Object Oriented Programming 2013/14

Tree augmented naive Bayes classifier

MEEC – IST

1 Problem

Classification is the problem of categorizing unseen examples into predefined classes based on a set of training examples. The learning algorithm generates a model from a complete set of independent instances and their corresponding categories; the model is then used to predict the categories of novel instances.

With the continuous expansion of data availability it is critical to provide automatic classification and analysis from raw data to support decision-making processes. Domains of application include surveillance, security, internet, finance, health care, just to name a few.

There are several approaches to perform classification, namely those based on neural networks, support vector machines, Bayesian network classifiers, regression, among others. In this project we focus our attention on discrete Bayesian network classifiers.

2 Bayesian network classifiers

Let X be a *discrete random variable* taking values in a finite set $\mathcal{X}$. We denote an $(n+1)$–dimensional *random vector* by $\mathbf{X} = (X_1, \ldots, X_n, C)$ where each component X_i is a random variable over $\mathcal{X}_i$ and C is a random variable over $\mathcal{C}$. For each variable X_i, we denote the elements of $\mathcal{X}_i$ by $x_{i1}, \ldots, x_{ir_i}$ where r_i is the number of values X_i can take. We say that x_{ik} is the k-th value of X_i, with $k \in \{1, \ldots, r_i\}$. In addition, the values in $\mathcal{C}$ are denoted by $z_1, \ldots, z_s$, where s is the number of values C can take.

The probability that a random vector $\mathbf{Y}$ takes value $\mathbf{y}$ is denoted by $P(\mathbf{y})$, conditional probabilities $P(\mathbf{y} \mid \mathbf{z})$ being defined correspondingly.

An **(augmented) Bayesian network classifier** (BNC) is a triple $B = (\mathbf{X}, G, \Theta)$ where:

- $\mathbf{X} = (X_1, \ldots, X_n, C)$ is a random vector where each random variable X_i and C range over by a finite domain, $\mathcal{X}_i$ and $\mathcal{C}$, respectively. The variables $X_1, \ldots, X_n$ are called **attributes**, or **features**, and C is called the **class variable**.

- $G = (\mathbf{X}, E)$ is a **directed acyclic graph** (DAG) with nodes $\{X_1, \ldots, X_n, C\}$ and edges E representing direct dependencies between the variables. This DAG is such that C has no parents, and C is a parent of all other nodes X_i.

- The **parameters**

$$\Theta = \{\theta_{ijkc}\}_{i\in\{1\dots n\},\, j\in\{1,\dots,q_i\},\, k\in\{1,\dots,r_i\}, c\in\{1,\dots,s\}} \cup \{\theta_c\}_{c\in\{1,\dots,s\}}$$

  encoding the *local distributions* of the network via

$$P_B(X_i = x_{ik} \mid \Pi_{X_i} = w_{ij}, C = z_c) = \theta_{ijkc} \text{ and } P_B(C = z_c) = \theta_c,$$

  where Π_{X_i} denotes the (possibly empty) set of **parents** of X_i in G, excluding the class variable C. Moreover, for each node X_i, the number of possible **parent configurations** (vectors of parent's values) is denoted by q_i. The actual parent configurations are ordered (arbitrarily) and denoted by $w_{i1}, \dots, w_{iq_i}$ and we say that w_{ij} is the j-th configuration of Π_{X_i}, with $j \in \{1, \dots, q_i\}$.

A Bayesian network classifier defines a joint probability distribution over $\mathbf{X}$ given by

$$P_B(X_1, \dots, X_n, C) = P_B(C) \times \prod_{i=1}^{n} P_B(X_i \mid \Pi_{X_i}, C). \tag{1}$$

The problem of learning a BNC given data T consists in finding the BNC that best fits the data T. In this project we focus our attention to **score-based learning** approaches, where a **scoring criterion** ϕ is considered in order to quantify the fitting of a BNC. In this context, the problem of learning a BNC can be paraphrased in the following optimization problem: Given a data

$$T = \{(x_{11}, \dots, x_{n1}, c_1), \dots, (x_{1N}, \dots, x_{nN}, c_N)\}$$

and a scoring criterion ϕ, the *problem of learning a Bayesian network classifier* is to find a BNC $B \in \mathcal{B}_n$ that maximizes the value $\phi(B, T)$, where $\mathcal{B}_n$ is the set of all BNC with n features.

