A Graph Theoretic Approach to Simulation and Classification

Michael A. Kouritzin[a], Fraser Newton[a,*], Biao Wu[a]

[a] *Department of Mathematical and Statistical Sciences*
University of Alberta, Edmonton, T6G 2G1 CANADA

Abstract

A new class of discrete random fields designed for quick simulation and covariance inference under inhomogenous conditions is introduced and studied. Simulation of these correlated fields can be done in a single pass instead of relying on multi-pass convergent methods like the Gibbs Sampler or other Markov Chain Monte Carlo algorithms. The fields are constructed directly from an undirected graph with specified marginal probability mass functions and covariances between nearby vertices in a manner that makes simulation quite feasible yet maintains the desired properties. Special cases of these correlated fields have been deployed successfully in data authentication, object detection and CAPTCHA[1] generation. Further applications in maximum likelihood estimation and classification such as optical character recognition are now given within.

Keywords: Optical Character Recognition, Random Field, Graph Theory, Spatial Correlation, Simulation

1. Introduction

Random fields are widely used in sciences and technologies to model spatially distributed random phenomena or objects. Within science, random fields

*Corresponding author
Email address: fenewton@ualberta.ca (Fraser Newton)

[1] An acronym for "Completely Automated Public Turing test to tell Computers and Humans Apart" that are widely used to protect online resources from abuse by automated agents.

are used in geophysics, astrophysics, statistical mechanics, underwater acoustics, structural biology and agriculture. Applications of random fields in technologies include TV signal processing, image processing in photography such as medical images (human brain imaging, functional magnetic resonance imaging, mammography), computer vision, web data extraction, clustering gene expression time series, natural language processing etc. Readers are referred to Ashburner et al. (2003), Chellappa and Jain (1993), Li et al. (2008), Li et al. (1995), Li (1995), Winkler (2003), Worsley (1995), Zhang et al. (2001), and Zhu et al. (2008) for those applications. Technologically, researchers of random fields have dealt either with the modeling of images (for synthesis, recognition or compression purposes) or with the resolution of various spatial inverse problems (image restoration and reconstruction, deblurring, classification, segmentation, data fusion, optical flow estimation, optical character recognition, stereo matching, finger print classification, pattern recognition, face recognition, intelligent video surveillance, sparse signal recovery, natural language processing like Chinese chunk and so on, see Blue et al. (1993), Chellappa et al. (1995), Li (1995), Sun et al. (2008), and Winkler (2003)).

Scientists and technicians are interested in the inverse problems such as image restoration, boundary detection, tomographic reconstruction, shape detection from shading, and motion analysis. Many precisely formulated mathematical models were constructed to model certain types of random fields, and various methods and estimators have been developed to make the proposed models work in application. There are diverse needs calling for simulating random fields. For example, simulation is employed to calculate minimum mean square (MMS) and maximum posterior marginal (MPM) estimators, see Winkler (2003). Simulation can also be a smoothing technique. In chapter 2 of Winkler (2003), various smoothing techniques were proposed to clean "dirty" pictures. Most of these methods involve simulation. The difficult problem is how to simulate random fields effectively. A typical simulation would involve many correlated random variables, and could easily exceed the capacity of modern computers if one tried to simulate the whole random field from the probability distribution directly.

Researchers frequently resort to imposing discrete Markov assumptions on their random fields to be simulated out of practical need. In this regard, the Gibbs sampler was proposed to ease this simulation difficulty. Briefly speaking, a Gibbs sampler starts with a given *initial configuration* (i.e. potential realization of the random field) or a configuration chosen at random from some initial distribution, and then updates its configuration vertex by vertex based on the local characteristics of the random field. Once all vertices of a configuration are sequentially updated, a **sweep** or a **pass** is finished. A Gibbs sampler usually takes hundreds of sweeps to produce a configuration closely consistent with a given distribution and there are still computational and convergence issues to deal with. Indeed, the number of possible random configurations within a general discrete random field can be enormous and simulation is further complicated when the vertices are correlated with one another. These factors can make Gibbs sampling and other Markov chain Monte Carlo simulation impractical.

In this paper, we propose a graph theoretic construction for simulation and introduce a new class of discrete **correlated random fields** which incorporate given probability mass functions (pmfs) corresponding to vertices in a graph and pairwise covariances corresponding to edges in a graph. These fields are designed with efficient simulation in mind. Proposition 1 on which our fields are based establishes a method to imbed desired covariances and marginal probabilities into a random field while maintaining simulation ease. Indeed, Proposition 1 is a simple means to construct *some* conditional probabilities consistent with given marginal probabilities and covariances in such a way that sampling the missing portion of a random field sequentially is very feasible. More precisely, when simulating a new vertex, we compute this conditional probability of its state conditioned on the known portion and the previously-simulated vertices. We can construct a random field in one pass based on this algorithm. This method is especially suited to problems where pairwise covariance capture the meaningful relationships between variables. (We explore the role of covariances further in Section 4.) For demonstration purposes, we discuss prior applications of our random fields to data authentication, object detection and CAPTCHA

generation, as well as develop new example applications in maximum likelihood estimation and classification. In particular, our experimental results suggest our algorithm may help improve optical character recognition.

The remainder of this note is laid out as follows. Section 2 contains our notation and the statement of our main results, Proposition 1; Section 3 provides several mathematical examples illustrating the method; and in Section 4, we develop new applications in maximum likelihood estimation and optical character recognition.

2. Notation and Background

Our goal is to simulate a random field so that desired properties (in our case, marginal probabilities and pair-wise covariances) are maintained. Specifically, we will give a method for computing conditional probabilities so that these properties are maintained. We begin by describing how the problem is constructed in Section 2.1; then, we will describe exactly how to compute the probabilities in Section 2.2; finally, illustrative examples are given in Section 3.

2.1. Problem Setup

Suppose we are given a desired probability mass function (pmf) for each random variable and a set of desired pairwise covariances for some set of pairs of the random variables. Our goal is to simulate the random variables so that the desired properties are met.

2.1.1. Definitions

We will be working with undirected and directed graphs in the following. We begin by providing the required definitions and notation.

An *undirected graph* $G = (V, E)$ is a set of vertices V and edges E between some of the vertices, where $(u, v) \in E$ if there is an edge between vertices u and v; in this case, u and v are called *neighbors*. A *directed graph* $D = (V, A)$ is a set of vertices V and arcs A from some of the vertices to others, where $(u, v) \in A$ if there is an arc from vertex u to vertex v; in this case, u is called a *parent* of v

and v is called *child* of u. More generally, we would say that v is an ancestor of u if there are a collection of arcs starting at v and going to u such that the first arc starts at v, the last arc ends at u and every arc in between starts where the previous one ends. The real difference between directed and undirected graphs is the former has a direction to its edges.

An undirected graph is called *connected* if there is a path of edges between every pair of vertices. The *open neighborhood* $N_G(v)$ of vertex $v \in V$ is the set of vertices $u \neq v$ such that there is an edge between u and v, i.e., $(u, v) \in E$. We denote the open neighborhood of v by $N_G(v)$ and the closed neighborhood $N_G(v) \cup \{v\}$ by $N_G[v]$. For a set of vertices B, we define the open neighborhood of B as $N_G(B) = \cup_{v \in B} N_G(v) \setminus B$ and closed neighborhood $N_G[B] = N_G(B) \cup B$. For convenience, we set $N_G(\emptyset) = V$.

A directed graph is called *acyclic* if there is no vertex v that is an ancestor of itself.

2.1.2. Setup

We begin by constructing an undirected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges between vertices; there is a vertex for every given random variable and an edge between vertices if there is a given covariance for the corresponding pair of random variables. If the graph G is not connected, we want this procedure to operate over each connected component of G. This will require modified open neighborhoods:

$$M_G(B) = \begin{cases} \cup_{v \in B} N_G(v) \setminus B & \text{if } \cup_{v \in B} N_G(v) \setminus B \neq \emptyset \\ V \setminus B & \text{otherwise} \end{cases},$$

and $M_G(\emptyset) = V$. Note that for non-empty V, $M_G(B) = \emptyset$ if and only if $B = V$.

We construct a directed acyclic graph $D = (V, A)$ from G by traversing G and adding vertices and arcs based on the order of traversal. More precisely, we use the following algorithm.

1. Initialise $V_0 = \emptyset$, $A_0 = \emptyset$, and let $N = |V|$ be the number of vertices in G.
2. For $i = 1, \ldots, N$

(a) Select an arbitrary vertex from $M_G(V_{i-1})$ and denote it as v_i.

(b) Set $V_i = V_{i-1} \cup \{v_i\}$.

(c) Let $A_{v_i} = \{(v_i, u) : (v_i, u) \in E, (u, v_i) \notin A_{i-1}\}$ be the set of arcs from v_i for every undirected edge involving v_i in G which have not yet been used. (Here, (v_i, u) is an arc from v_i to u. Since v_i was not used previously in the algorithm we know $(v_i, u) \notin A_{i-1}$. However, we still check that $(u, v_i) \notin A_{i-1}$ to ensure we do not have a cycle.)

(d) Set $A_i = A_{i-1} \cup A_{v_i}$.

