The Astrophysical Journal, 777:83 (10pp), 2013 November 10 doi:10.1088/0004-637X/777/2/83

AUTOMATIC CLASSIFICATION OF VARIABLE STARS IN CATALOGS WITH MISSING DATA

Karim Pichara[1,2,3] and Pavlos Protopapas[2,4]
[1] Computer Science Department, Pontificia Universidad Católica de Chile, Santiago, Chile
[2] Institute for Applied Computational Science, Harvard University, Cambridge, MA, USA
[3] The Milky Way Millennium Nucleus, Av. Vicuña Mackenna 4860, 782-0436 Macul, Santiago, Chile
[4] Harvard-Smithsonian Center for Astrophysics, 60 Garden Street, Cambridge, MA 02138, USA
Received 2013 February 4; accepted 2013 August 29; published 2013 October 17

ABSTRACT

We present an automatic classification method for astronomical catalogs with missing data. We use Bayesian networks and a probabilistic graphical model that allows us to perform inference to predict missing values given observed data and dependency relationships between variables. To learn a Bayesian network from incomplete data, we use an iterative algorithm that utilizes sampling methods and expectation maximization to estimate the distributions and probabilistic dependencies of variables from data with missing values. To test our model, we use three catalogs with missing data (SAGE, Two Micron All Sky Survey, and *UBVI*) and one complete catalog (MACHO). We examine how classification accuracy changes when information from missing data catalogs is included, how our method compares to traditional missing data approaches, and at what computational cost. Integrating these catalogs with missing data, we find that classification of variable objects improves by a few percent and by 15% for quasar detection while keeping the computational cost the same.

Key words: methods: data analysis – stars: statistics – stars: variables: general

Online-only material: color figures

1. INTRODUCTION

Classifying objects based on their features (e.g., color, magnitude, or any statistical descriptor) dates back to the nineteenth century (Rosenberg 1910). Recently, automatic classification methods have become much more sophisticated and necessary due to the exponential growth of astronomical data. In time-domain astronomy, where data are in the form of light curves, a typical classification method uses features[5] of the light curves and applies sophisticated machine learning to classify objects in a multidimensional feature space, provided there are enough examples to learn from (training). After almost a decade since the first appearance of automatic classification methods, many of those methods have produced and continue to produce high-fidelity catalogs (Kim et al. 2011, 2012; Bloom & Richards 2011; Richards et al. 2011; Debosscher et al. 2007; Wachman et al. 2009; Wang et al. 2010).

To take full advantage of all information available, it is best to use as many available catalogs as possible. For example, adding *u*-band or X-ray information while classifying quasars based on their variability is highly likely to improve the overall performance (Kim et al. 2011, 2012; Pichara et al. 2012). Because these catalogs are taken with different instruments, bandwidths, locations, times, etc., the intersection of these catalogs is smaller than any single catalog; thus, the resulting multi-catalog contains missing values. Traditional classification methods cannot deal with the resulting *missing* data problem because to train a classification model it is necessary to have all features for all training members. This can be solved either by selecting the complete intersection of the training members from all catalogs or by deleting the subset of features that are not common to all members. Unfortunately, both methods dramatically reduce the size of the training set because, in general, most of the features present missing values.

Alternatively, one can fill missing data using Monte Carlo approaches where each missing value is drawn from a distribution that is determined from all objects in the training set. However, this approach totally ignores the relationship amongst the features. Figure 1 demonstrates the drawbacks of ignoring such a relationship. If one draws from the marginal distributions of x and y (shown with solid blue lines), the fact that x and y are correlated is not taken into account. In principle, if we knew that x takes the value x_i, then we should be drawing from the conditional distribution, shown with the dashed red line.

One of the main characteristics that an imputation method must have to deal with astronomical catalogs is that the imputation time for new elements with missing data should be very fast, due to the amount of objects in catalogs. There are several data imputation methods presented in the literature (e.g., Troyanskaya et al. 2001; Stekhoven & Bühlmann 2012). In the method proposed by Troyanskaya et al. (2001), they used K nearest neighbors to impute the missing data. The basic idea is to select the K nearest neighbors in the space of the non-missing features and then predict the missing variable using a weighted average of the neighbors in that variable. Unfortunately, with this method, each time we ask for the imputation of one value we need to find the K nearest objects, which takes a considerable amount of time if we are dealing with millions of objects where we want to impute missing data. The method proposed by Stekhoven & Bühlmann (2012) uses one different Random Forest (RF; Breiman 2001) model to predict each of the features in the data set. Having n features, they fit n different RFs, where the ith RF is trained with the features $\{1, \ldots, i-1, i+1, \ldots, n\}$ as predictors and the variable i as a response. To train an RF with variable i as a response, they only use the observed part of variable i in the training set. Even though in the paper they propose the iterative model using all of the data (impractical for astronomical catalogs), we might iterate only using a training

[5] We use the term "features" for all the descriptors we may use to represent a light curve with a numerical vector.

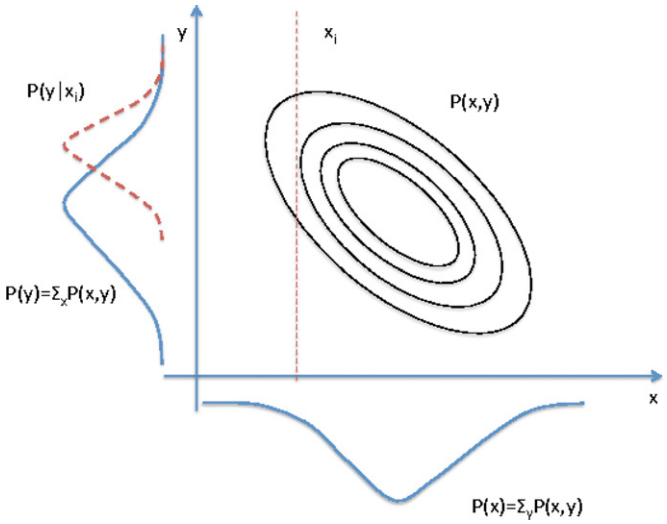


Figure 1. Joint distribution is shown as contours, the marginal distributions are shown as the dashed line, and conditional distributions in solid lines. Ignoring the joint distribution, one draws from the marginal distribution. However, knowledge of the joint distribution allows us to draw from the conditional distribution.

(A color version of this figure is available in the online journal.)

set and then use the set of forests to impute data in the big catalogs. In astronomical catalogs, features usually correspond to astrophysical variables of objects, and in many cases, astronomers may want to know an indicator of uncertainty in the prediction, or a probability distribution over the imputation value; unfortunately, from RFs it is hard to directly get uncertainty indicators for continuous responses, given that the model does not provide a distribution over the predictions.

In this work, we use (not naive) Bayesian networks (BNs). BNs are models that represent probabilistic dependency relationships among features using graphs (Pearl 1988), where nodes represent the features and connections provide information about the probabilistic dependency relationships between features. BNs belong to the family of graphical models, and they are very suitable to perform inference on a set of features given observations.

Some recent works in astronomy use BN models for automatic classification (Mahabal et al. 2008; Djorgovski et al. 2012). Also, Broos et al. (2011) propose a Machine Learning model to classify X-ray sources using a naive scheme, where all features are assumed to be independent given the class.