Score-based learning algorithms can be extremely efficient if the scoring criterion employed is decomposable, and in that case they are named **local score-based learning** algorithms. A scoring criterion ϕ is **decomposable** if the score assigned to each network decomposes over the network structure in such a way that it can be expressed as a sum of local scores ϕ_i that depends only on each node X_i and its parents, that is, scores of the following form:

$$\phi(B, T) = \sum_{i=1}^{n} \phi_i(\Pi_{X_i}, T).$$

Given a BNC B and a new instance $(y_1, \dots, y_n)$, we classify this novel instance as belonging to class z_c iff

$$P_B(C = z_c | X_1 = y_1, \dots, X_n = y_n) > P_B(C = z_{c'} | X_1 = y_1, \dots, X_n = y_n),$$

for all $c' \neq c$, where

$$P_B(C = z_c | X_1 = y_1, \dots, X_n = y_n) = \frac{P_B(X_1 = y_1, \dots, X_n = y_n, C = z_c)}{\displaystyle\sum_{c=1}^{s} P_B(X_1 = y_1, \dots, X_n = y_n, C = z_c)},$$

with joint probability distributions computed as in Equation (1).

3 An algorithm to learn Bayesian network classifiers

Learning unrestricted BNCs from data under typical scoring criteria is NP-hard. Indeed, exact polynomial-time bounded approaches for learning BNCs are only possible for a handful of network structures. In this project we will focus in tree-like network structures for which an exact polynomial time-bounded algorithm is known.

A **tree augmented naive Bayes classifier** (TAN) is a BNC where:

- there exists a root $R \in \{1, \ldots, n\}$ such that $\Pi_{X_R} = \emptyset$;
- for all $1 \leq i \leq n$, with $i \neq R$, there is a $j \neq i$ such that $\Pi_{X_i} = \{X_j\}$.

Learning BNCs is typically divided in *structure* and *parameter learning*. Structure learning is guided by a scoring criterion that performs a task of *model selection*.

3.1 Structure learning

In order to find the optimal TAN structure that maximizes a ϕ score for some data T, a complete weighted undirected graph is considered, where each edge between X_i and X_j is weighted with $\alpha_{ij}^T = \phi_j(\{X_i\}, T) - \phi_j(\emptyset, T)$.* Given this, the problem reduces to determining a **maximal weighted (undirected) spanning tree**.† After computing such spanning tree, a direction has to be assigned to each edge of the tree. This is done by choosing an arbitrary attribute as the tree root and then setting the direction of all edges to be outward from it. The TAN classifier is then built by adding a node labeled by C, and adding an arc from C to each tree node. The detailed algorithm follows:

1. Compute α_{ij}^T between each pair of attributes, with $1 \leq i < j \leq n$.
2. Build a complete undirected graph with attributes $X_1, \ldots, X_n$ as nodes. Annotate the weight of the edge connecting X_i to X_j by α_{ij}^T.
3. Build a maximal weighted (undirected) spanning tree.
4. Transform the resulting undirected tree to a directed one by choosing a root variable and setting the direction of all edges to be outward from it.
5. Construct a TAN classifier by adding a node labeled by C and adding an arc from C to each X_i, $i \leq n$.

3.2 Parameter learning

When the structure of the network is fixed in advance maximizing the likelihood of the data T reduces to estimating the parameters θ_{ijkc}. In this case, the *maximum likelihood* (ML) parameters that maximize LL are simply the **observed frequency estimates** (OFE) given by

$$\hat{\theta}_{ijkc} = \hat{P}_T(X_i = x_{ik} \mid \Pi_{X_i} = w_{ij}, C = z_c) = \frac{N_{ijkc} + N'}{N_{ijc}^K + r_i \times N'}$$

*For the scores we are going to consider in this project, $\alpha_{ij}^T = \alpha_{ji}^T = \phi_i(\{X_j\}, T) - \phi_i(\emptyset, T)$.