3. Set $A = A_N$. Then $D = (V, A)$ is a directed acyclic graph with the same vertices as G and an arc for each undirected edge in G.

Remark 1. The algorithm adds all vertices from G to D since it iterates N times and always adds a vertex from $M_G(V_{i-1})$, i.e., it never adds the same vertex twice. This also ensures that the resulting graph is acyclic.

Remark 2. Each vertex in D will have a set of parents (u is called a parent of v if there is an arc from u to v), which we will denote by $\mathrm{pa}(v)$. The first vertex added to a connected component of G has no parents; all other vertices have at least one parent.

Remark 3. The order of the vertices $\{v_1, \ldots, v_N\}$ is a topological sort since if $i < j$ and $(v_i, v_j) \in A$, there is an arc from v_i to v_j.

For each vertex $v \in V$, let $\mathbf{X}_v$ be a finite space of states at vertex v. For any nonempty $B \subset V$, denote the space of configurations $x_B = (x_v)_{v \in B}$ on B by Cartesian product $\mathcal{X}_B = \prod_{v \in B} \mathbf{X}_v$. We abbreviate $\mathcal{X}_V$ by $\mathcal{X}$, i.e., $\mathcal{X} = \prod_{v \in V} \mathbf{X}_v$. Finally, we denote our given discrete random variables by $X = (X_v)_{v \in V}$ indexed by V. In addition, we denote the desired pmf for each X_v, $v \in V$ by π_v and the desired covariances between X_u and X_v, $(u, v) \in E$ by $\beta_{u,v}$.

Let Π denote a *probability measure* or *distribution* on $\mathcal{X}$. If for every $x \in \mathcal{X}$, $\Pi(x) > 0$, i.e., Π is a strictly positive probability measure on $\mathcal{X}$, then Π is called a *random field*. We also call the random vector $X = (X_v)_{v \in V}$ on the probability

space $(\mathcal{X}, \Pi)$ a random field. For any nonempty $B \subset V$, define the projection map from $\mathcal{X}$ onto $\mathcal{X}_B$ as follows:

$$X_B : x \to x_B,$$

where $x \in \mathcal{X}$ and $x_B \in \mathcal{X}_B$.

Example 1. Suppose $V = \{(1,1),(1,2),(2,1),(2,2)\}$ is the space over which the random field is defined, $X_v = \{-1,0,1\}$ for all $v \in V$ is the alphabet for the random variables at the sites v and $B = \{(1,1),(2,2)\}$ is the collection of sites of interest. Then, $\mathcal{X} = \{-1,0,1\} \times \{-1,0,1\} \times \{-1,0,1\} \times \{-1,0,1\}$, $\mathcal{X}_B = \{-1,0,1\} \times \{-1,0,1\}$ and

$$x_B \in \{(-1,-1),(-1,0),(-1,1),(0,-1),(0,0),(0,1),(1,-1),(1,0),(1,1)\}.$$

2.2. Method for Computing Conditional Probabilities

Now, we work exclusively with the directed acyclic graph $D = (V, A)$ with N vertices and topological sort $\{v_i\}_{i=1}^N$ and we are ready to assign conditional probabilities.

We write the distribution Π of X as

$$\Pi(X = x) = \prod_{v \in V} \Pi(X_v = x_v | X_{\mathrm{pa}(v)} = x_{\mathrm{pa}(v)}). \tag{1}$$

so we are now working with a Bayesian network. Furthermore, we can find the probability of any set of vertices $B \subset V$. Let j be the maximal element of $\{i : 1 \leq i \leq N, v_i \in B\}$, i.e., v_j is the element of B which is last in the

topological sort; then, by the multiplication rule and (1),

$$\begin{aligned}
&\Pi(X_B = x_B) \\
= &\sum_{x_{v_k}:1\le k\le j, v_k\notin B} \Pi(X_{v_j} = x_{v_j}, X_{v_{j-1}} = x_{v_{j-1}}, \ldots, X_{v_1} = x_{v_1}) \\
= &\sum_{x_{v_k}:1\le k\le j, v_k\notin B} \big(\Pi(X_{v_j} = x_{v_j} | X_{v_{j-1}} = x_{v_{j-1}}, \ldots, X_{v_1} = x_{v_1}) \\
&\quad \times \Pi(X_{v_{j-1}} = x_{v_{j-1}}, \ldots, X_{v_1} = x_{v_1})\big) \\
= &\sum_{x_{v_k}:1\le k\le j, v_k\notin B} \prod_{i=1}^{j} \Pi(X_{v_i} = x_{v_i} | X_{v_{i-1}} = x_{v_{i-1}}, \ldots, X_{v_1} = x_{v_1}) \\
= &\sum_{x_{v_k}:1\le k\le j, v_k\notin B} \prod_{i=1}^{j} \Pi(X_{v_i} = x_{v_i} | X_{\mathrm{pa}(v_i)} = x_{\mathrm{pa}(v_i)}), \qquad (2)
\end{aligned}$$

where we used the convention $\Pi(X_{v_1} = x_{v_1} | X_{v_0} = x_{v_0}, \ldots, X_{v_1} = x_{v_1}) = \Pi(X_{v_1} = x_{v_1})$.

Herein, we simulate random fields with given marginal probabilities for vertices and given covariances between vertices. Our algorithm constructs X with the given marginal probabilities $\{\pi_{v_i}(x_{v_i}) : x_{v_i} \in \mathbf{X}_{v_i}\}_{i=1}^{N}$ and the given covariances between neighboring vertices $\{\beta_{v_i,u} : u \in \mathrm{pa}(v_i)\}$ for $1 \le i \le N$. (It is assumed a priori that these marginal conditions hold within V. It is also assumed that $\beta_{v_i,u} = \beta_{u,v_i}$ for $u \in \mathrm{pa}(v_i)$ $(1 \le i \le N)$ since $\beta_{v_i,u}$ will denote covariance between X_{v_i} and X_u.) Equation (3) (to follow) gives us the means to assign conditional probabilities $\Pi(X_{v_i} = x_{v_i} | X_{\mathrm{pa}(v_i)} = x_{\mathrm{pa}(v_i)})$ such that the random field X (with distribution Π) has the desired covariances $\{\beta_{u,v}\}$ and marginal probabilities $\{\pi_v\}$. There are, of course, many other ways to assign a distribution Π that is consistent with these marginal pmfs and covariances. Typically, in Markov chain Monte Carlo methods one has a particular joint distribution, must re-simulate all site random variables many times, and relies on convergence. This traditional approach is far from the realm of possibility with contemporary computers for the OCR problems discussed in a later section. Our Proposition 1 gives a construction for a collection of distributions with the correct site pmfs and site-to-site covariances that can be used to simulate the distribution with one pass in real time. This single-pass property allows us to

simulate much larger problems than possible using traditional Gibb's sampler or Markov chain Monte Carlo methods where a very large number of passes are usually necessary.

Proposition 1. Assume that $D = (V, A)$ is a directed acyclic graph with N vertices and that $X = (X_v)_{v \in V}$ denote discrete random variables indexed by V. Suppose further that $\{v_i\}_{i=1}^N$ is a topological sort of the vertices V, and $\{\tilde{\pi}_v(x_v) : x_v \in \mathbf{X}_v, v \in V\}$ and $\{\hat{\pi}_v(x_v) : x_v \in \mathbf{X}_v, v \in V\}$ are two sets of auxiliary pmfs. Assume that $\{\pi_v(x_v) : x_v \in \mathbf{X}_v, v \in V\}$ are pmfs and $\{\beta_{u,v} : (u, v) \in A \text{ or } (v, u) \in A\}$ are numbers such that the right hand side of (3) is in [0,1]. Form the conditional probabilities recusively, starting with $i = 1$, as

$$\begin{aligned} \Pi(X_{v_i} = x_{v_i} | X_{\mathrm{pa}(v_i)} = x_{\mathrm{pa}(v_i)}) &= \pi_{v_i}(x_{v_i}) + \frac{g(v_i)}{\Pi(X_{\mathrm{pa}(v_i)} = x_{\mathrm{pa}(v_i)})} \\ &\quad \times \sum_{u \in \mathrm{pa}(v_i)} \beta_{u,v_i} g(u) h(u, v_i) \end{aligned} \tag{3}$$

for each $x_{v_i} \in \mathbf{X}_{v_i}$ and $x_{\mathrm{pa}(v_i)} \in \mathbf{X}_{\mathrm{pa}(v_i)}$ $(1 \leq i \leq N)$, where $\mu_{\tilde{\pi}_v} = \sum_{x_v \in \mathbf{X}_v} \tilde{\pi}_v(x_v) x_v$, $\sigma^2_{\tilde{\pi}_v} = \sum_{x_v \in \mathbf{X}_v} \tilde{\pi}_v(x_v)(x_v - \mu_{\tilde{\pi}_v})^2$, $g(v) = \frac{\tilde{\pi}_v(x_v)(x_v - \mu_{\tilde{\pi}_v})}{\sigma^2_{\tilde{\pi}_v}}$, $h(u, v) = \prod_{w \in \mathrm{pa}(v) \setminus \{u\}} \hat{\pi}_w(x_w)$, and $\Pi(X_{\mathrm{pa}(v_i)} = x_{\mathrm{pa}(v_i)})$ is computed according to (2). Then, the random field X, defined by

$$\Pi(X = x) = \prod_{i=1}^{N} \Pi(X_{v_i} = x_{v_i} | X_{\mathrm{pa}(v_i)} = x_{\mathrm{pa}(v_i)})$$

has marginal probabilities $\{\pi_v\}$ and covariances $\mathrm{cov}(X_u, X_v) = \beta_{u,v}$ for all $u \in \mathrm{pa}(v)$.

