Undirected graphical models

Kevin P. Murphy

Last updated November 16, 2006

* Denotes advanced sections that may be omitted on a first reading.

1 Introduction

We have seen that conditional independence assumptions are a useful way to decompose joint probability distributions into a simpler form. For example, the "bag of words" model assumes all variables are independent

$$p(x_{1:D}) = \prod_{i=1}^{D} p(x_i) \tag{1}$$

The naive Bayes model assumes all the x variables (features) are independent conditional on y (class):

$$p(y, x_{1:D}) = p(y)p(x_{1:D}|y) = p(y)\prod_{i=1}^{D} p(x_i|y) \tag{2}$$

A Markov model assumes the variables only depend on their immediate "predecessor"

$$p(x_{1:D}) = p(x_1)\prod_{i=2}^{D} p(x_i|x_{i-1}) \tag{3}$$

Graphical models provide a convenient way to represent conditional independence assumptions using graphs. Nodes represent random variables and (lack of) edges represent independence assumptions, in a way to be described below. Such independence assumptions reduce the number of parameters in the model, and therefore the amount of data that is needed to learn the model from data. Independence assumptions can also be exploited computationally to speed up the inference process (i.e., estimating one variable given evidence on some of the others). We will give details later.

There are many kinds of graphical model, but the two most popular are based on **directed acylic graphs (DAGs)**, and on **undirected graphs**. In the directed graph, we can (informally) think of an edge from node X_i to node X_j as meaning X_i "causes" or "directly influences" X_j, whereas in an undirected graph, edges represent correlation or constraints rather than causation. In this chapter, we focus on undirected graphs, also called **Markov random fields (MRFs)** or **Markov networks**.

2 Conditional independence properties

In an MRF, we say that a set of nodes A is independent of another set B given a third set C (written more concisely as $X_A \perp X_B|X_C$) if C *separates* A from B in the graph. This means that, if we remove the nodes in C, there are no paths connecting any node in A to any node in B. For example, in Figure 1, we see that $C = \{2, 5\}$ separates $A = \{1\}$ from $B = \{4, 6\}$. We also see that $C = \{2\}$ separates $A = \{3\}$ from $B = \{4\}$. And so on. These are called the **global Markov properties** encoded by a graph. (There are various other kinds of Markov properties, discussed below, but these can be shown to be equivalent to the global Markov property under various assumptions.)

Let us now consider some more examples. In Figure 2(left), we see that

$$X_i \perp X_j|Y \tag{4}$$

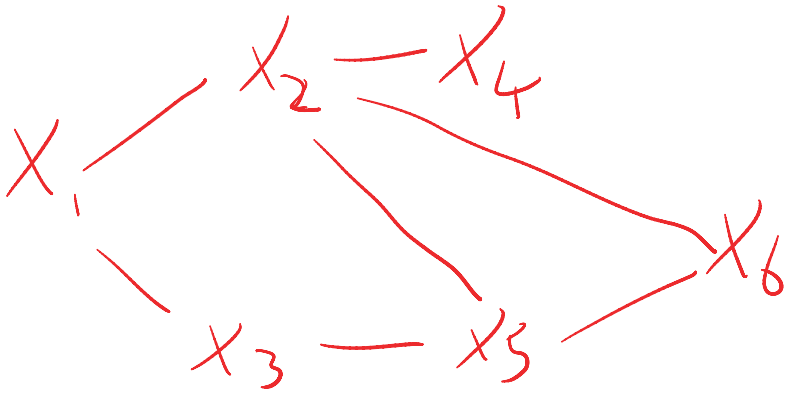

Figure 1: An example of an undirected graphical model.

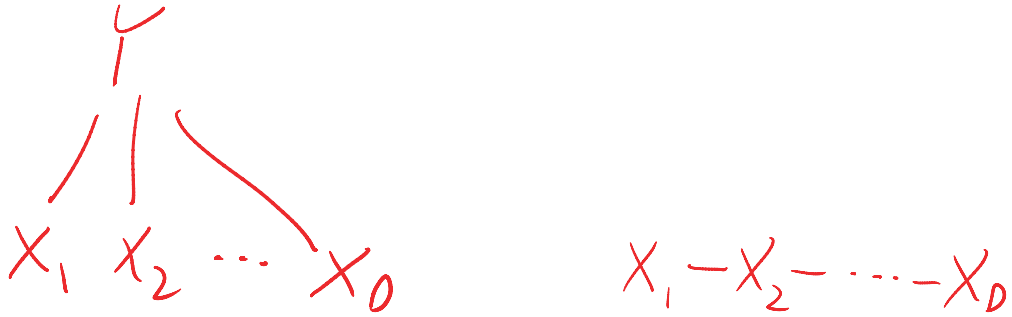

Figure 2: Left: a naive Bayes classifier represented as an MRF. Right: a Markov chain represented as an MRF.

