The *Independency tree* model: a new approach for clustering and factorisation

M. Julia Flores and José A. Gámez
Computing Systems Department / SIMD ($i^3\mathcal{A}$)
University of Castilla-La Mancha
Albacete, 02071, Spain

Serafín Moral
Departamento de Ciencias de la Computación e I. A.
Universidad de Granada
Granada, 18071, Spain

Abstract

Taking as an inspiration the so-called *Explanation Tree* for abductive inference in Bayesian networks, we have developed a new clustering approach. It is based on exploiting the variable independencies with the aim of building a tree structure such that in each leaf all the variables are independent. In this work we produce a structure called *Independency tree*. This structure can be seen as an extended probability tree, introducing a new and very important element: a list of probabilistic single potentials associated to every node. In the paper we will show that the model can be used to approximate a joint probability distribution and, at the same time, as a hierarchical clustering procedure. The Independency tree can be learned from data and it allows a fast computation of conditional probabilities.

1 Introduction

In the last years the relevance of unsupervised classification within data mining processing has been remarkable. When dealing with large number of cases in real applications, the identification of common features that allows the formation of groups/clusters of cases seems to be a powerful capability that both simplifies the data processing and also allows the user to understand better the trend(s) followed in the registered cases.

In the data mining community this *descriptive* task is known as *cluster analysis* (Anderberg, 1973; Duda et al., 2001; Kaufman and Rousseeuw, 1990; Jain et al., 1999), that is, getting a decomposition or partition of a data set into groups in such a way that the objects in one group are similar to each other but as different as possible from the objects in other groups. In fact, as pointed out in (Hand et al., 2001, pg. 293), we could distinguish two different objectives in this descriptive task: (a) *segmentation*, in which the aim is simply to partition the data in a *convenient* way, probably using only a small number of the available variables; and (b) *decomposition*, in which the aim is to see whether the data is composed (or not) of natural subclasses, i.e., to discover whether the overall population is heterogeneous. Strictly speaking cluster analysis is devoted to the second goal although, in general, the term is widely used to describe both segmentation and cluster analysis problems.

In this work we only deal with *categorical* or *discrete* variables and we are closer to segmentation than (strictly speaking) to cluster analysis. Our proposal is a method that in our opinion has several good properties: (1) it produces a tree-like graphical structure that allows us to visually describe each cluster by means of a configuration of the relevant variables for

that segment of the population; (2) it takes advantage of the identification of contextual independencies in order to build a decomposition of the joint probability distribution; (3) it stores a joint probability distribution about all the variables, in a simple way, allowing an efficient computation of conditional probabilities. An example of an independence tree is given in figure 1, which will be thoroughly described in the following sections.

The paper is structured as follows: first, in Section 2 we give some preliminaries in relation with our proposed method. Section 3 describes the kind of model we aim to look for, while in Section 4 we propose the algorithm we have designed to discover it from data. Section 5 describes the experiments carried out. Finally, Section 6 is devoted to the conclusions and to describe some possible lines in order to continue our research.

2 Preliminaries

Different types of clustering algorithms can be found in the literature differing in the type of approach they follow. Probably the three main approaches are: partition-based clustering, hierarchical clustering, and probabilistic model-based clustering. From them, the first two approaches yield a *hard* clustering in the sense that clusters are exclusive, while the third one yields a *soft* clustering, that is, an object can belong to more than one cluster following a probability distribution. Because our approach is somewhat related to hierarchical and probabilistic model-based clustering we briefly comment on these types of clustering.

Hierarchical clustering (Sneath and Sokal, 1973) returns a tree-like structure called dendrogram. This structure reflects the way in which the objects in the data set have been merged from single points to the whole set or split from the whole set to single points. Thus, there are two distinct types of hierarchical methods: *agglomerative* (by merging two clusters) and *divisive* (by splitting a cluster). Although our method does not exactly belong to hierarchical clustering, it is closer to hierarchical divisive methods than to any other clustering approach (known by us). In concrete, it works as *monothetic* divisive clustering algorithms (Kaufman and Rousseeuw, 1990), that split clusters using one variable at a time, but differs from classical divisive approaches in the use of a Bayesian score to decide which variable is chosen at each step, and because branches are not completely developed.

