Probabilistic Graphical Models

Sargur N. Srihari

Department of Computer Science and Engineering

University at Buffalo, The State University of New York

srihari@buffalo.edu

To appear in "Encyclopedia of Social Network Analysis and Mining, Springer 2014"

Contents

1 Title

Probabilistic Graphical Models

2 Affiliation

University at Buffalo, The State University of New York

3 Synonyms

Bayesian Networks, Markov Networks, Markov Random Fields

4 Glossary

- *Bayesian Network (BN)*: A directed graph whose nodes represent variables, and edges represent influences. Together with conditional probability distributions, a Bayesian network represents the joint probability distribution of its variables.
- *Conditional Probability Distribution*: Assignment of probabilities to all instances of a set of variables when the value of one or more variables is known.
- *Conditional Random Field (CRF)*: A partially directed graph that represents a conditional distribution.
- *Factor Graph*: A type of parameterization of PGMs in the form of bipartite graphs of factor nodes and variable nodes, where a factor node indicates that the variable nodes it is connected to form a clique in a PGM.
- *Graph*: A set of nodes and and edges, where edges connect pairs of nodes.
- *Inference*: Process of answering queries using the distribution as the model of the world.
- *Joint Probability Distribution*: Assignment of probabilities to each instance of a set of random variables.

- *Log-linear Model*: a Markov network represented using features and energy functions.
- *Markov Network (MN)*: An undirected graph whose nodes represent variables, and edges represent influences. Together with factors defined over subsets of variables, a Markov network represents the joint probability distribution of its variables.
- *Markov Random Field (MRF)*: Synonymous with Markov network. Term more commonly used in computer vision.
- *Partially Directed Graph*: A PGM with both directed and undirected edges.
- *Probabilistic Graphical Model (PGM)*: A graphical representation of joint probability distributions where nodes represent variables, and edges represent influences.

5 Definition

Probabilistic Graphical Models (PGMs), also known as graphical models, are representations of probability distributions over several variables. They use a graph-theoretic representation where nodes correspond to random variables and edges correspond to interactions between them. When the edges are directed, they are known as Bayesian networks (BNs). Since the edges of a BN typically represent causality between variables, they are also referred to as causal BNs. PGMs with undirected edges are known as Markov networks (MNs) or Markov Random Fields (MRFs).

6 Introduction

PGMs provide a powerful framework for modeling the joint distribution of a large-number n of random variables $\chi = \{X_1, X_2, \ldots, X_n\}$. PGMs use graphical representations that consists of nodes (also called vertices) and edges (also called links), where each node represents a random variable (or a group of random variables) and links express probabilistic relationships between variables. They allow distributions to be written tractably even when the explicit representation of the joint distribution is astronomically large: when the set of possible values of χ, $Val(\chi)$, are very large (exponential in n), PGMs exploit independencies between variables, thereby resulting in great savings in the number of parameters needed for representing full joint distributions.

PGMs are used to answer queries of interest, such as the probability of a particular assignment of the values of all the variables, i.e., $\xi \in Val(\chi)$. Other queries of interest are: conditional probability of latent variables given values of observable variables, maximum *a posteriori* probability of variables of interest, the probability of a particular outcome when a causal variable is set to a particular value, etc. Answers are produced using an inference procedure.

7 Key Points

PGMs provide: (i) a simple way to visualize structure of probabilistic model, (ii) insights into properties of model, e.g., conditional independence properties by inspecting graph, and (iii) complex computations required to perform inference and learning are expressed as graphical manipulations.

A powerful aspect of graphical models is that it is not necessary to state whether the distributions they represent are discrete or continuous: a specific graph can make probabilistic statements about a broad class of distributions. The theory of PGM representation and analysis is a marriage between graph theory and probability theory. The graph-theoretic representation augments analysis instead of using pure algebra.

8 Historical Background

Probability theory was developed to represent uncertainty. Gerolamo Cardano (1501-1576) was possibly the earliest to formulate a theory of chance. The French mathematicians Blaise Pascal (1623-1662) and Pierre-Simon Laplace (1749-1827) laid the foundations, with Laplace's major contribution to probability theory appearing in 1812. The English clergyman Thomas Bayes (1701-1761) stated the theorem named after him which relates conditional and marginal probabilities of variables by a simple application of the sum and product rules of probability.

The use of PGMs allows the application of principles of probability theory to large sets of variables which would otherwise be computationally infeasible. An early use of BNs, before the general framework was defined, was in genetic modeling of transmission of certain properties such as blood type from parent to child. BNs, as diagrammatic representations of causal probability distributions, was first defined by the computer scientist Judea Pearl [1]. A BN is not necessarily

based on the fully Bayesian approach of converting prior distributions of parameters to posterior distribution, although it becomes useful when data sets are limited.

A Markov process, named after the Russian mathematician Andrey Markov (1856-1952), describes the linear dependency of a variable on its previous states in a chain. Markov Random Fields were a generalization to model the two-dimensional dependency of a pixel on other pixels. MNs with log-linear representations have been around for a long time, with their origins in statistical physics. In the *Ising model*, which is due to the physicist Ernt Ising (1900-1998), the energy of a physical system of interacting atoms is determined from their spin, where each atom's spin is the sum of its electron spins. Each atom is characterized by a binary random variable $X_i \in \{+1, -1\}$ whose value x_i is the direction of its spin. Its energy function has the parametric form $\epsilon_{ij}(x_i, x_j) = -w_{ij}x_i x_j$ which is symmetric in X_i, X_j. They are used to answer questions concerning an infinite number of atoms, e.g., determine the probability of a configuration where majority of spins are +1 (or -1) versus more mixed ones. The answer depends on the strength of interactions, e.g., by multiplying all weights by a temperature parameter.

BNs are popular in AI and statistics. MNs, which are better suited to express soft constraints between variables, are popular in computer vision and text analytics.

9 Main Part (Probabilistic Graphical Models)

This discussion is divided into three parts: representation of PGMs, inference using PGMs and learning of PGMs.

9.1 Representation

The class of PGMs where the graphs are directed acyclic graphs and the directionality associated with edges express causal relationships between random variables are known as BNs. PGMs where links are undirected, i.e., do not have directionality, correspond to MNs, or Markov random fields (MRFs).

