Probabilistic Graphical Models: Recap, Examples
10708, Fall 2020
Pradeep Ravikumar

1 Introduction

So far we have studied: (a) undirected graphical models, together with its close cousin, factor graphical models, (b) directed graphical models, and (c) chain graphical models. In each of these, a graph specifies a set of distributions, in one of two ways.

In the first, the graph specifies a set of conditional independence *constraints*, and the associated set of distributions is all those distributions that satisfy these constraints. The constraints can take the form of so-called global Markov conditional independence constraints, local Markov, and finally pairwise Markov conditional independence constraints. Each of these classes of constraints in turn specify a different set of distributions, but under certain mild conditions (distributions are positive in the case of UGMs, and have densities in the case of DGMs), these different types of Markov constraints all specify the same set of distributions given a graph.

The other way in which a graph specifies a set of distributions is as all those distributions that factor according to the graph, in the sense that their density or PMF can be written as the product of "local factors", where the notion of locality depends on whether it is a UG, DAG, or CG.

The fascinating thing about PGMs is that under the erstwhile specified mild conditions, both of these specifications of sets of distributions — one via Markov conditional independence constraints, and one via algebraic factorization of the distribution — are equivalent. With respect to computational advantages, for inference i.e. probabilistic reasoning, for learning, the algebraic factorization characterization is most important. But as we will see, one could devise clever algorithms for approximate probabilistic reasoning as well as learning that heavily use the fact that the associated distributions satisfy the associated set of Markov conditional independence constraints, so one could also view the first characterization as having computational importance.

So far we have studied the representational theory of probabilistic graphical models. We will next study the other two legs: probabilistic reasoning given a specific PGM, and then, learning a PGM given data from some unknown distribution, ideally the unknown target PGM. Before we do so, let us briefly review some applications of PGMs.

2 Hierarchical Bayesian Models

In Bayesian infernce, all model quantities are considered random variables, not just the observation variables, but also the unknown model parameters. Let $X \sim P_X(\cdot|\theta)$, and $\theta \sim P_\theta(\cdot)$. We could express this a simple DGM with variables X and θ and an edge from θ to X. One caveat with Bayesian inference is that the requirement that we specify a prior P_θ. One approach to address this is via more priors: $\theta \sim \mathcal{P}_\theta(\cdot|\gamma)$, and $\gamma \sim P_\gamma(\cdot)$. One could stop here, or of course go deeper, which leads us to the broad class of hierarchical Bayesian models, all of which can be expressed very naturally as DGMs. The characterization as DGMs is not just as an aside, but has computational advantages when devising approximate inference probabilistic reasoning algorithms.

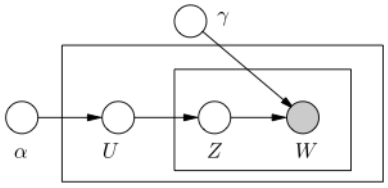

Figure 1: Topic Model

A popular example of this is a topic model, as shown in Figure 2. This is a so-called "bag of words" document model, where the document is essentially represented as a set of indicators of word presence, thus removing information about word ordering in the document. Here, each document is thus a set of observed word variables W. Each word W is drawn from a multinomial distribution specified by a word-specific "topic" variable $Z \in [K]$. Each of these word-specific topic variables is drawn from a document-specific multinomial distribution U. And these multinomial distributions U are drawn from a Dirichlet distribution with parameter α. Note that each word has its own mixture model over topics, where each document has its own mixture model weights, which in turn are drawn from a common Dirichlet prior distribution.

3 Constraint Satisfaction Problems

In a lot of industrial AI and operations research applications, and indeed when we wish to solve combinatorial optimization problems in general, one has to solve so-called constraint satisfaction, or satisfiability problems. The best known example is 3-SAT: we have a collection of boolean variables $(X_1, \ldots, X_p) \in \{0,1\}^p$, and a collection of clauses each of which is the OR of three literals (where a literal is any boolean variable or its negation), for instance $\bar{x}_1$ OR x_4 OR x_5. The 3-SAT problem asks whether there exists an assignment to the boolean variables such that all clauses are true i.e. satisfied. Suppose a clause C involves

variables X_s, X_t, X_u, and suppose we associate whether a variable is negated or not in the clause C via constants $z_s, z_t, z_u \in \{0, 1\}$ associated with the clause C, so that we can write the clause compactly as:

$$\psi_{stu}(x_s, x_t, x_u) = x_s^{z_s}(1 - x_s)^{1-z_s} \vee x_t^{z_t}(1 - x_t)^{1-z_t} \vee x_u^{z_u}(1 - x_u)^{1-z_u}.$$

Thus, if $z_s = z_t = z_u = 1$, we get the simple OR clause $x_s \vee x_t \vee x_u$. Thus each clause specifies a factor function over three variables. Let F denote the set of clauses. Now consider the factorization:

$$P(x) = \frac{1}{Z} \prod_{(s,t,u)\in F} \psi_{stu}(x_s, x_t, x_u).$$

It can be seen that if there is no satisfiable assignment, then at least one of the clause factor functions would be equal to zero for all x, so that $P(x) = 0$ for all x. But if there were a satisfiable assignment x, then all the clause factor functions would be equal to one for the assignment x, so that $P(X)$ specifies a uniform distribution over all satisfiable assignments. Thus performing probabilistic reasoning on the factor graph would allow us to solve 3-SAT.

