Proceedings of the 2016 Winter Simulation Conference
T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, eds.

RANDOM VECTOR GENERATION FROM MIXED-ATTRIBUTE DATASETS USING RANDOM WALK

Andrew Skabar

Department of Computer Science and Information Technology
La Trobe University
Kingsbury Drive
Bundoora, VIC 3083, AUSTRALIA

ABSTRACT

Given data in a matrix $\boldsymbol{X}$ in which rows represent vectors and columns comprise a mix of discrete and continuous variables, the method presented in this paper can be used to generate random vectors whose elements display the same marginal distributions and correlations as the variables in $\boldsymbol{X}$. The data is represented as a bipartite graph consisting of object nodes (representing vectors) and attribute value nodes. Random walk can be used to estimate the distribution of a target variable conditioned on the remaining variables, allowing a random value to be drawn for that variable. This leads to the use of Gibbs sampling to generate entire vectors. Unlike conventional methods, the proposed method requires neither the joint distribution nor the correlations to be specified, learned, or modeled explicitly in any way. Application to the Australian Credit dataset demonstrates the feasibility of the approach in generating random vectors on challenging real-world datasets.

1 INTRODUCTION

Often the need arises to generate correlated multivariate vectors which include both continuous and discrete variables. For example, simulation studies monitoring the quality of some product or service would typically deal with a number of quantities with correlated continuous and discrete characteristics. The problem addressed in this paper can be expressed as follows: given the data in some matrix $\boldsymbol{X}$, where rows represent vectors and columns comprise a mixture of discrete and continuous variables, how can we generate random vectors which are distributed just like the vectors in $\boldsymbol{X}$, where by "just like" we mean that the generated vectors should have the same marginal distributions as those in $\boldsymbol{X}$, and display the same correlations as the data in $\boldsymbol{X}$.

Various methods have been proposed for random vector generation. Many of these assume that the full joint distribution is known. (In our problem setting it would need to be estimated). For example, under the approach proposed by Johnson (1987), to generate a random vector $\boldsymbol{x} = (x_1,...,x_D)^T$, first generate x_1 from $F_1(x_1)$, the cumulative marginal distribution for x_1. Then generate x_p, $p = 2,...,D$, from the cumulative conditional distribution $F_p(x_p \mid x_1,...,x_{p-1})$. Even if the joint distribution is known, the cumulative conditional distribution functions may not be easy to derive (Law and Kelton 2000), and consequently the approach is typically applied only in situations involving a small number of variables with similar distributions (Niederreiter 1992).

Under the acceptance/rejection sampling approach (Devroye 1986, Johnson 1987), a joint probability density function (pdf) that dominates the original joint pdf is first selected. A random vector is then generated from the dominating pdf, and is either accepted or rejected based on whether or not it falls

between the two functions. As well as also requiring the joint distribution to be known, the method is sensitive to dimensionality, and typically becomes inefficient as the number of variables increases (Gilks and Wild 1992, Gentle 2002).

The NORmal-To-Anything (NORTA) method, detailed by Cario and Nelson (1997) but based on earlier work by Mardia (1970) and Li and Hammond (1975), requires only partially specified properties (a set of marginal distributions and a correlation matrix), but not the full distribution. A D-dimensional standard normal random vector $\boldsymbol{z} = (z_1, z_1, ..., z_D)^T$ with $z_i \sim N(0,1)$ and correlations $\rho_{ij}(\boldsymbol{z}) = Cor(z_i, z_j)$ is first generated. This vector is then transformed into a uniform random vector $\Phi(z_i) \sim U(0,1)$, where Φ is the standard normal cumulative distribution function. Finally, an inverse transformation is used to transform this uniform vector into a vector with the target marginal distributions; i.e., $x_i = F_i^{-1}(\Phi(z_i))$. While the approach is completely general, the difficulty is in finding the correlation matrix of the normal random vector such that it guarantees that the final random vectors will have the desired correlation matrix (Ghosh and Henderson 2003).

Closely related to the NORTA method is the generation of random vectors with *copulas*, where a copula is a multivariate probability distribution for which the marginal probability distribution of each variable is uniform. The usual approach is to first specify the marginal distributions, and then select an appropriate copula that is able to determine the dependence structure between the marginals (Hörmann, Leydold, and Derflinger 2004; Joe 1997; Nelsen 2007), however the copula is sometimes fitted directly to the data (Strelen and Nassaj 2007, Channouf and L'Ecuyer 2009). Vectors can then be generated from the copula, and transformed back to the original space using the inverse transformation, as per the NORTA method. Channouf and L'Ecuyer (2009) note that capturing the dependence can often be more important than capturing the correct marginals, but Blum, Dias and Embrechts (2002) note that fitting a family of copulas to a sample is often as difficult as estimating the joint distribution in the first place. Where the technique has been applied to mixed-attribute datasets, the dimensionality has usually been very small; for example, Channouf and L'Ecuyer (2009) use mixed datasets with only one continuous and one discrete variable, and de Leon and Wu (2011) use a dataset with two continuous variables and one discrete variable.

Another line of research appears within the literature on probabilistic graphical models (Jordan 2004, Koller and Friedman 2009), in which a graph is used to denote the conditional dependence structure between random variables, with nodes in the graph representing random variables, and the pattern of edges representing the dependencies. If the edge structure is known, sampling from directed acyclic graphs consisting only of discrete variables is relatively straightforward, and can be performed using ancestral sampling (Bishop 2006). In the case of mixed graphical models, assumptions are normally made regarding the conditional distributions of continuous variables; for example, Brewer, Aitken and Talbot (1996) use a Gibbs sampling procedure on mixed graphical models in which continuous variables have conditional Normal distributions (later extended to Gamma distributions (Brewer 2000)) where the mean and variance depend on the discrete and continuous parents. If the edge structure of the graph is not known, then it has to be learned. While most work in structure learning has focused on the learning of either discrete or Gaussian graphical models, Lee and Hastie (2013) have recently described a generalization of this to mixed models in which the conditional distributions for continuous and discrete variables are given respectively by Gaussian linear regression and multiclass logistic regressions.