9.1.1 Bayesian Networks

A BN represents a joint probability distribution P over multiple variables χ by means of a directed graph G. Edges in the graph represent causal influences between variables represented as nodes. Conditional probability distributions (CPDs) represent the local conditional distributions $P(X_i|pa(X_i))$, where pa are parent nodes. The joint distribution has the factorization: $P(\chi) = \prod_{i=1}^{n} P(X_i|pa(X_i))$, which is the chain rule of BNs.

A BN G implicitly encodes a set of conditional independence assumptions $I(G)$. Each independence is of the form $(X \perp Y|Z)$, which can be read as: X is independent of Y given Z. If P is a probability distribution with independencies $I(P)$, then G is an *I-map* of P if $I(G) \subseteq I(P)$. If P factorizes according to G then G is an I-map of P. This is the key property to allowing a compact representation, and crucial for understanding network behavior. G is a *minimal I-map* of P if removing a single edge renders it not an I-map. G is a *perfect map* for P if $I(G) = I(P)$. Unfortunately every distribution does not have a perfect map. When many variable independencies are present the complexity of the BN decreases.

A. Local Models

When the variables are discrete-valued, the CPDs, which define local distributions of the form $P(Y|X_1, .., X_n)$, can be represented as conditional probability tables (CPTs), where each entry is the probability of the value of Y given the values of its parents. While CPTs are commonly used, they have some disadvantages, e.g., when the random variables have infinite domains as in the case of continuous variables. Also, in the discrete case, when n is large, the CPTs grow exponentially. To alleviate this, CPDs can be viewed as functions that return the conditional probability when given the value of Y and its parents.

Structure in a CPD can be exploited using either *deterministic* or *context-specific* CPDs. An example of a deterministic CPD is one where the value taken by Y is a function of the values of its parents $\{X_i\}$, and the conditional distribution has value 1 when the function holds and 0 otherwise. In a context-specific CPD several values of $\{X_i\}$ define the same conditional distribution. Examples of context-specific CPDs are trees and rules. In a CPD represented as a tree, there are leaf nodes and interior nodes. Each leaf node is associated with the distribution of Y while the path to that

leaf node defines the values taken by $\{X_i\}$. In a rule-based CPD each assignment of values to $\{X_i\}$ specifies the probability of a value assignment to Y.

Another type of CPD structure arises with independence of causal influence, where the combined influence of $\{X_i\}$ is a simple combination of the influence of each X_i on Y in isolation. One such model is the *noisy-or* where Y is binary-valued and the parents have independent parameters to activate Y. Another is a *generalized linear model* which uses soft linear functions: if Y is binary-valued then we can define a logistic CPD as the sigmoid function with weights $\{w_i\}_{i=0}^n$, i.e., $\sigma\left(w_0 + \sum_i w_i X_i\right)$; if Y is multi-valued, a multinomial logistic function is defined using the softmax function.

The case of continuous variables can be handled well by BNs. The dependency of a continuous variable Y on a continuous parent X, can be modeled as one where Y is Gaussian and the parameters depend on X, e.g., the mean of Y is a linear function of X and the variance of Y does not depend on X. A *linear Gaussian model* generalizes this to several parents, i.e., the mean of Y is a weighted sum of the parent variables.

When parents are both discrete and continuous we have a *hybrid* CPD; its form depends on whether the child is continuous or discrete. In the case when the child Y is continuous, we can define a *conditional linear Gaussian (CLG)* CPD as follows: if $\{U_i\}$ are discrete parents, $\{V_i\}$ are continuous parents, and $\{a_{\mathbf{u},i}\}$ are coefficients, then $P(X|u,v) = \mathcal{N}\left(a_{\mathbf{u},0} + \sum_{i=1}^k a_{\mathbf{u},i} v_i; \sigma^2_{\mathbf{u}}\right)$. When the child Y is discrete and the parent is continuous, we can use a multinomial distribution where for each assignment y we have a different continuous distribution over the parent.

B. Independencies

Independence properties are exploited to reduce computation of inference, i.e., answering queries. Separation between nodes in a directed graph, called *d-separation*, allows one to determine whether an independence $(X \perp Y|Z)$ holds in a distribution associated with BN structure G. BNs have two types of independencies: (i) local independencies: each node is independent of its non-descendants given it parents, and (ii) global independencies induced by d-separation. These two sets of independencies are equivalent. *D-separation* refers to four cases involving three variables X, Y and Z as follows: indirect-causal effect $(X \to Z \to Y)$, indirect evidential effect $(Y \to Z \to X)$, common cause $(X \leftarrow Z \to Y)$ and common effect $(X \to Z \leftarrow Y)$. In the first three cases, if Z is observed

then it blocks influence between X and Y. In the last case, known as a *v-structure*, an observed Z enables influence.

Reasoning in a BN strongly depends on connectivity. Reasoning can be top-down, called causal reasoning, or bottom-up, called evidential reasoning. Another type of reasoning is inter-causal reasoning, one example of which is *explaining away* where different causes of the same effect can interact. In another type of inter-causal reasoning, parent nodes can increase the probability of a child node.

C. Causality

While a BN captures conditional independences in a distribution, the causal structure is not necessarily meaningful, e.g., the directionality can even be antitemporal. In a good BN structure, an edge $X \rightarrow Y$ should suggest that X causes Y either directly or indirectly. While BNs with causal structure are likely to be sparser and more natural, the answers we obtain to probabilistic queries are the same. While $X \rightarrow Y$ and $Y \rightarrow X$ are equivalent probabilistic models they are very different causal models.

Causal models are important when we need to make interventions. Examples of causal queries involving intervention are: will preventing smoking in public places likely to decrease frequency of lung cancer, will strengthening family interactions (social capital) result in increased student scores, etc. One approach to model causal relationships is to use ideal interventions. An ideal intervention, written as $\mathbf{do}(\mathbf{Z} := z)$, is one where the only effect is to force variable $\mathbf{Z}$ to take the value z and have no other effect on other variables. The answer to an intervention query $P(\mathbf{Y}|\mathbf{do}(z), \mathbf{X} = \mathbf{x})$ is generally quite different from the answer to the probabilistic query $P(\mathbf{Y}|\mathbf{Z} = z, \mathbf{X} = \mathbf{x})$.