Usually in catalogs with missing data, the missing features are different depending on the object. That is the main reason to use a model that can deal with evidences that change from object to object. We assume that the nature of missing data is MAR (Missing at Random) or MCAR (Missing Complete at Random). MCAR means that the probability that a feature is missing is independent of the other features in observations. MAR means that the missing features may depend on the values of the observed component. MAR and MCAR cases can be handled with our model because we find the dependencies (or independencies) between features with BNs. For NMAR cases (Not Missing at Random), the probability that a feature is missing may depend on the other missing values (for example, no detections when the observed objects are too faint). We do not know any method able to handle NMAR cases without ad hoc distribution for the missing values.

Note that even in the case where we have a complete training set, the resulting classification model will only be able to classify objects that have complete information. In other words, we cannot use a classification model to predict objects with missing features. One option to mitigate this problem is to abstain from predicting for those objects, but that hinders the completeness of the prediction. With the proposed method, we can impute the data while predicting based on the learned BN.

In this work, we use as a base catalog the MACHO catalog (Alcock et al. 1997) and extract 14 features from each light curve. We then combine the MACHO catalog with other catalogs containing magnitudes at different wavelengths. We show how a BN in combination with an RF classifier can overcome the missing data problem and outperform other methods. Applying this model on a real data set, we are able to generate catalogs of variable source with extremely high fidelity.

Section 2.1 summarizes BNs, and in Section 2.2 we show how BNs can be used to infer missing values. Sections 2.3 and 2.4 show how we build BNs, first with complete data and then with missing values. Section 3 contains the classification mechanism. Results from experiments with real data are shown in Section 4. Conclusions follow in Section 5.

2. THEORETICAL BACKGROUND

2.1. Bayesian Networks

A BN is a probabilistic model that belongs in the special class of graphical models. Graphical models deal with uncertain data in the presence of latent variables. Latent variables can include any information that is unobservable, such as the mass of a star. In short, anything that is relevant to explain the observed data but was itself not observed can become a latent variable. In a graphical model, one assumes certain local statistical dependencies between the random variables that correspond to the latent variables and observed data. BNs are directed graphical models, in which the statistical dependency between random variables is based on directional relationships. Another class of graphical modes, not relevant to this paper, are undirected graphical models, such as Markov random networks. Many models that are typically not described as graphical models can be reinterpreted within a graphical modeling framework. Similarly, many process models or stochastic processes can be couched as graphical models.

To better explain the fundamentals of a BN, we present a simple example. Considering a light curve of a source, we examine certain properties of the light curve and other available information such as point-spread function size, magnitude, color, etc. In this example, we want to determine if the star is periodic or not. If the star exhibits periodic behavior that is larger than the error standard and we assume that the source has been observed often and for long time, a simple periodogram would flag this source as periodic very reliably. Unfortunately, a faulty CCD or unreliable electronics can mimic the periodic behavior, which can fool the periodogram. Regular engineering reports can reveal such behavior. Finally, you want to confirm every detection with a visual inspection. To model this situation, we can use a BN as shown in Figure 2, where nodes are the random variables and arrows indicate conditional dependencies between variables. The network encodes the intuition that the status of the periodogram results depends on a faulty CCD or actual periodic variation, and that the final call depends on the results of the visual inspection that only happens if the periodogram indicates that the source is periodic. It is useful to think of

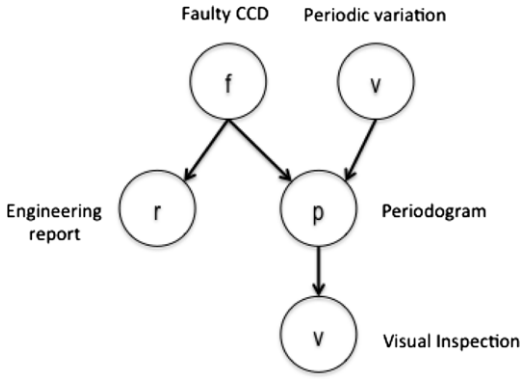


Figure 2. Bayesian network for the periodogram example.

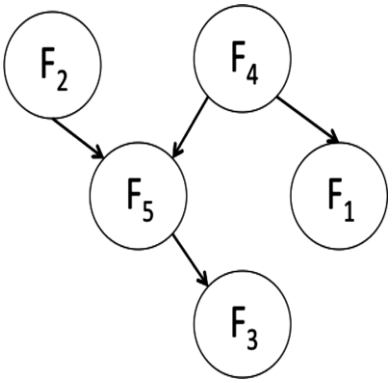


Figure 3. Example of a BN with features $\{F_1, \ldots, F_5\}$. The joint distribution can be factorized as the product of five probabilities, each one corresponding to the probability of the respective node variable given its parents in the network.

these conditional dependencies as causal relationships between variables; periodic behavior might cause the periodogram to flag a light curve as periodic, which in turn might pass the visual inspection. However, you should keep in mind that BNs can also be constructed in the absence of any causal interpretation. This is how Pearl (1994) originally thought of BNs as a way to reason probabilistically about causes and effects.

More formally, let $S = \{x_1, \ldots, x_n\}$ be a set of data instances (these are the light curves), each one described with a set of D features $\{F_1, \ldots, F_D\}$ (these are the light-curve features and/or brightness magnitudes). Each instance x_i is represented as a vector $x_i = \{F_1^i, \ldots, F_D^i\}$. BNs can represent the joint probability distribution $P(F_1, \ldots, F_D)$ of data set S as a product of factors, where each factor is a conditional probability distribution of each node given its parents in the BN:

$$P(S) = \prod_{i=1}^{n} P(x_i) = \prod_{i=1}^{n} P\left(F_1^i, \ldots, F_D^i\right) = \prod_{i=1}^{n} \prod_{j=1}^{D} P\left(F_j^i | \mathrm{Pa}_{\mathrm{BN}}^i(F_j)\right), \quad (1)$$

where $\mathrm{Pa}_{\mathrm{BN}}(F_j)$ represents the set of parents of variable F_j in the BN and $\mathrm{Pa}_{\mathrm{BN}}^i(F_j)$ indicate that parents of feature F_j are instantiated in the values of x_i.

One of the main advantages of the BN factorization is that each of the factors involves a smaller number of features, where it is easier to estimate.

For example, in Figure 3, we show a BN in a domain of five features $\{F_1, \ldots, F_5\}$. The joint probability distribution can be factorized according to the BN as

$$P(F_1, \ldots, F_5) = P(F_1|F_4)P(F_2)P(F_3|F_5) \times P(F_4)P(F_5|F_2, F_4).$$

In this case, instead of estimating a probability distribution over the five-dimensional space $(F_1, \ldots, F_5)$, we only need to estimate simpler distributions, such as $P(F_1|F_4)$, $P(F_2)$, $P(F_3|F_5)$, $P(F_4)$, and $P(F_5|F_2, F_4)$.

2.2. Inference in Bayesian Networks

BNs are useful to make inferences on any unobserved variable given a set of evidence. In our case, we aim to use BNs to predict values of missing features given the observed ones.