which is the naive Bayes assumption. Hence

$$p(X_{1:D}, Y) \propto \prod_{i=1}^{D} \psi_i(X_i, Y) \tag{5}$$

We can set $\psi_i(X_i, Y) = p(X_i|Y)$. What about the $p(Y)$ term? We can multiply that onto any of the clique potentials, say the first, so $\psi_1(X_1, Y) = p(X_1|Y)p(Y) = p(X_1, Y)$. Thus some of the potentials are conditional probabilities are some are joint probabilities. We will discuss this more below.

In Figure 2(right), we see that

$$X_{i+1} \perp X_{i-1}|X_i \tag{6}$$

which is the Markov assumption. Hence

$$p(X_{1:D}) \propto \prod_{i=2}^{D} \psi_i(X_{i-1}, X_i) \tag{7}$$

Again we can set $\psi_i(X_{i-1}, X_i) = p(X_i|X_{i-1})$, and $\psi_1(X_1, X_2) = p(X_1)p(X_2|X_1)$.

2.1 Varieties of conditional independencies *

An undirected graph can represent various kinds of conditional independences. Define the **pairwise Markov independencies** of a graph G to be

$$I_p(G) = \{X_i \perp X_j|X_{rest} : (i - j) \notin G\} \tag{8}$$

where $X_{rest} = V \setminus \{i, j\}$ are all the other variables apart from i and j, and V is all the nodes in the graph. In other words, for every missing edge $i - j$, we create a conditional independence assertion. For example, in Figure 1, we

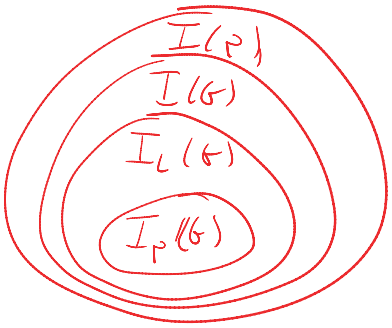

Figure 3: Nested sets of independence statements.

have

$$I_p(G) = \{(X_1 \perp X_5 | X_2, X_3, X_4, X_6), \;\; (X_1 \perp X_4 | X_2, X_3, X_5, X_6), \ldots, \} \tag{9}$$

(An alternative to listing all the independence properties is to think of the graph as an **oracle**, which can answer any conditional independence query.)

A second kind of independence are the **local Markov independencies**, defined as

$$I_l(G) = \{X_i \perp X_{rest} | X_{nbr(i)}\} \tag{10}$$

where $nbr(i)$ are the neighbors of node i, and $X_{rest} = V \setminus \{i\} \setminus X_{nbr(i)}$. (The neighbors of a node are also called its **Markov blanket**.) For example, in Figure 1, we have

$$I_l(G) = \{(X_3 \perp X_2, X_4, X_6 | X_1, X_5), \ldots\} \tag{11}$$

since 1 and 5 "shield" 3 from the other nodes.

The third kind of independence are the **global Markov independencies**, defined as

$$I(G) = \{X_A \perp X_B | X_C : \; C \text{ separates } A \text{ from } B\} \tag{12}$$

We can check if C separates A and B by removing all the nodes in C from the graph, and checking if there is a path from any node in A to any node in B. If so, they are not separated. For example, in Figure 1, $C = \{2, 5\}$ separates $A = \{1\}$ from $B = \{4, 6\}$, etc. Hence for this example

$$I(G) = \{(X_1 \perp X_4, X_6 | X_2, X_5), \ldots\} \tag{13}$$

Now we discuss the relationship between these three notions. Let $I(p)$ be the set of conditional independencies that are true of distribution p. We say that G is an **I-map** (independency map) for p if $I(G) \subseteq I(p)$. In otherwords, the graph does not make any false assertions of independence. Note that the fully connected graph is an I-map of all distributions, since it makes no assertions at all.