The identifiability of causality is complicated by the fact that correlation between two variables arise in multiple settings: when X causes Y, when Y causes X or when X and Y have a common cause. If the common cause W is observable, we can disentangle the correlation between X and Y induced by W and determine the residual correlation that is directly causal. However there usually are a large set of latent variables that we cannot observe. Fortunately, it is possible to answer, at least sometimes, causal questions in models with latent variables using only observed correlations. The intervention queries that can be answered using only conditional probabilities, which are said to be identifiable, can sometimes be determined using query simplification rules.

9.1.2 Markov Networks

When no natural directionality exists between variables, MNs offer a simpler perspective on directed graphs. Moreover there is no guarantee of perfect map in a BN since independences imposed may be inappropriate for the distribution; in a perfect map the graph precisely captures the independencies in the given distribution.

A. Parameterizations

A MN represents a joint probability distribution P over multiple variables χ by means of an undirected graph G whose nodes correspond to variables and edges correspond to direct probabilistic interactions. As in BNs parameterization of a MN defines local interactions. We combine local models by multiplying them, and convert it to a legal distribution by performing a normalization.

Affinities between variables can be captured using three alternative parameterizations: (i) MN as a product of potentials on cliques: good for discussing independence queries, (ii) Factor Graph, which is a product of factors that describes a Gibbs distribution: useful for inference, and (iii) Log-linear model with features, which is a product of features that describe all entries in each factor: useful for both hand-coded models and for learning.

1. Gibbs Parameterization

The first approach is to associate with each set of nodes a general purpose function called a *factor*, a function ϕ from $Val(D)$ to R where D is a subset of random variables. A factor captures compatibility between variables in its scope, and is similar to a CPD: for each combination there is a value. With attention restricted to non-negative factors: $Val(A, B)$ to $R+$, the value associated with a particular assignment (a, b) indicates affinity between the two values, with a higher value indicating higher compatibility. A Gibbs distribution generalizes the idea of a factor product.

A distribution P is a Gibbs distribution parameterized by a set of factors $\Phi = \{\phi_1(D_1), ..\phi_k(D_k)\}$ if it is defined as $P(\chi) = \frac{1}{Z}\tilde{P}(\chi)$, where $\tilde{P}(\chi) = \prod_{i=1}^{k} \phi_i(D_i), D_i \subseteq \chi$ is an unnormalized measure and $Z = \sum_{\chi} \tilde{P}(\chi)$ is known as the *partition function.* A Gibbs distribution factorizes over a Markov network G if each D_i is a complete subgraph (clique) of G. The *Hammersley-Clifford* theorem goes from independence properties of a distribution to its factorization, i.e., if P is a positive probability distribution, i.e., all probabilities are greater than zero, with independencies $I(P)$, then it factorizes

according to G. As with BNs G is an I-map of P.

Factors that parameterize the network are called *clique potentials*. The number of parameters is reduced by allowing factors only for maximal cliques, but it obscures the structure present. Factors do not represent marginal probabilities of the variables within their scope. A factor is only one contribution to the overall joint distribution. The distribution as a whole has to take into consideration contributions from all factors involved.

The subclass of MNs where interactions are only pairwise are commonly encountered, e.g., Ising model and Boltzmann machines are popular in computer vision. Here all factors are over single variables $\phi(X_j)$, called node potentials, or over pairs of variables $\phi(X_j, X_k)$, called edge potentials. Although simple they pose a challenge for inference algorithms.

2. Factor Graphs

The graph structure of a MN does not reveal all structure in a Gibbs parameterization, e.g., we cannot tell whether factors involve maximal cliques or their subsets. *Factor graphs* are undirected graphs that make the decomposition $p(\chi) = \prod_i \phi_i(D_i)$ explicit by using two types of nodes: variable nodes X_j denoted as ovals and factor nodes ϕ_i denoted as squares (See Figure 2(c)). They contain edges only between variable nodes and factor nodes. They are bipartite since there are two types of nodes with all links go between nodes of opposite type, and representable as two rows of nodes: variables on top and factor nodes at bottom. Other intuitive representations are used when derived from directed/undirected graphs. Steps in converting a distribution expressed as undirected graph are: create variable nodes corresponding to nodes in original, create factor nodes for maximal cliques D_i, and set factors $\phi_i(D_i)$ equal to clique potentials. Several different factor graphs are possible for the same distribution or graph. A directed graph can also be converted to a factor graph, where variable nodes correspond to variable nodes in factor graph, and factor nodes corresponding to conditional distributions.

3. Log-linear Models

While a factor graph makes the structure of the parameterization explicit, each factor is a complete table over its scope. We may wish to explicitly represent context-specific structure which involve particular values of the variables (as in BNs). Such patterns are more readily seen in log-space, by taking the negative natural logarithm of each potential.

If D is a set of random variables and $\phi(D)$ is a factor (consisting of values assigned to instances of D), $\epsilon(D) = -\ln \phi(D)$, thus $\phi(D) = \exp(-\epsilon(D))$. This has an analogy in statistical physics where the probability $\phi(D)$ of a physical state depends inversely on its energy $\epsilon(D)$, i.e., higher energy states have lower probability. In this representation $p(\chi) \propto \exp\left[-\sum_{i=1}^{k} \epsilon_i(D_i)\right]$. Logarithms of cell frequencies $\phi(D)$ are referred to as *log-linear* in statistics and the logarithmic representation ensures that the probability distribution is positive. Any MN parameterized using positive factors can be converted into a logarithmic representation.

If D is a subset of variables, *feature* $f(D)$ is a function from D to R (a real value). A feature is a factor without a non-negativity requirement.

The following log–linear model is a representation of a joint probability distribution over assignments to χ:

$$P(\chi : \theta) = \frac{1}{Z(\theta)} \exp\left(\sum_{i=1}^{k} \theta_i f_i(D_i)\right) \tag{1}$$

where $f_i(D_i)$ is a feature function defined over variables $D_i \subseteq \chi$, the set of all feature functions is denoted as $\mathcal{F} = \{f_i(D_i)\}_{i=1}^{k}$, k is the number of features in the model, $\theta = \{\theta_i : f_i \in F\}$ is a set of feature weights, $Z(\theta) = \sum_{\xi} \exp\left(\sum_{i=1}^{k} \theta_i f_i(\xi)\right)$ is a partition function, $f_i(\xi)$ is a shortened notation for $f_i\left(\xi \left\langle D_i \right\rangle\right)$ with a given assignment to the set of variables D_i.

