Machine Learning, assignment no. 1

ALAN RAMPONI, student ID: 179850, University of Trento

This assignment is about the use of Hugin, a Bayesian network based decision tool. First, I wrote a script that automatically splits the given dataset into two sets (training and test sets) randomly for three times. Second, I used the software to learn the Bayesian networks from data. Finally, I analyzed the results of the performed classification tasks in order to evaluate the performances and to extract some insights.

Categories and Subject Descriptors: [**Computing Methodologies**]: Machine Learning—*Bayesian network models*

1. INTRODUCTION

A Bayesian network is a probabilistic graphical model used to model a domain containing uncertainty. Formally, it is a directed acyclic graph where each node represents a random variable (*r.v.*). Each of them contains the states of the r.v. it represents and a conditional probability table, i.e. a table that contains probabilities of the node being in a specific state given the states of its parents. In this report it is described the process of sampling the given dataset, the learning of the Bayesian networks and the evaluation of the classification tasks.

2. THE DATA AND ITS SAMPLING

The first step was to analyze the data provided and to understand its own structure. The dataset used for this assignment is the **Iris flower data set** [Fisher 1936], a widely known and used multivariate dataset that consists of 50 samples from each of three species of Iris (*Iris setosa*, *Iris virginica* and *Iris versicolor*). Each sample has 4 features: the *sepal length*, the *sepal width*, the *petal length* and the *petal width*. An example of the structure just briefly explained can be seen in Table I.

Table I. Iris flower data set: a subset of the data contained within the file.

sepal length	**sepal width**	**petal length**	**petal width**	**TYPE**
5.1	3.5	1.4	0.2	*Iris-setosa*
7.0	3.2	4.7	1.4	*Iris-versicolor*
6.3	3.3	6.0	2.5	*Iris-virginica*
...	...	...	...	...

According to our purposes, the original dataset was splitted randomly into two subsets: a training set and a test set. This operation of sampling was repeated 3 times in order to generate three (random) different pairs of training and test sets. For this task it was implemented a Python script (named *sampler.py*[1]) that takes in input three parameters (n_iter, $train\_len$ and $test\_len$) to automatize and satisfy our needs. In particular, the script was purposely designed using the *scikit-learn* library [Cournapeau 2007] in order to practice one of the most famous ML libraries. To launch the script we just have to open a terminal and type $python\ sampler.py\ 3\ 100\ 50$ (i.e. the values requested for this task,

[1]Due to conflicts in the interpretation of the data by the Hugin software within the Linux environment a punctation translation was implemented in the code. In particular, the dots were replaced by commas and the commas were replaced by semicolons.

n_iter, $train\_len$ and $test\_len$ respectively). Thus, we can obtain $train\_1.dat$, $train\_2.dat$, $train\_3.dat$ and $test\_1.dat$, $test\_2.dat$, $test\_3.dat$.

3. THE LEARNING TASK

In order to learn the Bayesian networks from data it was used **Hugin Lite** [Hugin 1989], a limited version of Hugin Developer, a powerful Bayesian network based decision tool.

First, the learning wizard needed to gather the required information about the data. Thus, the training files were loaded into the environment and the features (i.e. the numeric nodes inferred from the data) were marked as continuous instead of numbered.

Second, the software was ready to learn the structure of the model for the specified data. The learning was performed using the NPC (*Necessary Path Condition*) algorithm [Steck 2001] offered within Hugin. Farther, the level of significance used by the structure learning algorithm was setted to 0.05, i.e. the typical value that is appropriate for most learning sessions. The probability of rejecting a true independence hypothesis is given by it.

Then, the GUI allowed to investigate the strengths of the marginal dependences between pairs of variables. The resulting DAG (*Directed Acyclic Graph*) showed the independences found in the data. We can see in Figure 1 and in Figure 2 the comparison between an high and a low threshold, that is a threshold such that only link in the current graph with marginal p-values less than the threshold are shown. For the third training set ($train\_3.dat$), in particular, the derived structure contains structural uncertainties, i.e. ambiguous regions and undirected links.

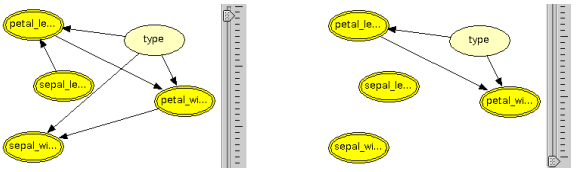

Fig. 1. Strengths of the dependences found in the data for high and low threshold (both $train\_1.dat$ and $train\_2.dat$).