With respect to probabilistic clustering, although our method produces a hard clustering, it is somehow related to it because probability distributions are used to complete the information about the discovered model. Probabilistic model-based clustering is usually modelled as a mixture of models (see e.g. (Duda et al., 2001)). Thus, a hidden random variable is added to the original observed variables and its states correspond with the components of the mixture (the number of clusters). In this way we move to a problem of learning from unlabelled data and usually EM algorithm (Dempster et al., 1977) is used to carry out the learning task when the graphical structure is fixed and *structural* EM (Friedman, 1998) when the graphical structure has also to be discovered (Peña et al., 2000). Iterative approaches have been described in the literature (Cheeseman and Stutz, 1996) in order to discover also the number of clusters (components of the mixture). Notice that although this is not the goal of the learning task, the structures discovered can be used for approximate inference (Lowd and Domingos, 2005), having the advantage over general Bayesian networks (Jensen, 2001) that the learned graphical structure is, in general, simple (i.e. naive Bayes) and so inference is extremely fast.

3 *Independency tree* model

If we have the set of variables $\mathbf{X} = \{X_1, \ldots, X_m\}$ regarding a certain domain, an independent probability tree for this set of variables is a tree such that (e.g. figure 1):

- Each inner node N is labelled by a variable $\mathsf{Var}(N)$, and this node has a child for each one of the possible values (states) of $\mathsf{Var}(N)$.

When a variable is represented by a node in the tree it implies that this variable partitions the space from now on (from this point to the farther branches in the tree) depending on its (nominal) value. So, a variable cannot appear again (deeper) in the tree.

- Each node N will have an associated list of potentials, $\mathsf{List}(N)$, with each of the potentials in the list storing a marginal probability distribution for one of the variables in $\mathbf{X}$, including always a potential for the variable $\mathsf{Var}(N)$.

  This list appears in Figure 1 framed into a dashed box.

- For any path from the root to a leaf, each one of the variables in $\mathbf{X}$ appears uniquely and exactly once in the the lists associated to the nodes along the path. This potential will determine the conditional probability of the variables given the values of the variables on the path from the root to the list containing the potential.

A configuration is a subset of variables $\mathbf{Y} \subseteq \{X_1, \ldots, X_m\}$ together with a concrete value $Y_j = y_j$ for each one of the variables $Y_j \in \mathbf{Y}$. Each node N has an associated configuration determined for the variables in the path from the root to node N (excluding $\mathsf{Var}(N)$) with the values corresponding to the children we have to follow to reach N. This configuration will be denoted as $\mathsf{Conf}(N)$.

An independent probability tree represents a joint probability distribution, p, about the variables in $\mathbf{X}$. If $\mathbf{x} = (x_1, \ldots, x_m)$, then

$$p(\mathbf{x}) = \prod_{i=1}^{m} \mathcal{P}_{\mathbf{x}(i)}(x_i)$$

where $\mathcal{P}_{x(i)}$ is the potential for variable X_i which is in the path from the root to a leaf determined by configuration $\mathbf{x}$ (following in each inner node N with variable $\mathsf{Var}(N) = X_j$, the child corresponding to the value of $X_j \in \mathbf{x}$).

This decomposition is based on a set of independencies among variables $\{X_1, \ldots, X_m\}$. Assume that $\mathsf{VL}(N)$ is the set of variables of the potentials in $\mathsf{List}(N)$, and that $\mathsf{DVL}(N)$[1] is the union of sets $\mathsf{VL}(N')$, where N' is a node in a path from N to a leaf, then the independencies are generated from the following statement: *Each variable X in $\mathsf{VL}(N)$ - $\mathsf{Var}(N)$ is independent of the variables $(\mathsf{VL}(N) - \{X\}) \cup \mathsf{DVL}(N)$ given the configuration $\mathsf{Conf}(N)$*; i.e. each variable in the list of a node is independent of the other variables in that list and of the variables in its descendants, given the configuration associated to the node.