B. Independencies

As in BNs, the graph structure of a MN encodes a set of independence assumptions. In a MN probabilistic influence flows along the undirected paths in the graph and blocked if we condition on intervening nodes.

There are three types of independencies in MNs. Two are local independencies: (i) pairwise independency $I_p(H)$ is the weakest type of independency: whenever two variables are directly connected, potential of being correlated not mediated by other variables, and (ii) Markov blanket $I_l(H)$: when two variables are not directly linked there must be a way of rendering them conditionally independent; this is analogous to local independencies in BNs. We can block all influences by conditioning on its immediate neighbors. A node is conditionally independent of all nodes given its immediate neighbors. For positive distributions (those with non-zero probabilities for all instantiations), all three are equivalent.

To determine global independency $I(H)$, identify three sets of nodes A, B and C. To test whether conditional independence property $A \perp B|C$, consider all possible paths from nodes in A to nodes in B. If all such paths pass through one or more nodes in C then path is blocked and independence holds.

MNs have a simple definition of independence: two sets of nodes are conditionally independent given a third set C if all nodes in A and B are connected through nodes in C. BN independence is more complex: it involves direction of arcs and inference problems. It is convenient to convert both to a factor graph representation.

In going from distributions to graphs, questions that arise are: how to encode independencies in given distribution P in a graph structure H? what sort of independencies to consider: global or local? are we looking for an I-map, minimal I-map, or perfect map?

9.1.3 Partially Directed Graphs

BNs with directed edges and MNs with undirected edges are both useful in different application scenarios. It is possible to convert BNs to MNs using *moralization* in which edges are introduced between unrelated parent nodes. Converting a MN into a BN introduces much higher network complexity. It is possible to unify both representations by incorporating both directed/undirected dependencies in the same PGM.

Conditional Random Fields (CRFs) are MNs with a directed dependency on some subset of variables. While a MN encodes a joint distribution over X, the same undirected graph can be used to represent a conditional distribution $P(Y|X)$, where Y is a set of target variables and X is a set of observed variables. It has an analog in directed graphical models: viz., conditional BNs.

CRF nodes correspond to $Y \cup X$ and parameterized as ordinary MNs. Can be encoded as a log-linear model with a set of factors $\phi_1(D_1), ..\phi_m(D_m)$. Instead of $P(Y, X)$ view it as representing $P(Y|X)$. To naturally represent a conditional distribution avoid representing a probabilistic model over X, and disallow potentials involving only variables in X.

To derive the CRF definition, from Bayes' rule we have $P(Y|X) = \frac{P(Y,X)}{P(X)}$. The numerator in terms of the Gibbs' definition of MN is $P(Y, X) = \frac{1}{Z(Y,X)}\tilde{P}(Y, X)$ where $\tilde{P}(Y, X) = \prod_{i-1}^{m} \phi_I(D_i)$ and $Z(Y, X) = \sum_{Y,X} \tilde{P}(Y, X)$. The denominator is $P(X) = \sum_Y P(Y, X) = \frac{1}{Z(Y,X)} \sum_Y \tilde{P}(Y, X)$.

Substituting back we get,

$$P(Y|X) = \frac{1}{Z(X)}\tilde{P}(Y, X). \tag{2}$$

where $Z(X) = \sum_Y \tilde{P}(Y, X)$, is the partition function which is a function of X. Whereas a Gibbs' distribution factorizes into factors and a partition function Z, a CRF has a different value in the partition function for every assignment x to X.

9.2 Inference

9.2.1 Query Types

There are three types of queries:

1. *Probability query:* given values of some variables, give distribution of another variable. This is the most common type of query. The query has two parts:

   - evidence, E, a subset of variables and their instantiation e, and
   - query variables, a subset Y of random variables in network.

   The inference task, which is to determine $P(Y|E = e)$, the posterior probability distribution over values y of Y conditioned on the fact $E = e$, can be viewed as marginal probability estimation over Y in the distribution we obtain by conditioning on e: $P(Y|E = e) = \sum_{\chi - Y} P(\chi|E = e)$.

2. *MAP (maximum a posteriori probability) query:* what is the most likely setting of variables. Also called MPE (most probable explanation). Most likely assignment to all non-evidence variables $W = \chi - E$ and $MAP(W|e) = \arg\max_w P(w, e)$ is the value w for which $P(W, e)$ is maximum. Instead of a probability we get the most likely value for all remaining variables.

3. *Marginal MAP query:* when some variables are known. Query does not concern all remaining variables but a subset of them. Given evidence $E = e$, task is to find most likely assignment to a subset of variables Y: $MAP(Y|e) = \arg\max_y P(y|e)$. If $Z = \chi - Y - E$ then $MAP(Y|e) = \arg\max_y \sum_z P(Y, Z|e)$. Inference of marginal MAP is more complex than MAP since it contains both summations (like in probability queries) and maximizations (like in MAP queries). Also, due to lack of MAP monotonicity, i.e., most likely assignment

$MAP(Y_1|e)$ might be completely different from assignment to Y_1 in $MAP(\{Y_1, Y_2\}|e)$, we cannot use a MAP query to give a correct answer to a marginal map query.

9.2.2 Computational Complexity

The probability of evidence $E = e$ can be determined from a BN, in principle, as follows: $P(E = e) = \sum_{X/E} \prod_{i=1}^{n} P(X_i|pa(X_i))|_{E=e}$. This is an intractable problem, one that is #P-complete. It is tractable when tree-width is less than 25, but most real-world applications have higher tree-width; where tree-width is defined as the number of variables in the largest clique. Approximations are usually sufficient (hence sampling), e.g., when $P(Y = y|E = e) = 0.29292$, approximation yields 0.3.

9.2.3 Inference algorithms

PGM structure can be exploited to find efficient algorithms. Algorithms are expressed as passing messages around graph. Two types of inference algorithms: exact and approximate methods which are useful when there are a large number of latent variables e.g., variational Bayes.

1. Exact Inference.

Consider graphs consisting of chains of random variables, also known as Markov chains, e.g., $N = 365$ days and X_i is weather (cloudy, rainy, snow..) on a particle day i. In this case directed and undirected graphs are exactly the same since there is only one parent per node (no additional links needed). The joint distribution has the form $p(\chi) = \frac{1}{Z}\Psi_{1,2}(X_1, X_2)\Psi_{2,3}(X_2, X_3)...\Psi_{N-1,N}(X_{N-1}, X_N)$. We wish to evaluate the marginal distribution $p(X_n)$ for a specific node part way along chain.e.g., what is the weather on November 11?