Theorem 1 *If $I(p)$ contains $I(G)$) (i.e., G is an I-map for p), then it will also contain $I_l(G)$ and $I_p(G)$: see Figure 3. In other words, global Markov properties imply local Markov properties imply pairwise Markov properties. If p is a positive distribution (i.e., $p(x) > 0$ for all x) then the converse is also true, hence $I_p(G) = I_l(G) = I(G)$.*

3 Factorization

Above we saw that a graph encodes conditional independence assumptions. An alternative way to view graphical models is as a means to define probability distributions in terms of a product of local factors. To explain this, we need some definitions.

A **clique** is a set of variables in a graph that are all connected with each other. A **maximal clique** is a clique which cannot be made larger without ceasing to be a clique. For example, in Figure 1, the maximal cliques are

$$\{X_1, X_2\}, \{X_1, X_3\}, \{X_2, X_4\}, \{X_3, X_5\}, \{X_2, X_5, X_6\} \tag{14}$$

A potential function $\psi(\cdot)$ is any non negative function of its arguments. We say that a probability distribution p **factorizes over** G, or p **is Markov wrt** G (wrt = with respect to) if it can be written in this form:

$$p(x) = \frac{1}{Z} \prod_{c \in C} \psi_c(x_c) \tag{15}$$

where C are the maximal cliques of the graph, $\psi_c(x_c)$ is a for the nodes in clique c and Z is the **partition function**

$$Z = \sum_{x_{1:D}} \prod_{c \in C} \psi_c(x_c) \tag{16}$$

The global normalization constant Z is needed because locally the ψ_c terms need not sum to one. For example, in Figure 1, we have

$$p(x_{1:6}) \propto \psi_{12}(x_1, x_2)\psi_{13}(x_1, x_3)\psi_{24}(x_2, x_4)\psi_{35}(x_3, x_5)\psi_{256}(x_2, x_5, x_6) \tag{17}$$

3.1 Hammersley-Clifford theorem *

The celebrated **Hammersley-Clifford theorem** states that, if the graph correctly captures the conditional independencies of the distribution, then there must exist potential functions ψ_c such that the distribution can be represented as $p(x) \propto \prod_c \psi_c(x_c)$. More precisely,

Theorem 2 *For any probability distribution p, if p factorizes over G, then G is an I-map of p (i.e., $I(G) \subseteq I(p)$). Conversely, if p is everywhere non-negative (i.e, $p(x) > 0$ for all x), and G is an I-map of p, then p must factorize over G, i.e., p must have the form*

$$p(x) = \frac{1}{Z} \prod_{c \in C} \psi_c(x_c) \tag{18}$$

See [KF06] for a proof.

3.2 MRFs are exponential family distributions

Let us assume a particular parametric form for each potential ψ_c, and denote the corresponding parameters by θ_c. Then we can write

$$p(x|\theta) = \frac{1}{Z(\theta)} \prod_{c \in C} \psi_c(x_c|\theta_c) \tag{19}$$

We can rewrite this as an **exponential family model** as follows

$$p(x|\theta) = \exp\left(\sum_{c \in C} \log \psi_c(x_c|\theta_c) - \log Z(\theta) \right) \tag{20}$$

If we restrict potentials to be strictly positive (which will imply that $p(x) > 0$ for all x, so we cannot encode hard constraints), then we can represent potentials using **energy functions** E_c

$$\psi_c(x_c) = \exp[-E_c(x_c)] \tag{21}$$

This is called the **Boltzmann distribution**. Low energy states are more probable than high energy ones. We will examples of this below.

3.3 Maxent models *

A **maximum entropy (maxent)** model is an MRF in which all the potentials have the form

$$\psi_c(x_c) = \exp[\sum_{j=1} \theta_{cj} f_{cj}(x_c)] \tag{22}$$

where $f_{cj}(x_c)$ is the j'th feature for clique c, applied to x_c, and θ_{cj} is the corresponding weight. In other words, the potentials are linear functions of a fixed set of features. Hence the joint is

$$p(x|\theta) = \frac{1}{Z(\theta)} \prod_c \exp[\sum_j \theta_{cj} f_{cj}(x_c)] \tag{23}$$

$$= \frac{1}{Z(\theta)} \exp[\sum_k \theta_k f_k(x)] \tag{24}$$

where we have combined all the features into one long feature vector f, and all the parameters into one long parameter vector θ. This is obviously a member of the exponential family. In machine, this model is sometimes called a **log-linear model**, although this means something slightly different in statistics, so we don't recommend this usage. We will explain the "maxent" term below.