Remark 4. Proposition 1 allows us to fully specify the Bayesian network, i.e., we are able to compute $\Pi(X_v = x_v | X_{\mathrm{pa}(v)} = x_{\mathrm{pa}(v)})$ $\forall v \in V, x_v \in \mathbf{X}_v, x_{\mathrm{pa}(v)} \in \mathbf{X}_{\mathrm{pa}(v)}$. This can be used to compute the probability of a particular realization of a network or to simulate realizations of networks. Indeed, while it also provides the conditional distributions required as input to a multi-pass approximate method like Gibbs sampling, it can be used for single-pass simulation to sample directly from the distribution as described in Section 2.3.

Remark 5. The auxiliary pmfs $\{\tilde{\pi}_v(\cdot)\}_{v\in V}$ and $\{\hat{\pi}_v(\cdot)\}_{v\in V}$ that appear in $g(\cdot)$ and $h(\cdot,\cdot)$ respectively can be chosen to give different distributions Π with the same desired marginal pmfs $\{\pi_v(\cdot)\}_{v\in V}$ and covariances $\{\beta_{u,v} : (u,v) \in A \text{ or } (v,u) \in A\}$. In some cases it is possible to take one or both of these auxiliary pmfs to be uniform to ease the compuational burden. This is illustrated in Section 2.4. Naturally, not all collections of site pmfs and site-to-site covariances are consistent. For example, if $V = 1, 2$, $\beta_{1,2} = 1$, $\pi_1 = \delta_0$ and $\pi_2 = \delta_1$, then there can be no joint distribution for this collection. Here, δ_x is the Dirac measure defined as

$$\delta_x(A) = \begin{cases} 0 & \text{if } x \notin A \\ 1 & \text{if } x \in A. \end{cases}$$

Moreover, even when the site marginals and covariances are consistent, there is a possibility that there is no joint distribution that can be constructed according to (3) with given auxiliary pmfs $\{\tilde{\pi}_v(\cdot)\}_{v\in V}$ and $\{\hat{\pi}_v(\cdot)\}_{v\in V}$. The second purpose of having the auxiliary pmfs is to increase the set of joint distributions that can be simulated. This later point will be studied in future work.

Remark 6. In Proposition 1, we assumed that $\pi_v(x_v) > 0 : \forall x_v \in \mathbf{X}_v$ for each $v \in V$. Note that $\mathbf{X}_v$ can be different for each $v \in V$. For given v, if there exists a $x_v \in \mathbf{X}_v$ such that $\pi_v(x_v) = 0$, we may deem it uninteresting and replace $\mathbf{X}_v$ with $\mathbf{X}_v \setminus \{x_v\}$. Therefore the positive probability mass function assumption of π_v is also a convention.

Remark 7. A random field generated by Proposition 1 is a correlated random field. Indeed, one value of this proposition is the assertion that there are correlated random fields that match a given collection of marginal probabilities and covariances.

Proof. It is clear by (3) that for $i = 1$, X_{v_1} has probability distribution $\pi_{v_1}(\cdot)$, since $\text{pa}(v_1) = \emptyset$.

We next prove that for $2 \leq i \leq N$, X_{v_i} has probability distribution $\pi_{v_i}(\cdot)$, and for any $u \in \text{pa}(v_i)$, $cov(X_{v_i}, X_u) = \beta_{v_i,u}$. To ease notation, we suppress the

subscript i. For $x_v \in \mathbf{X}_v$, by (3), one has that

$$\begin{aligned}
\Pi(X_v = x_v) &= \sum_{x_{\mathrm{pa}(v)} \in \mathbf{X}_{\mathrm{pa}(v)}} \Pi(\, X_v = x_v |\, X_{\mathrm{pa}(v)} = x_{\mathrm{pa}(v)}) \Pi(X_{\mathrm{pa}(v)} = x_{\mathrm{pa}(v)}) \\
&= \pi_v(x_v) + g(v) \sum_{u \in \mathrm{pa}(v)} \sum_{x_{\mathrm{pa}(v)} \in \mathbf{X}_{\mathrm{pa}(v)}} h(u,v) \beta_{v,u} g(u) \\
&= \pi_v(x_v) + g(v) \sum_{u \in \mathrm{pa}(v)} \beta_{v,u} \sum_{x_{\mathrm{pa}(v)\setminus\{u\}} \in \mathbf{X}_{\mathrm{pa}(v)\setminus\{u\}}} h(u,v) \sum_{x_u \in \mathbf{X}_u} g(u) \\
&= \pi_v(x_v), \qquad (4)
\end{aligned}$$

since for fixed $u \in \mathrm{pa}(v)$,

$$\sum_{x_u \in \mathbf{X}_u} g(u) = \sum_{x_u \in \mathbf{X}_u} \frac{\tilde{\pi}_u(x_u)(x_u - \mu_{\tilde{\pi}_u})}{\sigma^2_{\tilde{\pi}_u}} = \frac{1}{\sigma^2_{\tilde{\pi}_u}} (\mu_{\tilde{\pi}_u} - \mu_{\tilde{\pi}_u}) = 0.$$

Note that the above sum $\sum_{x_{\mathrm{pa}(v)} \in \mathbf{X}_{\mathrm{pa}(v)}}$ is a multiple sum, containing one summation for each parent.

Now fix $u \in \mathrm{pa}(v)$, we prove $cov(X_v, X_u) = \beta_{v,u}$. We compute the joint probability mass function of X_v and X_u. For $x_v \in \mathbf{X}_v, x_u \in \mathbf{X}_u$, we have that by (3) again

$$\begin{aligned}
&\Pi(X_v = x_v, X_u = x_u) \\
&\quad = \sum_{x_{\mathrm{pa}(v)\setminus\{u\}} \in \mathbf{X}_{\mathrm{pa}(v)\setminus\{u\}}} \Pi(X_v = x_v | X_u = x_u, X_{\mathrm{pa}(v)\setminus\{u\}} = x_{\mathrm{pa}(v)\setminus\{u\}}) \\
&\qquad \times \Pi(X_u = x_u, X_{\mathrm{pa}(v)\setminus\{u\}} = x_{\mathrm{pa}(v)\setminus\{u\}}) \\
&\quad = \pi_v(x_v) \sum_{x_{\mathrm{pa}(v)\setminus\{u\}} \in \mathbf{X}_{\mathrm{pa}(v)\setminus\{u\}}} \Pi(X_u = x_u, X_{\mathrm{pa}(v)\setminus\{u\}} = x_{\mathrm{pa}(v)\setminus\{u\}}) \\
&\qquad + g(v) \sum_{x_{\mathrm{pa}(v)\setminus\{u\}} \in \mathbf{X}_{\mathrm{pa}(v)\setminus\{u\}}} \sum_{w \in \mathrm{pa}(v)} h(w,v) \beta_{v,w} g(w)
\end{aligned}$$

since $\Pi(X_u = x_u, X_{\mathrm{pa}(v)\setminus\{u\}} = x_{\mathrm{pa}(v)\setminus\{u\}}) = \Pi(X_{\mathrm{pa}(v)} = x_{\mathrm{pa}(v)})$ for $u \in \mathrm{pa}(v)$.

Therefore,

$$
\begin{aligned}
&\Pi(X_v = x_v, X_u = x_u) \\
&\quad = \pi_v(x_v)\pi_u(x_u) + g(v) \\
&\qquad \times \sum_{x_{\mathrm{pa}(v)\setminus\{u\}} \in \mathbf{X}_{\mathrm{pa}(v)\setminus\{u\}}} \left[h(u,v)\beta_{v,u} g(u) + \sum_{w \in \mathrm{pa}(v)\setminus\{u\}} h(w,v)\beta_{v,w} g(w) \right] \\
&\quad = \pi_v(x_v)\pi_u(x_u) + g(v)\beta_{v,u} g(u) \prod_{w \in \mathrm{pa}(v)\setminus\{u\}} \left(\sum_{x_w \in \mathbf{X}_w} \hat{\pi}_w(x_w) \right) \\
&\qquad + g(v) \sum_{w \in \mathrm{pa}(v)\setminus\{u\}} \sum_{x_{\mathrm{pa}(v)\setminus\{u\}} \in \mathbf{X}_{\mathrm{pa}(v)\setminus\{u\}}} h(w,v)\beta_{v,w} g(w) \\
&\quad = \pi_v(x_v)\pi_u(x_u) + g(v)\beta_{v,u} g(u) \\
&\qquad + g(v) \sum_{w \in \mathrm{pa}(v)\setminus\{u\}} h(w,v)\beta_{v,w} \sum_{x_{\mathrm{pa}(v)\setminus\{u,w\}} \in \mathbf{X}_{\mathrm{pa}(v)\setminus\{u,w\}}} \sum_{x_w \in \mathbf{X}_w} g(w) \\
&\quad = \pi_v(x_v)\pi_u(x_u) + g(v)\beta_{v,u} g(u).
\end{aligned}
$$