As yet there are no observed nodes. The required marginal is obtained summing the joint distribution over all variables except X_n: $p(X_n) = \sum_{X_1} ... \sum_{X_{n-1}} \sum_{X_{n+1}} ... \sum_{X_N} p(\chi)$. This is referred to as the *sum-product* inference task. In the specific case of N discrete variables with K states each: potential functions are $K \times K$ tables, the joint distribution has $(N-1)K^2$ parameters and there are K^N values for χ. Evaluation of both joint and marginal is exponential with length N of chain (which makes it impossible for say $K = 10$ and $N = 365$).

Efficient evaluation involves exploiting conditional independence properties. Key concept used

is that multiplication is distributive over addition, i.e., $ab + ac = a(b + c)$, where the LHS involves 3 arithmetic operations, while the RHS involves only 2. Using this idea, rearrange order of summations/multiplications to allow marginal to be evaluated more efficiently. Consider summation over X_N. Potential $\Psi_{N-1,N}(X_{N-1}, X_N)$ is the only one that depends on X_N. So we can perform $\sum_{X_N} \Psi_{N-1,N}(X_{N-1}, X_N)$ to give a function of X_{N-1}. Use this to perform summation over X_{N-1}. Each summation removes a variable from distribution or removal of node from graph. The total cost is $O(NK^2)$, which is linear in chain length vs. exponential cost of naiive approach. Thus we are able to exploit many conditional independence properties of simple graph. This calculation is viewed as message passing in graph. The key insight is that the factorization of the distribution allows performing local operations on the factors rather than generating the entire distribution. It is implemented using the *variable elimination algorithm* which sums out variables one at a time, multiplying factors necessary for that operation.

The sum-product algorithm evaluates an expression for marginal probabilities expressed in the form $\sum_{\chi \neq X_n} \prod_i \phi_i$. Variable elimination can also be used for evaluating the setting of variables for the largest probability–an inference problem which takes the form $\arg\max_{\chi} \prod_i \phi_i$. It is known as the *max-sum* algorithm which can be viewed as an application of dynamic programming to PGMs.

The sum-product and max-sum algorithms provide efficient and exact solutions to tree-structured graphs. For many applications we have to deal with graphs having loops. An alternative implementation based on the same variable elimination insight uses a more global data structure for scheduling the operations. It is based on the idea of clique trees. If the staring point is a directed graph, it is first converted to an undirected graph by moralization. Next the graph is triangulated by finding chord-less cycles containing four or more nodes and adding extra links to eliminate such chord-less cycles. Next the triangulated graph is used to construct the clique-tree whose nodes correspond to maximal cliques. A clique tree maps a graph into a tree by introducing a node for each clique in the graph, where the maximum clique size is known as the tree-width. Finally a two-stage algorithm essentially equivalent to the sum-product algorithm is applied. However exact inference is exponential in space and time complexity with tree-width.

2. Approximate Inference.

Since exact inference may be intractable, we regard inference as optimization, where we con-

struct an approximation to the target factorized distribution $P_\Phi(\chi) = \frac{1}{Z}\prod_i \phi_i(D_i)$ that allows simpler inference. It involves searching through a class of "easy" distributions to find an instance Q that best approximates P_Φ, e.g., one that minimizes the Kullback-Leibler divergence (also known as relative entropy): $D(Q||P_\Phi) = E_Q\left[\ln \frac{Q}{P_\Phi}\right]$. This is equivalent to maximizing the *energy functional* $F[\tilde{P}_\Phi, Q] = \sum_i E_Q[\ln \phi_i] + H_Q(\chi)$, which has two terms, the first of which is known as the energy term and the second term is the entropy of Q. Assuming that inference is easy in Q, the expectations in the energy term should be relatively easy to evaluate and the entropy term depends on the choice of Q. Queries can then be answered using Q instead of P_Φ.

Finding a good approximation Q is one of maximizing the energy functional. Methods that approach inference as optimization can be classified as: (i) variational methods, which are deterministic, (ii) propagation-based approximation, such as loopy-belief propagation, and (iii) particle-based approximation which use stochastic numerical sampling from distributions.

Variational methods are strategies for optimizing the energy functional. Clique tree calibration optimizes the energy functional over a class of representations of Q.

In loopy belief propagation we apply the sum product algorithm even though there is no guaranty of good results. The message passing schedule is modified: a flood schedule simultaneously passes a message across every link in both direction, and a serial schedule passes one message at each time step.

Particle-based inference methods approximate the joint distribution as a set of instantiations, called particles. The particles can be full–involving complete assignments to all the network variables χ, or collapsed– which specifies an assignment only to a subset of the variables. The simplest method is forward sampling. It involves sampling the nodes of a BN in some order so that by the time we sample a node we have values for all of its parents. When the value of a variable is observed, it is inefficient to discard samples that do not agree with the value and incorrect to fix that value. Gibbs sampling is a Markov Chain Monte Carlo method that generates successive samples by fixing the values of all variables to the previous sample and generating the value of a new variable using the conditional distribution. Unlike forward sampling, Gibbs sampling applies equally well to BNs and MNs.

9.3 Learning

A PGM consists of a graphical structure and parameters. There are two approaches to constructing a model: (i) knowledge engineering: construct a network by hand with experts help (ii) machine learning: learn model from a set of instances. Hand-constructed PGMs have many limitations: time taken to construct them vary from hours to months, expert time can be costly or unavailable, the data may change over time, the data may be huge and errors may lead to poor answers.

Since inferring PGMs is an intractable problem, i.e., *NP*-hard, it is necessary to develop scaleable approximate learning methods. Existing methods for structure learning are either score-based or constraint-based approaches. Most existing solutions are applicable only to pairwise interactions, and their generalization to arbitrary size groupings of variables is needed.

