Lecture 1: Probabilistic Graphical Models

Tianpei Xie

Aug. 25th., 2022

Contents

1 Introduction **2**

2 Probabilistic Graphical Models **2**
2.1 Directed graphical models (Bayesian networks) 2
2.2 Undirected grapical models (Markov networks) 3
2.3 Factor graph representation 4
2.4 Important notes 4
2.5 Representation of factors 5
2.5.1 Tabular representation 5
2.5.2 Function approximation 6

3 Conditional independence **9**

4 Examples **11**
4.1 Ising Models (binary Markov random field) 11
4.1.1 Metric Labeling and Potts Model 12
4.2 Gaussian graphical models (GGM) 12
4.3 Hidden Markov Chain (HMM) 14
4.4 Conditional Random Field (CRF) 16
4.5 Latent Dirichlet allocation (LDA) 18
4.6 Restricted Boltzmann machine (RBM) 19

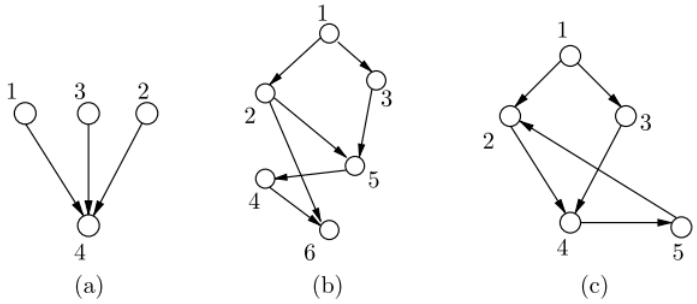


Fig. 2.1 (a) A simple directed graphical model with four variables (X_1, X_2, X_3, X_4). Vertices $\{1,2,3\}$ are all parents of vertex 4, written $\pi(4) = \{1,2,3\}$. (b) A more complicated directed acyclic graph (DAG) that defines a partial order on its vertices. Note that vertex 6 is a child of vertex 2, and vertex 1 is an ancestor of 6. (c) A forbidden directed graph (nonacyclic) that includes the directed cycle $(2 \to 4 \to 5 \to 2)$.

Figure 1: The directed graphical model and the ordering of its vertices based on directed edges.

1 Introduction

This chapter mainly covers books [Wainwright et al., 2008, Koller and Friedman, 2009]. **Probabilistic graphical models** bring together graph theory and probability theory in a powerful formalism for multivariate statistical modeling. In graphical models, the *graph* is

- a *representation* of *a set of conditional independence assertions* among variables; and
- a *data structure* that represent high-dimensional joint distribution via *factorization*.

PGMs are useful tool for ***multivariate*** and ***high dimensional statisticical analysis*** in which people need to understand the inter-dependencies among all variables of observations. We have notes on PGM before so this note is for quick reference of methods and ideas.

2 Probabilistic Graphical Models

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the set of edges, a *probabilistic graphical model (PGM)* defines a *joint probability distribution* $p(x_1, \ldots, x_m)$ that ***factorizes*** according to $\mathcal{G}$. Here each variable x_i corresponds to a node $v_i \in \mathcal{V}$ and the existence of an edge $(s,t) \in \mathcal{E}$ (or the ***absence*** of an edge (s,t)) defines a **statistical dependency** (or **conditional independency**) relation between x_s and x_t.

In particular, according to the type of edges, we can define two different types of PGMs:

2.1 Directed graphical models (Bayesian networks)

If the edge $(s,t) \in \mathcal{E}$ is **directed** (referred $s \to t$), i.e. there is a distinction between (s,t) and (t,s), the corresponding graphical models are referred as *directed graphical models*. Note that since a sequence of directed edges defines a logic flow, a circle in the graph would create undesireable contradiction. Therefore, we assume that $\mathcal{G}$ is a **directed acyclic graph (DAG)**, meaning that every edge is directed, and that the graph contains no directed cycles.

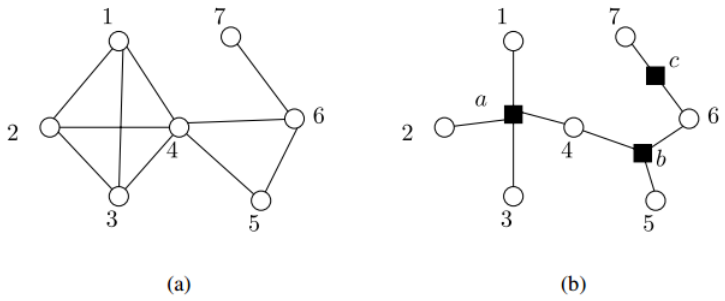


Fig. 2.2 Illustration of undirected graphical models and factor graphs. (a) An undirected graph on $m = 7$ vertices, with maximal cliques $\{1,2,3,4\}$, $\{4,5,6\}$, and $\{6,7\}$. (b) Equivalent representation of the undirected graph in (a) as a factor graph, assuming that we define compatibility functions only on the maximal cliques in (a). The factor graph is a bipartite graph with vertex set $V = \{1,\ldots,7\}$ and factor set $F = \{a,b,c\}$, one for each of the compatibility functions of the original undirected graph.

Figure 2: The undirected graphical model and its factor graph representation.

For any such DAG, we can define a **partial order** on the vertex set $\mathcal{V}$ by the notion of *ancestry*: we say that node s is an *ancestor* of u if there is a *directed path* $(s, t_1, t_2, \ldots, t_k, u)$. This is referred as ***topological ordering***. Given a DAG, for each vertex u and its parent set $\pi(u) = \{s \in \mathcal{V} : (s \to u) \in \mathcal{E}\}$, the conditional probability $p_s(x_s|x_{\pi(s)})$ is a **non-negative function** over $(x_s, x_{\pi(s)})$ and is **normalized** for all x_s, i.e. $\int p_s(x_s|x_{\pi(s)})dx_s = 1$.

The **directed graphical model** thus factorizes the joint distribution into a set of *local functions* $\{p_s(x_s|x_{\pi(s)}) : s \in \mathcal{V}\}$ according to the ancestor relations defined in $\mathcal{G}$

$$p(x_1, \ldots, x_m) = \prod_{s \in \mathcal{V}} p_s(x_s|x_{\pi(s)}). \tag{1}$$

This class of models are also referred as ***Bayesian networks*** [Koller and Friedman, 2009].

2.2 Undirected grapical models (Markov networks)

In the *undirected* case, the probability distribution factorizes according to functions defined on the ***cliques*** of the graph. A clique C is a ***fully connected subset*** of the vertex set $\mathcal{V}$, meaning that $(s,t) \in \mathcal{E}$ for all $s,t \in C$. Let us associate with each clique $\mathcal{C}$ a ***compatibility function*** $\psi_C : (\otimes_{s \in C} \mathcal{X}_s) \to \mathbb{R}_+$, where $\otimes_{s \in C} \mathcal{X}_s$ denotes the Cartesian product of the state spaces of the random vector X_C. Consequently, the compatibility function ψ_C is a local quantity, defined only for elements x_C within the clique.

With this notation, an *undirected graphical model* – also known as a ***Markov random field (MRF)***, *Markov Network*, or a *Gibbs distribution* – is a collection of distributions that *factorize* as

$$p(x_1, \ldots, x_m) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C), \tag{2}$$

where Z is a constant chosen to ensure that the distribution is normalized. The set $\mathcal{C}$ is often taken to be the *set of all* ***maximal cliques*** *of the graph*, i.e., the set of cliques that are *not* properly

contained within any other clique. Note that any representation based on nonmaximal cliques can always be converted to one based on maximal cliques by redefining the compatibility function on a maximal clique to be the *product* over the compatibility functions on the *subsets* of that clique.