Note that feature based potentials include tabular potentials as a special case: we create one binary feature for every possible value of x_c, and just set θ_{cj} to be the log of the original potential entries. For example, consider a clique $c = (i, j)$ on binary nodes. If the original potential

$$\psi_c = \begin{pmatrix} 0.1 & 0.5 \\ 1 & 20 \end{pmatrix} \tag{25}$$

then we set $f_{c1}(x_i, x_j) = I(x_i = 0, x_j = 0)$, $f_{c2}(x_i, x_j) = I(x_i = 0, x_j = 1)$, $f_{c3}(x_i, x_j) = I(x_i = 1, x_j = 0)$, and $f_{c4}(x_i, x_j) = I(x_i = 1, x_j = 1)$, and then set $\theta_{c1} = \log 0.1$, $\theta_{c2} = \log 0.5$, $\theta_{c3} = \log 1$, and $\theta_{c4} = \log 20$.

One example where a feature based representation of potentials is useful is when the number of states K is large and/or the cliques are large. In this case, the tabular representation of potentials may require too many parameters. For example, if we create a Markov model of language, we have one state per word, so $K \sim 50,000$. But K^2 parameters in each $\psi(X_i, X_{i+1})$ potential is too many to reasonably estimate. Instead of considering all possible word pairs, we can use log-linear potentials. Features might include "does the first word begin with a capital letter", "is the first word a noun and the second word a verb", etc. Typically these features are hand-constructed, but the **weights** (parameters) θ_c are learned from data (see below). Such models have been used for unsupervised learning of the rules of English spelling [PPL97].

We now explain the origin of the term "maximum entropy". Suppose we want to build a probability distribution that satisfies certain expectation constraints wrt some features f_j:

$$E[f_j(X)] = \alpha_j \tag{26}$$

For example, we might want our model to generate English words with a certain proportion of each letter of the alphabet. Also, we may want to enforce hard constraints like "q is always followed by u"; we just set $f_j(X) = 1$ if X satisfies the constraint, and $f_j(X) = 0$ otherwise, and set $\alpha_j = \infty$.

Apart from these constraints, we want pick as general a distribution as possible. In other words, we want to find a distribution p with maximum entropy, subject to the above constraints, plus the sum-to-one constraint. (We assume p_x is a finite length vector of elements.) We can formalize this as a convex optimization problem subject to linear constraints:

$$\max_p \quad -\sum_x p_x \log p_x \tag{27}$$

$$\text{subject to} \quad \sum_x p_x f_j(x) = \alpha_j \tag{28}$$

$$\sum_x p_x = 1 \tag{29}$$

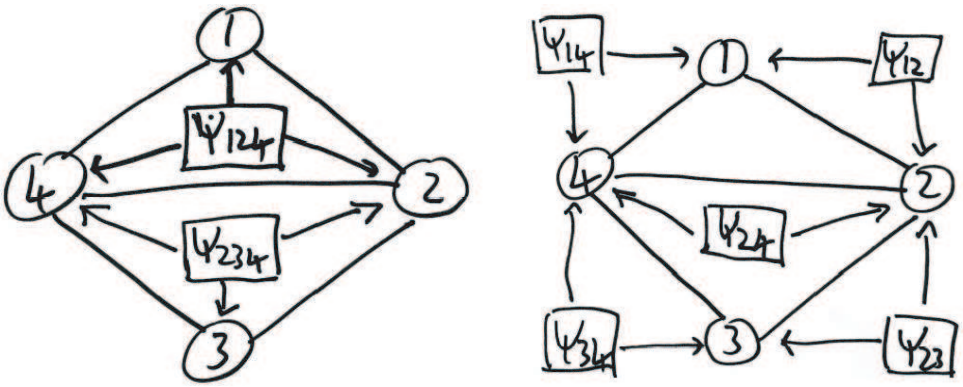

Figure 4: Chain graphs in which the clique potentials are explicitly represented as random variables (square nodes). Left: one potential per maximal clique. Right: one potential per edge.