The contribution of this paper is a novel graph-based method for the generation of random mixed-attribute vectors. The data in matrix $\boldsymbol{X}$ is represented as a bipartite graph consisting of object nodes (corresponding to rows of $\boldsymbol{X}$) and attribute value nodes (corresponding to values that attributes may take). Random walk on the graph can be used to estimate the distribution of any discrete or continuous target variable conditioned on the values of the remaining variables, thus allowing a random value to be sampled for the target variable. This leads to the use of a Gibbs sampling procedure to sample entire vectors.

We emphasize that while the method is graph-based, it is fundamentally different from probabilistic graphical models. In the proposed method the actual *data* is represented in the graph, whereas in the case of probabilistic graphical models the graph represents the conditional dependence structure between variables (nodes). Unlike both the analytic method proposed by Johnson (1987) and the

acceptance/rejection sampling approach, the proposed method does not require the full joint distribution to be specified or modelled explicitly in any way, and unlike the NORTA, copula and graphical modeling approaches, the method does not require the correlations or dependencies to be specified, learned, or modeled explicitly in any way. Rather, any modelling of the distribution is entirely implicit in, and inseparable from, the combination of the graph-based representation and the random walk that forms the basis of the Gibbs sampling procedure. This allows the method to reliably generate random vectors from challenging real-world mixed-attribute datasets such as the Australian Credit dataset, which contains 690 examples described over 10 discrete variables containing between 2 and 14 values, and 6 highly skewed continuous variables.

The paper is structured as follows. Section 2 describes the method in the limited setting of only discrete variables. Section 3 then extends this to include continuous variables. Section 4 presents results of applying the method to generating random vectors for the Australian Credit dataset. Section 5 concludes the paper.

2 GENERATING RANDOM VECTORS FROM DISCRETE DATA

2.1 Graph Representation

Assume a matrix $\boldsymbol{X}$ with N rows, representing vectors, and D columns, representing variables. Each D-dimensional row vector is denoted by $\boldsymbol{x}_n = (x_{n1},...,x_{nD})$, and each entry is denoted by x_{nd}. Since we are assuming that the attributes are discrete, x_{nd} takes values from a finite set $x_{nd} \in \{a_d^j\}_{j=1}^{N_d}$, where a_d^j is the j^{th} possible value for attribute d, and N_d is the total number of possible values for attribute d.

The graphs used to represent datasets contain two types of nodes: *object nodes*, which represent the objects corresponding to the rows of $\boldsymbol{X}$, and *attribute value nodes*, which represent an attribute value possessed by the objects. The graphs are bipartite, meaning that edges exist only between object nodes and attribute-value nodes, but not between nodes of the same type. More formally, $G = (O, A, E)$ where $O = \{o_1,...,o_N\}$ is the set of object nodes, $A = \{a_1^j\}_{j=1}^{N_1} \cup \{a_2^j\}_{j=1}^{N_2} \cup ... \cup \{a_D^j\}_{j=1}^{N_D}$ is the set of attribute value nodes, and E is the set of edges. The edge between object node o_i and attribute value node a_j^k has a weighting of 1 if $x_{ij} = a_j^k$ (i.e., if object i possesses the k^{th} value for attribute j), and 0 otherwise.

To illustrate, consider a Play Tennis dataset with attributes *Temperature* ('hot', 'mild', 'cool'), *Humidity* ('high', 'normal', 'low'), *Outlook* ('sunny', 'overcast', 'rainy'), and *Play* ('yes', 'no'). The graph in Figure 1 contains data for two objects: Monday (Temp = 'hot', Humidity = 'low', Outlook = 'sunny', Play = 'yes') and Tuesday (Temp = 'mild', Humidity = 'high', Outlook = 'rainy', Play = 'no').

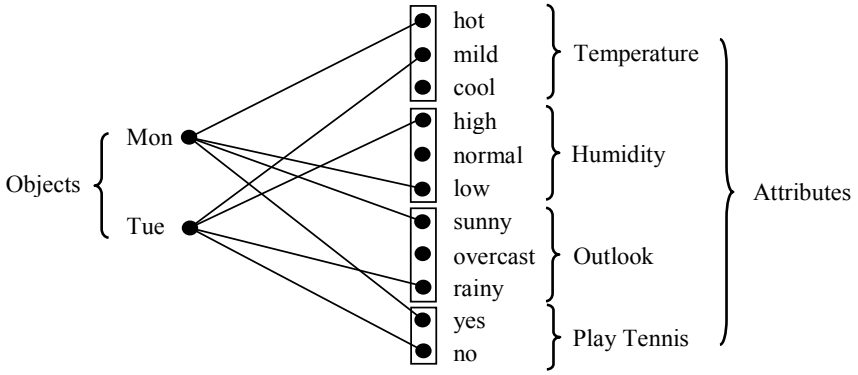

Figure 1: Bipartite graph showing object nodes and attribute value nodes. An edge of weight 1 connects an object node with nodes corresponding to attribute values possessed by the object.

So that random walk can be performed, it is convenient to represent the graph as a matrix. In order to do this we first construct the object-attribute table as shown in Table 1. We denote the matrix of values in the table as $\boldsymbol{B}$. Matrix $\boldsymbol{B}$ has N rows and $|A|$ columns, where $|A|$ is the cardinality of the set A.

Table 1: Object-attribute table for graph in Figure 1.