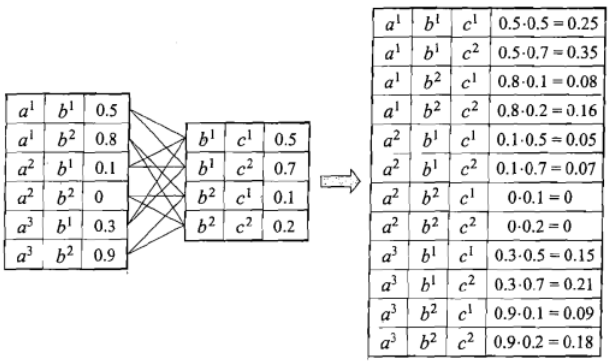
2.3 Factor graph representation

To order to emphasize the computational structure and the factorization of dependencis, we introduce the **factor graph representation**. A factor graph is a **bipartite graph** $\mathcal{G}' = (\mathcal{V}, \mathcal{F}, \mathcal{E}')$ where $\mathcal{V}$ is the same as the original graph. The set $\mathcal{F}$ is the index set of *factors*, e.g. the local functions for parent-child relationships or the compatibility function on cliques. $\mathcal{E}'$ is a new edge set, joining only vertices $s \in \mathcal{V}$ to factors $a \in \mathcal{F}$. In particular, edge $(s, a) \in \mathcal{E}$ if and only if x_s *participates* in the factor indexed by $a \in \mathcal{F}$. Figure 2 shows the undirected graphical model and its factor graph representation.

2.4 Important notes

There are several important notes when learning graphical models:

- A probabilistic graphical model is a model ***over*** graph $\mathcal{G}$, **not** a model ***of*** graph $\mathcal{G}$. The latter is covered by ***statistical network theory*** [Goldenberg et al., 2010, Jackson, 2010, Newman, 2018]. The difference is that while the graph in PGM is used to model statistical dependencies over variables and to develop inference algorithms, the sample data $\boldsymbol{X}_i := [X_{1,i}, \ldots, X_{m,i}]$ is still just a m-dimensional vector drawn i.i.d. from this distribution. On the other hand, a sample generated by a statistical network is a **graph itself**, i.e. $(\boldsymbol{X}_i, G_i)$ a pair of *node* samples, as well as graph topologies via *edge* samples. Sometimes this graph-like sample is also referred as a ***graph signal*** [Ortega et al., 2018]. The graph in PGM is a ***structure*** **of models** and **computation** not an **observation**. The graph of PGM also does not reflect the topolocial dependencis between two variables in the observation but only its statistical dependencies.
- A **deep learning model** [Goodfellow et al., 2016] is a probabilistic graphical model with ***hierarchical representation***. Unlike most of PGM we discussed, most of nodes of deep learning model represent **latent variables**, i.e. unobserved variables. These variables exist to decode the complicated relationship between observed input nodes. Deep learning models explore the *overparameterized function space* in order to find suitable compatibility function ψ.
- Given data samples, one need to first propose a graph representation of distribution before starting the inference and learning process. Usually, this graph comes from domain expertise. But most of time, it is unclear how to find a proper graph representation of the problem. The **structural learning** of graphical models is very **challenging**. Unlike a *deep learning model* which only cares for ***function approximation***, a PGM cares if the graph structure indeed represents the **true underlyiing conditional independencies**, which requires a series of testing.



a^1	b^1	0.5
a^1	b^2	0.8
a^2	b^1	0.1
a^2	b^2	0
a^3	b^1	0.3
a^3	b^2	0.9

b^1	c^1	0.5
b^1	c^2	0.7
b^2	c^1	0.1
b^2	c^2	0.2

a^1	b^1	c^1	$0.5 \cdot 0.5 = 0.25$
a^1	b^1	c^2	$0.5 \cdot 0.7 = 0.35$
a^1	b^2	c^1	$0.8 \cdot 0.1 = 0.08$
a^1	b^2	c^2	$0.8 \cdot 0.2 = 0.16$
a^2	b^1	c^1	$0.1 \cdot 0.5 = 0.05$
a^2	b^1	c^2	$0.1 \cdot 0.7 = 0.07$
a^2	b^2	c^1	$0 \cdot 0.1 = 0$
a^2	b^2	c^2	$0 \cdot 0.2 = 0$
a^3	b^1	c^1	$0.3 \cdot 0.5 = 0.15$
a^3	b^1	c^2	$0.3 \cdot 0.7 = 0.21$
a^3	b^2	c^1	$0.9 \cdot 0.1 = 0.09$
a^3	b^2	c^2	$0.9 \cdot 0.2 = 0.18$

Figure 4.3 An example of factor product

Figure 3: The tabular representation for each factor.

2.5 Representation of factors

A graphical model is a ***distributed system***. **Factors** are basic **building blocks** for graphical model. Since graphical models factorize over graph topology $\mathcal{G}$, they store their *information locally* within factors. Each factor covers a close *neighborhood* of each node. For Bayesian networks, it is the **conditional probability** $p_s(x_s|x_{\pi(s)})$ and for Markov random fields, it is the **compatiablity function** $\psi_C(x_C)$ for each clique C. The **key idea** when learning graphical models is to understand how local information are ***represented***, ***stored*** as well as to learn how they are ***propagated*** across graph to obtain global information.

2.5.1 Tabular representation

If the random variables are all **discrete with finite support** and the **number** of random variables involved in each *factor* is small (like $2-4$), a tabular representation of these local factors are available. This requires that the underlining graph $\mathcal{G}$ is ***simple*** such as ***tree****-structured* so that the **size of neighborhood** for a given node or a **clique** is very small. Figure 3 shows a tabular form of factor product.

The advantages of tabular representation:

- **Simple and natural**: A table representation is natural and simple for many applications in statistics such as ***contingency table analysis***. For these applications, the multi-dimensional data forms a **high-dimensional tensor**, each storing the occurrence frequency of each *possible outcome* under a set of configurations under many repeated experiments.
- **Easy for parallel computation**: A key step in inference is to compute the **product of factors** (e.g. the *marginalization*). Since each factor is represented by a tensor, this is simply a tensor product computation, which is easy to perform under parallel computing devices such as GPUs. Belief propagation also requires parallel computation during message passing.
- **Efficient inference**: Each factor is represented independently. This allows marginalization and maximization within each local factor. Tabular representation is suitable for efficient *dynamic programming algorithms* such as ***belief propagation*** (*sum-product* and *max-product*).

- **High discrimination provides theoretical guarantee**: For each random variables, the frequence for each outcome is stored in table independently. Therefore, updating the probability value for one variable wil *not affect* the probability of other variables. In other words, random variables are fully discriminated in the table. This allows us to develop theories that prove the optimality with convergence guarantee.

The disadvantages include

- **Not supporting continous variables**. Table can only represent discrete variables or discretized version of original function. Thus it easily introduces the quantization error in the process.
- **Not suitable for high dimensional factor**: Each factor in the table cannot be involve too many variables. Otherwise, the common operations within factors would be too costly.
- **High cost of storage** The size of tensor is determined by the **dimensionality** of factor as well as the **cardinality** of random variable domain. It can quickly explode when the graph is *growing* and forms a **big clique**. This is typical in *dynamic* graphical models.

2.5.2 Function approximation

The factors can be represented via **parameterization**: $\boldsymbol{\eta} \to \psi_{\boldsymbol{\eta}}$. There are mainly two ways: the *exponential family*, which is essentially **linear mapping**, and the *deep neural network*, which is overparameterized **nonlinear mapping**.

- The commonly used function representation for distributions are the exponential famlity. The joint distribution $p(\boldsymbol{x})$ follows the canonical form ***exponential famlity*** of distribution

$$\begin{aligned} p(x_1, \ldots, x_m) = p(\boldsymbol{x}; \boldsymbol{\eta}) &= \exp\left(\langle \boldsymbol{\eta}\,,\, \boldsymbol{\phi}(\boldsymbol{x}) \rangle - A(\boldsymbol{\eta})\right) h(\boldsymbol{x}) \nu(d\boldsymbol{x}) \\ &= \exp\left(\sum_{\alpha} \eta_\alpha \phi_\alpha(\boldsymbol{x}) - A(\boldsymbol{\eta})\right) h(\boldsymbol{x}) \nu(d\boldsymbol{x}) \end{aligned} \tag{3}$$

  where ϕ is a feature map and $\boldsymbol{\phi}(\boldsymbol{x}_C)$ defines a set of ***sufficient statistics*** (or ***potentials***). The normalization factor is defined as

$$A(\boldsymbol{\eta}) := \log \int \exp\left(\langle \boldsymbol{\eta}\,,\, \boldsymbol{\phi}(\boldsymbol{x}) \rangle\right) h(\boldsymbol{x}) \nu(d\boldsymbol{x}) = \log Z(\boldsymbol{\eta})$$

  $A(\boldsymbol{\eta})$ is also referred as ***log-partition function*** or *cumulant function*. The parameters $\boldsymbol{\eta} = (\eta_\alpha)$ are called ***natural parameters*** or *canonical parameters*. $\boldsymbol{\eta} \in \{\boldsymbol{\eta} \in \mathbb{R}^d : A(\boldsymbol{\eta}) < \infty\}$, which is called *natural parameter space*. Note that $A(\boldsymbol{\eta})$ is a convex function.