For example, consider the same BN as in Figure 3 and suppose we found an object with missing values F_5 and F_2 (we can observe F_1, F_3, F_4). If we want to estimate the most probable value for variable F_5 given the observed values for F_1, F_3, F_4, we can calculate $P(F_5|F_1, F_3, F_4)$ as

$$\begin{aligned} P(F_5|F_1, F_3, F_4) &= \frac{P(F_1, F_3, F_4, F_5)}{P(F_1, F_3, F_4)} \\ &= \frac{\sum_{F_2} P(F_1, F_2, F_3, F_4, F_5)}{\sum_{F_2, F_5} P(F_1, F_2, F_3, F_4, F_5)} \\ &= \frac{\sum_{F_2} P(F_1|F_4)P(F_2)P(F_3|F_5)P(F_4)P(F_5|F_2, F_4)}{\sum_{F_2, F_5} P(F_1|F_4)P(F_2)P(F_3|F_5)P(F_4)P(F_5|F_2, F_4)} \\ &= \frac{P(F_1|F_4)P(F_3|F_5)P(F_4)\sum_{F_2} P(F_2)P(F_5|F_2, F_4)}{P(F_4)P(F_1|F_4)\sum_{F_2, F_5} P(F_3|F_5)P(F_2)P(F_5|F_2, F_4)}. \end{aligned} \quad (2)$$

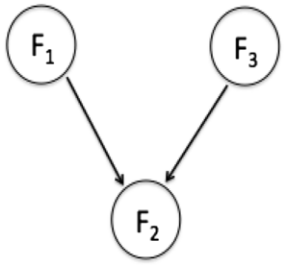
Summing out the unobserved features and "pushing in" the factors in the sums is known as *variable elimination* (Pearl 1994), which is the simplest exact inference algorithm.

In this work, we use Gaussian node inference (Shachter & Kenley 1989). Gaussian nodes are commonly used for continuous data; each variable is modeled with a Gaussian distribution where its parameters are a linear combination of the parameters of the parent nodes in the BN. Let F_j be a node with p parents, where each parent has a Gaussian distribution with mean μ_i and variance σ_i ($i \in [1 \ldots p]$). We model F_j with a Gaussian distribution with mean $\mu = [\mu_1, \ldots, \mu_k]$ and covariance matrix $\Sigma = [\sigma_{ik}]$, where σ_{ik} is the covariance between the ith parent of F_j and the kth parent of F_j. The probability distribution for node F_j is

$$P(F_j) = \mathcal{N}(\beta_0 + \beta^T \mu; \sigma^2 + \beta^T \Sigma \beta), \quad (3)$$

where β and σ are the parameters of the linear combination (which need to be estimated in the learning process). In Section 2.3.2, we explain details about Gaussian node representation and how to estimate the parameters. For the scope of this section, we assume that the parameters are known.

The simple idea behind inference with Gaussian nodes is that features that are not involved in the calculus of a probability can be eliminated from the BN (barren nodes). The easiest barren nodes to be eliminated are leaf nodes because they can be deleted without making any other change in the network. Unfortunately, not all barren nodes are leaves. The key idea is to perform arc reversals in order to let barren nodes be leaves. When such a reversal is performed, to preserve the



Features	Possible Values
F_1	1,2,3
F_2	1,2,3,4
F_3	1,2

Figure 4. Example of a simple BN with features $\{F_1, F_2, F_3\}$ and the values each feature can take.

joint distribution the network has to be adjusted, or re-learned. Shachter & Kenley (1989) describe a methodology for adjusting the network parameters that we adapted for this work too (see Appendix A for details).

2.3. Learning Bayesian Networks with Complete Data

In the previous section, we showed how to make inferences once the BN is known. Learning the network involves learning the structure (edges) and the parameters (probability distributions on each of the factors). We explain both cases separately in the next subsections.

2.3.1. Structure Learning with Complete Data

Given that the number of possible network structures grows exponentially with the number of nodes or features (Cooper & Herskovits 1992), it is not possible to do an exhaustive search. Usually, a greedy search strategy is necessary to find a suitable solution; in this work, we use the K2 algorithm (Cooper & Herskovits 1992). Starting with an initial random order of features (nodes) and an empty network (no edges), we start adding parents to each variable, such that the next parent we add is the one who creates the highest improvement in the network score; we keep adding parents until we complete the maximum allowed (parameter given by the user). In our work, we use a maximum of three parents. Note that if one node already has two parents and we attempt to add the third one, it might be possible that keeping two parents is better than adding a third one; in that case the node stays with two parents.

The score of a network structure is related to how the structure fits data. To calculate the score, we evaluate the probability of the structure given the data, which corresponds to applying the same factorization imposed by the structure and using multinomial distributions over each factor ($P(F_j|\mathrm{Pa_{BN}}(F_j))$ in Equation (1)). We estimate each probability by first discretizing the possible values that each feature F_j can take, $(f_{j1}, \ldots, f_{jr_j})$, and then creating a multi-dimensional histogram for $P(F_j|\mathrm{Pa_{BN}}(F_j))$.

Consider the feature F_j. Let q_j be the number of possible instantiations of the parent set $\mathrm{Pa_{BN}}(F_j)$. Recall that r_j is the maximum number of values that variable F_j can take. Let $N^j_{k,m}$ be the number of cases in data where variable F_j has the value f_{jk} ($k \in [1 \ldots r_j]$) when its set of parents $\mathrm{Pa_{BN}}(F_j)$ is instantiated to some value w^j_m, and let $N^j_m = \sum_{k=1}^{r_j} N^j_{k,m}$. For example, in Figure 4, if $j = 2$, $[f_{j1} = 1, f_{j2} = 2, \ldots, f_{j4} = 4]$, $q_j = 6$ (3×2 possible values of the joint combination of parents), $w^j_1 = \{1 \quad 1\}$, $w^j_2 = \{1 \quad 2\}, \ldots, w^j_6 = \{3 \quad 2\}$. Note that m is an index moving in the possible combinations of values of the joint set of parents ($m \in [1 \ldots 6]$).

Then the probability of a given structure B_s can be shown to be given by (see Appendix B for a derivation)

$$P(B_s|\mathrm{data}) = P(B_s) \prod_{j=1}^{D} \prod_{m=1}^{q_j} \frac{(r_j - 1)!}{(N_{jm} + r_j - 1)!} \prod_{k=1}^{r_j} N_{jmk}!, \quad (4)$$

where the term $P(B_s)$ is the prior on the network structure B_s.

2.3.2. Parameter Learning with Complete Data

Learning the parameters of a BN means learning the distribution of each of the factors on the right-hand side of Equation (1). The factorization of Equation (1) is given by the structure of the BN. This tells us that to learn the parameters it is necessary first to know the structure.

Given that features involved in our work are all continuous, we again use Gaussian nodes (Shachter & Kenley 1989).

Let F_j be a node in the network and $\mathrm{Pa_{BN}}(F_j)$ be the set of parents for F_j. Let us assume that F_j has k parents ($|\mathrm{Pa_{BN}}(F_j)| = k$), and we model F_j as a linear Gaussian of its parents:

$$P(F_j) = \mathcal{N}(\beta_0 + \beta^T \mu; \sigma^2 + \beta^T \Sigma \beta), \quad (5)$$

where the set of parents $\mathrm{Pa_{BN}}(F_j)$ are jointly Gaussian $\mathcal{N}(\mu; \Sigma)$, and μ and Σ are calculated from the data. Note that μ and β are k-dimensional vectors and the matrix Σ is $k \times k$.