	Temp			Humidity			Outlook			Play	
	'h'	'm'	'c'	'h'	'n'	'l'	's'	'o'	'r'	'y'	'n'
Mon	1	0	0	0	0	1	1	0	0	1	0
Tue	0	1	0	1	0	0	0	0	1	0	1

From $\boldsymbol{B}$ we then construct the adjacency matrix $\boldsymbol{W}$:

$$\boldsymbol{W} = \begin{bmatrix} \mathbf{0} & \boldsymbol{B} \\ \boldsymbol{B}^{\boldsymbol{T}} & \mathbf{0} \end{bmatrix}.$$

Matrix $\boldsymbol{W}$ is a square matrix with $N + |A|$ rows and columns. The first N rows and columns represent objects, and the remaining $|A|$ rows and columns represent attribute values. The blocks of zeros in the upper left and lower right reflect the bipartite nature of the graph.

2.2 Random Walks and Graph Centrality

The concept of random walk is important, because the amount of time a random walker spends visiting some node on a graph provides a measure of the relative importance of that node. The probability of moving from node i to node j is calculated by dividing the edge weight w_{ij} by the sum of the weights of all of i's outgoing edges:

$$p_{ij} = \frac{w_{ij}}{\sum_{k \in N(i)} w_{ik}}$$

where $N(i)$ denotes the nodes that are the immediate neighbors of i. The larger the value of an outgoing weight, the larger is the probability of traversing that edge as opposed to the other outgoing edges.

The relative importance, or *centrality*, of nodes can be determined by finding the stationary distribution of the graph. For a graph with adjacency matrix $\boldsymbol{W} = \{w_{ij}\}$, the stationary distribution can be found by solving the eigenvector equation

$$\boldsymbol{c} = e\boldsymbol{W}\boldsymbol{c} + (1-e)\mathbf{1}/N \tag{1}$$

in which case the dominant eigenvector $\boldsymbol{c}$ will represent the stationary distribution (Newman 2010). We will refer to vector $\boldsymbol{c}$ as the *centrality vector*, and its components as *centrality scores*. Note that the particular form of Eigenvector Centrality expressed in (1) is known as *Katz Centrality* (Katz 1953), and differs from PageRank (Brin and Page 1998) centrality in that it does not factor in an influence penalty for nodes with a large number of ingoing/outgoing connections; i.e., the matrix $\boldsymbol{W}$ is not row normalized. The second term on the right hand side of (1) can be thought of as the probability that at each step the walker is teleported with probability $(1-e)$ to a random node rather than following an edge (typically $e \cong 0.85$). This solves the problem of the walker getting stuck in disconnected parts of the graph, and means that the walker has a finite chance of reaching any node from any other node.

An important variation of this idea is to allow specification of where the walker should be teleported to. To specify that the walker be teleported to some particular node we can simply modify the equation to

$$c = e\boldsymbol{W}\boldsymbol{c} + (1-e)\boldsymbol{\theta}/|\boldsymbol{\theta}|$$

where $\boldsymbol{\theta}$ is a vector in which $\theta_i = 1$ for the node to which the walker is to be teleported, and 0 for all other nodes. An alternative way of conceptualizing this is to think of the vector $\boldsymbol{\theta}$ as specifying the source of the walk. When used in conjunction with PageRank this is sometimes referred to as *Personalized PageRank* (Jeh and Widom 2003). Note, however, that this idea is quite general. The vector $\boldsymbol{\theta}$ can be set to any weighted combination of nodes, and is crucial to the Gibbs sampling procedure to be introduced shortly. Henceforth we will refer to the vector $\boldsymbol{\theta}$ as the *personalization vector*, and the ensuing walk as *personalized random walk*.

In order to find the dominant eigenvector $\boldsymbol{c}$, an eigenvector equation needs to be solved. A general and robust approach is *power iteration*, which begins with a random vector $\boldsymbol{c}_k$, and iterates the step $\boldsymbol{c}_{k+1} = e\boldsymbol{W}\boldsymbol{c}_k + (1-e)\boldsymbol{\theta}/|\boldsymbol{\theta}|$ until convergence, when $\boldsymbol{c}$ will be the dominant eigenvector (Newman 2010).

2.3 On the Utility of Random Walk

Personalized random walk allows us to answer a number of questions. For example we may be interested in knowing which objects are most similar to some given object, in which case we could set the personalization vector to the node corresponding to the object of interest, perform random walk, and then rank objects according to their centrality scores. Or we may have an object for which one of the attribute values is missing, and wish to impute a value for that attribute. In this case we could again set the personalization vector to the object of interest, perform random walk, and then impute the value with highest conditional probability. (How the conditional probabilities can be calculated from the centrality values is explained in Section 2.4). Or further still, instead of imputing the most likely value for some missing attribute, we may have reason to sample a random value for that attribute, again based on the centrality scores corresponding to that attribute.

While the examples in the preceding paragraph involved investigating the properties of *given* objects (i.e., objects whose properties were specified in the supplied data), we can also investigate properties of hypothetical objects in the attribute space. For example, suppose that we are presented with a new object, and wish to determine its class. (This is really just a special case of missing value imputation, since class is just an attribute). Importantly, we do not need to introduce a new node to represent this novel object; we can simply set the personalization vector to represent the known attribute values of this object, and then perform missing value imputation as described above. These ideas form the basis of our random-walk based Gibbs sampling procedure.

2.4 Gibbs Sampling Procedure

Gibbs sampling (Geman and Geman 1984) is a Markov Chain Monte Carlo (MCMC) technique that is applicable in a wide range of sampling problems. Suppose that $p(\boldsymbol{x}) = p(x_1,...,x_D)$ is a distribution from which we wish to sample. At each step of the Gibbs sampling procedure, the value of one of the variables, say x_i, is replaced by a new value, drawn from the distribution of x_i conditioned on the values of the remaining variables; that is, x_i is replaced by a value drawn from $p(x_i|\boldsymbol{x}_{\setminus i})$, where $\boldsymbol{x}_{\setminus i}$ denotes $x_1,...,x_D$, but with x_i omitted. This procedure is then repeated by cycling through the remaining variables. The Gibbs sampling procedure is described in Algorithm 1.