4 Error Correcting Coding

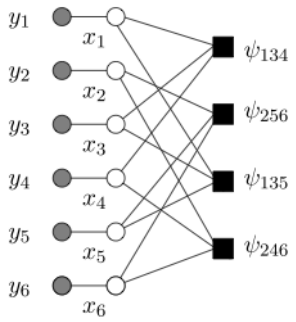

Figure 2: Example: Error Correcting Code Factor Graph

Suppose we wish communicate a series of bits $(X_1, \ldots, X_p) \in \{0, 1\}^p$. Unfortunately, the communication medium will add noise, so that the receiver receives potentially flipped bits $(Y_1, \ldots, Y_p) \in \{0, 1\}^p$. The task the receiver then faces is: given Y, what was the actual message sent X? This is an important question to answer for most distant communications through noisy channels, as with cell-phones, etc. One approach to ensure that we recover the exact X is to add redundancy to the bits we send, so that we could correct for the randomly flipped bits. We could for instance repeat each bit three times. But as it turns out there is a much more efficient approach, based on what is called error correcting codes. The simplest of these are so-called parity check codes. It is a set of constraints, each of which entails

that a group of bits sum up to one modulo two. This can be represented by a simple factor function:

$$\psi_{stu}(x_s, x_t, x_u) = I[x_s \oplus x_t \oplus x_u = 1],$$

where the specific code above checks for the parity of bits X_s, X_t, X_u. Thus each parity check corresponding to a factor function over the corresponding bits. Let F denote the set of parity check factors. Suppose further that we have a noise model for how bits can be corrupted: $P(Y_s|X_s)$, for all $s \in V$. We can then have the following factor graph model for the unknown message X, conditioned on observing the noisy transmission Y:

$$P(x|y) = \frac{1}{Z} \prod_{s \in V} P(y_s|x_s) \prod_{f \in F} \psi_{(s,t,u) \in F} \psi_{stu}(x_s, x_t, x_u).$$

The decoding problem can then be cast as probabilistic reasoning with the above factor graph. Figure 4 shows one such factor graph model with multiple parity check factors.

5 Image Processing

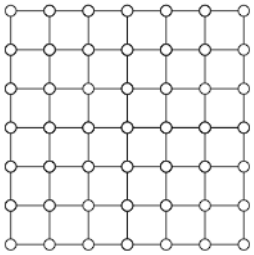

Figure 3: Example: 2D Grid UG

Any 2D image could be viewed as a matrix of pixel intensities, with one observation per pixel. We could thus associate a random variable per pixel to get a corresponding random vector $Y = (Y_{st})_{s,t \in [p]}$ for a $p \times p$ image. A natural graph associated with such an image is a 2D lattice grid graph as in Figure 5. Here each pixel is connected to its four nearest neighbors with respect to the 2D pixel ordering. In certain contexts, we could condition on the observed pixel intensities Y, and associate a different random vector $X = (X_{st})_{s,t \in [p]}$ with the set of pixels, depending on the tasks, and we are interested in $P(X|Y)$. For instance, with image denoising, X is the "true" image, while the observed Y is the noisy or corrupted image, and we are interested in the distribution of the true image. Similarly for image segmentation, each $X_{st} \in [K]$ is the segment label associated with the corresponding pixel. The UGMs in this case are simple pairwise models, with cliques over nodes and edges:

$$P(X|Y) = \prod_{s \in [p]} \phi_s(X_s, Y) \prod_{s \neq t} \phi_{st}(X_s, X_t),$$

where the node clique potentials $\phi_s(X, Y_s)$ are typically set to the score of a local classifier $P(Y_s|X)$, while the edge clique potentials acts as a global regularizer for instance encouraging that nearby pixels have the same value e.g. $\phi_{st}(X_s, X_t) = -\beta d(X_s, X_t)$, where $d(\cdot, \cdot)$ is some distance metric.

Most tasks in image processing can be cast as probabilistic reasoning over such UGMs.

6 Bioinformatics

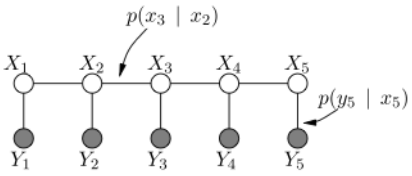

Figure 4: Example: HMMs

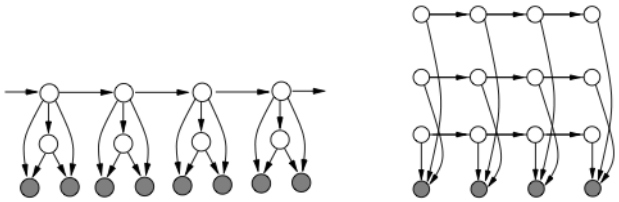

Figure 5: Example: More Complex HMMs

A lot of bioinformatics data consist of sequences, and a class of models which we will be studying shortly, Hidden Markov Models (HMMs), which have a sequence of hidden states, which in turn result in observed sequence. Instances of sequences include genomic sequence, which is a sequence of nucleotides (an example task: figuring out which parts of the sequence are genes and which are not: a counterpart of image segmentation). Figures 6 and 6 show some instances of such HMMs.

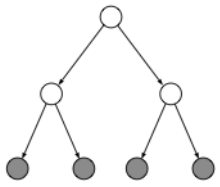

Figure 6: Example: Phylogenetic Trees

Tree-structured models are also very critical in bio-informatics. For example phylogenetic trees is a tree-structured graphical model in which a set of biological variables e.g. species are at the leaves, and the latent internal nodes correspond to latent variables e.g. ancestral species.

7 Language and Speech Processing

HMMs are also heavily used in natural language processing, since a lot of data in NLP consists of sequences e.g. a sentence is a sequence of words. And the different tasks have corresponding latent variables. For instance, Part Of Speech (POS) tagging of words in a sequence would have latent variables associated with each word corresponding to the POS tags.