To learn a Gaussian node, we learn the set of parameters $\{\beta_0, \ldots, \beta_k; \sigma\}$ of the linear combination. Let $\mathrm{Pa_{BN}}(F_j) = \{\tilde{F}_1, \ldots, \tilde{F}_k\}$ be the parent nodes with respective means $\{\mu_1, \ldots, \mu_k\}$; then $P(F_j|\mathrm{Pa_{BN}}(F_j)) = \mathcal{N}(\beta_0 + \beta_1 \tilde{F}_1 + \cdots + \beta_k \tilde{F}_k; \sigma^2)$.

Our task is to learn the set of parameters $\theta_{F_j} = \{\beta_0, \ldots, \beta_k; \sigma\}$. To learn those parameters, we optimize the log likelihood, expressed as

$$l_{F_j}(\theta_{F_j}|\mathrm{data}) = \sum_{i=1}^{n} \left[-\frac{1}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} (\beta_0 + \beta_1 x_{i\tilde{F}_1} + \cdots + \beta_k x_{i\tilde{F}_k} - x_{ij})^2 \right], \quad (6)$$

by setting its derivative with respect to β_0 to zero. We have

$$E[F_j] = \beta_0 + \beta_1 E[\tilde{F}_1] + \cdots + \beta_k E[\tilde{F}_k], \quad (7)$$

where $E[F_j] = \mu_j$ is the expectation of the variable F_j in the data. Setting the derivative of Equation (6) with respect to $\beta_1, \ldots, \beta_k$ to zero, we have the following k equations:

$$E[F_j \cdot \tilde{F}_1] = \beta_0 E[\tilde{F}_1] + \beta_1 E[\tilde{F}_1 \cdot \tilde{F}_1] + \cdots + \beta_k E[\tilde{F}_k \cdot \tilde{F}_1] \quad (8)$$

$$\vdots$$

$$E[F_j \cdot \tilde{F}_k] = \beta_0 E[\tilde{F}_k] + \beta_1 E[\tilde{F}_1 \cdot \tilde{F}_k] + \cdots + \beta_k E[\tilde{F}_k \cdot \tilde{F}_k]. \quad (9)$$

Setting the derivatives to zero, we end with $k + 1$ linear equations with $k+1$ unknowns. We can solve the equations using standard linear algebra to find the $k + 1$ solutions $\beta^*_0, \ldots, \beta^*_k$. To find σ, we replace the values of $\beta^*_0, \ldots, \beta^*_k$ in Equation (6)

and set the derivative of the log likelihood with respect to σ^2 to zero; then we have

$$\sigma^2 = \mathrm{cov}[F_j, F_j] - \sum_{p=1}^{k}\sum_{q=1}^{k} \beta_p^* \beta_q^* \,\mathrm{cov}[\tilde{F}_p, \tilde{F}_q], \quad (10)$$

where $\mathrm{cov}[\tilde{F}_p, \tilde{F}_q] = E[\tilde{F}_p \cdot \tilde{F}_q] - E[\tilde{F}_p]\,E[\tilde{F}_q]$. Note that if parent nodes are root nodes, they are just modeled with a unidimensional normal distribution and Equations (7) and (10) return the mean and variance of that variable.

2.4. Learning Bayesian Networks with Missing Data

Now that we know how to learn the structure and the parameters of a BN under complete data, we turn our attention to how to learn both the structure and the parameters under incomplete data. To learn the parameters with missing data, we need to know the structure previously, and to learn the structure, we need to guess the missing values. This is done in an iterative method.

We start by first describing the parameter-learning algorithm and then the structure-learning model.

2.4.1. Learning Parameters with Missing Data

We assume that we already know the structure of the BN before we start learning the distribution parameters with missing data. The basic idea is to start estimating the joint distribution of the set of (root) parent nodes from incomplete data using a multivariate Gaussian distribution. Then we estimate the distribution of children nodes like in the complete data case (Section 2.3.2) sampling the missing values of the parents from the distributions learned at the beginning.

To learn the parameters of a multivariate Gaussian distribution in the incomplete data case, we use the method proposed in Ghahramani & Jordan (1995). To optimize the log likelihood of the model given the data under the missing data case, each data point x_i can be written as $x_i = \{x_i^o, x_i^m\}$, using the superscripts m and o to indicate the features that are observed or missing. Let $\Sigma = \{\Sigma^{oo}, \Sigma^{om}, \Sigma^{mo}, \Sigma^{mm}\}$ and $\mu = \{\mu^m, \mu^o\}$ be the covariance matrix and vector mean of the multivariate Gaussian distribution we are estimating. The log likelihood for incomplete data, including the new notation for x_i and using a mixture of Gaussian distributions, can be written as

$$\begin{aligned} l(\theta|x^o, x^m) = \sum_{i=1}^{n} \Bigg[& \frac{n}{2}\log 2\pi + \frac{1}{2}\log|\Sigma| \\ & - \frac{1}{2}\left(x_i^o - \mu^o\right)^T \Sigma^{-1,oo}\left(x_i^o - \mu^o\right) \\ & - \left(x_i^o - \mu^o\right)^T \Sigma^{-1,om}(x_i^m - \mu^m) \\ & - \frac{1}{2}\left(x_i^m - \mu^m\right)^T \Sigma^{-1,mm}\left(x_i^m - \mu^m\right) \Bigg]. \end{aligned} \quad (11)$$

Given that we have the likelihood expressed in terms of unknown latent features (the unobserved part of data), we optimize it using the expectation maximization (EM) algorithm (Dempster et al. 1977). EM optimizes the likelihood function of a model, which depends on latent or unobserved features. The optimization procedure is a two-step iteration. The first step, the expectation step (E-step), calculates the expected value of the latent features to be used in the likelihood function.

In other words, the E-step creates a function to be optimized, using the expected value of the latent features estimated from the current value of the unknown parameters. The maximization step (M-step) is the maximization of the likelihood function created by the E-step, generating a new value of the current parameters (to be used again in the E-step).

In the E-step, we need to estimate the unobserved part x_i^m. We can express the expected value of x_i^m as

$$E\left[x_i^m \middle| x_i^o, \mu, \Sigma\right] = \mu^m + \Sigma^{mo}\Sigma^{-1,oo}\left(x_i^o - \mu^o\right). \quad (12)$$

Starting for an initial guess of the parameters μ and Σ, we calculate the expected value of missing data using Equation (12), and then we optimize the values of μ and Σ and continue iterating until μ and Σ do not change substantially.

After estimating the joint Gaussian distribution of the set of parents, we can estimate the normal distribution of the children like in the complete data case (Section 2.3.2) where the missing values of the parent are sampled from the learned multivariate Gaussian.

2.4.2. Learning the Network Structure with Missing Data

To learn the structure of a BN with missing data, we complete the missing values and then iterate to improve these values using the structure learned so far (Singh 1997). Algorithm 1 shows the main steps to construct a BN structure from missing data.