An independent probability tree also defines a partition of the set of possible values of variables in $\mathbf{X}$. The number of clusters is the number of leaves. If N is a leaf with associated configuration $\mathsf{Conf}(N)$, then this group is given by all set of values $\mathbf{x}$ that are compatible with $\mathsf{Conf}(N)$ (they have the same value for all the variables in $\mathsf{Conf}(N)$). For example, in figure 1 the configuration $\mathbf{X} = \{0, 1, 0, 1, 0\}$ would fall on the second leaf since X_1=0 and X_2=1. It is assumed that the probability distribution of the other variables in the cluster are given by the potentials in the path defined by the configuration, for example, P(X_3=0)=1.

In this way, with an independence tree we are able of accomplishing two goals in one: (1)The variables are partitioned in a hierarchical way that gives us at the same time a clustering result; (2)The probability value of every configuration of the variables.

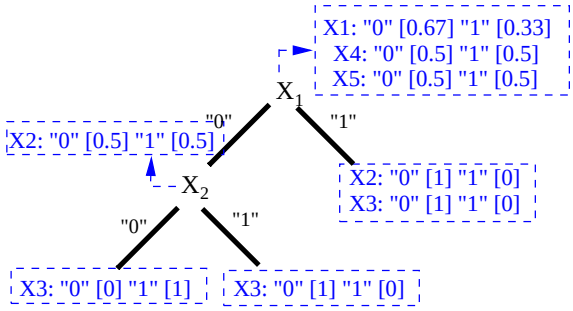

Figure 1: Illustrative independence tree structure learned from the exclusive dataset.

[1] D stands for "Descendants".

Our structure seeks to keep in leaves those variables that remain independent. At the same time, if the distribution of one variable is shared by several leaves, we try to store it in their common ascendant, to avoid repeating it in all the leaves. For that reason, when one variable appears in a list for a node N_j it means that this distribution is common for all levels from here to a leaf. For example, in figure 1, the binary variable X_4 has a uniform (1/2,1/2) distribution for all leaf cases (also for intermediate ones), since it is associated to the root node. On the other hand, we can see how X_2 distribution varies depending on the branch (left or right) we take from the root, that is, when $X_1 = 0$ the behaviour of variable X_2 is uniform whereas being 1 the value of X_1, X_2 is determined to be 0.

The intuition underlying this model is based on the idea that inside each cluster the variables are independent. When we have a set of data, groups are defined by common values in certain variables, having the other variables random variations. Imagine that we have a database with characteristics of different animals including mammals and birds. The presence of these two groups is based on the existence of dependencies between the variables (two legs is related with having wings and feathers). Once these variables are fixed, there can be another variables (size, colour) that can have random variations inside each group, but that they do not define new subcategories.

Of course, there are some other possible alternatives to define clustering. This is based on the idea that when all the variables are independent, then to subdivide the population is not useful, as we have a simple method to describe the joint behaviour of the variables. However, if some of the variables are dependent, then the values of some basic variables could help to determine the values of the other variables, and then it can be useful to divide the population in groups according to the values of these basic variables.

Another way of seeing it is that having independent variables is the simplest model we can have. So, we determine a set of categories such that each one is described in a very simple way (independent variables). This is exploited by another clustering algorithms as EM-based AutoClass clustering algorithm (Cheeseman and Stutz, 1996).

An important fact of this model is that it is a generalisation of some usual models in classification, the Naive Bayes model (a tree with only one inner node associated to the class and in its children all the rest of variables are independent), and the classification tree (a tree in which the list of each inner node only contains one potential and the potential associated to the class variable always appears in the leaves). This generalization means that our model is able of representing the same conditional probability distribution of the class variable with respect to the rest of the variables, taking as basis the same set of associated independencies.