ALGORITHM 1. Gibbs Sampling Procedure

Initialize $\{x_i : i = 1, \ldots, D\}$

for $\tau = 1,...,T$

Sample $x_1^{(\tau+1)} \sim p(x_1 | x_2^{(\tau)}, x_3^{(\tau)}, ..., x_D^{(\tau)})$.

Sample $x_2^{(\tau+1)} \sim p(x_2 | x_1^{(\tau+1)}, x_3^{(\tau)}, ..., x_D^{(\tau)})$.

$\vdots$

Sample $x_j^{(\tau+1)} \sim p(x_j | x_1^{(\tau+1)}, ..., x_{j-1}^{(\tau+1)}, x_{j+1}^{(\tau)}, ..., x_D^{(\tau)})$.

Sample $x_D^{(\tau+1)} \sim p(x_D | x_1^{(\tau+1)}, x_2^{(\tau+1)}, ..., x_{D-1}^{(\tau+1)})$.

end for

In order to draw a value from the conditional distribution $p(x_i|\boldsymbol{x}_{\setminus i})$, we construct an appropriate personalization vector, perform random walk using this personalization vector, and finally select a value for x_i on the basis of the resulting centrality values.

The personalization vector $\boldsymbol{P}$ is the concatenation of an *object personalization vector* $\boldsymbol{P}_{obj}$ and an *attribute personalization vector* $\boldsymbol{P}_{att}$. We write this concatenation as $\boldsymbol{P} = \boldsymbol{P}_{obj} \hat{} \boldsymbol{P}_{att}$. The vector $\boldsymbol{P}_{obj}$ is the zero vector of length N (i.e., $\boldsymbol{P}_{obj} = \boldsymbol{0}_{N\times 1}$), and vector $\boldsymbol{P}_{att}$ is defined as $\boldsymbol{P} = \boldsymbol{P}_{att_1} \hat{} \boldsymbol{P}_{att_2} \hat{} \ldots \hat{} \boldsymbol{P}_{att_D}$, where $\boldsymbol{P}_{att_d} = (p_d^1, ..., p_d^{N_d})^T$ and $p_d^j = 1$ if $x_d = a_d^j \ (d \neq i)$, and 0 otherwise. Put simply, the personalization vector has a value of 1 corresponding to each attribute value possessed by $\boldsymbol{x}$ (with the exception of the variable for

which we intend to draw a sample), and zeros elsewhere. Random walk is then performed by solving the eigenvector equation $\boldsymbol{c} = e\boldsymbol{W}\boldsymbol{c} + (1-e)\boldsymbol{P}/|\boldsymbol{P}|$.

In order to sample a value for target variable x_i we require an estimate of the conditional distribution $p(x_i \mid \boldsymbol{x}_{\setminus i})$. Let the centrality values for attribute *i*, normalized to sum to 1, be represented by the vector $(c_i^1, ..., c_i^{N_i})^T$, where c_i^j is the centrality value of the node corresponding to the j^{th} value of attribute *i*, and N_i is the number of attribute values for attribute *i*. Also, let the empirical marginal distribution for attribute *i* (i.e., the distribution of attribute *i* in the supplied examples) be represented by the vector $(m_i^1, ..., m_i^{N_i})^T$, where m_i^j is the marginal probability of attribute *i* taking the j^{th} value. The values $m_i^1, ..., m_i^{N_i}$ will sum to 1, and can be calculated directly from the supplied data.

The values in vector $(c_i^1, ..., c_i^{N_i})^T$ can be interpreted as conditional probabilities for variable x_i on the basis that they sum to unity, and reflect the amount of time the walker spends at the node corresponding to each attribute value. However, the values in the vector will depend on a number of factors, including, for example, the value of *e* used in the eigenvector centrality computation. So they should not be interpreted as the single correct probabilities. We estimate the conditional probability that variable x_i takes the j^{th} value of attribute *i* as follows:

$$P(x_i = a_i^j \mid \boldsymbol{x}_{\setminus i}) = K \cdot m_i^j \left(c_i^j / m_i^j \right)^{\alpha} \tag{2}$$

where *K* is a constant selected to ensure that these values sum to 1 across all attribute values; i.e.,

$$K \cdot \sum_{k=1}^{N_i} \left(m_i^k \left(c_i^k / m_i^k \right)^{\alpha} \right) = 1 .$$

The intuition behind (2) is that the greater the divergence of the c_i values from the m_i values, the greater is the dependency of attribute *i* on the values of the other attributes. The ratio c_i^j / m_i^j provides a measure of this divergence, and the parameter α determines the extent to which the centrality values influence the estimate of the conditional probabilities. An α value of 0 will result in the conditional probabilities being set equal to the empirical marginals; increasing the value of α will place greater emphasis on the inter-variable dependencies. This is best illustrated with an example.

Suppose that an attribute can take one of three possible values, and that the empirical marginals and normalized centralities are as follows:

Empirical Marg. Probs (*m*):	[0.500, 0.300, 0.200]
Norm. Centralities (*c*):	[0.470, 0.340, 0.190]

Note that for Attribute 2 the normalized centrality score is higher than the empirical marginal probability, whereas for Attributes 1 and 3 the centrality scores are less than the empirical marginal probabilities. The conditional distributions calculated using (2) for various values of parameter α are:

Cond. Prob. ($\alpha = 0$):	[0.500, 0.300, 0.200]
Cond. Prob. ($\alpha = 2$):	[0.439, 0.382, 0.179]
Cond. Prob. ($\alpha = 5$):	[0.339, 0.518, 0.143]
Cond. Prob. ($\alpha = 10$):	[0.187, 0.729, 0.083]
Cond. Prob. ($\alpha = 15$):	[0.088, 0.871, 0.041]

When $\alpha = 0$, the conditional probabilities are equal to the empirical marginal probabilities. As α increases the difference between the conditional probabilities and the empirical marginal probabilities is accentuated, with conditional probabilities for Attribute 2 increasing, and those for Attributes 1 and 3 decreasing.