Therefore, since

$$
\begin{aligned}
cov(X_v, X_u) &= cov(X_v - \mu_{\tilde{\pi}_v}, X_u - \mu_{\tilde{\pi}_u}) \\
&= E[(X_v - \mu_{\tilde{\pi}_v})(X_u - \mu_{\tilde{\pi}_u})] - E[X_v - \mu_{\tilde{\pi}_v}]E[X_u - \mu_{\tilde{\pi}_u}]
\end{aligned}
$$

and

$$
\begin{aligned}
E[(X_v - \mu_{\tilde{\pi}_v})(X_u - \mu_{\tilde{\pi}_u})] &= \sum_{x_v \in \mathbf{X}_v} \sum_{x_u \in \mathbf{X}_u} \Big[(x_v - \mu_{\tilde{\pi}_v})(x_u - \mu_{\tilde{\pi}_u}) \\
&\qquad \times \Pi(X_v = x_v, X_u = x_u) \Big] \\
&= \sum_{x_v \in \mathbf{X}_v} \sum_{x_u \in \mathbf{X}_u} \Big[(x_v - \mu_{\tilde{\pi}_v})(x_u - \mu_{\tilde{\pi}_u}) \\
&\qquad \times [\pi_v(x_v)\pi_u(x_u) + g(v)\beta_{v,u} g(u)] \Big] \\
&= \sum_{x_v \in \mathbf{X}_v} \sum_{x_u \in \mathbf{X}_u} (x_v - \mu_{\tilde{\pi}_v})(x_u - \mu_{\tilde{\pi}_u})\pi_v(x_v)\pi_u(x_u) \\
&\qquad + \sum_{x_v \in \mathbf{X}_v} \sum_{x_u \in \mathbf{X}_u} (x_v - \mu_{\tilde{\pi}_v})(x_u - \mu_{\tilde{\pi}_u}) g(v)\beta_{v,u} g(u) \\
&= E[X_v - \mu_{\tilde{\pi}_v}]E[X_u - \mu_{\tilde{\pi}_u}] \\
&\qquad + \beta_{v,u} \sum_{x_v \in \mathbf{X}_v} (x_v - \mu_{\tilde{\pi}_v}) g(v) \sum_{x_u \in \mathbf{X}_u} (x_u - \mu_{\tilde{\pi}_u}) g(u) \\
&= E[X_v - \mu_{\tilde{\pi}_v}]E[X_u - \mu_{\tilde{\pi}_u}] + \beta_{v,u}, \qquad (5)
\end{aligned}
$$

we have that

$$cov(X_v, X_u) = \beta_{v,u} \tag{6}$$

□

2.3. Algorithm for Simulating Random Fields

Now, simulating the random field becomes a simple matter of computing the required probabilities and generating uniform random numbers to select a value for each vertex. For $v \in V$, we denote $\mathbf{X}_v = \{x_v^1, ..., x_v^{d_v}\}$, where $d_v \in \mathbb{N}$ is the cardinality of $\mathbf{X}_v$.

Do for $i = 1, \ldots, N$:

1. Compute $\Pi(\, X_{v_i} = x_{v_i}^j \,\big|\, X_{\mathrm{pa}(v_i)} = x_{\mathrm{pa}(v_i)})$ for $1 \le j \le d_{v_i}$ using (3).
2. Generate a $[0,1]$-uniform random variable U. For the given U, there exists unique $1 \le j \le d_{v_i}$ such that

$$\sum_{u=1}^{j-1} \Pi(\, X_{v_i} = x_{v_i}^u \,\big|\, X_{\mathrm{pa}(v_i)} = x_{\mathrm{pa}(v_i)}) \le U < \sum_{u=1}^{j} \Pi(\, X_{v_i} = x_{v_i}^u \,\big|\, X_{\mathrm{pa}(v_i)} = x_{\mathrm{pa}(v_i)}).$$

   Then set $X_{v_i} = x_{v_i}^j$. For notational convenience, we suppress superscript j and use x_{v_i} to indicate the simulated value $x_{v_i}^j$ of X_{v_i}.

2.4. Special Cases

Proposition 1 has a few interesting and important special cases:

1. If $\hat{\pi}_v(x_v) = \tilde{\pi}_v(x_v), x_v \in \mathbf{X}_v$ for all $v \in V$, then (3) becomes

$$\begin{aligned} \Pi(X_{v_i} = x_{v_i} | X_{\mathrm{pa}(v_i)} = x_{\mathrm{pa}(v_i)}) &= \pi_{v_i}(x_{v_i}) \\ &\quad + \Bigg(\prod_{w \in \mathrm{pa}(v_i) \bigcup \{v_i\}} \tilde{\pi}_w(x_w) \Bigg) \\ &\quad \cdot \sum_{u \in \mathrm{pa}(v_i)} \frac{(x_{v_i} - \mu_{\tilde{\pi}_{v_i}}) \beta_{v_i,u} (x_u - \mu_{\tilde{\pi}_u})}{\sigma^2_{\tilde{\pi}_{v_i}} \sigma^2_{\tilde{\pi}_u} \Pi(X_{\mathrm{pa}(v_i)} = x_{\mathrm{pa}(v_i)})}. \end{aligned} \tag{7}$$

   In (7), two auxiliary collections of pmfs are reduced to one.

2. If $\hat{\pi}_v(x_v) = \tilde{\pi}_v(x_v) = \pi_v(x_v), x_v \in \mathbf{X}_v$ for all $v \in V$, then (3) is

$$\begin{aligned}\Pi(X_{v_i} = x_{v_i} | X_{\mathrm{pa}(v_i)} = x_{\mathrm{pa}(v_i)}) = \\ \times \pi_{v_i}(x_{v_i}) \Bigg[1 + \Bigg(\prod_{w \in \mathrm{pa}(v_i)} \pi_w(x_w) \Bigg) \\ \cdot \sum_{u \in \mathrm{pa}(v_i)} \frac{(x_{v_i} - \mu_{\pi_{v_i}}) \beta_{v_i,u} (x_u - \mu_{\pi_u})}{\sigma^2_{\pi_{v_i}} \sigma^2_{\pi_u} \Pi(X_{\mathrm{pa}(v_i)} = x_{\mathrm{pa}(v_i)})} \Bigg]\end{aligned} \tag{8}$$

   There are no auxiliary collections of pmfs in (8).

3. If $\hat{\pi}_v(x_v) = \tilde{\pi}_v(x_v) = \frac{1}{d_v}, x_v \in \mathbf{X}_v$ for all $v \in V$ where d_v is the cardinality of $\mathbf{X}_v$, then (3) has the following form

$$\begin{aligned}\Pi(X_{v_i} = x_{v_i} | X_{\mathrm{pa}(v_i)} = x_{\mathrm{pa}(v_i)}) = \pi_{v_i}(x_{v_i}) \\ + \sum_{u \in \mathrm{pa}(v_i)} \frac{(x_{v_i} - \bar{\mu}_{v_i}) \beta_{v_i,u} (x_u - \bar{\mu}_u)}{d_{\mathrm{pa}(v_i) \cup \{v_i\}} \bar{\sigma}^2_{v_i} \bar{\sigma}^2_u \Pi(X_{\mathrm{pa}(v_i)} = x_{\mathrm{pa}(v_i)})}\end{aligned} \tag{9}$$

   where $\bar{\mu}_v = \dfrac{1}{d_v} \displaystyle\sum_{x_v \in \mathbf{X}_v} x_v$ and $\bar{\sigma}^2_v = \dfrac{1}{d_v} \displaystyle\sum_{x_v \in \mathbf{X}_v} (x_v - \bar{\mu}_v)^2$ for all $v \in V$ and $d_B = \displaystyle\prod_{w \in B} d_w$. Here two collections of auxiliary pmfs take same discrete uniform pmfs respectively. The simplicity in (9) reduces the computation of conditional probabilities. Notice the $\bar{\mu}_v$ and $\bar{\sigma}^2_v$ calculations are simplified.

4. If $\hat{\pi}_v(x_v) = \frac{1}{d_v}$ and $\tilde{\pi}_v(x_v) = \pi_v(x_v)$, $\forall x_v \in \mathbf{X}_v$ for all $v \in V$ with d_v being the cardinality of $\mathbf{X}_v$, then (3) is changed to

$$\begin{aligned}\Pi(X_{v_i} = x_{v_i} | X_{\mathrm{pa}(v_i)} = x_{\mathrm{pa}(v_i)}) = \\ \pi_{v_i}(x_{v_i}) \Bigg[1 + \sum_{u \in \mathrm{pa}(v_i)} \frac{(x_{v_i} - \mu_{\pi_{v_i}}) \beta_{v_i,u} \pi_u(x_u) (x_u - \mu_{\pi_u})}{d_{\mathrm{pa}(v_i) \setminus \{u\}} \sigma^2_{\pi_{v_i}} \sigma^2_{\pi_u} \Pi(X_{\mathrm{pa}(v_i)} = x_{\mathrm{pa}(v_i)})} \Bigg].\end{aligned} \tag{10}$$

   Here one collection of auxiliary pmfs take discrete uniform pmfs and another collection is identitical to the prescribed $\{\pi_v\}$.