  In exponential family, due to property of exponent, we can formulate the (unnormalized) local functions as a exponential family too

$$\phi_C(\boldsymbol{x}_C; \boldsymbol{\eta}_C) = \exp\left(\sum_{k_C} \eta_{k_C} \phi_{k_C}(\boldsymbol{x}_C)\right)$$

  Commonly known distribution and there natural parameterization:

– Bernoulli distribution $\mathrm{B}(x;p)$: ν = Counting measure, $\eta = \log(p/(1-p))$, $\phi(x) = x$

$$\begin{aligned}\langle \eta\,,\, \phi(x)\rangle &= \log\left(\frac{p}{1-p}\right)x \\ A(\eta) &= -\log(1-p) = \log(1+\exp(\eta))\end{aligned}$$

– Gaussian distribution $\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma})$: ν = Lebesgue measure $\mathbb{R}^d$, $h(\boldsymbol{x}) = \frac{1}{(2\pi)^d}$,

$$\boldsymbol{\eta} = \left(\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}, -\frac{1}{2}\mathrm{vec}(\boldsymbol{\Sigma}^{-1})\right) := \left(\boldsymbol{\theta}, -\frac{1}{2}\mathrm{vec}(\boldsymbol{\Theta})\right) \tag{4}$$

$$\boldsymbol{\phi}(\boldsymbol{x}) = (\boldsymbol{x}, \mathrm{vec}(\boldsymbol{x}\boldsymbol{x}^T)) \tag{5}$$

$$\begin{aligned}\langle \boldsymbol{\eta}\,,\, \boldsymbol{\phi}(x)\rangle &= \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - \frac{1}{2}\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x} = \boldsymbol{x}^T\boldsymbol{\theta} - \frac{1}{2}\boldsymbol{x}^T\boldsymbol{\Theta}\boldsymbol{x} \\ A(\boldsymbol{\eta}) &= \frac{1}{2}\left(\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \log\det|\boldsymbol{\Sigma}|\right) = \frac{1}{2}\left(\boldsymbol{\theta}^T\boldsymbol{\Theta}^{-1}\boldsymbol{\theta} - \log\det|\boldsymbol{\Theta}|\right)\end{aligned} \tag{6}$$

– Poisson distribution Possion(λ): ν = Counting measure $h(x) = 1/(x!)$, $\eta = \log(\lambda)$, $\phi(x) = x$

$$\begin{aligned}\langle \boldsymbol{\eta}\,,\, \boldsymbol{\phi}(x)\rangle &= \log(\lambda)x \\ A(\eta) &= \lambda = \exp(\eta)\end{aligned}$$

– Gamma distribution $\Gamma(\alpha,\lambda)$: ν = Lebesgue measure $(0,\infty)$, $\boldsymbol{\eta} = (-\lambda, \alpha-1)$ and $\boldsymbol{\phi}(x) = (x, \log(x))$

$$\begin{aligned}\langle \boldsymbol{\eta}\,,\, \boldsymbol{\phi}(x)\rangle &= -\lambda x + (\alpha-1)\log(x) \\ A(\eta) &= \log(\Gamma(\alpha)) - \alpha\log(\lambda) = \log(\Gamma(\eta_2+1)) - (\eta_2+1)\log(-\eta_1)\end{aligned}$$

- We can *re-parameterize* the exponential family by choosing mean parameter $\boldsymbol{\theta}$ as parameter when $\boldsymbol{\eta} := \boldsymbol{\eta}(\boldsymbol{\theta})$. In (3), if $\boldsymbol{\eta} := \boldsymbol{\eta}(\boldsymbol{\theta}) = \boldsymbol{\theta}$, we call it canonical form.

$$\begin{aligned}p(x_1,\ldots,x_m) = p(\boldsymbol{x};\boldsymbol{\theta}) &= \exp\left(\langle \boldsymbol{\eta}(\boldsymbol{\theta})\,,\, \boldsymbol{\phi}(\boldsymbol{x})\rangle - A(\boldsymbol{\eta}(\boldsymbol{\theta}))\right) \\ &= \exp\left(\sum_\alpha \eta_\alpha(\boldsymbol{\theta})\phi_\alpha(\boldsymbol{x}) - A(\boldsymbol{\eta}(\boldsymbol{\theta}))\right)\end{aligned} \tag{7}$$

From [Wainwright et al., 2008] we can see that the form in (3) and (7) are both ***conjugate*** to each other based on convex analysis.

– A special form of exponential family is the ***generalized linear models (GLMs)***, when $p(x_s|x_C)$ follows exponential family, $\phi(\boldsymbol{x}) = \langle \boldsymbol{\theta}\,,\, \boldsymbol{x}\rangle$,

$$\boldsymbol{\mu} = \mathbb{E}_{\boldsymbol{\eta}}\left[\boldsymbol{\phi}(\boldsymbol{x})\right] = g^{-1}\left(\langle \boldsymbol{\theta}\,,\, \boldsymbol{x}\rangle\right) \tag{8}$$

where g is called the ***link function***, $\langle \boldsymbol{\theta}\,,\, \boldsymbol{x}\rangle$ is referred as linear predictor or system components. $g = (\nabla A)^{-1}$ where $A(\boldsymbol{\eta})$ is the log-partition function.

- The other way to parameterize the local function is via ***deep neural network***. See [Bengio et al., 2009, Goodfellow et al., 2016, Kingma et al., 2019] for detailed discussion on neural networks.

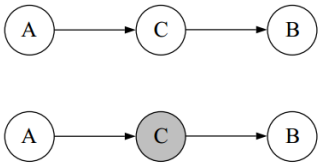
$$p_s(\boldsymbol{x}_s|\boldsymbol{x}_{\pi(s)}) := p_s(\boldsymbol{x}_s|\boldsymbol{\eta}) \quad \text{where } \boldsymbol{\eta} = \text{deep-net}(\boldsymbol{x}_{\pi(s)}). \tag{9}$$

Compare to tabular representation, the function approximation has several advantages:

- **Efficient factor representation**: compare to the size of tensor for each factor, the parameterized function just need to maintain its parameter, which is much less compared to the dimensionality and cardinality of random vectors.
- **Support continuous variables**: Tabular representation is not for continous variables. For parameterized function approximator, one can build factors using continous input.
- **Convex analysis**: Learning exponential families involves solving **convex optimization** problems. The concept of *conjugate duality* allows us to formulate it as *maximum entropy estimation*. The *smoothness* of *log-partition function*, its *variational form* via *negative entropy function* and its gradient map etc. provide critical tools to analyze and develop efficient *variational inference* algorithms with convergence guarantee.
- **Universal function approximator**: Using simple nonliner functions, one can build a complex network that are capable of representing any functions. This is an important theorem behind the *artificial neural networks* which is a type of graphical model (with hierarchical bipartite graph) using simple non-linear function to approximate factors.

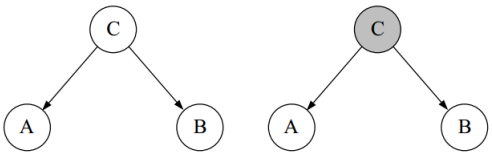
The disadvantages are

- **Integration results lack explicit forms**. *Integration* over a large space is required for tasks such as *margalization*, the *log-partition function* A and the *entropy function* $-A^*$ etc. Except for a few cases, the result of integration in general graphical model cannot be represented in **explicit function form**. *Approximation* via ***Monte Carlo sampling methods*** (such as *MCMC, Gibbs sampling*) [Shapiro, 2003] has been introduced. The other way is via ***variational methods***. Finally, one can rely on auto-differentiation such as the ***backpropagtion*** to find local optimal solution for inference.
- **Approximate inference**: Graphical models such as exponential families are defined using joint distributions. Both the log-partition function A and the negative entropy function A^* lack of explicit form. Moreover, they are not decomposable according to graph topology $\mathcal{G}$. This makes inference algorithms such as belief propagation very difficult to implment. One has to rely on *variational methods* to **approximate** the objective functions, adding additional apprximation errors.
- **Loss global convergence**: Many variational methods are **non-convex** at the expense of being decomposable according to graph. These methods can only reach to local optima.
- **Generalization vs. discrimination**. Using function means that the update of one probabilty value will affect the other probability value when all of them share the same sufficient statistics. This makes it hard to implement iterative algorithms that based on the local graph topology.