The convergence criterion is that the score of the network $\langle B^{(t)}, \theta^{(t)}\rangle$ does not change substantially. Note that in the first iteration, we just fill the missing values using an independent Gaussian mixture model. Although independency is a very strong assumption, in our case it does not affect the final result given that in all subsequent steps we refill the missing values with data sampled from the current BN; in other words, we use all probabilistic dependencies between features given by the network structure.

3. THE AUTOMATIC CLASSIFICATION MODEL

In previous sections, we show how to fill missing values using probabilistic dependencies between features. After we infer the missing values using the BN, we proceed to train the automatic classifier using the new training completed set. In this work, we use an RF classifier (Breiman 2001), which is a popular and very efficient algorithm based on decision tree models (Quinlan 1993) and bagging for classification problems (Breiman 1996, 2001).[6] It belongs to the family of ensemble methods, appearing in machine learning literature at the end of the 1990s (Dietterich 2000), and has been used recently in the astronomical journals (Pichara et al. 2012; Carliles et al. 2010; Richards et al. 2011). We give a very brief explanation here of how RF works; the reader can find detailed description in Breiman (2001).

The process of training or building an RF given training data is as follows.

1. Let P be the number of trees in the forest (model parameter) and F be the number of features describing data.

[6] Other models can be used for classification, but we found that RF gives superior results.

Algorithm 1: Algorithm to Learn BN Structure with Missing Data

- *Learn for each variable in* $\{F_1, \ldots, F_D\}$ *a univariate Gaussian Mixture* $GM(i)$ $i \in [1 \ldots D]$;
- *Create* M *complete data sets* D_s^1, $s \in [1 \ldots M]$ *filling the missing values of each variable* F_i *with values sampled from* $GM(i)$;
- $t = 1$;

while *Convergence criteria are not achieved* **do**

  for $s = 1$ *to* M **do**

    From each complete data set in $D_s^{(t)}$, learn a BN, $B_s^{(t)}$ (Section 2.3.1).

  - *Create one Bayesian network structure* $B^{(t)}$ *as the union of all the BNs* [a];
  - *Learn the parameters* $\theta^{(t)}$ *using the original incomplete data and the network structure* $B^{(t)}$ *(Section 2.4.1)*;
  - *Use the network* $\langle B^{(t)}, \theta^{(t)} \rangle$ *to sample new values and create new completed data sets* $D_s^{(t+1)}$;
  - $t = t + 1$.

[a]The union is performed in two steps: (1) make all structures $B_s^{(t)}$ consistent with the feature order used in the algorithm from Section 2.3.1 by performing arc reversals and (2) create the arc union of all the consistent structure from the previous step.

2. Build P sets of n samples taken with replacement from the training set; this is called bagging. Note that each of the P bags has the same number of elements with the training set but less different examples, given that the samples are taken with replacement (the training set also has n samples).
3. For each of the P sets, train a decision tree (without pruning) using at each node a random sample of $F' \leqslant F$ possible features to select the one that optimizes the split (F' is a model parameter).

The RF classifier creates many linear separators, attempting to separate between elements of different classes using some features (the ones given by the nodes of each decision tree) and some data points (the ones given by each of the bags).

Each of the decision trees creates one decision, and the final decision is the most voted class among the set of P decision trees (see Breiman 2001 for more details). It is worth noting that Breiman (2001) showed that as the number of trees goes to infinity, the classification error of the RF becomes bounded and the classifier does not overfit the data.

4. EXPERIMENTAL RESULTS

In this section, we show the results from the application of the model on four astronomical catalogs, three with missing values. We not only show the advantages of the model, but we produce a catalog of variable stars within the LMC that is available for downloading.[7]

First, we prove the imputation accuracy of our model performing imputation tests in real data sets. Then, to test the advantage of the model, we proved three main facts: (1) it is possible to learn an automatic classification model that is able to deal with missing data, (2) information with missing data can be useful for automatic classification, in other words the model outperforms any model that uses a subset of training set with complete data, and (3) the proposed model overcomes the case where missing data are filled using traditional statistical methods that model each variable independently.

Fortunately, the main computational cost of the algorithm occurs during the training phase, where the model needs to

[7] http://iic.seas.harvard.edu/research/time-series-center

Table 1
NRMSE Using Bayesian Networks and Gaussian Mixtures in the MACHO Data Set

	5%	10%	15%	20%
BN	0.396	0.437	0.514	0.523
Gaussian	0.624	0.701	0.764	0.790

Note. Missing values were artificially generated completely at random.

learn the BN structure and parameters. After training the model, performing the inference on missing values for a light curve takes a fraction of a second.

We use three astronomical catalogs with missing data, SAGE (Meixner et al. 2006), *UBVI* (Piatti et al. 2011), and Two Micron All Sky Survey (2MASS; Skrutskie et al. 2006). We also use the MACHO catalog (Alcock et al. 1997), with no missing values, but useful in comparing the light-curve classification accuracy between the MACHO features with the additional incomplete extra features from SAGE, *UBVI*, and 2MASS. We process around 20 million objects. The MACHO light curves are described using 14 variability features: CAR σ, Mean Mag, CAR τ, σ, η, Con, Stetson L, CuSum, $B - R$, Period, Period SNR, Stetson K AC, N above 4, and N below 4. See Pichara et al. (2012) for a description of the MACHO features.

4.1. Imputation Tests

To calculate the imputation error, we use the MACHO data set where we randomly delete 5%, 10%, 15%, and 20% of data entries in order to simulate missing values. We run our model and measure the NRMSE (Normalized Root Mean Squared Error) over the predicted values, defined as

$$\text{NRMSE} = \sqrt{\frac{\text{Mean}([x_{\text{imp}} - x_{\text{true}}]^2)}{\text{var}(x_{\text{true}})}},$$

where x_{imp} is the imputed value and x_{true} is the true value. When the estimation is accurate, NRMSE approaches 0.0, whereas when the estimation is equivalent to a random guess, NRMSE approaches 1.0. We compare our imputation results with an imputation method using mixture of Gaussians. Table 1 shows our results.

We can see from Table 1 that the BN method presents less NRMSE compared with Gaussian mixture imputation.

4.2. Interpreting the BN

One of the advantages of BNs is that they provide a conditional probability structure of features, which can be interpreted to attain deeper insight into your data. Figure 5 shows the BN structure our model found for the MACHO data set. Connection among features indicates a degree of probabilistic dependency among features. Nodes that are not connected with any other nodes are estimated independently from the others. We added colors to the nodes to indicate groups of features that belong to the same "type" of features. For example, features related to the magnitude level of the object are in red; as we can see, there are many connections among these kind of nodes, showing that the learning algorithm despite the missing data was able to detect most of the dependency relationships. There are some relationships that the model could not find, for example, the feature CAR τ was modeled as independent given that it is not connected with any other feature in the network structure.