3 EXTENSION TO CONTINUOUS VARIABLES

At first sight it might appear that a continuous attribute could be represented using a single node, with the object's value for that attribute being represented by the weight of the connection between the object node and attribute node; however, this leads to difficulties. To illustrate, consider a random walk on such a graph.

The probability of moving from one node to another depends on the weight of the edge connecting those two nodes. Since object nodes are not directly connected with one another, walk from one object node to another can only proceed via an attribute value node. If two objects have a high value for some continuous attribute, then the probability that walk proceeds through that connection will be higher than what would be the case if the objects both had low values for that same attribute. A general treatment of continuous attributes must allow for the case that the implicit similarity between two objects might be a result of their sharing *similar*, but necessarily *high*, values for some numeric attribute.

To solve this problem we use a distributed representation by which continuous attributes are discretized into a finite number of bins, each corresponding to a unique attribute value node. For example, values taken by the continuous attribute *Temperature* might be discretized into categories of *low*, *medium* or *high*, and then represented as per the scheme for discrete attributes described above. However, while this scheme comes some way towards solving the problem (since random walk can now proceed through any of the three nodes corresponding to that attribute), two problems remain: (i) information can be lost at the boundaries (e.g., two objects may have similar values for the attribute, but one may be discretized to *low*, the other to *medium*), and (ii) sampling a value for the variable will result in a discrete value, whereas we wish to generate continuous values. To solve the first problem, we discretize in such a way as to allow partial membership to the categories, as shown in Figure 2. Note that prior to computing membership values, it is assumed that continuous attributes have been normalized to the interval [0,1].

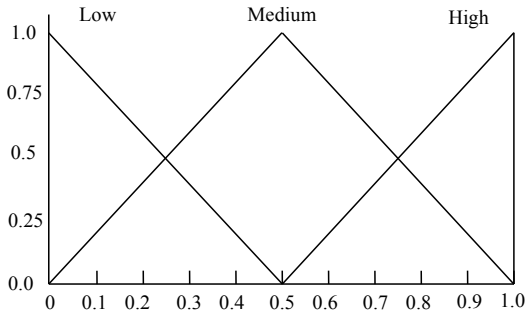

Figure 2: Membership functions for the distributed representation of continuous attributes.

We illustrate with a revised version of the Play Tennis dataset in which the attributes for *Temperature* and *Humidity* are now continuous, and each in the range [0, 100]. Consider the following objects: Monday (Temp = 80, Humidity = 75, Outlook = 'sunny', Play = 'yes') and Tuesday (Temp = 50, Humidity = 40, Outlook = 'rainy', Play = 'no'). Discretizing as described above results in the object-attribute table shown in Table 2, from which we can then construct the adjacency matrix $\boldsymbol{W}$ exactly as described in Section 2.1.

Table 2: Object-Attribute table for mixed Play Tennis dataset.

	Temp			Humidity			Outlook			Play	
	'h'	'm'	'l'	'h'	'm'	'l'	's'	'o'	'r'	'y'	'n'
Mon	0.6	0.4	0.0	0.5	0.5	0.0	1	0	0	1	0
Tue	0.0	1.0	0.0	0.0	0.8	0.2	0	0	1	0	1

There are several points to note. Firstly, there is no loss of information in discretizing continuous variables in this way, since the original value can be retrieved exactly from the distributed representation. Secondly, the representation maintains an equal balance between the contribution of discrete and continuous variables in calculating centrality values; i.e., for each discrete variable the sum of weights connecting an object node to its attribute value nodes will be one, and in the case of continuous variables the sum of outgoing values for each attribute will also be one. Thirdly, the representation allows the random walk procedure described in the previous section to be maintained unchanged.

3.1 Sampling Continuous Variables

In order to sample a value for the continuous-valued target variable x_i we again require an estimate of the conditional distribution $p(x_i|\boldsymbol{x}_{\setminus i})$. As for discrete variables, the personalization vector $\boldsymbol{P}$ is again defined as $\boldsymbol{P} = \boldsymbol{P}_{obj} \hat{} \boldsymbol{P}_{att}$ where $\boldsymbol{P} = \boldsymbol{P}_{att_1} \hat{} \boldsymbol{P}_{att_2} \hat{} \ldots \hat{} \boldsymbol{P}_{att_D}$, however, if attribute d is continuous, then $\boldsymbol{P}_{att_d} = (p_d^l, p_d^m, p_d^h)^T$, where p_d^l, p_d^m and p_d^h are the membership values for x_d.

We perform random walk, and let $(c_i^l, c_i^m, c_i^h)^T$ be the vector of centrality values (normalized to sum to 1) for the three nodes used in the distributed representation of x_i. We then convert from this distributed representation back to a continuous value as follows, treating the resulting value as the expected value of x_i:

$$E(x_i) = (0.0,\ 0.5,\ 1.0) \times (c_i^l, c_i^m, c_i^h)^T$$

We draw a sample for x_i based on some distribution around this expected value. Since the distribution of the random variable x_i is limited to an interval of finite length, a suitable distribution is the Beta distribution, which is defined on the interval [0, 1], and whose shape is determined by parameters p and q.

$$\text{Sample } x_i \sim \beta eta(p,q)$$

The mean of the Beta distribution is $p/(p+q)$, and the width is controlled by $p+q$. Figure 3 shows Beta distributions with means 0.1, 0.5 and 0.8 for three different widths. Note that the larger the value of $p + q$, the narrower is the distribution.

It is convenient to introduce a parameter β to represent the width of the distribution (i.e., $\beta = p + q$). Note that distribution parameters p and q can be determined since the mean of the distribution $p/(p+q)$ will equal the expected value $E(x_i)$, and the value of β will be specified. The parameter β can be considered analogous to the parameter α used in the case of discrete attributes.