5. If we assume $\hat{\pi}_v(x_v) = \pi_v(x_v), \forall x_v \in \mathbf{X}_v$ for all $v \in V$, then (3) becomes

$$\begin{aligned} & \Pi(X_{v_i} = x_{v_i} | X_{\mathrm{pa}(v_i)} = x_{\mathrm{pa}(v_i)}) \\ = \; & \pi_{v_i}(x_{v_i}) + \frac{\tilde{\pi}_{v_i}(x_{v_i})(x_{v_i} - \mu_{\tilde{\pi}_{v_i}})}{\sigma^2_{\tilde{\pi}_{v_i}}} \\ & \times \sum_{u \in \mathrm{pa}(v_i)} \left(\prod_{w \in \mathrm{pa}(v_i) \setminus \{u\}} \pi_w(x_w) \right) \cdot \frac{\beta_{v_i,u} \tilde{\pi}_u(x_u)(x_u - \mu_{\tilde{\pi}_u})}{\sigma^2_{\tilde{\pi}_u} \Pi(X_{\mathrm{pa}(v_i)} = x_{\mathrm{pa}(v_i)})} \end{aligned} \tag{11}$$

6. Assume that the space of states $\mathbf{X}_v$ for all $v \in V$ is same and is denoted by $\mathbf{X}$. If $\hat{\pi}_v(x_v) = \hat{\pi}(x_v)$ and $\tilde{\pi}_v(x_v) = \tilde{\pi}(x_v)$, $x_v \in \mathbf{X}$ for all $v \in V$, then (3) is adapted to

$$\begin{aligned} & \Pi(X_{v_i} = x_{v_i} | X_{\mathrm{pa}(v_i)} = x_{\mathrm{pa}(v_i)}) \\ = \; & \pi_{v_i}(x_{v_i}) + \frac{\tilde{\pi}(x_{v_i})(x_{v_i} - \mu_{\tilde{\pi}})}{\sigma^2_{\tilde{\pi}}} \\ & \times \sum_{u \in \mathrm{pa}(v_i)} \left(\prod_{w \in \mathrm{pa}(v_i) \setminus \{u\}} \hat{\pi}(x_w) \right) \cdot \frac{\beta_{v_i,u} \tilde{\pi}(x_u)(x_u - \mu_{\tilde{\pi}})}{\sigma^2_{\tilde{\pi}} \Pi(X_{\mathrm{pa}(v_i)} = x_{\mathrm{pa}(v_i)})} \end{aligned} \tag{12}$$

where $\mu_{\tilde{\pi}} = \sum_{x_v \in \mathbf{X}} \tilde{\pi}(x_v) x_v$ and $\sigma^2_{\tilde{\pi}} = \sum_{x_v \in \mathbf{X}} \tilde{\pi}(x_v)(x_v - \mu_{\tilde{\pi}})^2$ $(v \in V)$. In this case, auxiliary pmfs $\{\hat{\pi}_v\}$ are identically distributed, and so are $\{\tilde{\pi}_v\}$.

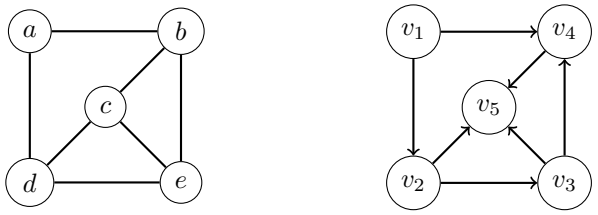
7. If we combine the assumptions for formula (9) and (12) together, i.e., $\mathbf{X}_v = \mathbf{X}$ and $\hat{\pi}_v(x_v) = \tilde{\pi}_v(x_v) = \frac{1}{d}, x_v \in \mathbf{X}$ for all $v \in V$ where d is the cardinality of $\mathbf{X}$, then (3) takes the following simple form

$$\begin{aligned} \Pi(X_{v_i} = x_{v_i} | X_{\mathrm{pa}(v_i)} = x_{\mathrm{pa}(v_i)}) &= \pi_{v_i}(x_{v_i}) \\ &+ \sum_{u \in \mathrm{pa}(v_i)} \frac{(x_{v_i} - \bar{\mu}) \beta_{v_i,u} (x_u - \bar{\mu})}{d^{|\,\mathrm{pa}(v_i)|+1} (\bar{\sigma}^2)^2 \Pi(X_{\mathrm{pa}(v_i)} = x_{\mathrm{pa}(v_i)})} \end{aligned} \tag{13}$$

where $\bar{\mu} = \frac{1}{d} \sum_{x_v \in \mathbf{X}} x_v$, $\bar{\sigma}^2 = \frac{1}{d} \sum_{x_v \in \mathbf{X}} (x_v - \bar{\mu})^2$ $(v \in V)$ and $|\,\mathrm{pa}(v_i)|$ is the cardinality of $\mathrm{pa}(v_i)$. Formula (13) is used in Kouritzin et al. (2013) for CAPTCHA generation.

3. Examples

In the following, we illustrate how the conditional probabilities are calculated as well as how different problems can be represented as a graph with associated marginal probabilities and pairwise covariances.



(a) Undirected Graph G (b) Directed Graph D

Figure 1: Graph Construction Example

Example 2. Suppose we have random variables and pairwise covariances such that our undirected graph $G = (V, E)$ is as given in Figure 1a. Then, our procedure for constructing the directed graph in Section 2.1.2, $D = (V, A)$, could result in the graph shown in Figure 1b.

Let the common space of states for each vertex be $\mathbf{X} = \{-1, 0, 1\}$, i.e., $\mathbf{X}_v = \{-1, 0, 1\}$ for all $v \in V$. To construct a probability measure Π on $\mathcal{X} = \prod_{i=1}^{5}\{-1, 0, 1\}$, we use Proposition 1 to compute the conditional probabilities $\Pi(X_{v_i} = x_{v_i}|X_{\mathrm{pa}(v_i)} = x_{\mathrm{pa}(v_i)})$ for $1 \leq i \leq 5$. To compute, for example, $\Pi(X_{v_4} = x_{v_4}|X_{\mathrm{pa}(v_4)} = x_{\mathrm{pa}(v_4)})$, we have to compute $\Pi(X_{\mathrm{pa}(v_4)} = x_{\mathrm{pa}(v_4)})$ as a prerequisite, using Proposition 1. So we have

$$\begin{aligned}
\Pi(X_{\mathrm{pa}(v_4)} = x_{\mathrm{pa}(v_4)}) &= \Pi(X_{v_3} = x_{v_3}, X_{v_1} = x_{v_1}) \\
&= \sum_{x_{v_2} \in \{-1,0,1\}} \Pi(X_{v_3} = x_{v_3}, X_{v_2} = x_{v_2}, X_{v_1} = x_{v_1}) \\
&= \sum_{x_{v_2} \in \{-1,0,1\}} \big[\Pi(X_{v_3} = x_{v_3}|X_{v_2} = x_{v_2}) \\
&\qquad \times \Pi(X_{v_2} = x_{v_2}|X_{v_1} = x_{v_1})\Pi(X_{v_1} = x_{v_1})\big].
\end{aligned}$$

Remark 8. As can be seen in this example, the cost of computing the conditional probabilities depends heavily on computing $\Pi(X_{\mathrm{pa}(v_i)} = x_{\mathrm{pa}(v_i)})$ in the denominator of the right hand side of (3), which in turn depends on the exact structure of the graph. In the trivial case, $\mathrm{pa}(v_i)$ is empty and no operations are required. In the above example, $\Pi(X_{v_3} = x_{v_3}, X_{v_2} = x_{v_2}, X_{v_1} = x_{v_1})$ must be calculated for every value of x_{v_2} in $\mathbf{X}$ by (2). In general, the cost grows with the size of $x_{v_k} : 1 \leq k \leq j, v_k \notin B$ in (2). This cost can be mitigated via dynamic

programming and simplifying the structure of the graph when possible.

Example 3. Suppose we are given an image (i.e., a two dimensional grid of pixels) of size $M \times N$. We can construct an undirected graph representing this image by placing a vertex at each pixel location, labeled with the pixel's coordinates in the image (e.g., $(1, 2)$). With this convention, the set of vertices would be $V = \{(m, n) : m \in \{1, ..., M\}, n \in \{1, ..., N\}\}$ and we could consider the distance between vertices to be $|(m_1, n_1)(m_2, n_2)| = \sqrt{|m_2 - m_1|^2 + |n_2 - n_1|^2}$. The alphabet $\mathbf{X}_v$ for the pixel v would be some finite collection of colors (or grey levels) $\{1, 2, ..., d\}$ that could even depend upon $v \in V$. Furthermore, we could add an edge between two vertices whenever they are within a certain distance from each other. Then, given a set of sample images, we can estimate both the marginal probabilities for each vertex and pairwise covariances for each edge, and can then simulate new images. This is quite related to what was done in the CAPTCHA generation work of Kouritzin et al. (2013).