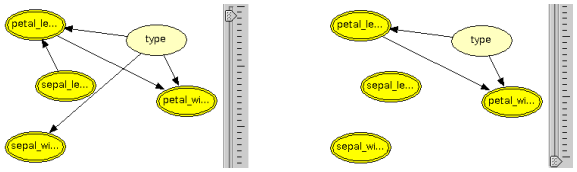

Fig. 2. Strengths of the dependences found in the data for high and low threshold ($train\_3.dat$).

In addition to this, all experience tables were initialized to 1. Thus, by including such knowledge before the learning, the resulting probabilities will be based on both the prior knowledge and the data.

Finally, the process of learning the conditional probabilities for the variables from the data was started (*EM learning*). The number of iterations was setted to 5 with a convergence threshold setted to 1.0E-4. In this process the software extracted the conditional distributions from the data. The results can be viewed in Table II[2], and the details of the networks can be found in the Appendix (Figure 4, Figure 5 and Figure 6).

Table II. EM learning: final results for each of the training file.

training file	**Log-likelihood**	**AIC**	**BIC**
$train\_1.dat$	-220,593	-251,593	-291,973
$train\_2.dat$	-234,301	-265,301	-305,681
$train\_3.dat$	-249,78	-277,78	-314,252

[2]It is important to note that the results don't change if the number of iterations is setted to a number ≥ 5.

4. THE EVALUATION

After each learning task, the resulting Bayesian networks were tested using the analysis wizard of Hugin. Thus, for each learned network its own test set (composed by 50 samples) was loaded in the environment (e.g. for $train\_1.dat$ was loaded $test\_1.dat$) and after a quick check of the imported samples in the "data source" tab, the accuracy of the generated data was analyzed. In particular, as we can see in Table III, the average classification error rate is 6%, that is quite satisfactory for our purposes. In addition, the confusion matrices are shown in Figure 3.

Table III. Evaluation: data accuracy for each of the test file.

test file	error rate	avg. Euclidian distance	avg. Kulbach-Leibler div.
$test\_1.dat$	**8.00**	-251,593	-291,973
$test\_2.dat$	**6.00**	-265,301	-305,681
$test\_3.dat$	**4.00**	-277,78	-314,252

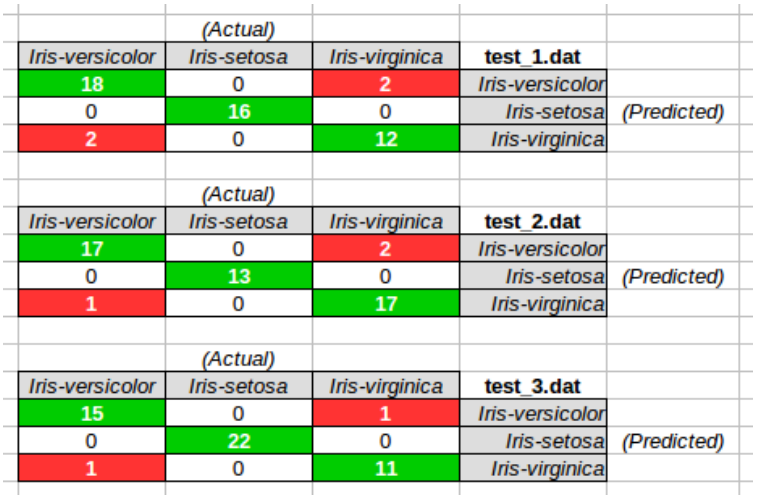

Fig. 3. Confusion matrices that summarize the performances of the classification tasks. The misclassified samples are shown in red while the correctly classified samples are shown in green.

More details of the classification tasks can be found in Table IV and Table V in the Appendix.

5. DISCUSSION

As mentioned in Section 4, the error rate is acceptable for this kind of classification. In fact, in more delicate classification tasks, such as the discrimination between edible and poisonous mushrooms, the very same average error rate would have been too high. Among the other things, it is important to note that the misclassified Iris flowers are all in the *Iris-versicolor* and *Iris-virginica* categories. This means that the *Iris-setosa* samples are so different with respect to the others in terms of the values of their features.

In conclusion, it is possible to state that such kind of classification problem gave good results and it allowed to learn better the concept of Bayesian network as a probabilistic graphical model.

APPENDIX

In this section there are some details about the learned Bayesian networks and their evaluation.

