others are treated as hidden).

We have hence built the CPT for each node. Since our network has all discrete nodes and TemperatureDoor, TemperatureWindow, Humidity and Z-plug have not binary domain, we have chosen to model CPD with a *Dirichlet Distribution*, that guarantees a better precision in multinomial distributions [1].

The functions `createDataBnet`, `createDataZplug` perform query on the database, set up the timeline and process data to create the matrix for the learning function.

The function `calculateCPT` creates the network and calls the learning function `learn_params_em` which populate the node's CPT.

3.3 Inference in BNT Toolbox

Once the CPT of each node has been defined we can make inference by using either exact or approximate inference algorithms. Inference is based on a BNT *engine*, which is software object that allows to compute marginal probability. Every engine use a different inference algorithm each of which is a different trade-offs between speed, accuracy, complexity and generality. We have chosen

Junction Tree engine i.e., an exact inference algorithm typically efficient when working discrete nodes. To make inference we need to:

- set up the engine with our network. (`jtree_inf_engine`),
- set up evidence nodes and theirs values. (`enter_evidence`),
- make inference calling `marginal_nodes` on query variables.

4 Testing

4.1 Accuracy of Bayesian Network Model

In order to test the accuracy of the model we calculated its probability with this formula:

$$\mathbf{P}(X) = \alpha \mathbf{P}(X|parents(X))\mathbf{P}(parents(X))$$

For example for the `TemperatureDoor` node we calculated:

$$\begin{aligned}\mathbf{P}(TemperatureDoor) = \\ \mathbf{P}(TemperatureDoor|Window = close)\mathbf{P}(Window = close) \\ + \mathbf{P}(TemperatureDoor|Window = open)\mathbf{P}(Window = open)\end{aligned}$$

We created a histogram with the result and we compare it with the Golden Model. The Golden Model represent the real probability, we computed it using classic probability formula:

$$\mathbf{P}(X) \approx \frac{n_x}{n_t}$$

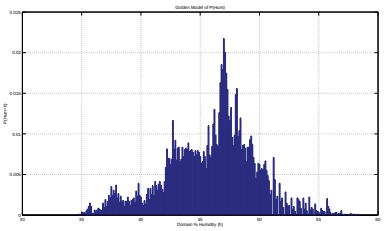
where n_t is the total number of trials and n_x is the number of trials where the event occurred. In the `TemperatureDoor` node for every value of temperature collected we calculated this probability and we create a histogram. We have done this for all nodes. As you see in Figure 2, this test confirms the correctnes of our implementation, since the Golden Model and Bayesian Network Model give the very similar.

4.2 Fault Detection

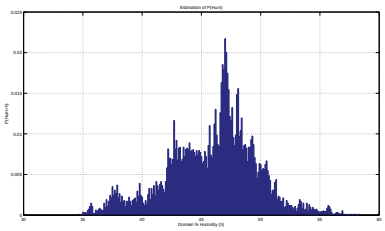
Finally, we have exploited our Bayesian Network as a fault-detection system of the energy consumption.

First of all, we have trained the system for 22 days from the 26th of June to 17th of July, 2012. In that way, we have obtained the CPT of each network node. Hence, we have used the data gathered from 18th to 27th of July, 2012 to simulate an unexpected energy consumption during periods without movement detection: we selected the range where the movement sensor remains off for at least 30 minutes so that we could assume there is no person inside. In this range we have hence injected a fault value into the Z-plug data, distinguishing three different faults:

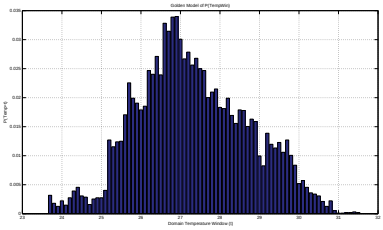
1. fault-injection of all data inside all ranges;
2. fault-injection of all data inside some ranges, selected randomly;