From this model one may want to apply two kinds of operations:

• The production of clusters and their characterisation. A simple in-depth from root until every leaf node access of the tree will give us the corresponding clusters, and also the potential lists associated to the path nodes will indicate the behaviour of the other variables not in the path.

• The computation of the probability for a certain complete configuration with a value for each variable: $\mathbf{x} = \{x_1, x_2, \ldots, x_m\}$. This algorithm is recursive and works as follows:

getProb(Configuration $\{x_1, x_2, \ldots, x_m\}$,Node N)
$Prob \leftarrow 1$
1 For all Potential $\mathcal{P}_j \in$ List(N)
1.1. $X_j \leftarrow$ Var($\mathcal{P}_j$)
1.2. $Prob \leftarrow Prob \cdot \mathcal{P}_j(X_j = x_j)$
2 If N is a leaf then return $Prob$
3 Else
3.1. $X_N \leftarrow$ Var(N)
3.2. Next Node $N' \leftarrow$ Branch_child($X_N = x_N$)
3.3. return $Prob$·getProb($\{x_1, x_2, \ldots, x_m\}$,$N'$)

If we have a certain configuration we should go through the tree from root until leaves taking the corresponding branches. Every time we reach a node, we have to use the single potentials in the associated list to multiply the value of probability by the values of the potentials corresponding to this configuration.

It is also possible to compute the probability for an interest variable Z conditioned to

a generic configuration (a set of observations): $\mathbf{Y} = \mathbf{y}$. This can be done in two steps: 1) Transform the independent probability tree into a probability tree (Salmerón et al., 2000); 2) Make a marginalisation of the probability tree by adding in all the variables except in Z as describe in (Salmerón et al., 2000).

In the following we describe the first step. It is a recursive procedure that visits all nodes from root to leaves. Each node can pass to its children a float, $Prob$ (1 in the root) and a potential $\mathcal{P}$ which depends of variable Z (empty in the root). Each time a node is visited, the following operations are carried out:

• All the potentials in $\mathsf{List}(N)$ are examined and removed. For any potential, depending on its variable X_j we proceed as follows:

- If $X_j \neq Z$, then if $X_j = Y_k$ is in the observations configuration, $Prob$ is multiplied by the value of this potential for $Y_k = y_k$.

- If $X_j \neq Z$ and X_j does not appear in the observations configuration, then the potential is ignored.

- If $X_j = Z$ and X_j is the one associated to node N, then we transform each one of the children of Z, but multiplying $Prob$ by the value of potential in $Z = z$, before transforming the child corresponding to this value.

- If $X_j = Z$ and X_j is not the one associated to node N, then the potential of X_j is stored in $\mathcal{P}$.

• After examining the list of potentials, we proceed:

- If N is not a leaf node, then we transform its children.

- If N is a leaf node and $\mathcal{P} = \emptyset$, we assign the value $Prob$ to this node.

- If N is a leaf node and $\mathcal{P} \neq \emptyset$, we make N an inner node with variable Z: For each value $Z = z$, build a node N'_z which is a leaf node with a value equal to $Prob \odot \mathcal{P}(z)$. Make N'_z a child of N.

This procedure is fast: it is linear in the the size of the independent probability tree (considering the number of values of Z constant). The marginalisation in the second step has the same time complexity. In fact, this marginalization can be done by adding for each value $Z = z$ the values of the leaves that are compatible with this value (compatibility means that to follow this path we do not have to assume $Z = z'$ with $z \neq z'$), which is linear too.

4 Our clustering algorithm

In this section we are going to describe how an independent probability tree can be learned from a database D, with values for all the variables in $\mathbf{X} = \{X_1, \ldots, X_m\}$.

The basics of the algorithm are simple. It tries to determine for each node, the variable with a strongest degree of dependence with the rest of remaining variables. This variable will be assigned to this node, repeating the process with its children until all the variables are independent.