We write the Lagrangian as

$$J = -\sum_x p_x \log p_x + \lambda(\sum_x p_x - 1) + \sum_j \theta_j(\sum_x p_x f_j(x) - \alpha_j) \tag{30}$$

Taking derivatives wrt a specific element p_x

$$\frac{\partial J}{\partial p_x} = -p_x[\frac{\partial}{\partial p_x}\log p_x] - [\frac{\partial}{\partial p_x}p_x]\log p_x + \lambda + \sum_j \theta_j f_j(x) \tag{31}$$

$$= -1 - \log p_x + \sum_j \theta_j f_j(x) \tag{32}$$

$$= 0 \tag{33}$$

Hence

$$\log p_x = -1 + \lambda + \sum_j \theta_j f_j(x) \tag{34}$$

$$p_x = e^{\lambda - 1}\exp[\sum_j \theta_j f_j(x)] \tag{35}$$

Since $\sum_x p_x = 1$, we have

$$e^{\lambda-1} = \frac{1}{\exp[\sum_j \theta_j f_j(x)]} = Z \tag{36}$$

Hence

$$p_x = \frac{1}{Z}\exp[\sum_j \theta_j f_j(x)] \tag{37}$$

So we have derived the exponential family using the maximum entropy principle, where the weights (parameters) θ_j are the Lagrange multipliers. Typically the α_j values are derived from data, as we discuss below.

4 Factor graphs *

In a Bayesian model, the parameters are treated as random variables, and can therefore be viewed as nodes in the graph. Although the "main" graph of an MRF is undirected, the edges from the parameter nodes will be directed, since the priors on the parameters will usually be **locally normalized**. For example, referring to Figure 4(left), we

have

$$p(x_{1:4}, \psi) = p(x_{1:4}|\theta)p(\psi) \quad (38)$$

$$= \left[\frac{1}{Z(\psi)}\psi_{124}(x_{124}) \times \psi_{234}(x_{234})\right][p(\psi_{124})p(\psi_{234})] \quad (39)$$

$$Z(\psi) = \sum_{x_1,x_2,x_3,x_4} \psi_{124}(x_{124}) \times \psi_{234}(x_{234}) \quad (40)$$

where $\psi = (\psi_{234}, \psi_{124})$ are the parameters. This is an example of a **chain graph**, which (loosely speaking) is a graph in which some nodes have directed arrows to sets of nodes connected together in undirected maximal cliques.

Now imagine that instead of associating a potential with each maximal clique, we associate potentials with non maximal cliques. For example, referring to Figure 4(right), suppose $\psi_{124} = \psi_{12}\psi_{14}$ and $\psi_{234} = \psi_{23}\psi_{34}\psi_{24}$, so we have one potential for each edge. (This is called a **pairwise MRF**.) Then

$$p(x_{1:4}|\psi) = \frac{1}{Z}\prod_{<ij>} \psi_{ij}(x_{ij}) \quad (41)$$

$$= \frac{1}{Z}\psi_{12}(x_{12})\psi_{14}(x_{14})\psi_{23}(x_{23})\psi_{24}(x_{24})\psi_{34}(x_{34}) \quad (42)$$

Note that the ψ_{24} edge potential is shared (tied) across different max cliques, so the clique potentials are no longer independent.

One way to make the parameterization of the MRF graphically explicit, without any committment to a Bayesian approach, is to use a **factor graph**. A factor graph is an undirected bipartite graph with two kinds of nodes. Round nodes represent variables, square nodes represent factors (potentials), and there is an edge from each variable to every factor that mentions it. In Figure 5, we show factor graphs for the two models mentioned above. It is possible to extend the conditional independence semantics of graphical model to include factor graphs [Fre03], but here we just use them as a convenient notation for making the parameterization more explicit.

5 Decomposable graphs *

An undirected graph is said to be **chordal** or **triangulated** if every undirected loop/ cycle $X_1 - X_2 \cdots X_k - X_1$ of length $k \geq 4$ has a chord, i.e., an edge connects X_i, X_j for all non-adjacent nodes i,j in the cycle. The process of converting a non-chordal graph to chordal form is called **triangulation**. The importance of triangulated graphs will be explained later.