Table 2
Percentage of Missing Values in the SAGE/2MASS Catalogs

Variables	% of Missing Values
J	54%
H	54%
K	58%
m_{36}	1%
m_{45}	13%
m_{58}	68%
m_{80}	74%

Table 3
Percentage of Missing Values in the *UBVI* Catalog

Variables	% of Missing Values
U	49%
B	0%
V	0%
I	14%

Table 4
Number of Objects Per Class in the SAGE–*UBVI* Training Set

Class	Number of Training Objects
Non-variables	1136
QSO	45
Be star	76
Cepheid	70
RR Lyrae	69
EB	100
LPV	337

4.3. Classification Results in Missing Data Catalogs

For SAGE and 2MASS we used a training set of 1955 objects described in seven features (J, H, K, m_{36}, m_{45}, m_{58}, m_{80}). Table 2 shows the percentage of missing values for different features in SAGE/2MASS catalogs.

For the *UBVI* catalog (Piatti et al. 2011) we used 4193 training instances described in four features (U, V, B, I). Table 3 shows the percentage of missing values for different features in the *UBVI* training set.

We created one training set gathering all SAGE, 2MASS, and *UBVI* training sets. The resulting training set contains seven classes of stars: non-variables, quasars (QSO), Be stars, Cepheids, RR Lyraes, eclipsing binaries (EBs), and long periodic variables (LPV). The number of objects per class on the SAGE–2MASS–*UBVI* training set is described in Table 4.

To evaluate the capabilities of our model dealing with missing data, we compared the results of classification accuracy of our model versus filling the missing data with the independent Gaussian mixture model on each variable. Then we take samples from that distribution to replace the missing values.

These experiments allows us to show the importance of analyzing the dependency relationship between features in order to make inferences on missing values. To measure the accuracy, we use precision, recall, and F-Score, defined as

$$\text{F-Score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}},$$

where precision and recall are defined as

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \qquad \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

where TP, FP, and FN are the number of true positives, false positives, and false negatives, respectively. Note that all these values are obtained using a 10-fold cross-validation process.

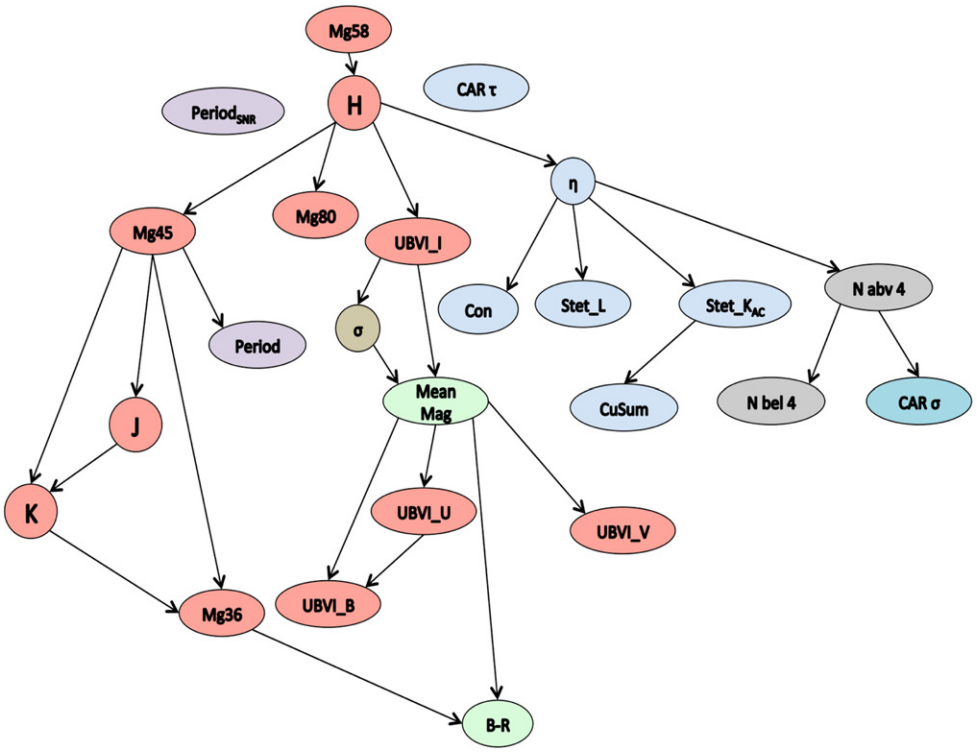


Figure 5. BN structure for the MACHO data set; the colors indicate that the nodes belong to the same type of features.
(A color version of this figure is available in the online journal.)

Table 5
Precision, Recall, and F-score for Different Classes Using Two Different Methods for Filling Missing Values in the SAGE–2MASS–*UBVI* Training Set

	Gaussian Mixture			Our Model		
Class	Precision	Recall	F-Measure	Precision	Recall	F-Measure
None-variables	0.857	0.952	0.902	**0.878**	0.942	**0.909**
Quasar	0.9	0.8	0.847	0.878	**0.956**	**0.915**
Be	0.679	0.5	0.576	**0.724**	**0.553**	**0.627**
Cepheid	0.805	0.886	0.844	0.785	0.886	0.832
RR-Lyrae	0.333	0.014	0.028	**0.583**	**0.203**	**0.301**
EB	0.5	0.25	0.333	**0.525**	**0.31**	**0.39**
LPV	0.919	0.938	0.928	**0.925**	**0.947**	**0.935**
Weighted average:	0.821	0.851	0.826	**0.846**	**0.863**	**0.848**

Note. (1) Independent mixtures of Gaussian and (2) our model.

Table 5 shows the accuracy for the model that uses Gaussian mixtures and the accuracy of the proposed model. We can see that our model presents better results in most of the classes (bold numbers), for example, we can see significant improvement in the recall of quasars, which indicates that the model is able to detect more quasars than the model that fills features independently. We also increase the precision and recall for Be stars, RR-Lyraes, and LPVs.

After evaluating our proposed method with SAGE, 2MASS, and *UBVI* catalogs, we aim to probe that all the information encoded by these catalogs is useful to classify variable stars after missing data were imputed. To probe that, we combined again the SAGE–2MASS–*UBVI* training set with a training set used by Pichara et al. (2012) in a previous work, the MACHO catalog (Alcock et al. 1997). In the previous work, an automatic classifier was built in order to detect quasars in the MACHO database. This training set was created extracting 14 time series features per band (see Pichara et al. 2012 for further details).

Table 6
Precision, Recall, and F-score to Compare the Model Used by Pichara et al. (2012) with and without the Information of the SAGE–*UBVI* Training Set

	Adding SAGE–2MASS–*UBVI* Features	Only MACHO Features
Quasar precision	0.843	0.857
Quasar recall	**0.956**	0.8
Quasar F-Score	**0.896**	0.828

To evaluate the contribution of our model, we trained a new classifier that uses the previous 14 features from Pichara et al. (2012) and the new features from the SAGE–2MASS–*UBVI* training set, after our model processed the missing values. We expect that the new classifier improves the quasar classification, showing higher recall and precision values in the training set and getting a new high-quality list of quasar candidates. Table 6 shows the results of both training sets, with and without the

SAGE–2MASS–*UBVI* catalog, showing that we improve the value of F-Score in quasar detection.