For this, we need a measure of the degree of dependence of two variables X_i and X_j in database D. The measure should be centered around 0, in such a way that the variables are considered dependent if and only if the measure is greater than 0. In this paper, we consider the K2 score (Cooper and Herskovits, 1992), measuring the degree of dependence as the difference in the logarithm of the K2 score of X_j conditioned to X_i minus the logarithm of K2 score of marginal X_j, i.e. the difference between the logarithms of the K2 scores of two networks with two variables: one in which X_i is a parent of X_j and other in which the two variables are not connected. Let us call this degree of dependence $\mathsf{Dep}(X_i, X_j|D)$. This is a non-symmetrical measure and should be read as the influence of X_i on X_j, however in practice the differences between $\mathsf{Dep}(X_i, X_j|D)$ and $\mathsf{Dep}(X_j, X_i|D)$ are not important.

In any moment, given a variable X_i and a database D, we can estimate a potential $\mathcal{P}_i(D)$ for this variable in the database. This potential is the estimation of the marginal probability of X_i. Here we assume that this is done by counting the absolute frequencies of each one of the values in the database.

The algorithm starts with a list of variables L which is initially equal to $\{X_1, \ldots, X_m\}$ and a database equal to the original one D, then it

determines the root node N and its children in a recursive way. For that, for any variable X_i in the list, it computes

$$\mathsf{Dep}(X_i|D) = \sum_{X_j \in L} \mathsf{Dep}(X_i, X_j|D)$$

Then, the variable X_k with maximum value of $\mathsf{Dep}(X_i|D)$ is considered.

If $\mathsf{Dep}(X_k|D) > 0$, then we assign variable X_k to node N and we add potential $\mathcal{P}_k(D)$ to the list $\mathsf{List}(N)$, removing X_k for L. For all the remaining variables X_i in L, we compute $\mathsf{Dep}(X_i, X_j|D)$ for $X_j \in L(j \neq i)$, and $X_j = X_k$. If all these values are less or equal than 0, then we add potential $\mathcal{P}_i(D)$ to $\mathsf{List}(N)$ and remove X_i from L; i.e. we keep in this node the variables which are independent of the rest of variables, including the variable in the node. Finally, we build a child of N for each one of the values $X_k = x_k$. This is done by calling recursively to the same procedure, but with the new list of nodes, and changing database D to $D[X_k = x_k]$, where $D[X_k = x_k]$ is the subset of D given by those cases in which variable X_k takes the value x_k.

If $\mathsf{Dep}(X_k|D) \leq 0$, then the process is stopped and this is a leaf node. We build $\mathsf{List}(N)$, by adding potential $\mathcal{P}_i(D)$ for any variable X_i in L.

In this algorithm, the complexity of each node computation is limited by $O(m^2.n)$ where m is the number of variables and n is the database size. The number of leaves is limited by the database size n multiplied by the maximum number of cases of a variable, as a leaf with only one case of the database is never branched. But usually the number of nodes is much lower.

In the algorithm, we make some approximations with respect to the independencies represented by the model. First, we only look at one-to-one dependencies, and not to joint dependencies. It can be the case that X_i is independent of X_j and X_k and it is not independent of (X_j, X_k). However, testing these independents is more costly and we do not have the possibility of a direct representation of this in the model. This assumption is made by other Bayesian networks learning algorithms such as PC (Spirtes et al., 1993), where a link between two nodes is deleted if these nodes are marginally independent.

Another approximation is that it is assumed that all the variables are independent in a leaf when $\mathsf{Dep}(X_k|D) \leq 0$, even if some of the terms we are adding are positive. We have found that this is a good compromise criterion to limit the complexity of learned models.