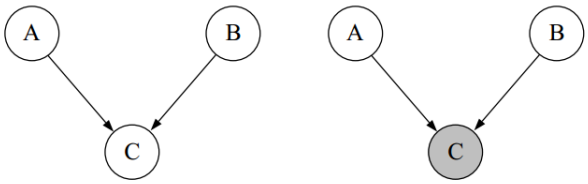
- A and B are marginally *dependent*
- A and B are conditionally *independent*

(a)



- A and B are marginally *dependent*
- A and B are conditionally *independent*

(b)



- A and B are marginally *independent*
- A and B are conditionally *dependent*

(c)

Figure 4: The conditional independency structure in directed graph. The shaded nodes are observed. In (c), the variable A and B are marginally *dependent* but conditionally *independent*.

3 Conditional independence

The most important property in graphical models is the ***Markov properties*** over graph. In particular, let A, B, and S be an arbitrary triple of mutually disjoint subsets of vertices. Let us stipulate that X_A be ***conditional independent*** of X_B given X_S, denoted as $X_A \perp\!\!\!\perp X_B | X_S$, if there is **no path** from a vertex in A to a vertex in B when we **remove** the vertices S from the graph. The set S is referred as ***cut-set***, and we say that node set A and B are **seperated** given S, denoted as $\text{sep}_{\mathcal{G}}(A; B|S)$ [Koller and Friedman, 2009]. The **Markov property** over graph $\mathcal{G}$ is written as

$$p(x_A|x_S, x_B) = p(x_A|x_S) \quad \text{if } \text{sep}_{\mathcal{G}}(A; B|S)$$

For directed graph, the definition of path is replaced by the notion of *active trail*, which not only consider directed path from $A \leftrightarrow B$ but also that of a *v-structure* $X_{i-1} \rightarrow X_i \leftarrow X_{i+1}$, when X_i or one of its descendant is in S. See Figure 4 for illustration of the active trails (a path of all dependent variables). The existence of **v-structure dependency** indicates that sometimes dependent variables may not directly related but may related to a common variables. This is the effect of "**explaining away**". (like "an increase in activation of Earthquake leads to a decrease in activation of Burglar") Using this, for directed graph we have concept of d-separation [Koller and Friedman, 2009].

For undirected graphical models, the ***conditional independency*** is *equivalent* to ***graph reachability***. This equivalence is a fundamental mathematical result, linking an algebraic concept (**factorization**) and a graph-theoretic concept (**reachability**). This result also has algorithmic

consequences, in that it reduces the problem of *assessing conditional independence* to the problem of *assessing reachability* on a graph, which is readily solved using simple breadth-first search algorithms.

Note that graphical model reprsentation is **not unique** since different models may represent the same set of conditional independence assertions among variables (i.e. **I-equivalence**). I-equivalence of two graphs immediately implies that any distribution P that can be factorized over one of these graphs can be factorized over the other. This observation has important implications with respect to our ability to determine the directionality of influence (i.e. the direction of an edge $X \to Y$ or $X \leftarrow Y$). The direction can only be inferred via **causal analysis**.

Some useful properties for conditional independency:

- The set of neighbors is the separation set of a singular node

$$X \perp\!\!\!\perp Y \,|\, (\mathcal{V} - \{X, Y\})$$
$$X \perp\!\!\!\perp (\mathcal{V} - \{X\} - \mathcal{N}_{\mathcal{G}}(X)) \,|\, \mathcal{N}_{\mathcal{G}}(X)$$

  where $\mathcal{N}_{\mathcal{G}}(X)$ is the set of all neighboring nodes of X in the graph.

- **Symmetry**:

$$(X \perp\!\!\!\perp Y \,|Z) \Leftrightarrow (Y \perp\!\!\!\perp X \,|Z).$$

- **Decomposition**:

$$(X \perp\!\!\!\perp (Y, W) \,|Z) \Rightarrow (X \perp\!\!\!\perp Y \,|Z). \tag{10}$$

- **Weak Union**:

$$(X \perp\!\!\!\perp (Y, W) \,|Z) \Rightarrow (X \perp\!\!\!\perp Y \,|(Z, W)). \tag{11}$$

  Addition evidence from a conditional independent factor will not change the existing conditional independency with other factors.

- **Contraction**:

$$(X \perp\!\!\!\perp W \,|(Y, Z)) \wedge (X \perp\!\!\!\perp Y \,|Z) \Rightarrow (X \perp\!\!\!\perp (Y, W) \,|Z). \tag{12}$$

- **Strong Union**:

$$(X \perp\!\!\!\perp Y \,|Z) \Rightarrow (X \perp\!\!\!\perp Y \,|(Z, W)). \tag{13}$$

  In other words, additional evidence W cannot induce dependence.

- **Transitivity**: For all disjoint sets X, Y, Z and all variables A:

$$\neg(X \perp\!\!\!\perp A \,|Z) \wedge \neg(A \perp\!\!\!\perp Y \,|Z) \Rightarrow \neg(X \perp\!\!\!\perp Y \,|Z) \tag{14}$$

  Intuitively, this statement asserts that if X and Y are both correlated with some A (given Z), then they are also correlated with each other (given Z). We can also write the **contrapositive** of this statement, which is less obvious but easier to read. For all X, Y, Z, A:

$$(X \perp\!\!\!\perp Y \,|Z) \Rightarrow (X \perp\!\!\!\perp A \,|Z) \vee (A \perp\!\!\!\perp Y \,|Z) \tag{15}$$

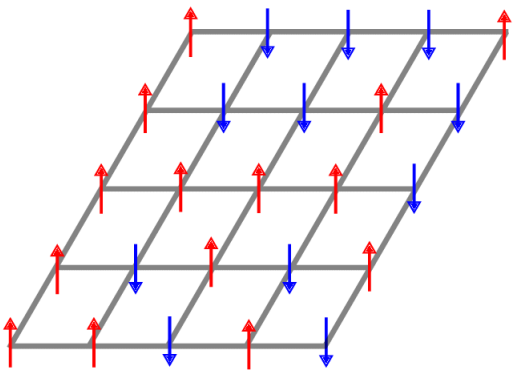


Figure 5: The Ising model which is built on a grid graph.

4 Examples

4.1 Ising Models (binary Markov random field)

The ***Ising model*** from statistical physics is a classical example of a graphical model in exponential form. Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and suppose that the random variable X_s associated with node $s \in \mathcal{V}$ is ***Bernoulli***, say taking the spin values $\{-1, +1\}$. In the context of statistical physics, these values might represent the *orientations* of magnets in a field, or the *presence/absence* of particles in a gas. The Ising model and variations on it have also been used in image processing and spatial statistics.

Ising model is a ***pairwise Markov random field***. Its factors involve at most two variables. Therefore, its underlying is graph is usually a grid network. Ising model is a part of exponential family

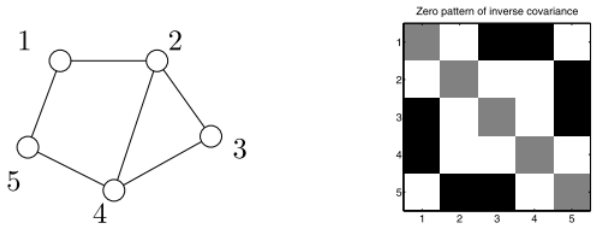
$$p(x_1, \ldots, x_m) = \exp\left(\sum_{s\in\mathcal{V}} w_s\, x_s + \sum_{(s,t)\in\mathcal{E}} w_{s,t}\, x_s\, x_t - A(\boldsymbol{w})\right), \tag{16}$$

where the base measure ν is the counting measure restricted to $\{0, 1\}^m$. Here $w_{s,t} \in \mathbb{R}$ is the *strength* of edge (s, t), and $w_s \in \mathbb{R}$ is a *potential* for node s, which models an "external field" in statistical physics, or a noisy observation in spatial statistics. Strictly speaking, the family of densities (16) is more general than the classical Ising model, in which $w_{s,t}$ is constant for all edges. The log-parition function

$$A(\boldsymbol{w}) = \log \sum_{\boldsymbol{x}\in\{0,1\}^m} \exp\left(\sum_{s\in\mathcal{V}} w_s\, x_s + \sum_{(s,t)\in\mathcal{E}} w_{s,t}\, x_s\, x_t\right)$$

Note that the local function (factor) is of the form

$$\phi_s(x_s, x_{\mathcal{N}(s)}) = \exp\left(w_s\, x_s + \sum_{t\in\mathcal{N}(s)} w_{s,t}\, x_s\, x_t\right)$$



(a) (b)

Fig. 3.1 (a) Undirected graphical model on five nodes. (b) For a Gaussian Markov random field, the zero pattern of the inverse covariance or precision matrix respects the graph structure: for any pair $i \neq j$, if $(i,j) \notin E$, then $\Theta_{ij} = 0$.