†Maximal weigthed (undirected) spanning trees can be computed with Prim's or Kruskal's algorithms:
http://en.wikipedia.org/wiki/Prim%27s_algorithm
http://en.wikipedia.org/wiki/Kruskal%27s_algorithm

and

$$\hat{\theta}_c = \hat{P}_T(C = z_c) = \frac{N_c + N'}{N + s \times N'},$$

where:

- N_{ijkc} is the number of instances in the data T where the variable X_i takes its k-th value x_{ik}, the variables in Π_{X_i} take their j-th configuration w_{ij}, and the class variable C takes its c-th value;
- N^K_{ijc} is the number of instances in the data T where the variables in Π_{X_i} take their j-th configuration w_{ij} and the class variable takes its c-th value;
- N_c is the number of instances where the class variable takes its c-th value;
- N is the total number of instances in data T; and
- N' are *pseudo-counts* that avoid the common mistake of assigning probability zero to an event that is extremely unlikely, but not impossible. In this project we always take N' to be 0.5.

3.3 Model selection

In this project we consider only two different scoring criteria, the *log-likelihood* (LL) score and the *minimum description length* (MDL) score.

In the LL score, the weight of an edge between X_i and $X_{i'}$ is given by the *conditional mutual information* between X_i and $X_{i'}$ conditioned on C, denoted by $I_{\hat{P}_T}(X_i; X_{i'} \mid C)$, and so

$$\begin{aligned} \alpha^T_{ii'} &= I_{\hat{P}_T}(X_i; X_{i'} \mid C) \\ &= \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \sum_{c=1}^{s} \frac{N_{ijkc}}{N} \log_2 \frac{N_{ijkc} N_c}{N^J_{ikc} N^K_{ijc}}, \end{aligned}$$

where:

- we assume without loss of generality that $\Pi_{X_i} = \{X_{i'}\}$ and $q_i = r_{i'}$, and
- N^J_{ikc} is the number of instances in the data T where the variable X_i takes its k-th value x_{ik} and the class variable takes its c-th value.

For the case of the MDL score, the weight of an edge between X_i and $X_{i'}$ is a penalized version of the previous LL weight, defined as

$$\begin{aligned} \alpha^T_{ii'} &= I_{\hat{P}_T}(X_i; X_{i'} \mid C) - \frac{s(r_i - 1)(r_{i'} - 1)}{2} \ln(N) \\ &= \left(\sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \sum_{c=1}^{s} \frac{N_{ijkc}}{N} \log_2 \frac{N_{ijkc} N_c}{N^J_{ikc} N^K_{ijc}} \right) - \frac{s(r_i - 1)(q_i - 1)}{2} \ln(N), \end{aligned}$$

where, once again, we assume without loss of generality that $\Pi_{X_i} = \{X_{i'}\}$ and $q_i = r_{i'}$.

4 An example

The example given in this section considers that the network structure of the BNC is fixed. From that fixed network structure the counts needed to compute the network parameters (namely, N_{ijkc}, N_{ijc}^K and N_c, for all i, j, k, c) as well as those needed to compute the network score (namely, N_{ijkc}, N_{ijc}^K,N_{ikc}^J, and N_c for all i, j, k, c) are computed for a given dataset T.

As an example consider $\mathbf{X} = (X_1, X_2, X_3, C)$ where X_1, X_3, and C, are binary random variables, and X_2 is a ternary random variable. In this case, the number of values each variable can take is given by:

- $r_1 = 2$, with $x_{11} = 0$ and $x_{12} = 1$;
- $r_2 = 3$, with $x_{21} = 0$, $x_{22} = 1$, $x_{23} = 2$;
- $r_3 = 2$, with $x_{31} = 0$ and $x_{32} = 1$;
- $s = 2$, with $z_1 = 0$ and $z_2 = 1$.

Consider the BNC B with the following network structure

$$B \equiv X_1 \to X_2 \to X_3,$$

where the class variable C has also an edge to X_i, for all $i = 1, 2, 3$. The number of parent configurations each variable can take is given by:

- $q_1 = 1$, with $w_{11} = \epsilon$, where ϵ is the empty configuration;
- $q_2 = 2$, with $w_{21} = x_{11} = 0$ and $w_{22} = x_{12} = 1$, as $\Pi_{X_2} = \{X_1\}$ and X_1 is binary;
- $q_3 = 3$, with $w_{31} = x_{21} = 0$, $w_{32} = x_{22} = 1$ and $w_{33} = x_{23} = 2$, as $\Pi_{X_3} = \{X_2\}$ and X_2 is ternary.