Moreover, after training the model, we run it on the whole MACHO catalog, in order to generate a new quasar candidate list. To evaluate the quality of the new quasar candidate list, we calculate the matching level of our list of candidates with the previous known lists. We use the recent works (Kim et al. 2012; Pichara et al. 2012) to compare their candidates with the list proposed in this work. Kim et al. (2012) found a list of 2566 candidates and a refined list of 663 strong candidates. In Pichara et al. (2012), we found a list of 2551 candidates, with 74% of matches with the 663 refined strong candidates. In this work, we improve the list, getting a list of 1730 candidates, from where we got 562 matches with the previous list of 2566 candidates and 502 matches with the previous list of 663 strong candidates (75.7%). We can see that our list of 1730 candidates has about the same level of matching with the previous strong candidate list, but reducing the size of the list by 32%.

5. CONCLUSIONS

We show a new way of dealing with missing data, testing on real astronomical data sets, showing that catalogs with missing data can be useful for automatic classification. One of the main advantages of our model is that it makes it possible to integrate catalogs in order to increase the available information for the training process. We improve the accuracy of our results in previous work on quasar detection due to the integration of new catalogs with missing data. Our model considers probability dependencies between features that make it possible to take advantage of the observed values, in order to increase the accuracy of the estimation when the number of observed values increases. Most of the computational time required is during the training time that makes it possible to run the model in complete catalogs because the model just needs to perform inferences on the missing values, which takes less than a second per object.

This work is supported by Vicerrectoría de Investigación (VRI) from Pontificia Universidad Católica de Chile, Institute of Applied Computer Science at Harvard University, and the Chilean Ministry for the Economy, Development, and Tourism's Programa Iniciativa Científica Milenio through grant P07-021-F, awarded to The Milky Way Millennium Nucleus.

APPENDIX A

ARC REVERSAL

To understand the meaning of arc reversal, it helps to think as if each node propagates its variance downstream to its successors (these variances are the elements of the covariance matrix in Equation (5) as we describe in Section 2.3.2). Suppose we reverse the arc $F_i \rightarrow F_j$. Before reversing, part of the variance in F_j was explained by F_i, and then after the reversal we have to compensate this by adding an arc from the parents of F_i to F_j. Also, part of the variance of F_i is now explained by F_j, so F_i's new variance must be discounted in order to adjust for that value. Figure 6 shows an example of the arc reversal procedure and updates of variances. In that example, F_4 is a barren node but cannot be removed from the network for it is not a leaf node. By reversing the arc $\{F_4 \rightarrow F_5\}$ to $\{F_5 \rightarrow F_4\}$, the variances and edge parameters are adjusted as described in the figure. With the arc reversed, F_4 is now a leaf node and can be removed from the network.

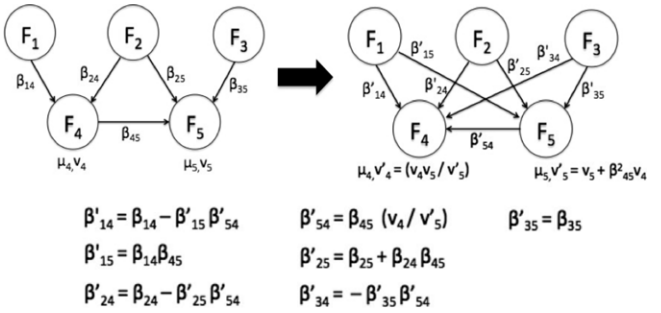


Figure 6. Arc reversal example for Gaussian nodes. β values correspond to the coefficients of the linear combination.

Formally, suppose we want to perform inference to calculate $P(F_J|F_K)$, where F_J and F_K are sets of features such that $F_J \cap F_K = \emptyset$. Let N be the total set of nodes in the network $((F_J \cup F_K) \subset N)$. We first create an ordered sequence s of nodes in N such that $F_K \prec_s F_J \prec_s N \setminus (F_J \cup F_K)$ (note that "\" is the set subtraction operator). We use the notation $\prec_s$ just to define an order relationship between the elements inside the sequence s; then $F_K \prec_s F_J$ means that F_K is before F_J in s. The idea of this order is just to leave evidence nodes before in the sequence, which ensures that they will be ancestors in the network, making easier the flow of information in the graph. The steps to perform inference are given below.

The last step of the algorithm simply uses the BN factorization to calculate the desired probability $P(F_J|F_K)$. Given that the barren nodes are no longer available in the BN, the joint probability is expressed only through features in $F_J \cup F_K$. Note that nodes in F_J are descendants of nodes in F_K, and nodes in F_K are all observed, so then we just instantiate them to their values and calculate directly the probability of nodes in F_J given the values in F_K.

APPENDIX B

BN SCORE

In this section, we show a derivation of Equation (4) for BN score (Cooper & Herskovits 1992). The score of a network

---

Given the ordered sequence $\prec_s$: **for** *each arc* $F_i \rightarrow F_j$ *in the BN* **do**

  if $F_j \prec_s F_i$, then reverse the arc $F_i \rightarrow F_j$.

Delete all nodes in $N \setminus (F_J \cup F_K)$ from the resulting network ;

$$P(F_J|F_K) = \frac{P(F_J,F_K)}{P(F_K)}$$

$$= \frac{\prod_{j=1}^{N\setminus(F_J\cup F_K)} P(F_j|\mathrm{Pa}_{\mathrm{BN}}(F_j))}{\sum_{N\setminus F_K} \prod_{j=1}^{N\setminus(F_J\cup F_K)} P(F_j|\mathrm{Pa}_{\mathrm{BN}}(F_j))}.$$

---

structure is related to how the structure fits data. To calculate the score, we evaluate the probability of the structure given the data, which corresponds to applying the same factorization imposed by the structure and using multinomial distributions over each factor ($P(F_j|\mathrm{Pa}_{\mathrm{BN}}(F_j))$ in Equation (1)). We estimate each probability by first discretizing the possible values that each feature F_j can take, $(f_{j1}, \ldots, f_{jr_j})$, and then creating a multi-dimensional histogram for $P(F_j|\mathrm{Pa}_{\mathrm{BN}}(F_j))$.

Consider the feature F_j. Let q_j be the number of possible instantiations of the parent set $\mathrm{Pa}_{\mathrm{BN}}(F_j)$. Recall that r_j is the maximum number of values that variable F_j can take.

Let $N^j_{k,m}$ be the number of cases in data where variable F_j has the value f_{jk} ($k \in [1 \ldots r_j]$) when its set of parents $\mathrm{Pa}_{\mathrm{BN}}(F_j)$ is instantiated to some value w^j_m, and let $N^j_m = \sum_{k=1}^{r_j} N^j_{k,m}$.

To decide for the best structure, we need an expression for the probability of a given structure under presence of data ($P(B_s, \mathrm{data})$). Given that for each structure we can have a different set of parameters (θ_s), we need to condition the parameters and integrate them out.