In most applications of PGMs, the graphical structures are assumed to be either known or designed by human experts, thereby limiting the machine learning problem is one of parameter estimation. Structure learning is a model selection problem which requires defining a set of possible structures and a measure to score each structure. Learning as optimization is the predominant approach with a hypothesis space consisting of set of candidate models and an objective function which is a criterion for quantifying preference over models. The learning task is to find a high-scoring model within its model class. Different choices of objective functions have ramification to results of learning. The hypothesis space is super-exponential ($2^{O(n^2)}$), with the situation worse for MNs since cliques can be of size greater than two.

9.3.1 Parameter Estimation

Parameter estimation is a building block for more advanced PGM learning: structure learning and learning from incomplete data. The data set consists of fully observed instances of the network variables $\mathcal{D} = \{\xi[1], , \xi[M]\}$.

1. Bayesian Networks

In the case of a fixed BN the parameter estimation problem is decomposed into a set of unrelated problems. Two main approaches to determine the CPDs are: maximum likelihood estimation and Bayesian parameter estimation. In the maximum likelihood approach, the likelihood function is the

probability that the model assigns to the training data. For example, in the multinomial case, where a variable X can take values $x^1, .., x^K$, the likelihood function has the form $L(\theta : \mathcal{D}) = \prod_k \theta_k^{M[k]}$, where $M[k]$ is the number of times the value x^k appears among M samples, and the maximum likelihood estimate is $\hat{\theta_k} = \frac{M[k]}{M}$.

The Bayesian approach becomes useful when the number of samples is limited. We begin with a prior distribution for the parameters and convert it to a posterior distribution based on the likelihood of observed samples. For CPTs with multi-valued discrete variables a Dirichlet prior is useful since it is conjugate to the multinomial distribution. It has the form $P(\theta) = Dirichlet(\alpha_1, ..., \alpha_K)$ where α_k are hyper-parameters and $\alpha = \sum_k \alpha_k$ with $E[\theta_k] = \frac{\alpha_k}{\alpha}$. The posterior has the form $Dirichlet(\alpha_1 + M[1], ..., \alpha_K + M[K])$. The hyper parameters play the role of virtual samples to avoid zero probabilities due to lack of samples.

2. Markov Networks

For MNs, the global partition function induces entanglements of parameters. The problem is stated as one of determining the set of parameters θ from $\mathcal{D}$ when the features $\mathcal{F}$ are known. For fixed structure problems the optimization problem is convex. i.e., (i) local minimum is the global minimum. (ii) set of all (global) minima is convex and (iii) if strict convexity holds the minimum is unique. Thus it is possible to use iterative numerical optimization, but at each step it requires inference which is expensive.

9.3.2 Structure Learning

1. Bayesian Networks

BN structure learning methods rely on three measures: (i) *deviance* from independence between variables, (ii) a decision rule that defines a *threshold* for the deviance measure to determine whether the hypothesis of independence holds, and (iii) a *score* for the structure.

Deviance from independence between a pair of variables is provided by the *chi-squared* test of independence. The Pearson's chi-squared statistic between variables X_i, X_j given a data set $\mathcal{D}$ of M samples is

$$d_{\chi_2}(\mathcal{D}) = \sum_{X_i, X_j} \frac{\Big(M[X_i, X_j] - M \cdot \hat{P}(X_i) \cdot \hat{P}(X_j)\Big)^2}{M \cdot \hat{P}(X_i) \cdot \hat{P}(X_j)} \tag{3}$$

When the variables are independent $d_{\chi 2}(\mathcal{D}) = 0$. It has a larger value when the joint count $M[X_i, X_j]$ and the expected count (under the independence assumption) differ. Another deviance measure is mutual information (which is equivalent to the *Kullback-Leibler* distance) between the joint distribution and the product of the marginals:

$$d_I(\mathcal{D}) = \frac{1}{M} \sum_{X_i, X_j} M[X_i, X_j] \log \frac{M[X_i, X_j]}{M[X_i] \cdot M[X_i]} \tag{4}$$

When the variables are independent $d_I(\mathcal{D}) = 0$ and a larger value otherwise.

A decision rule accepts the hypothesis that the variables are independent if the deviance measure is less than a threshold and rejects the hypothesis otherwise, The threshold is chosen such that the false rejection probability has a given value, say 0.05 (called the p-value).

Examples of structure scores over a data set are the log-likelihood $score_L(G : \theta : \mathcal{D}) = \ell(\hat{\theta}_G : \mathcal{D}) = \sum_{\mathcal{D}} \sum_{i=1}^{m} \log \hat{P}(X_i | paX_i)$, and the Bayesian Information Criterion (BIC) which penalizes more complex structures: $score_{BIC}(G : \theta : D) = \ell(\hat{\theta}_G : \mathcal{D}) - \frac{\log M}{2} Dim(G)$, where Dim is the number of independent parameters in G.

Approaches to BN structure learning are: constraint-based, score-based and Bayesian model averaging. In constraint-based learning the BN is viewed as a representation of independencies, but it is sensitive to failures of individual independence tests, i.e., if one test returns a wrong answer it misleads the network construction procedure. In score-based learning, the BN is viewed as specifying a statistical model where each structure given a score, with optimization to find highest score; but search may not have an elegant and efficient solution. Bayesian Model Averaging generates an ensemble of possible structures and averages the prediction of all possible structures; due to the immense number of structures, approximations are needed.

2. Markov Networks

The problem is to identify the MN structure with a bounded complexity, which most accurately represents a given probability distribution, based on a set of samples from the distribution. MN complexity is the number of features in the log–linear representation of the MN. This problem, which is *NP*-hard, has several sub-optimal solutions which may be characterizes as either constraint-based or score-based.

In the constraint-based approach, conditional independences of variables are tested on a given data set. A simple algorithm for structure learning is to determine the empirical mutual information between all pairs of variables and to keep only those edges whose values exceed a threshold. Since the constraint-based approach lacks noise robustness, requires many samples and only considers pairwise dependencies, the score-based approach is considered.

The score-based approach computes a score for a given model structure, e.g. log-likelihood with the maximum likelihood parameters. One such score is $\ell(\mathcal{F}, \theta, \mathcal{D}) - ||\theta||_1$, where the second term is L_1 regularization to prevent over-fitting. The goal is to determine the set of features as well as the parameters. A search algorithm can then be used to obtain the MN structure with the optimal score. The greedy algorithm starts from the MN without any features (the model where all variables are disjoint). Features are then introduced to the MN one by one. At each iteration, a feature is selected that brings maximum increase in the objective function value. The search can be speeded by limiting the number of candidate features to enter the MN, e.g., features whose empirical probability differs most from their expected value with respect to the current MN.