Figure 6: The Gaussian graphical model on undirected graph.

4.1.1 Metric Labeling and Potts Model

Consider a generalization of the Ising model: suppose that the random variable X_s at each node $s \in \mathcal{V}$ takes values in the **discrete space** $\mathcal{X} := \{0, 1, \ldots, r-1\}$, for some integer $r > 2$. One interpretation of a state $j \in \mathcal{X}$ is as a label, for instance defining membership in an image segmentation problem. Each pairing of a node $s \in \mathcal{V}$ and a state $j \in \mathcal{X}$ yields a sufficient statistic

$$\phi_{s;j}(x_s) = \mathbb{1}\{x_s = j\} \tag{17}$$

Moreover, for each edge (s,t) and pair of values $(j,k) \in \mathcal{X} \times \mathcal{X}$, define the sufficient statistics

$$\phi_{st;jk}(x_s, x_t) = \mathbb{1}\{x_s = j \wedge x_t = k\} \tag{18}$$

Note that $\sum_{j \in \mathcal{X}} \phi_{s;j}(x_s) = 1$. Similar to (16), we have

$$p(x_1, \ldots, x_m) = \exp\left(\sum_{s \in \mathcal{V}} \sum_{j \in \mathcal{X}} w_{s;j}\, \phi_{s;j}(x_s) + \sum_{(s,t) \in \mathcal{E}} \sum_{(j,k) \in \mathcal{X} \times \mathcal{X}} w_{st;jk}\, \phi_{st;jk}(x_s, x_t) - A(\boldsymbol{w}) \right), \tag{19}$$

When $\phi_{st;jk}(x_s, x_t) := -\rho(j,k)$ for some metric function $\rho : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$, we have the ***metric labeling problem***. Consequently, the canonical parameters satisfy the relations $w_{st;kk} = 0$ for all $k \in \mathcal{X}$, $w_{st;jk} < 0$ for all $j \neq k$, and satisfy the *reversed triangle inequality* (that is, $w_{st;jl} \geq w_{st;jk} + w_{st;kl}$ for all triples (j,k,l)). Another special case is the ***Potts model*** from statistical physics, in which case $w_{st;kk} = \alpha$ for all for all $k \in \mathcal{X}$, and $w_{st;jk} = \beta$ for all $j \neq k$,.

The local function (factor) is of the form

$$\phi_s(x_s, x_{\mathcal{N}(s)}) = \exp\left(\sum_{j \in \mathcal{X}} w_{s;j} \phi_{s;j}(x_s) + \sum_{t \in \mathcal{N}(s)} \sum_{(j,k) \in \mathcal{X} \times \mathcal{X}} w_{st;jk}\, \phi_{st;jk}(x_s, x_t) \right)$$

4.2 Gaussian graphical models (GGM)

Given an *undirected* graph $\mathcal{G}$ with vertex set $\mathcal{V} = \{1, \ldots, m\}$, a *Gaussian Markov random field (GMRF)* (or *Gaussian graphical models*) [Wainwright et al., 2008] consists of a multivariate *Gaussian* random vector $(X_1, \ldots, X_m)$ that respects the Markov properties of $\mathcal{G}$. It can be represented in

exponential form using the collection of sufficient statistics $\boldsymbol{\phi}(\boldsymbol{x}) := (x_s, x_s^2, s \in \mathcal{V};\ x_s\, x_t,\ (s,t) \in \mathcal{E})$. We define the inverse of covariance matrix $\boldsymbol{\Theta} = \boldsymbol{\Sigma}^{-1}$, which is referred as **precision matrix**.

A **key property** that distinguishes Gaussian graphical model from other models is that the ***graph structure*** **is fully encoded in the** ***precision matrix*** $\boldsymbol{\Theta}$. That is,

$$(s,t) \notin \mathcal{E} \Leftrightarrow X_s \perp\!\!\!\perp X_t | X_{\mathcal{V}-\{s,t\}} \Leftrightarrow \boldsymbol{\Theta}_{s,t} = 0 \tag{20}$$

As shown in (4) and (5), the form of Gaussian graphical model is

$$p(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\Theta}) = \exp\left(\langle \boldsymbol{\theta}\,,\, \boldsymbol{x} \rangle - \frac{1}{2} \left\langle \boldsymbol{\Theta}\,,\, \boldsymbol{x}\boldsymbol{x}^T \right\rangle - A(\boldsymbol{\theta}, \boldsymbol{\Theta}) \right) \tag{21}$$

Note that $\left\langle \boldsymbol{\Theta}\,,\, \boldsymbol{x}\boldsymbol{x}^T \right\rangle := \mathrm{tr}(\boldsymbol{\Theta}\boldsymbol{x}\boldsymbol{x}^T)$ is the inner product btw matrices. The natural parameter space

$$\Omega = \{ \boldsymbol{\Theta} \in \mathrm{Sym}_m, \boldsymbol{\theta} \in \mathbb{R}^m, \boldsymbol{\Theta} \succ \mathbf{0} \} \tag{22}$$

Unlike most of graphical models, the **log-partition function** of GGM has **closed-form**

$$A(\boldsymbol{\theta}, \boldsymbol{\Theta}) = \frac{1}{2} \left(\boldsymbol{\theta}^T \boldsymbol{\Theta}^{-1} \boldsymbol{\theta} - \log\det |\boldsymbol{\Theta}| \right) \tag{23}$$

Another important property for Gaussian graphical model is that the **conditional distribution** $p_s(\boldsymbol{x}_S | \boldsymbol{x}_{\mathcal{N}})$, i.e. the **local function**, is also Gaussian distributed

$$p_s(\boldsymbol{x}_S | \boldsymbol{x}_{\mathcal{N}}) = \exp\left(\left\langle \boldsymbol{\theta}_{S|\boldsymbol{x}_{\mathcal{N}}}\,,\, \boldsymbol{x}_S \right\rangle - \frac{1}{2} \left\langle \boldsymbol{\Theta}_S\,,\, \boldsymbol{x}_S \boldsymbol{x}_S^T \right\rangle - A(\boldsymbol{\theta}_{S|\boldsymbol{x}_{\mathcal{N}}}, \boldsymbol{\Theta}_S) \right) \tag{24}$$

$$\text{where } \boldsymbol{\theta}_{S|\boldsymbol{x}_{\mathcal{N}}} = \boldsymbol{\Theta}_S \boldsymbol{\mu}_{S|\boldsymbol{x}_{\mathcal{N}}} = \boldsymbol{\theta}_S - \boldsymbol{\Theta}_{S,\mathcal{N}} \boldsymbol{x}_{\mathcal{N}}$$

$$\boldsymbol{\Theta} = \begin{bmatrix} \boldsymbol{\Theta}_S & \boldsymbol{\Theta}_{S,\mathcal{N}} \\ \boldsymbol{\Theta}_{\mathcal{N},S} & \boldsymbol{\Theta}_{\mathcal{N},\mathcal{N}} \end{bmatrix} \tag{25}$$

where S is the clique and $\mathcal{N} = \mathcal{V} - S$ is the complentary set to S, $\boldsymbol{\Theta}_S$ is the diagonal block of precision matrix $\boldsymbol{\Theta}$ corresponding to S. $\boldsymbol{\theta}_S$ is the S block in $\boldsymbol{\theta}$. Also