Example 4. Suppose we are given an undirected graph $G = (V, E)$ where the vertices are divided into a known part H^C and an unknown part H, (i.e., V is partitioned into H and H^C). In this case, our goal is to simulate H while using the information contained in H^C. For example, we want to restore a portion of an image given some known set of pixels. We want to order the vertices so that $V = \{v_1, v_2, \ldots, v_k, v_{k+1}, \ldots, v_n\}$, where $H^C = \{v_1, v_2, \ldots, v_k\}$ and $H = \{v_{k+1}, \ldots, v_n\}$. This allows us to start simulating immediately from v_{k+1} since all preceding vertices are already known. To order the vertices in this way, we begin by working with G_{H^C}, the subgraph induced by H^C, and apply the algorithm in Section 2.1.2 to obtain directed subgraph $D_{H^C} = (V_{H^C}, A_{H^C})$. Next, we apply the algorithm in Section 2.1.2 to G with one slight difference: instead of initializing $V_0 = \emptyset, A_0 = \emptyset$, we initialize $V_0 = V_{H^C}$ and $A_0 = A_{H^C}$. Then, we've ordered the vertices so that $H^C = \{v_1, v_2, \ldots, v_k\}$ and $H = \{v_{k+1}, \ldots, v_n\}$, as desired, and we can begin simulating from v_{k+1}.

For example, consider Figure 2a which shows G with vertices from H^C with dashed outlines and vertices from H with solid outlines. The subgraph G_{H^C} is

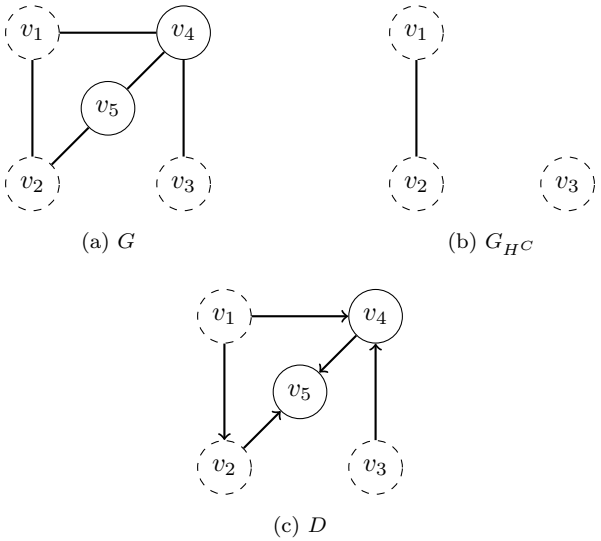


Figure 2: Known and Unknown Vertices Example

shown in Figure 2b, and Figure 2c shows the final directed graph $D = (V, A)$.

4. Applications to Classification and Image Recognition

Many image recognition algorithms rely on site-based pattern recognition to decide which of a collection of categories $\Theta = \{\theta_i\}_{i=1}^{d}$ an image belongs to. For example, if we are interested in classifying an image as that of a dog, cat, or mouse, then $\theta_1, \theta_2, \theta_3$ would be a set of parameters, like site intensities or pmfs, that might distinguish the image. These pattern algorithms usually ignore correlations between site variables, possibly due to the apparent computational complexity that might be required by algorithms that use correlations to help distinguish. However, covariances can capture pairwise relationships between variables so including them in recognition algorithms could potentially lead to improved classification. Our algorithm facilitates real-time covariance-included classification and recognition.

In the following, we will show that including more covariances increases the power to distinguish by increasing the likelihood ratio of the "correct"

parameters to alternative parameters; thereby, demonstrating the worth of computationally-efficient classification methods that include covariance information. We begin with a few definitions, taken in our setting of interest.

4.1. Basic Definitions

An *observation equation* is a stochastic equation $X = M(\theta)$ that mimicks the process of transforming a set of parameters $\theta \in \Theta$ into an observation X. For example, in the animal image recognition problem the observation equation would transform the parameters θ_1, θ_2 and θ_3 into random images of a dog, cat and mouse respectively. Ideally, each generation would produce a distinct version of the animal with the noise that would typically be encountered in a real image. In our case, each θ_i is a set of marginal probabilities and pairwise covariances so θ_1 would consist of site pmfs and site-to-site covariances that would characterize an image of a dog. With a particular set of parameters the observation equation becomes a *model*. In our current example, there would be three models: $X = M(\theta_1)$ for a dog, $X = M(\theta_2)$ for a cat, and $X = M(\theta_3)$ for a mouse. The objective of recognition or classification can then be that of *model selection*, where one statistically determines which of the d models best fits the *observed data* X_V of a given image or more generally record. Within our framework, X_V is a random field over a set of vertices V and the model selection classification problem is equivalent to identifying which set of parameters $\theta_i \in \Theta$ best represents the observed data.

When possible, model selection is effectively done by likelihood methods. *Likelihood* is the probability of the observed data given a set of parameters θ_i, which we will denote by $L(\theta_i)$. It is a function of the parameters θ_i, and we are interested in finding the best parameters in the parameter space Θ for the observed data. Maximum likelihood model selection selects the parameters which make the observed data the most likely, i.e.,

$$\theta^* = \text{argmax}_{\theta_i \in \Theta} L(\theta_i).$$

We work with the *log likelihood* $l(\theta_i) = \log L(\theta_i)$, which also has its maximum

at θ^*. Our Bayesian network framework is convenient for computation of $l(\theta_i)$:

$$L(\theta) = \prod_{v \in V} \Pi(X_v = x_v | X_{\mathrm{pa}(v)} = x_{\mathrm{pa}(v)})$$

by (1), which we can compute directly from (3) [2]. Then, the log likelihood is

$$l(\theta) = \sum_{v \in V} \log \Pi(X_v = x_v | X_{\mathrm{pa}(v)} = x_{\mathrm{pa}(v)}) \tag{14}$$

and model-selection-based classification is just a matter of maximizing $l(\theta_i)$ over $\theta_i \in \Theta$.

Simulated observations are images (or records) generated (using a computer) from the models rather than from the physical process that we are modeling like the act of taking pictures of animals. Simulated observations are useful in testing the power of a method because: a) one knows the *ground truth* e.g. exactly what the image is and b) modeling error does not come to bare since the images are generated by the model. However, in order to generate simulated observations one must have the model and the parameters. In our case, our model will boil down to a graph and the parameters pmfs for the vertices and covariances along the edges. Suppose we are given a graph G as in Section 2.1.2, where edges represent pairwise covariances of varying magnitudes. Then, one can construct D from G, and this directed acyclic graph will encode the true parameters of interest. We apply the algorithm in Section 2.3 to simulate $n+1$ realizations of $\{x_v\}_{v \in V}$.

Generally, the parameters are not given so we will need to estimate our site pmf and covariance parameters. Suppose that $\{X^{i,j}\}_{j=1}^n$ is a collection of real images for model i. Our *unbiased pmf estimator* and *unbiased covariance*

[2] The result computed from (3) is occassionally outside of $[0, 1]$ due to: i) numerical issues in implementing our simulation algorithm, ii) numerical issues in estimating our site pmfs and site-to-site covariances, and iii) the inability of our algorithm to produce a distribution with the exactly desired site pmfs and site-to-site covariances with the chosen auxiliary pmfs. In these cases, we use 0.01 and 0.99 in place of negative numbers and numbers greater than 1 respectively and find that the algorithm continues work well provided these exception only happen occassionally. A theoretical investigation of these exceptions is a topic of future work.

estimator are:

$$\hat{\pi}_v^{i,n}(x_v) = \frac{1}{n}\sum_{j=1}^{n}\delta_{x_v}(X_v^{i,j}) \text{ and } \hat{\beta}_{u,v}^{i,n} = \frac{1}{n-1}\sum_{j=1}^{n}(X_u^{i,j} - \bar{X}_u^i)(X_v^{i,j} - \bar{X}_v^i), \quad (15)$$

where

$$\delta_{x_v}(X_v^{i,j}) = \begin{cases} 1 & \text{if } X_v^{i,j} = x_v \\ 0 & \text{otherwise} \end{cases} \text{ and } \bar{X}_u^i = \frac{1}{n}\sum_{j=1}^{n} X_u^{i,j}.$$

The adjective unbiased refers to the fact that you get the actual pmf or covariance upon taking expectation for any fixed n under the assumption of $\{X^{i,j}\}_{j=1}^n$ independence. In our first example to follow next, the model parameters (covariances and pmfs) could be set by the authors. However, that would be cheating since they would never be known in a real application. Therefore, we only use these set parameters to generate an initial fixed collection of observations. Then, we procede to estimate parameters treating this initial collection of simulated observations as if they were real observations. In the second example to follow after that, we apply our method to Optical Character Recognition where the real observations are various character font images that are readily available from books and the internet. In both examples then, we are treating the pmf and covariance estimation as part of recognition process.