10 Key Applications

PGMs have been widely used in several fields for modeling and prediction, e.g., text analytics, image restoration and computational biology. They are a natural tool for handling uncertainty and complexity which occur throughout applied mathematics and engineering .

PGMs can account for model uncertainty, measurement noise and integrate diverse sources of data. PGMs can be used predict the probability of observed and unobserved relationships in a network. Fundamental to the idea of a graphical model is the notion of modularity where a complex system is built by combining simpler parts. Probability theory provides the glue whereby the parts are combined, ensuring that the system as a whole is consistent providing ways to interface models to data. The graph-theoretic side provides an intuitively appealing interface by which humans can model highly-interacting sets of variables. The resulting data structure lends itself naturally to designing efficient general-purpose algorithms. PGMs provide the view that classical multivariate probabilistic systems are instances of a common underlying formalism, e.g., mixture models, factor

analysis, hidden Markov models, Kalman filters and Ising models. PGMs are encountered in systems engineering, information theory, pattern recognition and statistical mechanics. Other benefits of the PGM view are that specialized techniques in one field can be transferred between communities and exploited and they provide a natural framework for designing new systems.

10.1 Visualization

PGMs are useful to visualize structure of probabilistic models. Joint distributions can be factored into conditional distributions using product rule and expressed as BNs.

10.2 Generative Models

PGMs can be used to generate samples, e.g., ancestral sampling is a systematic way of generating samples from BNs. They can be used as generative models for data, thereby circumventing often stringent privacy regulations.

10.3 Genetic Inheritance

BNs can be used to model genetic inheritance. Consider transmission of certain properties such as blood type from parent to child. Blood type of a child $B(c)$ is an observable quantity, called phenotype, that depends on the genetic makeup $G(c)$, of a person called a genotype. There are three types of CPDs for genetic inheritance. The penetrance model $P(B(c)|G(c))$ describes probabilities of different phenotypes given a person's genotype: it is deterministic for blood type. The transmission model is $P(G(c)|G(p), G(m))$, where c is a person, p and m are are the person's father and mother, respectively: each parent is equally likely to transmit either of two alleles to child. Genotype priors are $P(G(c))$. Real models are more complex. Phenotypes for late-onset diseases are not a deterministic function of genotype. A particular genotype may have a higher probability of a disease. The genetic makeup of an individual is determined by many genes. Some phenotypes depend on many genes. Multiple phenotypes depend on many genes. An example BN representing genetic inheritance of DNA is shown in Fig. 1.

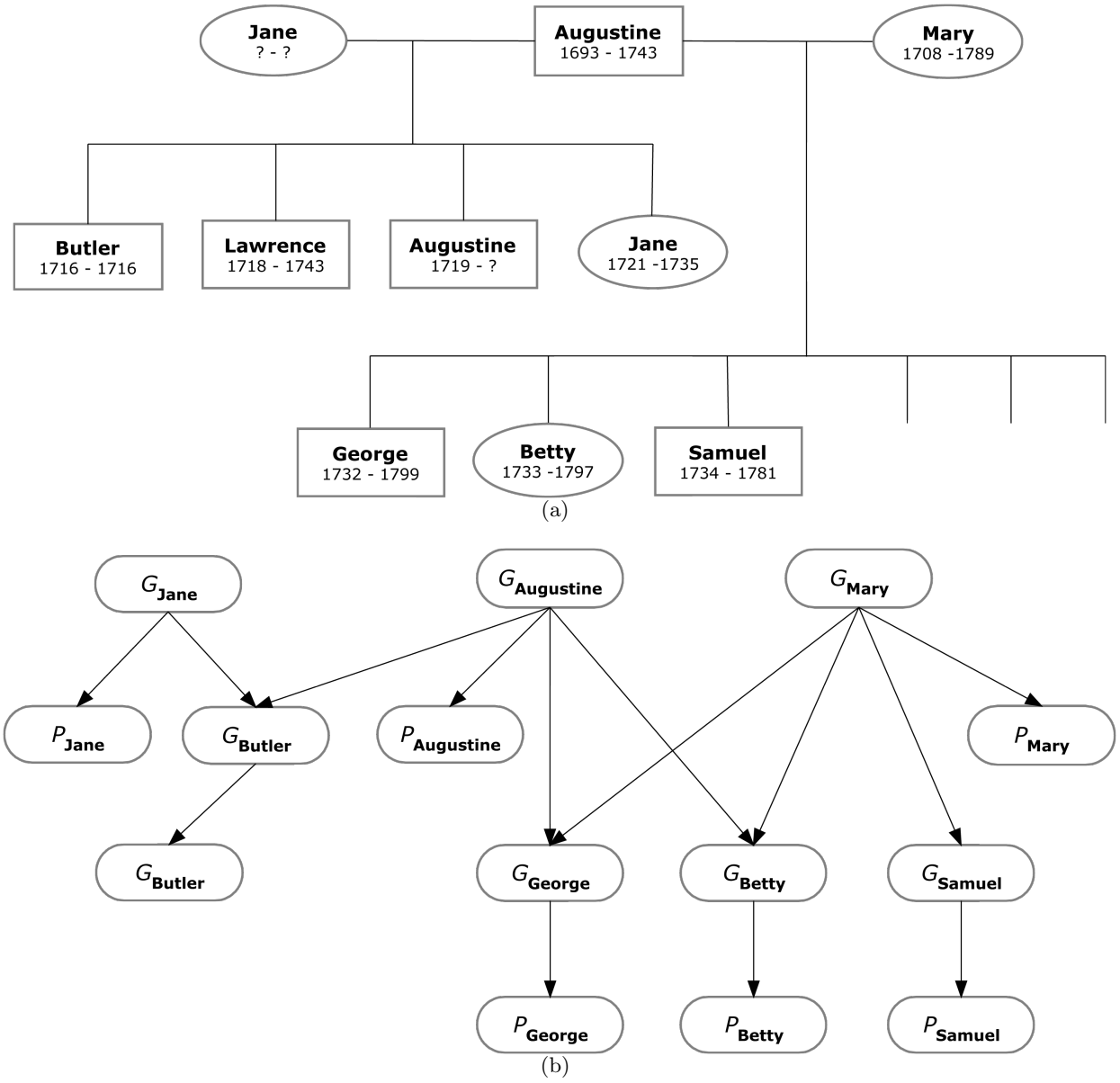
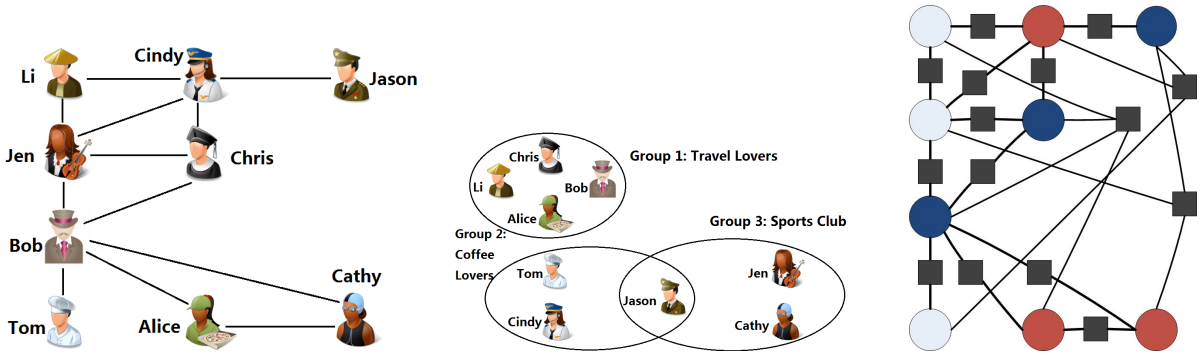