$$\begin{aligned} p_s(\boldsymbol{x}_S | \boldsymbol{x}_{\mathcal{N}}) &= \mathcal{N}(\boldsymbol{\mu}_{S|\boldsymbol{x}_{\mathcal{N}}}, \boldsymbol{\Sigma}_{S|\mathcal{N}}) \\ \text{where } \boldsymbol{\mu}_{S|\boldsymbol{x}_{\mathcal{N}}} &= \boldsymbol{\mu}_S + \boldsymbol{\Sigma}_{S,\mathcal{N}} \boldsymbol{\Sigma}_{\mathcal{N}}^{-1} (\boldsymbol{x}_{\mathcal{N}} - \boldsymbol{\mu}_{\mathcal{N}}) \\ &= \boldsymbol{\mu}_S - \boldsymbol{\Theta}_S^{-1} \boldsymbol{\Theta}_{S,\mathcal{N}} (\boldsymbol{x}_{\mathcal{N}} - \boldsymbol{\mu}_{\mathcal{N}}) \\ \boldsymbol{\theta}_S &= \boldsymbol{\Theta}_S \boldsymbol{\mu}_S + \boldsymbol{\Theta}_{S,\mathcal{N}} \boldsymbol{\mu}_{\mathcal{N}} \\ \boldsymbol{\Sigma}_{S|\mathcal{N}} &= \boldsymbol{\Sigma}_S - \boldsymbol{\Sigma}_{S,\mathcal{N}} \boldsymbol{\Sigma}_{\mathcal{N}}^{-1} \boldsymbol{\Sigma}_{\mathcal{N},S} \\ &= \boldsymbol{\Theta}_S^{-1} \end{aligned} \tag{26}$$

(27)

Because of (20), **parameter estimation** in Gaussian graphical model with sparse regularization is equivalent to **structure learning**, i.e. learning the underlying graph $\mathcal{G}$ itself. Efficient ***sparse inverse covariance estimation*** methods such as ***Graphical Lasso***, *neighborhood regression* etc. [Wainwright, 2019] are available for estimating the graph structure.

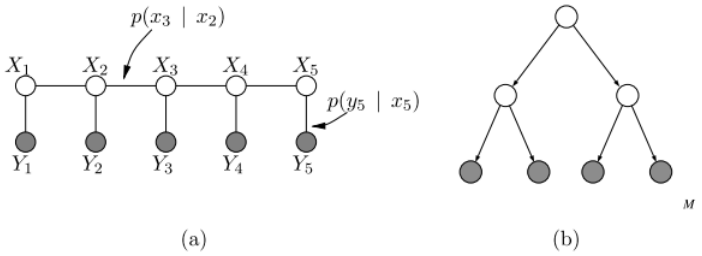


Fig. 2.4 (a) The graphical model representation of a generic hidden Markov model. The shaded nodes $\{Y_1, \ldots, Y_5\}$ are the observations and the unshaded nodes $\{X_1, \ldots, X_5\}$ are the hidden state variables. The latter form a Markov chain, in which X_s is independent of X_u conditional on X_t, where $s < t < u$. (b) The graphical model representation of a phylogeny on four extant organisms and M sites. The tree encodes the assumption that there is a first speciation event and then two further speciation events that lead to the four extant organisms. The box around the tree (a "plate") is a graphical model representation of replication, here representing the assumption that the M sites evolve independently.

Figure 7: The Hidden Markov chain on directed chain graph is a generative model.

4.3 Hidden Markov Chain (HMM)

The ***hidden Markov model (HMM)*** is a *generative* ***Bayesian network*** on a directed chain-structured graph. Due to its simple structure, it is widely used in ***sequential modeling*** such as *part-of-speech tagging*, *name entity recognition* in natural language processing, speech recognition, *gene finding* in bioinformatics and *decoding* in digital communications.

As a Bayesian network, HMM has two *type* of nodes: the *unobserved* one $\{x_t \in \mathcal{X}\}_{t=1}^T$, referred as **state**, and the *observed* one $\{y_t \in \mathcal{Y}\}_{t=1}^T$, referred as **observation**. The **state** variables are connected with each other via a **chain graph structure**, i.e. each state (except for initial state) has one and only one parent, and each observation only connect to one state.

The size of *state space* $r = |\mathcal{X}|$ is usually **small**, and the *state variables* are **discrete**. Therefore it is common to use the **tabular representation** to encode the HMM. Following Markov property, it is seen that HMM has **only two different local functions**:

- The ***state transition probabilty*** $p(x_t|x_{t-1})$. Let $\boldsymbol{P} = [p_{i,j}] = [p(x_t = j|x_{t-1} = i)] \in \mathbb{R}^{|\mathcal{X}|\times|\mathcal{X}|}$
- The ***emission probability*** $p(y_t|s_t)$. $\boldsymbol{B} = [b_i(y_t)] = [p(y_t|x_t = i)] \in \mathbb{R}^{|\mathcal{Y}|\times|\mathcal{X}|}$

Note that these two functions are identical for all t. This is because the *ergodic markov process* will reach steady-state in the long run, and at steady-state, the state transition probabilty $p(x_t|x_{t-1}) = p(x_{t+1}|x_t) = \ldots$. This ***recurrent structure*** allow HMM to cover sequences of different length.

The joint distribution of *HMM* is thus easy to follow

$$p(x_0, x_1, \ldots, x_T; y_1, \ldots, y_t) = \prod_{t=1}^{T} p(y_t|x_t)p(x_t|x_{t-1})p(x_0) \tag{28}$$

There are two tasks associated with HMM:

- **Inference** *states* given observations: The inference in HMM is to find $x_{1:T} := \{x_t\}_{t=1}^T$ that

maximize the log-likehood function given observations $y_{1:T} := \{y_t\}_{t=1}^T$

$$\begin{aligned}\hat{x}_{1:T} &= \text{argmax}_{x_{1:T}}\, p(x_{1:T}; y_{1:T}) \\ &= \text{argmax}_{x_{1:T}} \prod_{t=1}^{T} p(y_t|x_t)p(x_t|x_{t-1})p(x_0)\end{aligned} \tag{29}$$

The algorithm to solve (29) is based on ***dynamic programming***, called ***Viterbi decoding***. In particular, it utilizes the recursive structure of the model and defind the *local objective* as **value function:**

$$v_t(i) := \max_{x_{1:t-1}} p(x_{1:t-1}, o_{1:t-1}, x_t = i), \quad i = 1, \ldots, r \tag{30}$$

It follows the following **Bellman equation** as recursion formula.

$$v_t(i) = \max_{j=1,\ldots,r} v_{t-1}(j)\; p_{j,i}\; b_i(o_t), \quad i = 1, \ldots, r \tag{31}$$

This is essentially a **max-product algorithm**.

- **Learning**, i.e. parameter estimation on $\boldsymbol{P}$ and $\boldsymbol{B}$: The estimation task for HMM need to find $\boldsymbol{P}$ and $\boldsymbol{B}$ that maximize the **marginalized likelihood function** over observation $y_{1:T} := \{y_t\}_{t=1}^T$. Let $\boldsymbol{\theta} := (\boldsymbol{P}, \boldsymbol{B})$

$$\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta}} p(y_{1:T}|\boldsymbol{\theta}) = \prod_{x_{1:T}} p(x_{1:T}; y_{1:T}|\boldsymbol{\theta}) \tag{32}$$

This is done via an ***expectation-maximization (EM)*** algorithm called ***Baum-Welch algorithm***. Define the following quantities

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(l-1)}) = \mathbb{E}_{p(x_{1:T}|y_{1:T}, \boldsymbol{\theta}^{(l-1)})} \left[\log p(x_{1:T}; y_{1:T}|\boldsymbol{\theta})\right] \tag{33}$$

$$\gamma_i(t) = p(x_t = i|y_{1:T}, \boldsymbol{\theta}^{(l-1)}) \tag{34}$$

$$\xi_{i,j}(t) = p(x_t = i, x_{t+1} = j|y_{1:T}, \boldsymbol{\theta}^{(l-1)}) \tag{35}$$

  - **E-step**, i.e. estimation of $p(x_{1:T}|y_{1:T}, \boldsymbol{\theta}^{(l-1)})$. This quantity can be split into two part at time t, one with past events and one with future events. Then we use the dynamic programming with a **forward-backward procedure**.