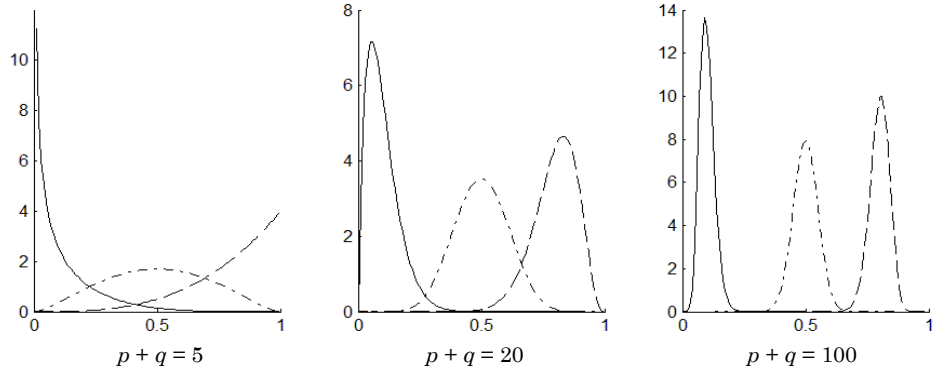

Figure 3: Beta distributions with means 0.1, 0.5 and 0.8 for different values of $p+q$.

4 CASE STUDY: THE AUSTRALIAN CREDIT DATASET

The Australian Credit dataset **(**Quinlan 1987) contains 690 examples, described over 16 attributes (10 discrete and 6 continuous). The discrete variables take between 2 and 14 values, and the continuous attributes are highly skewed. The dataset has traditionally been used as a classification dataset, with 15 of the attributes being used as input, and a binary attribute used as the class variable whose value is to be predicted. In this section we provide results of generating random 16-dimensional vectors for this dataset.

In order for the set of generated vectors to be considered to be from the same distribution as the set of 690 supplied vectors we require that: (i) the marginal distributions of variables in each set be the same, and (ii) that the correlations between variables be the same. While comparison of marginal distributions is relatively straightforward, direct comparison of correlations between 16 variables is difficult due to the number of combinations (120 correlations at just the pairwise level), and this is compounded by the difficulty in measuring correlations between continuous and discrete variables.

As a surrogate to the direct comparison of correlations, we propose the following procedure: (1) construct a training set consisting of all the generated vectors, and a test set consisting of all the supplied

vectors; (2) train a classifier using the examples in the training set; (3) test the performance of the classifier on the test set. If the generated vectors (training set) have not successfully captured the dependency structure of the original data, then classifier performance on the supplied vectors (test set) should be poor; by contraposition, successful classification on the supplied vectors will support the hypothesis that the generated examples (training set) reflect the correlation structure of the original data.

Figure 4 shows training and test accuracies, averaged over 100 trials, of a logistic regression classifier trained on 10,000 randomly generated vectors and tested on the 690 supplied examples. (Logistic regression was chosen on the grounds that it is much less likely to overfit in comparison to classifiers such as MLPs, and when used without regularization does not require specification of any hyper-parameters that may influence results). Results are shown for values of α ranging from 0 to 20. The value of the parameter β was held constant at 10 (discussed later). Also shown is the value of the leave-one-out cross-validation baseline of 84.9%, obtained by applying conventional leave-one-out cross-validation using only the 690 supplied examples. If the generated vectors are from the same distribution as the supplied examples, then we would expect that the test accuracy of a classifier trained on the generated examples and tested on the supplied examples be in the vicinity of this baseline.

It can be seen from Figure 4 that low values of α yield poor classification performance on both the training (generated) examples and the test (supplied) examples. This is because the α value is too small for the generated vectors to capture the dependencies in the data. (Recall from Section 2.4 that for $\alpha = 0$, the conditional probabilities for discrete attributes are equal to the empirical marginal probabilities, and hence do not capture any of the correlations). As the value of α is increased, classification performance increases for both the training and test examples, but more rapidly for the test examples. At $\alpha = 6$, classification performance on the test examples reaches the baseline, and as α is further increased, accuracy on the test examples plateaus at approximately 86%, while accuracy on the training examples continues to rise.

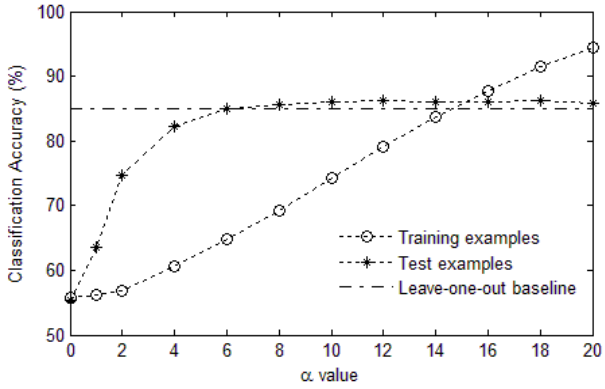

Figure 4: Classification accuracy on Australian Credit dataset using Logistic Regression classifier trained on generated examples and tested on supplied examples ($\beta = 10$).

The observation that training accuracy is lower than test accuracy for smaller values of α can be explained using the simple 1-dimensional example in Figure 5. Each plot shows the same five positive and fives negative examples, described over one continuous input attribute. The curves represent estimates of the distributions of these examples. Also shown is the decision boundary; all examples to the left of the decision boundary would be classified as positive, and those to the right as negative. Consider drawing samples from each of the estimated distributions, and then training a classifier using these samples. If the distributions from which the samples are drawn are very broad (as in the figure on the left), there will be a large amount of overlap between regions of the input space occupied by positive and negative training examples, and hence the accuracy on the training examples will be lower than that on the ten test examples. Conversely, if the estimated distributions are narrow, then there will be little or no overlap between positive and negative training examples, and thus training accuracy will exceed test accuracy. The parameter α in our model has the same effect: low values of α result in the generated samples being more broadly distributed around the supplied examples; high α values result in more compact estimates.