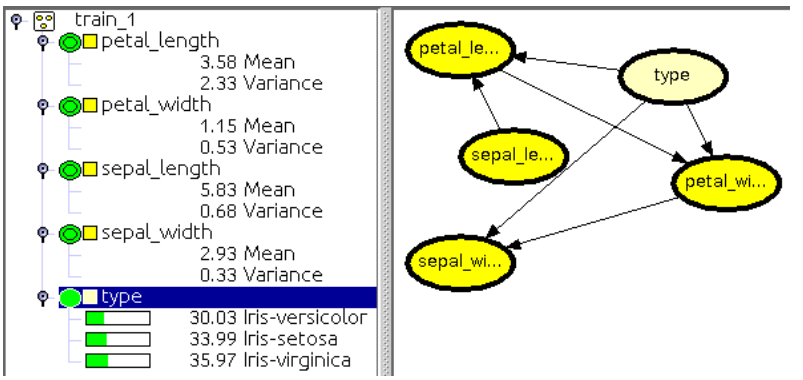

Fig. 4. Learning task: details of the network 1.

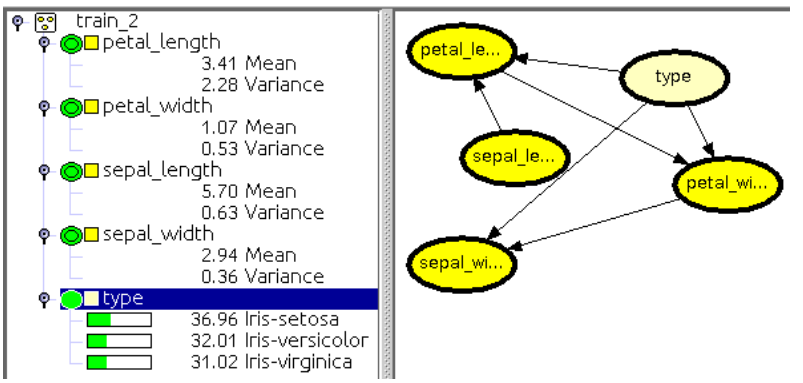

Fig. 5. Learning task: details of the network 2.

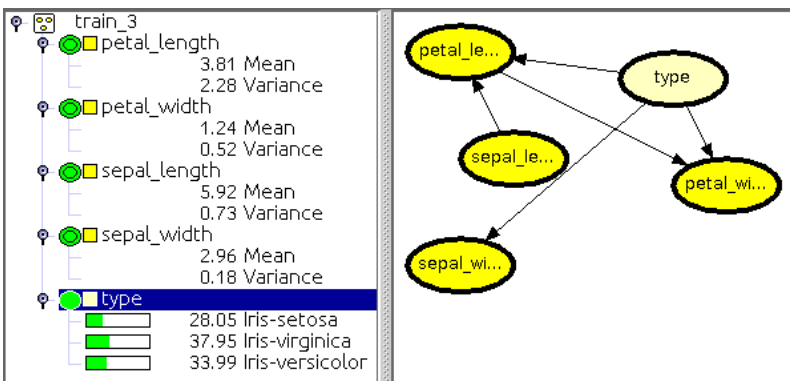

Fig. 6. Learning task: details of the network 3.

Table IV. Evaluation: results for each of the test file.

test file	**Log-likelihood**	**AIC**	**BIC**
$test\_1.dat$	-129,305	-189,942	-160,305
$test\_2.dat$	-119,543	-150,543	-180,180
$test\_3.dat$	-123,303	-151,303	-178,071

Table V. Evaluation: AUC of each Iris type for each of the test file.

test file	**Iris-setosa**	**Iris-versicolor**	**Iris-virginica**
$test\_1.dat$	0,93842	0,93583	0,91171
$test\_2.dat$	0,92412	0,9401	0,94312
$test\_3.dat$	0,95536	0,93658	0,91557

REFERENCES

David Cournapeau. 2007. Scikit-Learn: Machine Learning in Python. Python library. (2007). http://scikit-learn.org/stable/

Ronald Fisher. 1936. Iris Flower Data Set. UCI Machine Learning Repository. (1936). http://archive.ics.uci.edu/ml/datasets/Iris

Expert Hugin. 1989. Hugin Lite: The Leading Decision Support Tool (limited version). Hugin Expert website. (1989). http://www.hugin.com/productsservices/demo/hugin-lite

Harald Steck. 2001. *Constrained-Based Structural Learning in Bayesian Networks Using Finite Data Sets*. Ph.D. Dissertation. Mnchen, Germany.