    * **Forward step**: Define the *forward value function*

$$\alpha_i(t) := p(y_{1:t}, x_t = i\,|\boldsymbol{\theta}) \tag{36}$$

      That is the probability of observing *past events* $y_1, \ldots, y_t$ **and** being state $x_t = i$. This value function also has a **forward Bellman equation**:

$$\begin{aligned}\alpha_i(t+1) &= p(y_{t+1}|x_{t+1} = i) \sum_j p(x_{t+1} = i|x_t = j)\alpha_j(t) \\ &= b_i(y_{t+1}) \sum_j p_{j,i}\alpha_j(t)\end{aligned} \tag{37}$$

∗ **Backward step**: Similarly, define the *backward value function*

$$\beta_i(t) := p(y_{(t+1):T} | x_t = i, \boldsymbol{\theta}) \tag{38}$$

That is the probability of observing *future events* $y_{t+1}, \ldots, y_T$ **given** state $x_t = i$. This value function also has a **backward Bellman equation**:

$$\begin{aligned} \beta_i(t) &= \sum_j p(x_{t+1} = j | x_t = i) p(y_{t+1} | x_{t+1} = j) \beta_j(t+1) \\ &= \sum_j p_{i,j} b_j(y_{t+1}) \beta_j(t+1) \end{aligned} \tag{39}$$

This is essentially a **sum-product algorithm** to compute marginal distribution.

– **M-step**, i.e. maximization of $\mathbb{E}_{p(x_{1:T}|y_{1:T})} [\log p(x_{1:T}; y_{1:T} | \boldsymbol{\theta})]$. Update the $\gamma_i(t)$ and $\xi_{i,j}(t)$ via

$$\gamma_i(t) = \frac{\alpha_i(t) \beta_i(t)}{\sum_j \alpha_j(t) \beta_j(t)} \tag{40}$$

$$\xi_{i,j}(t) = \frac{\alpha_i(t)\ p_{i,j}\ \ \beta_j(t+1)\, b_j(y_{t+1})}{\sum_k \sum_s \alpha_k(t)\ p_{k,s}\ \ \beta_s(t+1)\, b_s(y_{t+1})} \tag{41}$$

Finally, the estimate $\hat{\boldsymbol{P}} = [\hat{p}_{i,j}]$ and $\hat{\boldsymbol{B}} = [\hat{b}_j(y)]$

$$\hat{p}_{i,j} = \frac{\sum_{t=1}^{T} \xi_{i,j}(t)}{\sum_{t=1}^{T} \gamma_i(t)} \tag{42}$$

$$\hat{b}_i(y) = \frac{\sum_{t=1}^{T} \mathbb{1}\{y_t = y\} \gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)} \tag{43}$$

4.4 Conditional Random Field (CRF)

HMM is a generative model. It is hard for HMM to add arbitrary features directly into the model. A discriminative model can be better. The **conditional random field (CRF)** [Sutton et al., 2012] can be used to solve this issue. The most commonly used CRF is linear chain CRF, which is close to HMM. Similar to HMM, CRF optimize the posterior probability

$$\hat{x}_{1:T} = \text{argmax}_{x_{1:T}}\, p(x_{1:T} | y_{1:T}).$$

Unlike HMM, CRF define the conditional probability directly as a **log-linear** model:

$$\begin{aligned} p(x_{1:T} | y_{1:T}) &= \frac{1}{Z(y_{1:T})} \exp\left(\sum_{k=1}^{K} w_k F_k(\boldsymbol{x}, \boldsymbol{y}) \right) \\ &= \frac{1}{Z(y_{1:T})} \exp\left(\sum_{k=1}^{K} \sum_{t}^{T} w_k f_k(x_t, x_{t-1}, y_{1:T}, t) \right) \end{aligned}$$

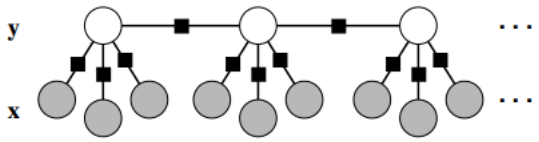


Fig. 2.4 Graphical model of an HMM-like linear-chain CRF.

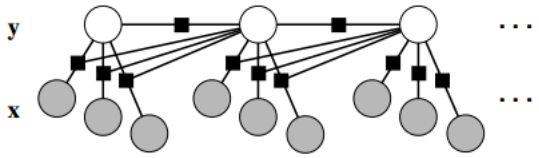


Fig. 2.5 Graphical model of a linear-chain CRF in which the transition score depends on the current observation.

Figure 8: Comparison of generative graphical models and discriminative graphical models [Sutton et al., 2012] .

Each f_k is called a **local feature**, which is accumulated over time to form a **global feature** F_k. Each of local feature is a linear-chain CRF which makes use of the current state/tag x_t and previous state/tag x_{t-1} and the entire observations $y_{1:T}$ and the current position t. It is linear chain CRG since f_k only depends on x_t and x_{t-1}. An example pos feature $f_k := \mathbb{1}\{y_t = \text{"the"}, x_t = \text{DET}\}$ and $f_s := \mathbb{1}\{y_{t+1} = \text{"Street"}, x_t = \text{PROPN}, x_{t-1} = \text{NUM}\}$. For NER feature, could be "identity of y_i, identity of neighboring words".

The linear structure of CRF makes it easy to build **binary feature** and extend to different special cases. These feature can be manually designed or automatically generated via **feature templates**: $< x_t, y_t >, < x_t, x_{t-1} >, < x_t, y_{t-1}, y_{t+1} >$. These features can be generated from training data. For unknown words, we can define the **word shape** features, which represent the ***abstract letter pattern*** of the word by mapping lower-case letters to x, upper-case to X, numbers to d, and retaining punctuation, e.g. $I.M.F := X.X.X$. Prefix and suffix features are also useful. For example some feature templates with unknown words such as "y_i contains a particular prefix/suffix" or "y_i's word shape" etc.

The *known-word templates* are computed for every word seen in the training set; the *unknown word features* can also be computed for all words in training, or only on training words whose frequency is below some threshold. The result of the known-word templates and word-signature features is *a very large set of features.*

In NER, one feature that is especially useful for **locations** is a **gazetteer**, a list of place names, often providing millions of entries for locations with detailed geographical and political information. This can be implemented as a **binary feature** indicating a phrase appears in the list. Other related resources like name-lists can be used, as can other entity dictionaries like lists of corporations or products, although they may not be as helpful as a gazetteer

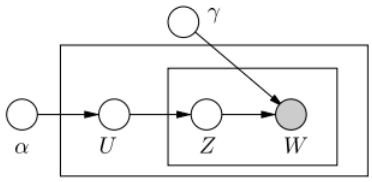


Fig. 3.3 Graphical illustration of the Latent Dirichlet allocation (LDA) model. The word variable W is multinomial conditioned on the underlying topic Z, where γ specifies the topic distributions. The topics Z are also modeled as multinomial variables, with distributions parameterized by a probability vector U that follows a Dirichlet distribution with parameter α. This model is an example of a hierarchical Bayesian model. The rectangles, known as plates, denote replication of the random variables.

Figure 9: The Latent Dirichlet allocation is a generative Bayesian network model.

4.5 Latent Dirichlet allocation (LDA)

The latent Dirichlet allocation model [Wainwright et al., 2008] is a particular type of hierarchical Bayes model for capturing the statistical dependencies among words in a corpus of documents. It involves three different types of random variables: ***documents*** U, ***topics*** Z, and ***words*** W.