In addition, consider the dataset T to be given by

X_1	X_2	X_3	C
0	0	0	0
0	1	1	1
1	2	0	1
1	1	1	1
0	1	0	0
1	2	0	0
1	0	1	1

and, so, $n = 3$ and $N = 7$.

For the given dataset T and BNC B, we have that the N_{ijkc} counts, for all i, j, k, c, are:

	1st parent config.			2nd parent config.			3rd parent config.	
X_1	$N_{1111} = 2$	$N_{1121} = 1$						
	$N_{1112} = 1$	$N_{1122} = 3$						
X_2	$N_{2111} = 1$	$N_{2121} = 1$	$N_{2131} = 0$	$N_{2211} = 0$	$N_{2221} = 0$	$N_{2231} = 1$		
	$N_{2112} = 0$	$N_{2122} = 1$	$N_{2132} = 0$	$N_{2212} = 1$	$N_{2222} = 1$	$N_{2232} = 1$		
X_3	$N_{3111} = 1$	$N_{3121} = 0$		$N_{3211} = 1$	$N_{3221} = 0$		$N_{3311} = 1$	$N_{3321} = 0$
	$N_{3112} = 0$	$N_{3122} = 1$		$N_{3212} = 0$	$N_{3222} = 2$		$N_{3312} = 1$	$N_{3322} = 0$

In addition, the $N^K_{ijc} = \sum_{k=1}^{r_i} N_{ijkc}$ counts, for all i, j, c, are given by:

	1st parent config.	2nd parent config.	3rd parent config.
X_1	$N^K_{111} = 3$ $N^K_{112} = 4$		
X_2	$N^K_{211} = 2$ $N^K_{212} = 1$	$N^K_{221} = 1$ $N^K_{222} = 3$	
X_3	$N^K_{311} = 1$ $N^K_{312} = 1$	$N^K_{321} = 1$ $N^K_{322} = 2$	$N^K_{331} = 1$ $N^K_{332} = 1$

Moreover, the $N^J_{ikc} = \sum_{j=1}^{q_i} N_{ijkc}$ counts, for all i, k, c, are given by:

	1st value	2nd value	3rd value
X_1	$N^J_{111} = 2$ $N^J_{112} = 1$	$N^J_{121} = 1$ $N^J_{122} = 3$	
X_2	$N^J_{211} = 1$ $N^J_{212} = 1$	$N^J_{221} = 1$ $N^J_{222} = 2$	$N^J_{231} = 1$ $N^J_{232} = 1$
X_3	$N^J_{311} = 3$ $N^J_{312} = 1$	$N^J_{321} = 0$ $N^J_{322} = 3$	

and the N_c counts, for all c, are given by:

$$N_1 = 3 \text{ and } N_2 = 4.$$

The parameters stored in the BNC C at node X_1 are those given by:

params for $z_1 = 0$	$x_{11} = 0$	$x_{12} = 1$
$w_{11} = \epsilon$	$\hat{\theta}_{1111} = \frac{N_{1111}+N'}{N^K_{111}+r_1 \times N'}$ $= \frac{2+0.5}{3+2\times 0.5} = \frac{2.5}{4}$	$\hat{\theta}_{1121} = \frac{N_{1121}+N'}{N^K_{111}+r_1 \times N'}$ $= \frac{1+0.5}{3+2\times 0.5} = \frac{1.5}{4}$

params for $z_2 = 1$	$x_{11} = 0$	$x_{12} = 1$
$w_{11} = \epsilon$	$\hat{\theta}_{1112} = \frac{N_{1112}+N'}{N^K_{112}+r_1 \times N'}$ $= \frac{1+0.5}{4+2\times 0.5} = \frac{1.5}{5}$	$\hat{\theta}_{1122} = \frac{N_{1122}+N'}{N^K_{112}+r_1 \times N'}$ $= \frac{3+0.5}{4+2\times 0.5} = \frac{3.5}{5}$