5 Experiments

To make an initial evaluation of the *Independence Tree* (IndepT) model, we decided to compare it with other well known unsupervised classification techniques that are also based on Probabilistic Graphical Models: learning of a Bayesian network (by a standard algorithm as PC) and also Expectation-Maximisation with a Naive Bayes structure, which uses cross-validation to decide the number of clusters. Because we produce a probabilistic description of the dataset, we use the log-likelihood (logL) of the data given the model to score a given clustering. By using the logL as goodness measure we can compare our approach with other algorithms for probabilistic model-based unsupervised learning: probabilistic model-based clustering and Bayesian networks. This is a direct evaluation of the procedures as methods to encode a complex joint probability distribution. At the same time, it is also an evaluation of our method from the clustering point of view, showing whether the proposed segmentation is useful for a simple description of the population.

Then, the basic steps we have followed for the three procedures [IndepT,PC-Learn,EM-Naive] are:

1. Divide the data cases into a training or data set (S_D) and a test set (S_T), using (2/3,1/3).
2. Build the corresponding model for the cases in S_D.
3. Compute the log-likelyhood of the obtained model over the data in S_T.

Among the tested cases, apart from easy synthetical data bases created to check the expected and right clusters, we have looked for real sets of cases. Some of them have been taken

from the UCI datasets repository (Newman et al., 1998) and others from real applications related to our current research environment.

We will indicate the main remarks about all the evaluated cases:

• Case 1: *exclusive*: This is a very simple dataset with five binary variables, from X_1 to X_5. The three first variables have an *exclusive* behaviour, that is, if one of them is 1 the other two will be 0. On the other hand, X_4 and X_5 are independent with the three first ones and also with each other. This example is interesting not especially for the LogL comparison, but mainly to see the interpretation of the tree given in figure 1.

• Case 2: *tic-tac-toe*: Taken from UCI repository and it encodes the complete set of possible board configurations at the end of tic-tac-toe games. It presents 958 cases and 9 attributes (one per each game square).

• Cases 3: *greenhouses*: Data cases taken from real greenhouses located in Almería (Spain). There are four distinct sets of cases, here we indicate them with denotation (case)={num_cases, num_attributes}: ($3A$)= {1240,8}, ($3B$)= {1465,17}, ($3C$)= {1318,33}, ($3D$)= {1465,6}.

• Cases 4: *sheep*: Datasets taken from the work developed in (Flores and Gámez, 2005) where the historical data of the different animals were registered and used to analyse their genetic merit for milk production directed to Manchego cheese. There are two datasets with 3087 cases, $4A$ with 24 variables/attributes and $4B$, where the attribute `breeding value` (that could be interpreted as class attribute) has been removed.

• Case 5: *connect4*: Also downloaded from the UCI repository and it contains all legal 8-play positions in the game of connect-4 in which neither player has won yet, and in which the next move is not forced. There are 67557 cases and 42 attributes, each corresponding to one connect-4 square[2].

The results of the experiments can be found in figure 1 and in table 1. Figure 1 presents the tree learned in the *exclusive* case. As it can be observed, the tree really captures what is happening in the problem: X_4 and X_5 are independent and placed in the list of root node and then one of the other variables is looked up, if its value is 1, then the values of the other two variables is completely determined and we stop. If its value is 0, then another variable has to be examined.

case	IndepT	PC-Learn	EM-Naive
1	-61.33	-62.88	-119.67
2	-1073.94	-2947.85	-3535.11
$3A$	-3016.92	-2644.38	-4297.21
$3B$	-2100.40	-3584.06	-6266.71
$3C$	-4956.04	-7630.98	-15145.15
$3D$	-1340.04	-2770.15	-4223.45
$4A$	-6853.12	-18074.69	-32213.29
$4B$	-6802.94	-17357.80	-31244.28
5:	-465790.73	-349631.35	-794415.07

Table 1: Comparison in terms of the log-likelyhood value for all cases (datasets).