Figure 1: Genetic inheritance based on DNA represented as a BN: (a) a family tree, and (b) Bayesian network of genotypes and phenotypes.



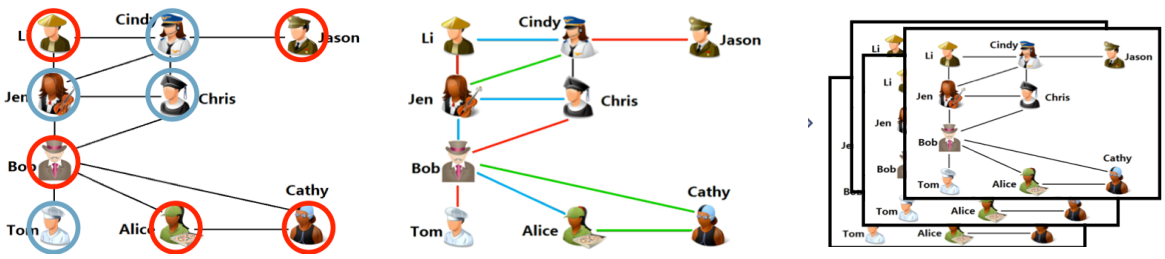
(a) (b) (c)

Figure 2: Statistical dependencies between variables in an OSN: (a) pairwise dependencies, (b) higher-order groupings in an affiliation network, and (c) a possible MN factor graph where filled circles represent actors with known gender and squares represent factors.

10.4 Social Networks

On-line social networks (OSNs) provide a clear way of analyzing the structure of whole social entities[2]. An (OSN) can be viewed as a graph where the *nodes* represent individuals or organizations (actors) and the *edges* are dyadic ties that represent the dependencies between them (Fig. 2(a)). Actors have a set of attributes, e.g. gender, age, hobbies. OSNs usually have higher order groupings of actors, e.g., travel lovers, sports club and coffee lovers (Fig. 2(b)). They can be represented as MNs using a factor graph representation (Fig. 2(c)). The dependencies need not follow the OSN links. Nodes can also be links between actors taking values {0,1} or $[0, 1]$ and dependencies are connections to same actors.

Some inference (predictive modeling) tasks with OSNs are: (i) *Predictive Modeling*: strength of a given connection in the future given current state of the network (structure, attributes) and its previous history, (ii) *Group/Community Detection*: reveal groups of users that are interconnected according to a given criteria given current network structure and attributes, (iii) *Behavior Pattern Extraction*: reveal hidden connections between attribute values of users or between some of their actions and attributes given current structure of the OSN and user attribute values and history of changes in the network, (iv) *Actor Classification*: label each user according to some criteria given network structure and user attribute values (Fig. 3(a)), (v) *Link Classification*: label each link according to some criterion given network structure and user attribute values and history of



(a) (b) (c)

Figure 3: Some statistical inference problems in OSN big data: (a) actor classification, (b) link classification, and (c) data generation.

changes (Fig. 3(b)), and (vi) *Artificial Network Data Generation*: construct a generative model to produce artificial data sets resolving privacy issues that occur while working with real OSN data (Fig. 3(c)).

11 Future Directions

With the enormous amounts of data being generated from instruments, cameras, internet transactions, email, genomics, etc., statistical inference with big heterogeneous data sets is becoming increasingly important. When the number of variables become large, the amount of data needed for exact statistical modeling becomes impractical. Heterogeneity of attributes describing complex relations gives rise to a number of unique statistical and computational challenges, e.g., the number of parameters needed to model the distributions becomes exponential and the parameter inference algorithms become intractable. This is where PGMs become useful as they provide approximations of exact distributions. An example of big data that can be naturally analyzed using PGMs are OSNs of kinship, email, affiliation groups, mobile communication devices, bibliographic citations and business interactions.

12 Cross-References

Gibbs Sampling (00146)

Markov Monte Carlo Model (00150)

Models of Social Networks (00182)

Probabilistic Analysis (00155)

13 Acknowledgements

The author wishes to thank his teaching and research assistants for the PGM course (CSE 674 at the University at Buffalo). In particular Dmitry Kovalenko, Yingbo Zhao, Chang Su and Yu Liu for many discussions.

14 References

[1] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[2] Stanley Wasserman and Katherine Faust. Social network analysis in the social and behavioral sciences. In *Social Network Analysis: Methods and Applications*, page 127. Cambridge University Press, 1994.

15 Recommended Reading

- C. Bishop, *Pattern Recognition and Machine Learning*. Springer, New York, 2006; has a chapter on graphical models which provides a good introduction.
- D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*, MIT Press, 2009; a detailed treatise on PGMs.

- S. Srihari, *Lecture slides and videos on machine learning and PGMs* at `http://www.cedar.buffalo.edu/~srihari/CSE574` .