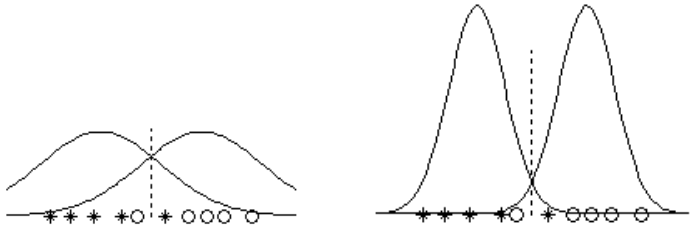

Figure 5: 1D classification example. Dotted line represents decision boundary.

While the classification experiment on the Australian Credit dataset was performed as a surrogate to the direct comparison of correlations, from a machine learning perspective it is quite remarkable that a classifier trained entirely on randomly generated data is able to achieve classification performance superior to that of using leave-one-out cross-validation in the conventional supervised learning setting. Machine learning practitioners familiar with the Australian Credit dataset will know that a classification accuracy of 86% is at the very upper limits of performance achievable on this dataset, yet we have been able to achieve this accuracy through training using only randomly generated examples. This leads us to conclude that the correlations and dependencies in the original vectors have indeed been captured in the generated vectors.

We now compare the marginal distributions between the 690 supplied vectors and a sample of 10,000 generated vectors. Figure 6 shows the marginal distributions for supplied and generated vectors corresponding to $\alpha = 6$ and $\beta = 10$. For all continuous variables (i.e., Variables 2, 3, 8, 11, 14 and 15), the highly skewed nature of the distributions is well-captured by the generated examples. For categorical variables the distributions are also very similar, even for variables such as 6 and 7, which can take a large number of possible values.

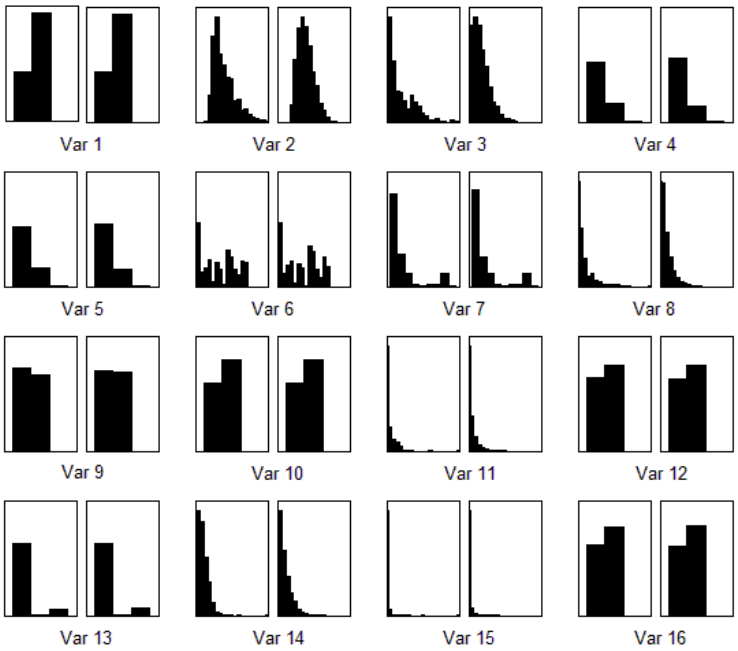

Figure 6: Marginal distributions for supplied (left) and generated (right) vectors of Australian Credit dataset.

Finally we consider the effect of the parameter β, which controls the width of the Beta distribution from which continuous attributes are sampled as part of the Gibbs sampling procedure. Figure 7 compares the marginal distribution of Variable 2 for the supplied examples with the marginal distribution resulting from different values of β. Small values of β (corresponding to broad Beta distributions) result in broad marginal distributions; large values of β result in narrower distributions. A β value of 10 gives the best match.

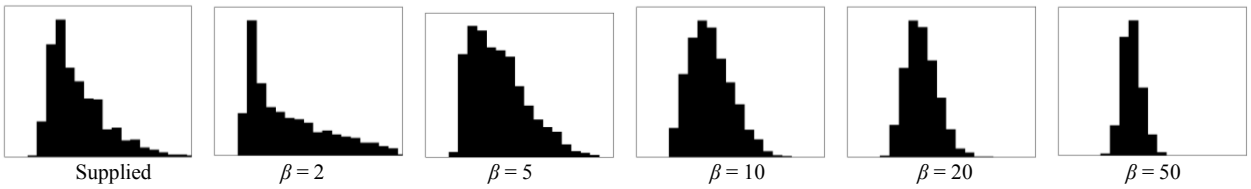

Figure 7: Effect of β on marginal distribution for continuous variable Var 2 ($\alpha = 6$).

5 CONCLUDING REMARKS

The paper has presented a new approach for the random generation of correlated vectors based on data contained in a set of supplied vectors. Whereas most methods decompose the problem into two separate steps—a modelling step in which the joint distribution is estimated, and a generation step in which a random vector is generated—the proposed method does not involve any explicit modelling of the joint distribution. Rather, any modelling of the distribution is entirely implicit in, and inseparable from, the combination of the graph-based representation and the random walk that forms the basis of the Gibbs sampling procedure.

The method is controlled by two parameters—one for discrete variables (α), and one for continuous variables (β). These need to be selected to ensure that the marginal distributions and the correlation structure are correct. Parameter β can be set by comparing the marginals of continuous variables in the generated vectors with those in the supplied examples. To set α, we suggest commencing with a small value, and increasing α while monitoring the marginal distributions and correlations. For low-dimensional datasets it may be possible to monitor pairwise correlations directly; for higher-dimensional datasets it may be more convenient to use a surrogate (e.g., classification performance). Since α and β jointly affect performance, these cannot be set independently, and some tuning may be required.