In addition, the parameters stored in the BNC C at node X_2 are those given by:

params for $z_1 = 0$	$x_{21} = 0$	$x_{22} = 1$	$x_{23} = 2$
$w_{21} = 0$	$\hat{\theta}_{2111} = \frac{N_{2111}+N'}{N^K_{211}+r_2 \times N'}$ $= \frac{1+0.5}{2+3\times 0.5} = \frac{1.5}{3.5}$	$\hat{\theta}_{2121} = \frac{N_{2121}+N'}{N^K_{211}+r_2 \times N'}$ $= \frac{1+0.5}{2+3\times 0.5} = \frac{1.5}{3.5}$	$\hat{\theta}_{2131} = \frac{N_{2131}+N'}{N^K_{211}+r_2 \times N'}$ $= \frac{0+0.5}{2+3\times 0.5} = \frac{0.5}{3.5}$
$w_{22} = 1$	$\hat{\theta}_{2211} = \frac{N_{2211}+N'}{N^K_{221}+r_2 \times N'}$ $= \frac{0+0.5}{1+3\times 0.5} = \frac{0.5}{2.5}$	$\hat{\theta}_{2221} = \frac{N_{2221}+N'}{N^K_{221}+r_2 \times N'}$ $= \frac{0+0.5}{1+3\times 0.5} = \frac{0.5}{2.5}$	$\hat{\theta}_{2231} = \frac{N_{2231}+N'}{N^K_{221}+r_2 \times N'}$ $= \frac{1+0.5}{1+3\times 0.5} = \frac{1.5}{2.5}$

params for $z_2 = 1$	$x_{21} = 0$	$x_{22} = 1$	$x_{23} = 2$
$w_{21} = 0$	$\hat{\theta}_{2112} = \frac{N_{2112}+N'}{N^K_{212}+r_2\times N'} = \frac{0+0.5}{1+3\times 0.5} = \frac{1.5}{2.5}$	$\hat{\theta}_{2122} = \frac{N_{2122}+N'}{N^K_{212}+r_2\times N'} = \frac{1+0.5}{1+3\times 0.5} = \frac{1.5}{2.5}$	$\hat{\theta}_{2132} = \frac{N_{2132}+N'}{N^K_{212}+r_2\times N'} = \frac{0+0.5}{1+3\times 0.5} = \frac{0.5}{2.5}$
$w_{22} = 1$	$\hat{\theta}_{2212} = \frac{N_{2212}+N'}{N^K_{222}+r_2\times N'} = \frac{1+0.5}{3+3\times 0.5} = \frac{1.5}{4.5}$	$\hat{\theta}_{2221} = \frac{N_{2222}+N'}{N^K_{222}+r_2\times N'} = \frac{1+0.5}{3+3\times 0.5} = \frac{1.5}{4.5}$	$\hat{\theta}_{2231} = \frac{N_{2232}+N'}{N^K_{222}+r_2\times N'} = \frac{1+0.5}{3+3\times 0.5} = \frac{1.5}{4.5}$

Moreover, the parameters stored in the BNC C at node X_3 are those given by:

params for $z_1 = 0$	$x_{31} = 0$	$x_{32} = 1$
$w_{31} = 0$	$\hat{\theta}_{3111} = \frac{N_{3111}+N'}{N^K_{311}+r_3\times N'} = \frac{1+0.5}{1+2\times 0.5} = \frac{1.5}{2}$	$\hat{\theta}_{3121} = \frac{N_{3121}+N'}{N^K_{311}+r_3\times N'} = \frac{0+0.5}{1+2\times 0.5} = \frac{0.5}{2}$
$w_{32} = 1$	$\hat{\theta}_{3211} = \frac{N_{3211}+N'}{N^K_{321}+r_3\times N'} = \frac{1+0.5}{1+2\times 0.5} = \frac{1.5}{2}$	$\hat{\theta}_{3221} = \frac{N_{3221}+N'}{N^K_{321}+r_3\times N'} = \frac{0+0.5}{1+2\times 0.5} = \frac{0.5}{2}$
$w_{33} = 2$	$\hat{\theta}_{3311} = \frac{N_{3311}+N'}{N^K_{331}+r_3\times N'} = \frac{1+0.5}{1+2\times 0.5} = \frac{1.5}{2}$	$\hat{\theta}_{3321} = \frac{N_{3321}+N'}{N^K_{331}+r_3\times N'} = \frac{0+0.5}{1+2\times 0.5} = \frac{0.5}{2}$