With respect to the capacity of approximating the joint probability distribution, we can see in table 1 that our method always provides greater values of logL than EM-Naive in all the datasets. With respect to PC algorithm, the independent tree wins in all the situations except in two of them (cases $3A$ and 5). This is a remarkable result as our model has some limitations in representing sets of independencies that can be easily represented by a Bayesian network (for example a Markov chain), however its behaviour is usually better. We think that this is due to two main aspects: it can represent asymmetrical independencies and it is a simple model (which is always a virtue).

6 Concluding remarks and future work

In this paper we have proposed a new model for representing a joint probability distribution in a compact way, which can be used for fast computation of conditional probability distributions. This model is based on a partition of the state space, in such a way that in each group all the

[2]Actually, only the half of the cases have been used.

variables are independent. In this sense, it is at the same time a clustering algorithm.

In the experiments with real and synthetical data we have shown its good behaviour as a method of approximating a joint probability, providing results that are better (except for two cases for the PC algorithm) to the factorisation provided by an standard Bayesian networks learning algorithm (PC) and by EM clustering algorithm. In any case, more extensive experiments are necessary in order to compare it with another clustering and factorisation procedures.

We think that this model can be improved and exploited in several ways: (1) Some of the clusters can have very few cases or can be very similar in the distributions of the variables. We could devise a procedure to joint these clusters; (2) we feel that the different number of values of the variables can have some undesirable effects. This problem could be solved by considering binary trees in which the branching is determined by a partition of the set of possible values of a variable in two parts. This could allow to extend the model to continuous variables; (3) we could determine different stepping branching rules depending of the final objective: to approximate a joint probability or to provide a simple (though not necessarily exhaustive) partition of the state space that could help us to have an idea of how the variables take their values; (4) to use this model in classification problems.

Acknowledgments

This work has been supported by Spanish MEC under project TIN2004-06204-C03-{02,03}.

References

M.R. Anderberg. 1973. *Cluster Analysis for Applications*. Academic Press.

P. Cheeseman and J. Stutz. 1996. Bayesian classification (AUTOCLASS): Theory and results. In U. M. Fayyad, G. Piatetsky-Shapiro, P Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI Press/MIT Press.

G.F. Cooper and E.A. Herskovits. 1992. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347.

A. P. Dempster, N. M. Laird, and D.B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 1:1–38.

R.O. Duda, P.E. Hart, and D.J. Stork. 2001. *Pattern Recognition*. Wiley.

M.J. Flores and J. A. Gámez. 2005. Breeding value classification in manchego sheep: A study of attribute selection and construction. In *Kowledge-Based Intelligent Information and Enginnering Systems. LNAI*, volume 3682, pages 1338–1346. Springer Verlag.

N. Friedman. 1998. The Bayesian structural EM algorithm. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 129–138. Morgan Kaufmann.

D. Hand, H. Mannila, and P. Smyth. 2001. *Principles of Data Mining*. MIT Press.

A.K. Jain, M.M. Murty, and P.J. Flynn. 1999. Data clustering: a review. *ACM Computing Surveys*, 31:264–323.

F. V. Jensen. 2001. *Bayesian Networks and Decision Graphs*. Springer Verlag.

L. Kaufman and P. Rousseeuw. 1990. *Finding Groups in Data*. John Wiley and Sons.

D. Lowd and P. Domingos. 2005. Naive Bayes models for probability estimation. In *Proceedings of the 22nd ICML conference*, pages 529–536. ACM Press.

D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. 1998. UCI repository of machine learning databases.

J.M. Peña, J.A. Lozano, and P. Larrañaga. 2000. An improved Bayesian structural EM algorithm for learning Bayesian networks for clustering. *Pattern Recognition Letters*, 21:779–786.

A. Salmerón, A. Cano, and S. Moral. 2000. Importance sampling in Bayesian networks using probability trees. *Computational Statistics and Data Analysis*, 34:387–413.

P.H. Sneath and R.R. Sokal. 1973. *Numerical Taxonomy*. Freeman.

P. Spirtes, C. Glymour, and R. Scheines. 1993. *Causation, Prediction and Search*. Springer Verlag.