Words W are drawn from a multinomial distribution, $p(W = j|Z = i; \gamma) = \exp(\gamma_{ij})$, for $j = 0, 1, ..., k1$, where γ_{ij} is a parameter encoding the probability of the j-th word under the i-th *topic*. This conditional distribution can be expressed as an *exponential family* in terms of indicator functions as follows:

$$p_\gamma(w|z) \propto \exp\left(\sum_{i=0}^{k-1}\sum_{j=0}^{r-1}\gamma_{ij}\mathbb{1}\{w=j\}\,\mathbb{1}\{z=i\}\right) \tag{44}$$

At the next level of the hierarchy (see FIgure 9) the topic variable Z also follows a multinomial distribution whose parameters are determined by the **Dirichlet variable** as follows:

$$p(z|u) \propto \exp\left(\sum_{i=0}^{k-1}\mathbb{1}\{z=i\}\log(u_i)\right) \tag{45}$$

Finally, at the top level of the hierarchy, the ***Dirichlet variable*** U has a density with respect to Lebesgue measure of the form

$$p_\alpha(u) \propto \exp\left(\sum_{i=0}^{r-1}\alpha_i\log(u_i)\right). \tag{46}$$

The overall joint distribution factorizes as below:

$$\begin{aligned} p(w,z,u) &= p_\gamma(w|z)p(z|u)p_\alpha(u) \\ &\propto \exp\left(\sum_{i=0}^{r-1}\alpha_i\log(u_i) + \sum_{i=0}^{k-1}\mathbb{1}\{z=i\}\log(u_i) + \sum_{i=0}^{k-1}\sum_{j=0}^{r-1}\gamma_{ij}\mathbb{1}\{w=j\}\,\mathbb{1}\{z=i\}\right) \end{aligned} \tag{47}$$

The sufficient statistics consist of the collections of functions

$$\boldsymbol{\phi}(w,z,u) = ((\log(u_i))_i, (\mathbb{1}\{z=i\})_i, (\mathbb{1}\{w=j\}\,\mathbb{1}\{z=i\})_{i,j}).$$

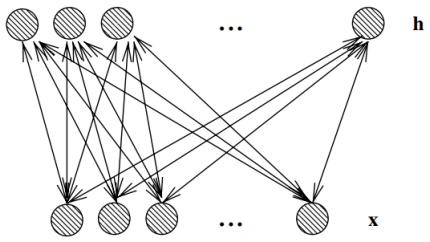


Fig. 4.5 Undirected graphical model of a Restricted Boltzmann Machine (RBM). There are no links between units of the same layer, only between input (or visible) units $\mathbf{x}_j$ and hidden units $\mathbf{h}_i$, making the conditionals $P(\mathbf{h}|\mathbf{x})$ and $P(\mathbf{x}|\mathbf{h})$ factorize conveniently.

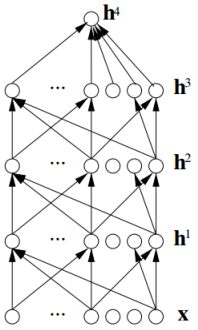


Fig. 4.1 Multi-layer neural network, typically used in supervised learning to make a prediction or classification, through a series of layers, each of which combines an affine operation and a non-linearity. Deterministic transformations are computed in a feedforward way from the input $\mathbf{x}$, through the hidden layers $\mathbf{h}^k$, to the network output $\mathbf{h}^\ell$, which gets compared with a label y to obtain the loss $L(\mathbf{h}^\ell, y)$ to be minimized.

Figure 10: The restricted Boltzmann machine is a generative Bayesian model based on layerwise graph. [Bengio et al., 2009]

4.6 Restricted Boltzmann machine (RBM)

Traditionally, each conditional probability distribution $p_s(x_s|x_{\pi(s)})$ is represented by a tabular form or a linear model [Koller and Friedman, 2009]. As we explained above, a more flexible way to parameterize such conditional distributions is with neural networks.

For a ***Deep Belief Network (DBN)*** with $l+1$ layers, the input observed variables are denoted as $\boldsymbol{x}$, and the hidden variables at layer k are denoted $\boldsymbol{h}^{(k)}$. The joint distribution is factorized as

$$p(\boldsymbol{x}, \boldsymbol{h}^{(1)}, \ldots, \boldsymbol{h}^{(l)}) = p(\boldsymbol{x}|\boldsymbol{h}^{(1)}) \left(\prod_{k=1}^{l-2} p(\boldsymbol{h}^{(k)}|\boldsymbol{h}^{(k+1)}) \right) p(\boldsymbol{h}^{(l-1)}, \boldsymbol{h}^{(l)}) \tag{48}$$

The ***restricted Boltzmann machine (RBM)*** serves a prior distribution $p(\boldsymbol{h}^{(l-1)}, \boldsymbol{h}^{(l)})$ in a deep belief network and is a basic building block of many generative neural network. RBM is an *undirected* graphical model. It is built upon a **bipartite graph** where variables (neurons) are connected **between** two layers while variables within each layer are **conditional independent** given the other layer. It is a part of exponential family with **binary variables** $\boldsymbol{x} \in \{0,1\}^{d_x}$ and $\boldsymbol{h} \in \{0,1\}^{d_h}$. The joint distribution is as below:

$$p(\boldsymbol{x}, \boldsymbol{h}; \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{W}) \propto \exp\left(\langle \boldsymbol{b}\,,\, \boldsymbol{x} \rangle + \langle \boldsymbol{c}\,,\, \boldsymbol{h} \rangle + \left\langle \boldsymbol{W}^T\,,\, \boldsymbol{x}\,\boldsymbol{h}^T \right\rangle \right) \tag{49}$$

where $\left\langle \boldsymbol{W}^T\,,\, \boldsymbol{x}\,\boldsymbol{h}^T \right\rangle = \boldsymbol{h}^T \boldsymbol{W} \boldsymbol{x}$ and $\boldsymbol{W} \in \mathbb{R}^{d_h \times d_x}$, $\boldsymbol{b} \in \mathbb{R}^{d_x}$ and $\boldsymbol{c} \in \mathbb{R}^{d_h}$.

Note that due to the structure of bipartite graph, each neuron in RBM is **conditionally independent given the alternative layer**. Therefore, the local function in RBM is easily obtained as

$$\begin{aligned} p(\boldsymbol{h}|\boldsymbol{x}) &= \frac{\exp\left(\langle \boldsymbol{b}\,,\, \boldsymbol{x} \rangle + \langle \boldsymbol{c}\,,\, \boldsymbol{h} \rangle + \left\langle \boldsymbol{W}^T\,,\, \boldsymbol{x}\,\boldsymbol{h}^T \right\rangle \right)}{\sum_{\boldsymbol{h}} \exp\left(\langle \boldsymbol{b}\,,\, \boldsymbol{x} \rangle + \langle \boldsymbol{c}\,,\, \boldsymbol{h} \rangle + \left\langle \boldsymbol{W}^T\,,\, \boldsymbol{x}\,\boldsymbol{h}^T \right\rangle \right)} \\ &= \prod_i \frac{\exp\left\{ h_i \left(c_i + \boldsymbol{W}_{i,\cdot} \boldsymbol{x} \right) \right\}}{\sum_{h_i} \exp\left\{ h_i \left(c_i + \boldsymbol{W}_{i,\cdot} \boldsymbol{x} \right) \right\}} \end{aligned}$$

$$= \prod_{i=1}^{d_h} p(h_i|\boldsymbol{x}) \tag{50}$$

$$\text{where } p(h_i = 1|\boldsymbol{x}) = \frac{\exp\{(c_i + \boldsymbol{W}_{i,\cdot}\boldsymbol{x})\}}{\sum_{h_i} \exp\{h_i (c_i + \boldsymbol{W}_{i,\cdot}\boldsymbol{x})\}}$$

$$= \sigma(c_i + \boldsymbol{W}_i \boldsymbol{x}) \tag{51}$$

and $\sigma(x) = e^{x_i}/(\sum_i e^{x_i})$ is the non-linear *smooth* **activation function** and $W_{i,\cdot}$ is the i-row of $\boldsymbol{W}$. Similarly

$$p(\boldsymbol{x}|\boldsymbol{h}) = \prod_{j=1}^{d_x} p(x_j|\boldsymbol{h}) \tag{52}$$

$$\text{where } p(x_j = 1|\boldsymbol{h}) = \sigma\left(b_j + \boldsymbol{h}^T \boldsymbol{W}_{\cdot,j}\right) \tag{53}$$

and $\boldsymbol{W}_{\cdot,j}$ is j-th column of $\boldsymbol{W}$.

References

Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

Anna Goldenberg, Alice X Zheng, Stephen E Fienberg, Edoardo M Airoldi, et al. A survey of statistical network models. *Foundations and Trends® in Machine Learning*, 2(2):129–233, 2010.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Matthew O Jackson. *Social and economic networks*. Princeton university press, 2010.

Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Mark Newman. *Networks*. Oxford university press, 2018.

Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106 (5):808–828, 2018.

Alexander Shapiro. Monte carlo sampling methods. *Handbooks in operations research and management science*, 10:353–425, 2003.

Charles Sutton, Andrew McCallum, et al. An introduction to conditional random fields. *Foundations and Trends® in Machine Learning*, 4(4):267–373, 2012.

Martin J Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge University Press, 2019.

Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.