params for $z_2 = 1$	$x_{31} = 0$	$x_{32} = 1$
$w_{31} = 0$	$\hat{\theta}_{3112} = \frac{N_{3112}+N'}{N^K_{312}+r_3\times N'} = \frac{0+0.5}{1+2\times 0.5} = \frac{0.5}{2}$	$\hat{\theta}_{3122} = \frac{N_{3122}+N'}{N^K_{312}+r_3\times N'} = \frac{1+0.5}{1+2\times 0.5} = \frac{1.5}{2}$
$w_{32} = 1$	$\hat{\theta}_{3212} = \frac{N_{3212}+N'}{N^K_{322}+r_3\times N'} = \frac{0+0.5}{2+2\times 0.5} = \frac{0.5}{3}$	$\hat{\theta}_{3222} = \frac{N_{3222}+N'}{N^K_{322}+r_3\times N'} = \frac{2+0.5}{2+2\times 0.5} = \frac{2.5}{3}$
$w_{33} = 1$	$\hat{\theta}_{3312} = \frac{N_{3312}+N'}{N^K_{332}+r_3\times N'} = \frac{1+0.5}{1+2\times 0.5} = \frac{1.5}{2}$	$\hat{\theta}_{3322} = \frac{N_{3322}+N'}{N^K_{332}+r_3\times N'} = \frac{0+0.5}{1+2\times 0.5} = \frac{0.5}{2}$

Finally, the parameters $\hat{\theta}_c$, for all c, stored in the BNC B at node C, are given by:

$$\hat{\theta}_1 = \frac{N_1 + N'}{N + s \times N'} = \frac{3 + 0.5}{7 + 2 \times 0.5} = \frac{3.5}{8} \text{ and } \hat{\theta}_2 = \frac{N_2 + N'}{N + s \times N'} = \frac{4 + 0.5}{7 + 2 \times 0.5} = \frac{4.5}{8}.$$

5 Parameters and results

The program should receive the following parameters:

train	filename of a dataset from which a TAN classifier is going to be learned
test	filename of a test set from which the learned classifier is going to be tested
score	the score to be used to build the TAN classifier

and it should return the classification for the instances in the *test* set according the TAN classifier built from the *train* dataset. The score used to build the TAN classifier is determined in the parameter *score* and it could be one of the following strings: LL or MDL (case sensitive).

5.1 Input files format

The *train* dataset must be provided as a .csv file[‡] formatted as:

[‡]Comma separated value (.csv) files can be opened in excel or in any text editor.

$$\begin{array}{llll} X_1, & \dots, & X_n, & C \\ x_{11}, & \dots, & x_{n1}, & c_1 \\ \dots, & \dots, & \dots, & \dots \\ x_{1N}, & \dots, & x_{nN}, & c_N \end{array}$$

where the first line corresponds to the name of the random variables $(X_1, \dots, X_n, C)$ and the following lines to their values in each instance of the training data. For the sake of simplicity, we assume that variable X_i, with $1 \leq i \leq n$, ranges from 0 to $r_i - 1$ and the class variable C ranges from 0 to $s - 1$. Recall notation introduced in previous sections to understand the size of the data.

The *test* set is also a .csv file that provides new instances that we want to classify, based on the classifier learned from the *train* dataset as explained in the previous sections, and complies with the following format:

$$\begin{array}{lll} X_1, & \dots, & X_n \\ y_{11}, & \dots, & y_{n1} \\ \dots, & \dots, & \dots \\ y_{1t}, & \dots, & x_{nt} \end{array}$$

where t is the number of instances in the test set. In the *test* set we also assume that variable X_i, with $1 \leq i \leq n$, ranges from 0 to $r_i - 1$.

A few examples of *train* and *test* sets will be provided in the "Project section" of the OOP website. Stay tuned!