Based upon the above discussion and definitions, our example problems to follow have the same basic form: Given a collection of $n+1$ observations from each of d models, construct and test a classification algorithm to determine which model each observation came from. The dilemma is that we need these $d(n+1)$ observations both to do the model selection and to estimate the parameters. It would be cheating to deem an observation to be simultaneously both of known type i say so it could be used to estimate parameters θ_i and to be of unknown type so it is good for model selection testing to determine if our classification method is working. Hence, each observation should only be used one way at a time, which leads us to the known pessimistic *leave-one-out-cross-validation* testing method. Here, one observation of each type is reserved for validation while the rest are used for training i.e. for estimating the pmfs

and covariances in θ_i. Then, we rotate to leave out a different observation and continue until all observations have been left out once. Naturally, the parameter estimates and model selection would depend upon the observation left out for validation so it is important to cycle through all observations as we do and report on average results.

4.2. Simple Graph Example

To illustrate the importance of covariance information, we first directly consider a graph example. Let $G = (V, E)$ be as depicted in Figure 3 so there are 20 vertices and 16 edges, and let $\mathbf{X}_v = \{1, -1\}$ for all $v \in V$. Let $\Theta_T = \{\theta_T^1, \theta_T^2\}$,

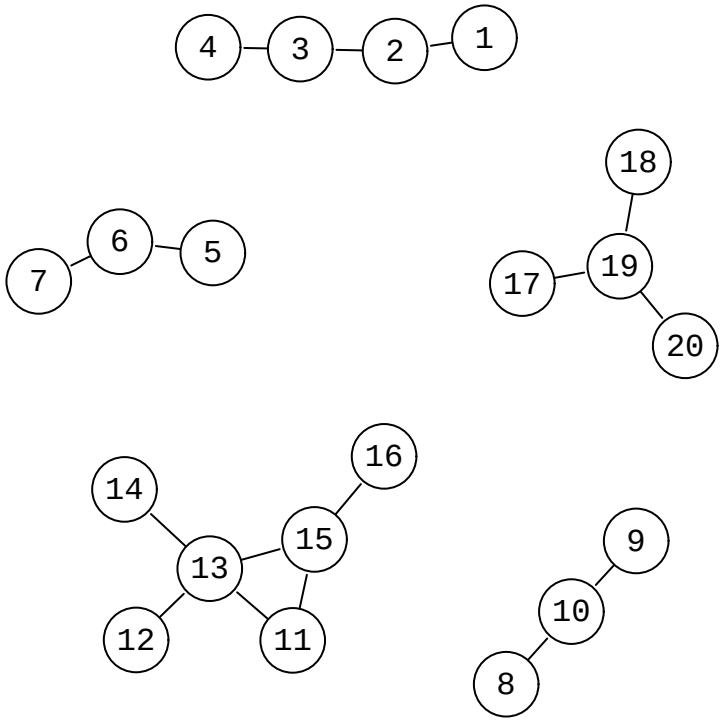


Figure 3: $G = (V, E)$

where

$$\theta_T^i = \{\pi_1^i, \ldots, \pi_{20}^i; \beta_{1,2}^i, \beta_{2,3}^i, \ldots, \beta_{19,20}^i\}$$

and

$$\pi_j^i(k) = \begin{cases} \frac{1}{2} & \text{if } k = 0 \\ \frac{1}{2} & \text{if } k = 1 \end{cases}$$

for all $i = 1, 2; j = 1, 2, \ldots, 20$. Moreover, $\beta_{j,k}^1 = 0.2$ for all j, k and $\beta_{j,k}^2 = -0.2$ for all j, k i.e. the sets have covariances of equal but opposite magnitude of 0.2 and -0.2 respectively. These are the true sets of parameters. We simulate $n + 1 = 10,000$ observations under each $\theta_T^i \in \Theta_T$ and apply leave-one-out cross-validation to the observations under θ_T^1.

To form the thirty-two candidate model parameters $\Theta = \{\theta_i^{N_E}\}_{\substack{i=1,2 \\ N_E=1,2,\ldots,16}}$, we use the unbiased estimators of the marginal probabilities $\{\hat{\pi}_v^i\}_{\substack{i=1,2 \\ v\in\{1,\ldots,20\}}}$ and the covariances $\{\hat{\beta}_j^i\}_{\substack{i=1,2 \\ j=1,\ldots,N_E}}$ on the $n = 9,999$ non-left-out observations. (We are particularly interested in the effect of the number of covariances i.e. edges included in the model parameters, which we denote by N_E, so we do the model selection repeatedly with different values of $N_E \in \{0, 1, ..., 16\}$.) The N_E covariance estimates of greatest magnitude $\{\hat{\beta}_j^i\}_{\substack{i=1,2 \\ j=1,\ldots,N_E}}$ are used to determine the edges in the undirected graph G; the marginal probability estimates and N_E covariance estimates are denoted as $\hat{\theta}_i^{N_E}$. Finally, the log likelihood of the validation (left out) observation is computed using (14) under each $\theta_i^{\hat{N}_E} \in \Theta$, and the $\hat{\theta}_i^{N_E}$ which yields the greatest log likelihood is used as the classification.

Figure 4 graphs the mean log likelihoods of each $\theta_i^{N_E} \in \Theta$ across the $n+1$ trials (i.e. each different left out observation) with various values of N_E, and shows the divergence of the log likelihoods as N_E increases (i.e., as more covariances are used in the inference). In particular, it demonstrates that increasing N_E increases the likelihood under the correct parameters while decreasing the likelihood under the incorrect parameters; in other words, including the covariances increases the effectiveness of the inference.

4.3. Optical Character Recognition

The above detailed a hypothetical experiment illustrating the utility of covariances in inference. We now briefly explore how the hypothetical can be

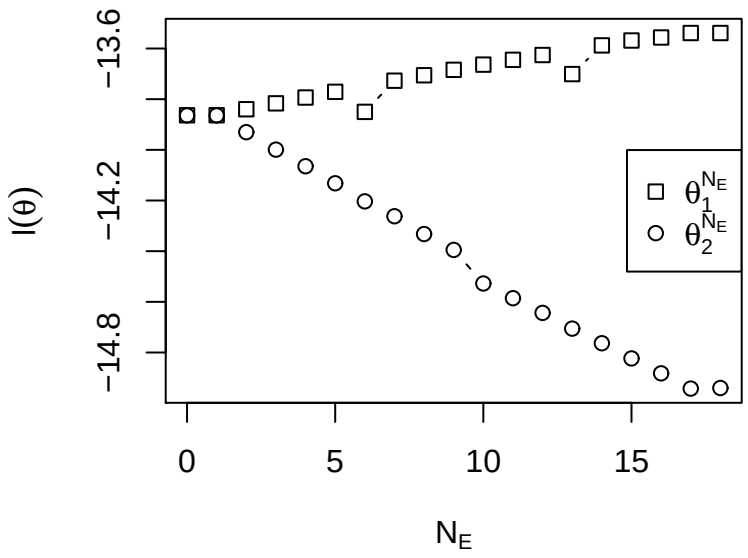


Figure 4: Mean $l(\theta)$

extended to a practical problem: optical character recognition (OCR). In OCR, the goal is to recognize characters or words in an image. In particular, we consider images which can be represented as a matrix of values corresponding to the colour of each pixel, though it is also easy to represent it as a graph $G = (V, E)$ where each vertex represents a pixel. Suppose our goal is to determine which letter is represented in a novel blurry or corrupted image. This letter is of unknown font so we have to work with probabilities and covariances for the various pixels in the picture.

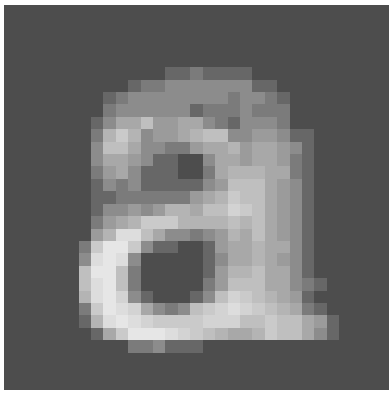
As before, each $\theta_i^{N_E} \in \{\theta_1^{N_E}, \ldots, \theta_{26}^{N_E}\}$ is a parameter vector corresponding to a different category; in this case, a letter in the alphabet. A $\theta_i^{N_E}$ consists of the marginal pmfs at each vertex over the possible colours at that vertex, and the N_E strongest covariances between pixel pairs for the edges in the graph.

The parameters $\theta_i^{N_E}$ could be estimated from images of the i^{th} letter in various fonts. Then, classification of a novel image is done as before: find the parameters which maximize the probability of the data.

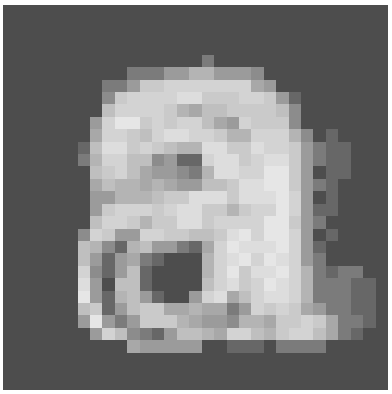
To illustrate the importance of both the pmf and covariance parameters within each vector, we consider 52 black and white images of the letter "a". Figure 5a is a heat map of the marginal probability of each pixel being black; Figure 5b is a heat map of the greatest positive covariance between each pixel and any other pixel; Figure 5c is a heat map of the greatest negative covariance between each pixel and any other pixel. Together, the figures illustrate that both marginal probabilities and pairwise covariances capture meaningful information about the letter, which indicates the applicability of our approach to OCR.