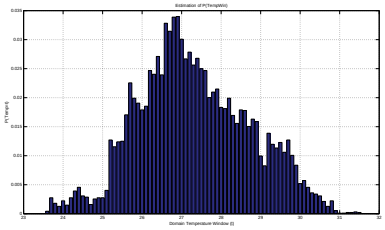
(a) G.M. Humidity



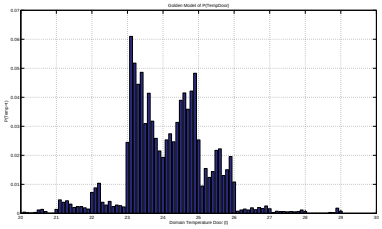
(b) B.N. Model Humidity



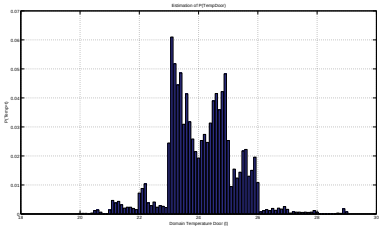
(c) G.M. TemperatureWindow



(d) B.N. Model TemperatureWindow



(e) G.M. TemperatureDoor



(f) B.N. Model TemperatureDoor

Figure 2: Golden Model versus Bayesian Network Model

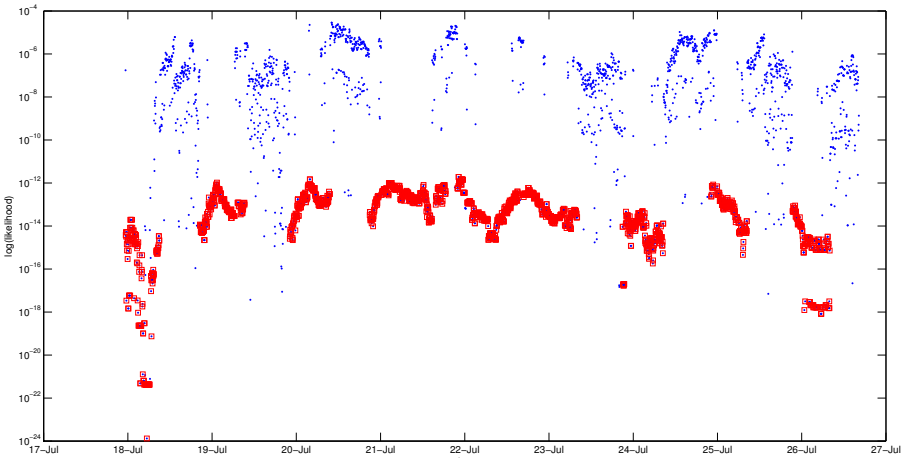


Figure 3: Full Joint Distribution in case 1

3. fault-injection of one datum inside all ranges.