5.2 Running in the command line

A .jar file must be created so that the program runs by typing in the terminal

`java -jar <<YOUR-JAR-NAME>>.jar` *train test score*

where *train* and *test* are the names of the input files, and *score* is the string that identifies the score to be used to build the TAN model. These input files should be found outside of the `.jar` file. An absolute or a relative path may be used.

5.3 Results

When running the program in the command line the following output must be printed to the terminal:

`Building the classifier:`	$TIME$ `time`
`Testing the classifier:`	
`-> instance 1:`	*class_of_instance_1*
`...`	...
`-> instance` t`:`	*class_of_instance_t*

where:

- $TIME$ is the time spent to build the TAN classifier; and
- *class_of_instance_l*, with $1 \leq l \leq t$, is the result of classifying the l-th instance of the *test* set, according to the learned classifier.

6 Bonus point

If a Swing GUI is provided, a bonus point may be added to the final mark of the project. Therefore, as the project assesses 6 points out of the final grade, with a **fully-functional and correctly implemented** GUI, the project may evaluate to 7 points (instead of 6).

Notwithstanding, even if a Swing GUI is provided, the program must also run from the command line as described in Section 5.3.

7 Deadlines and material for submission

The **deadline for submitting the project is May 19th, before 18:00**. The submission is done via fenix, so ensure that you are registered in a project group.

The following files must be submitted:

1. An UML specification including classes and packages (as detailed as possible), in `.pdf` or `.jpg` format. Place the UML files inside a folder named UML.
2. An executable `.jar` (with the respective source files `.java`, compiled classes `.class`, and `MANIFEST.MF` correctly organized into directories).
3. Documentation (generated by the javadoc tool) of the application. Place the documentation inside a folder named JDOC.
4. A final report (up to 10 pages, in .pdf format) containing information that complements the documentation generated by the javadoc tool. Place the final report inside a folder named DOCS.
5. A self assessment form (in `.pdf` format) that will be made available in due time in the course webpage. Place the self assessment form inside the folder named DOCS (the same folder as the final report).

The UML folder, executable (the `.jar` file with the source files, besides the compiled files and `MANIFEST.MF`), the JDOC folder and the DOCS folder, should be **submitted via fenix in a single `.zip` file**.

The final discussion will be held from May 20 to May 30. The distribution of the groups for final discussion will be available in due time. All group members must be present during the discussion. The final grade of the project will depend on this discussion, and it will not necessarily be the same for all group members.

8 Grading

The assessment will be based on the following 6-point scale:

1. **(1 point)**: UML. The UML will be evaluated, in a 1-point scale as: 0–very bad, 0.25–bad, 0.5–average, 0.75–good and 1–excellent.
2. **(5 points)**: A solution that provides an extensible and reusable framework for learning TAN classifiers. The implementation of the requested features in Java are also an important evaluation criteria and the following discounts, on a 5-point scale, are pre-established:

(a) **(-2 points):** OOP ingredients are not used or they are used incorrectly; this includes polymorphism, open-close principle, etc.

(b) **(-1 points):** Java features are handled incorrectly; this includes incorrect manipulation of methods from `Object`, `Collection`, etc.

(c) **(-0.5 points):** Prints outside the format requested in Section 5.3.

(d) **(-0.5 points):** A non-executable jar file, or a jar file without sources or with sources out of date.

(e) **(-0.5 points):** During the discussion, problems with Java versions; the Java executable should run properly in the laboratory PCs.

(f) **(-0.5 points):** During the discussion, problems in extracting/building a jar file, as well as compiling/running the executable in Java, both from the command line.

Finally, in a 6-point scale:

1. Files submitted outside of the required format will have a penalty of 5% over the respective grade.

2. Projects submitted after the established date will have the following penalty: for each day of delay there will be a penalty of 2^n points of the grade, where n is the number of days in delay. That is, reports submitted up to 1 day late will be penalized in $2^1 = 2$ points, incurring in a penalty of 0.6 points of the final grade; reports submitted up to 2 days late will be penalized in $2^2 = 4$ points, incurring in a penalty of 1.2 points of the final grade; and so on. Per day of delay we mean cycles of 24h from the day specified for submission.