We now further establish the utility of this approach with a simple experiment. We collected a total of 1239 unique low resolution 20x20 images representing the 26 characters in the English alphabet. See Figure 6 for examples. (The number of unique images per character varies, with a minimum of 28 and a maximum of 53, because some characters in different fonts at such a low resolution reduce to identical black and white pixels. The correct letter is known for each image.) Here, the common state space for each pixel is $\mathbf{X} = \{-1, 1\} = \{\text{black}, \text{white}\}$. To construct a model parameter set $\theta_i^{N_E}$ we estimate the marginal probabilities and pairwise covariances using (15) again for each site v and potential edge (u, v). Then, we take the strongest N_E covariances to form the edges in G with one modification: to ensure the problem remains feasible, we only add edges to G if the edge does not form a connected component of more than 10 vertices[3]. For example, in the case $i = 23$ (so the letter is "w") and $N_E = 5$ there are 400 site pmfs and 5 covariances $\beta_{152,153} = 0.85, \beta_{130,153} = 0.79, \beta_{252,253} = 0.78, \beta_{273,274} = 0.78, \beta_{130,191} = 0.76$. We perform 1065 leave-one-out cross-validation trials, with at least 40 trials

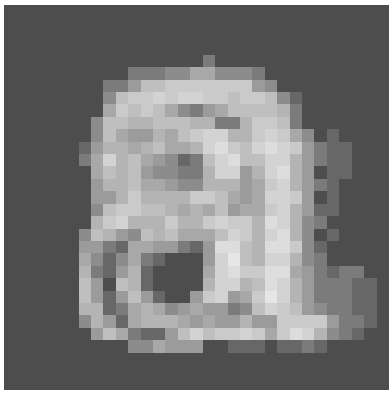
[3]Depending on the exact structure of the graph, $\Pi(X_{\text{pa}(v)} = x_{\text{pa}(v)})$, computed according to (2), may become expensive. In this case, a simple heuristic ensures this does not become an issue.



(a) Marginal Probabilities



(b) Largest Positive Covariances



(c) Largest Negative Covariances

Figure 5: Parameters for "a"

per character except for "l", which has only 28 unique images. In each trial, we determine the log likelihood of the image under $\theta_i^{N_E} \in \Theta$ corresponding to each character in the English alphabet with each $N_E \in \{0, 5, 25, 50, 100\}$ and $i = 1, \ldots, 26$. We hypothesize that both the accuracy of the OCR and the difference of log likelihoods between the correct character and the best alternative letter increase with N_E.

To demonstrate this, we determine the maximum log likelihood across $\theta_i^{N_E} \in \Theta^{N_E^*}$ for some maximum number of edges N_E^* in G, where $\Theta^{N_E^*} = \{\theta_i^{N_E} \in \Theta : N_E \leq N_E^*\}$. If the covariances aid in OCR, then we expect that the accuracy of our OCR will increase with N_E^*. Indeed, this is exactly the outcome we obtain

in Figure 7, which shows the empirical probability of picking the correct letter using log likelihood and N_E covariances. In addition, the difference between the log likelihood of the correct response (denoted θ_T) and the best alternative response (denoted θ_A^*)[4], given in Figure 8, also increases with N_E^*, which agrees with our simple simulation results in Figure 4. Here, the log likelihood ratio $l(\theta_T) - l(\theta_A^*)$ is computed by substituting θ_T and θ_A^* into (14).

Taken together, these empirical results support the notion that marginal probabilities and pairwise covariances, in combination with the method given in Section 2, are an effective inference and classification tool.

Figure 6: Example images of "a"

5. Conclusion

In this work, we introduced a method of efficiently constructing conditional probabilities such that marginal probabilities and pairwise covariances over a graph are maintained, and this method is used to generate correlated random fields in a computationally feasible manner. Several important special cases are addressed and illustrative examples are provided. Furthermore, several appli-

[4]In this scenario, we do not have true parameters for each letter; instead, we use the unbiased estimates for the site pmfs and covariances.

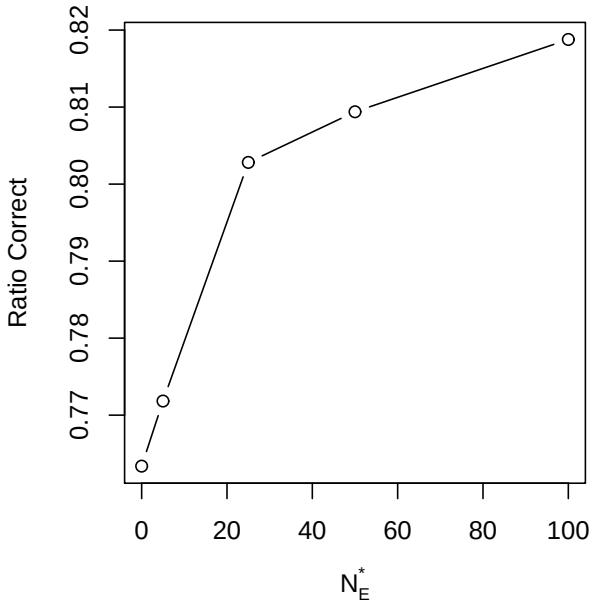


Figure 7: OCR Accuracy by N_E^*

cations using earlier one and two-dimensional versions of the random field are detailed, and new applications in inference and classification such as optical character recognition are empirically demonstrated.

Future work will include exploring the constraints on the values of the covariances so that the right hand side of (3) is in $[0, 1]$, investigating the constraints under which the random field does not depend upon the order of simulation (in terms of joint field probability) as well as further development of new applications in simulation and inference to areas such as image restoration, optical character recognition, and speech recognition.

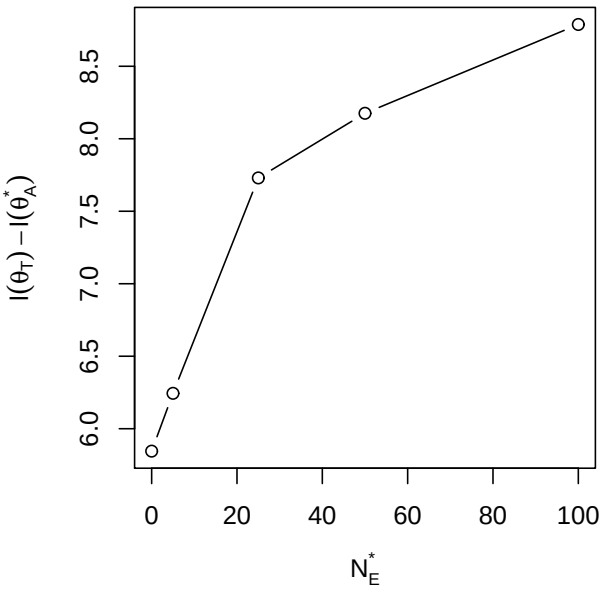


Figure 8: Mean $l(\theta_T) - l(\theta_A^*)$ by N_E^*

6. References

Ashburner, J., Friston, K., Penny, W., 2003. Human brain function, 2nd Edition. Academic Press, edited by Ashburner, J.; Friston, K.; Penny, W.

Blue, J., Candela, G., Grother, P., Chellappa, R., Wilson, C., 1993. Evaluation of pattern classifiers for fingerprint and ocr applications. Pattern Recognition 27 (4), 485–501.

Chellappa, R., Jain, A., 1993. Markov random fields: theory and application. Boston: Academic Press, 1993, edited by Chellappa, Rama; Jain, Anil.

Chellappa, R., Wilson, C., Sirohey, S., 1995. Human and machine recognition of faces: A survey. Proceedings of the IEEE 83 (5), 705–741.

Kouritzin, M., Newton, F., Wu, B., 2013. On random field completely automated public turing test to tell computers and humans apart generation. Image Processing, IEEE Transactions on 22 (4), 1656–1666.

Li, C., Yuan, Y., Wilson, R., 2008. An unsupervised conditional random fields approach for clustering gene expression time series. Bioinformatics 24 (21), 2467–2473.

Li, H., Kallergi, M., Clarke, L., Jain, V., Clark, R., 1995. Markov random field for tumor detection in digital mammography. IEEE Transactions on Medical Imaging 14 (3), 565–576.

Li, S., 1995. Markov Random Field Modeling in Computer Vision. Springer-Verlag, Tokyo.

Sun, G., Liu, Y., Qiao, P., Lang, F., 2008. Chinese chunking algorithm based on cascaded conditional random fields. In: Proceedings of the 11th Joint Conference on Information Sciences. Atlantis Press.

Winkler, G., 2003. Image analysis, random fields and markov chain monte carlo methods: A mathematical introduction.

Worsley, K., 1995. Boundary corrections for the expected euler characteristic of excursion sets of random fields, with an application to astrophysics. Advances in Applied Probability 27, 943–959.

Zhang, Y., Brady, M., Smith, S., 2001. Segmentation of brain mr images through a hidden markov random field model and the expectation-maximization algorithm. IEEE Transactions on Medical Imaging 20 (1), 45–57.

Zhu, J., Nie, Z., Zhang, B., Wen, J., 2008. Dynamic hierarchical markov random fields for integrated web data extraction. The Journal of Machine Learning Research 9, 1583–1614.