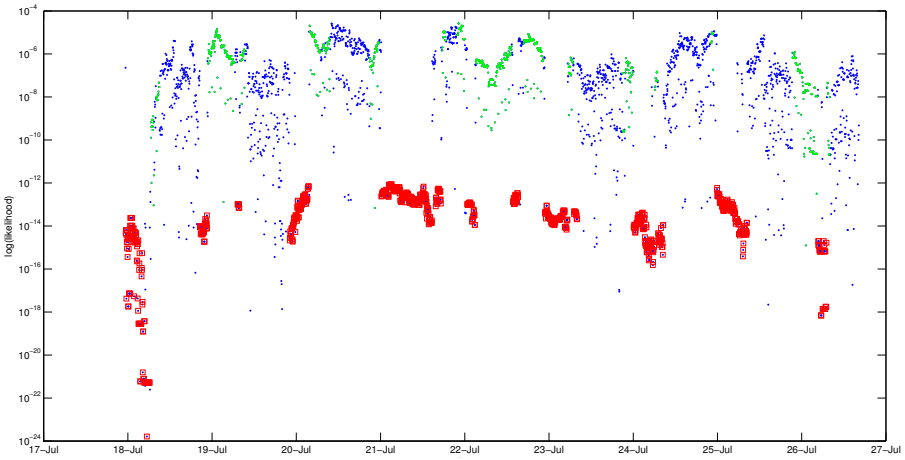


Figure 4: Full Joint Distribution in case 2

The result of these tests are shown in Figure 3, 4 and 5. The x-axis of graph shows time and the y-axis is the log likelihood of the combined probability over the whole network - i.e., the full joint probability distribution over $U = \{X_1, X_2, \ldots, X_k\}$ where X is the variable expressed by the node of the network (see formula 1) – with the lower the likelihood, the higher the likelihood that a fault is occurring.

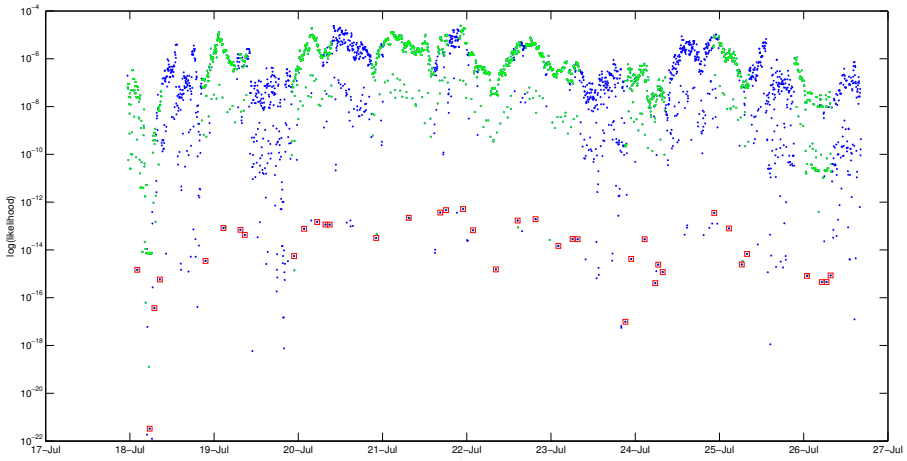


Figure 5: Full Joint Distribution in case 3

As can be seen, fault values (in red square) have a less log likelihood i.e., around 10^{-15}, respect to the others i.e., around 10^{-10}. Hence it is possible to detect a fault in the Z-Plug every time the log likelihood has a lowering in a not negligible time interval (such as at least three detections).

Furthermore, the three different tests show that the log-likelihood is *temporal independent* i.e, inserting a fault does not affect the likelihood of the next samples. That is because we modeled a static Bayesian Network and hence, after learning, each sample is independent from the others.

Although there are some false positive about faults, we can suggest a method to create a fault alarm of that system: namely, it is enough to compute the full joint distribution of each data sample and to alert if the last n full joint distributions are all around $10^{-}15$. Obviously, the exact value of n will be regulated through several tests in the real environment.

5 Conclusion

In this paper we presented a Bayesian Network data-structure, a well-developed representation for uncertain knowledge. Through this data-structure we are able to learn the Conditional Probability Distribution of the network random variables through statistical method such as Maximum-Likelihood Estimation and Expectation-Maximization. We hence used this model for detecting faults on the energy consuming, by developing a simple method based on the full joint distribution of the whole network.

References

[1] Kevin Murphy. *How to use the Bayes Net Toolbox*. 2007. URL: `http://bnt.googlecode.com/svn/trunk/docs/usage.html` (cit. on p. 5).

[2] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. 3rd. Prentice Hall, 2009 (cit. on pp. 1, 2).

[3] J. Li S. West Y. Guo J. Wall. "A Machine Learning Approach for Fault Detection in Multi-variable Systems". In: *Proceedings of ATES in conjunction with Tenth Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*. 2011, pp. 217–298 (cit. on p. 1).