Computationally, the method is linear in both the number of vectors generated and the dimensionality of the dataset, with a single random walk required for every attribute value sampled. The convergence time for the random walk will depend on the size of the graph.

The method can also be used to generate vectors from any marginal or conditional distribution. To generate vectors from a marginal distribution, simply generate vectors as described above (i.e., over the full joint distribution), but keep only the variables of interest. To generate vectors from conditional distributions, simply clamp the variables to be conditioned on, sampling only the remaining variables.

There are many potential uses for the method, the most obvious being for generating data for simulation studies. In the context of machine learning, it can be applied to dataset expansion for supervised learning. A special case of this may be in addressing the class imbalance problem, in which case we can generate vectors from the conditional distribution of the underrepresented class. The method might also be used to generate entirely synthetic data, in which case any number of vectors can be randomly generated based on a relatively small number of manually created examples.

REFERENCES

Bishop, C. M. 2006. *Pattern Recognition and Machine Learning*. New York: Springer-Verlag.

Blum, P., A. Dias, and P. Embrechts. 2002. "The ART of Dependence Modelling: the Latest Advances in Correlation Analysis." In *Alternative Risk Strategies*, edited by M. Lane, 339–356. London: Risk Waters Group.

Brewer, M. J., C. G. G. Aitken, and M. Talbot. 1996. "A Comparison of Hybrid Strategies for Gibbs Sampling in Mixed Graphical Models." In *Computational Statistics & Data Analysis* 21(3):343–365.

Brewer, M. J. 2000. "A Bayesian Model for Local Smoothing in Kernel Density Estimation." In *Statistics and Computing* 10(4):299–309.

Brin, S., and L. Page, 1998. "The Anatomy of a Large-Scale Hypertextual Web Search Engine." In *Computer Networks and ISDN Systems* 30:107–117.

Cario, M. C., and B. L. Nelson. 1997. "Modeling and Generating Random Vectors with Arbitrary Marginal Distributions and Correlation Matrix." Technical Report, Department of Industrial Engineering and Management Science, Northwestern University, Evanston, IL.

Channouf, N., and P. L'Ecuyer. 2009. "Fitting a Normal Copula for a Multivariate Distribution with both Discrete and Continuous Marginals." In *Proceedings of the 2009 Winter Simulation Conference*, edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, 352–358. Piscataway, New Jersey: Institute of Electrical and Electronic Engineers, Inc.

de Leon, A. R., and B. Wu. 2011. "Copula-Based Regression Models for a Bivariate Mixed Discrete and Continuous Outcome." In *Statistics in Medicine* 30(2):175–185.

Devroye, L. 1986. *Non-uniform Random Variate Generation*, New York: Springer-Verlag.

Geman, S., and D. Geman. 1984. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images." In *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6:721–741.

Gentle, J. E. 2002. *Elements of Computational Statistics*. New York: Springer-Verlag.

Ghosh, S., and S. G. Henderson. 2003. "Behavior of the NORTA Method for Correlated Random Vector Generation as the Dimension Increases." In *ACM Transactions on Modeling and Computer Simulation* 13(3):276–294.

Gilks, W. R., and P. Wild. 1992. "Adaptive Rejection Sampling for Gibbs Sampling." In *Applied Statistics*, 337–348.

Hörmann, W., J. Leydold, and G. Derflinger. 2004. *Automatic Nonuniform Random Variate Generation*. Berlin: Springer.

Jeh, G., and J. Widom. 2003. "Scaling Personalized Web Search." In *Proceedings of the 12th International Conference on World Wide Web*, 271–279, ACM.

Joe, H. 1997. *Multivariate Models and Multivariate Dependence Concepts*. London: Chapman & Hall/CRC.

Johnson, M. E. 1987. *Multivariate Statistical Simulation*. New York: John Wiley & Sons.

Jordan, M. I. 2004. "Graphical Models." In *Statistical Science* 19:140–155.

Katz, L. 1953. "A New Status Index Derived from Sociometric Index." In *Psychometrika* 18:39–43.

Koller, D., and N. Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. Cambridge, MA: MIT press.

Law, A. M., and W. D. Kelton. 2000. *Simulation Modeling and Analysis*. New York: McGraw-Hill, Inc.

Lee, J., and T. Hastie. 2013. "Structure Learning of Mixed Graphical Models." In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, 388–396.

Li, S.T., and J. L. Hammond. 1975. "Generation of Pseudorandom Numbers with Specified Univariate Distributions and Correlation Coefficients." In *IEEE Transactions on Systems, Man and Cybernetics* 5:557–561.

Mardia, K. 1970. "A Translation Family of Bivariate Distributions and Frechet's Bounds." In *Sankhyā: The Indian Journal of Statistics, Series A*:119–122.

Niederreiter, H. 1992. *Random Number Generation and Quasi-Monte Carlo Methods*. Philadelphia: Society for Industrial and Applied Mathematics.

Nelsen, R. B. 2007. *An Introduction to Copulas*. New York: Springer.

Newman, M. 2010. *Networks: an Introduction*. New York: Oxford University Press, Inc.

Quinlan, J. R. 1987. "Simplifying Decision Trees." In *International Journal of Man-Machine Studies* 27(3):221–234.

Strelen, J. C. and F. Nassaj. 2007. "Analysis and Generation of Random Vectors with Copulas." In *Proceedings of the 2007 Winter Simulation Conference*, edited by S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, 488-496. Piscataway, New Jersey: Institute of Electrical and Electronic Engineers, Inc.

AUTHOR BIOGRAPHY

ANDREW A. SKABAR is Senior Lecturer in the School of Engineering and Mathematical Sciences at La Trobe University, Australia. He holds a PhD in Electrical and Electronic Engineering from Queensland University of Technology. His email address is a.skabar@latrobe.edu.au.