Using multinomial distributions, we have

$$\begin{aligned} P(B_s, \mathrm{data}) &= \int_{\theta_s} P(\mathrm{data}|B_s, \theta_s) P(\theta_s|B_s) P(B_s) d\theta_s \\ &= P(B_s) \int_{\theta_s} \left[\prod_{j=1}^{D} \prod_{m=1}^{q_j} \prod_{k=1}^{r_j} \theta_{jkm}^{N^j_{k,m}} \right] P(\theta_s|B_s) d\theta_s \\ &= P(B_s) \int_{\theta_{jkm}} \cdots \int \left[\prod_{j=1}^{D} \prod_{m=1}^{q_j} \prod_{k=1}^{r_j} \theta_{jkm}^{N^j_{k,m}} \right] \\ &\quad \times \left[\prod_{j=1}^{D} \prod_{m=1}^{q_j} P(\theta_{j1m}, \ldots, \theta_{jr_jm}) \right] \\ &\quad \times d\theta_{111}, \ldots, d\theta_{jkm}, \ldots, d\theta_{Dr_jq_j}. \end{aligned} \tag{B1}$$

Assuming a uniform distribution for $P(\theta_{j1m}, \ldots, \theta_{jr_jm})$, we have that $P(\theta_{j1m}, \ldots, \theta_{jr_jm}) = C_{jm}$ (for some constant C_{jm}). Given that C_{jm} is also a density function,

$$\int_{\theta_{jkm}} \cdots \int C_{jm}\, d\theta_{111}, \ldots, d\theta_{Dr_jq_j} = 1. \tag{B2}$$

Solving Equation (B2) yields $C_{jm} = (r_j - 1)!$ (see Appendix of Cooper & Herskovits 1992). Substituting this result and using independence of terms in Equation (B1), we have that

$$\begin{aligned} P(B_s, \mathrm{data}) = P(B_s) \prod_{j=1}^{D} \prod_{m=1}^{q_j} \int_{\theta_{jkm}} \cdots \int \left[\prod_{k=1}^{r_j} \theta_{jkm}^{N^j_{k,m}} \right] (r_j - 1)! \\ \times d\theta_{111}, \ldots, d\theta_{jkm}, \ldots, d\theta_{Dr_jq_j}. \end{aligned} \tag{B3}$$

The multiple (Dirichlet) integral in Equation (B3) has the following solution (Wilks 1962):

$$\int_{\theta_{jkm}} \cdots \int \prod_{k=1}^{r_j} \theta_{jkm}^{N^j_{k,m}} d\theta_{111}, \ldots, d\theta_{Dr_jq_j} = \frac{\prod_{k=1}^{r_j} N^j_{k,m}!}{(N^j_m + r_j - 1)!}. \tag{B4}$$

Substituting the result of Equation (B4) in Equation (B1), we have that

$$P(B_s|\mathrm{data}) = P(B_s) \prod_{j=1}^{D} \prod_{m=1}^{q_j} \frac{(r_j - 1)!}{(N^j_m + r_j - 1)!} \prod_{k=1}^{r_j} N^j_{k,m}!, \tag{B5}$$

where the term $P(B_s)$ is the prior on the network structure B_s. In this work, we assume that all possible network structures are equally likely, so we use the same prior for all of them. The expression $P(B_s|\mathrm{data})$ is the probability of the network structure given data, in other words, how good is the fit of the network structure with data. The better the structure fits the data, the higher the score $P(B_s|\mathrm{data})$. Then using the previously mentioned greedy search method, we select the structure that presents higher probability among the searched ones.

REFERENCES

Alcock, C., Allsman, R. A., Alves, D., et al. 1997, ApJ, 479, 119
Bloom, J. S., & Richards, J. W. 2011, in Advances in Machine Learning and Data Mining for Astronomy, ed. M. J. Way, J. D. Scargle, K. M. Ali, & A. N. Srivastava (Boca Raton, FL: CRC Press), 89
Breiman, L. 1996, in Proc. of International Conference on Machine Learning, Vol. 24 (Hingham, MA: Kluwer Academic Publishers), 123
Breiman, L. 2001, in Proc. of International Conference on Machine Learning, Vol. 45 (Hingham, MA: Kluwer Academic Publishers), 5
Broos, P., Getman, K., Povich, M., et al. 2011, ApJS, 194, 4
Carliles, S., Budavri, T., Heinis, S., Priebe, C., & Szalay, A. 2010, ApJ, 712, 511
Cooper, G. F., & Herskovits, E. 1992, Mach. Learn., 9, 309
Debosscher, J., Sarro, L., Aerts, C., et al. 2007, A&A, 475, 1159
Dempster, A., Laird, N., & Rubin, D. 1977, J. R. Stat. Soc. Ser. B (Methodological), 39, 1
Dietterich, T. 2000, in Proceedings of the First International Workshop on Multiple Classifier Systems, ed. J. Kittler & F. Roli (Berlin: Springer-Verlag), 1
Djorgovski, S. G., Mahabal, A. A., Drake, A. J., et al. 2012, in IAU Symp. 285, New Horizons in Time Domain Astronomy, ed. R. E. M. Griffin, R. J. Hanisch, & R. Seaman (Cambridge: Camrbidge Univ. Press), 1743
Ghahramani, Z., & Jordan, M. 1995, Learning from Incomplete Data, Tech. Rep., Lab Memo No. 1509, CBCL Paper No. 108, MIT AI Lab
Kim, D.-W., Protopapas, P., Byun, Y.-I., et al. 2011, ApJ, 735, 68
Kim, D.-W., Protopapas, P., Trichas, M., et al. 2012, ApJ, 747, 107
Mahabal, A., Djorgovski, S., Turmon, M., et al. 2008, AN, 329, 288
Meixner, M., Gordon, K. D., Indebetouw, R., et al. 2006, AJ, 132, 2268
Pearl, J. 1988, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference (San Mateo, CA: Morgan Kaufmann Publishers)
Pearl, J. 1994, Artificial Intelligence in Perspective (Cambridge, MA: MIT Press), 49
Piatti, A. E., Claria, J. J., & Ahumada, A. V. 2011, NewA, 16, 161
Pichara, K., Protopapas, P., Kim, D., Marquette, J., & Tisserand, P. 2012, MNRAS, 427, 1284
Quinlan, J. 1993, C4.5: Programs for Machine Learning (San Mateo, CA: Morgan Kaufmann Publishers)
Richards, J. W., Starr, D. L., Butler, N. R., et al. 2011, ApJ, 733, 10
Rosenberg, H. 1910, AN, 186, 71
Shachter, R., & Kenley, R. 1989, Manage. Sci., 35, 5
Singh, M. 1997, in Proceedings of the 14th National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence (Providence, RI: AAAI Press), 534
Skrutskie, M. F., Cutri, R. M., Stiening, R., et al. 2006, AJ, 131, 1163
Stekhoven, D. J., & Bühlmann, P. 2012, Bioinformatics, 28, 112
Troyanskaya, O., Cantor, M., Sherlock, G., et al. 2001, Bioinformatics, 17, 520
Wachman, G., Khardon, R., Protopapas, P., & Alcock, C. 2009, in Machine Learning and Knowledge Discovery in Databases, ed. W. Buntine, M. Grobelnik, D. Mladenic, & J. Shawe-Taylor (Lecture Notes in Computer Science, Vol. 5782; Berlin: Springer), 489
Wang, Y., Khardon, R., & Protopapas, P. 2010, in Machine Learning and Knowledge Discovery in Databases, ed. J. L. Balcázar, F. Bonchi, A. Gionis, & M. Sebag (Lecture Notes in Computer Science, Vol. 6323; Berlin: Springer), 418
Wilks, S. S. 1962, Mathematical Statistics (New York: Wiley), 1