Some simple examples of chordal and non chordal graphs are shown in Figure 6. A less obvious example is shown in Figure 7: this illustrates that triangulation is more than covering the graph with little triangles. In particular, we may have to add a lot of edges to ensure the graph is triangulated (such edges are called **fill-in edges**). Furthermore, there may be many ways to do this. Let us define the **optimal triangulation** as the one which adds the least number of fill-in edges. Let $C(G)$ be the size of the largest maximal clique in the optimal triangulation of graph G. Then we define the **tree width** of the graph as $T(G) = C(G) - 1$. (It is defined this way so that for trees, $T(G) = 1$.) For example, in Figure 7(right), we see that $C(G) = 4$ (look at the 2-4-5-6 clique or the 4-5-6-8 clique), which is larger than the maximal cliques in the non-triangulated graph in Figure 7(left). In this case, the max clique size is not very large, but in general, one can show that for planar 2D grids of size $D = d \times d$, the treewidth is $O(d)$ [LT79]. The importance of this result will be discussed later.

Triangulated graphs are also called **decomposable graphs**. A simple example of a decomposable graph is a chain, $X_1 - X_2 - X_3 - X_4$. In this case the cliques are $C_1 = \{X_1, X_2\}$, $C_2 = \{X_2, X_3\}$, and $C_3 = \{X_3, X_4\}$. Define the **separators** as the intersection of neighboring cliques: $S_{ij} = C_i \cap C_j$. For the chain, the separators are the internal nodes, $S_{1,2} = \{X_2\}$, $S_{2,3} = \{X_3\}$. See Figure 8.

Let us denote the clique potentials by ψ_c and the separator potentials by ψ_s. If we define $\psi_c = p(X_c)$ and $\psi_s = p(X_s)$, then we can write the joint as

$$p(X_{1:D}) = \frac{\prod_c \psi_c(x_c)}{\prod_s \psi_s(x_s)} \quad (43)$$

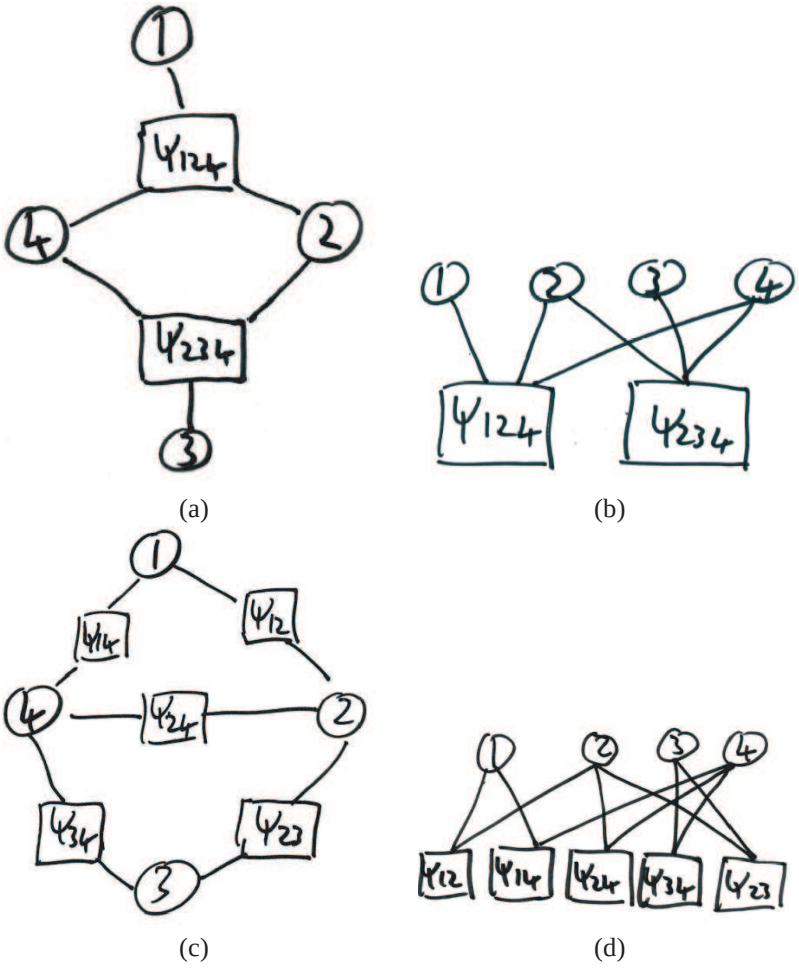

Figure 5: Some factor graphs. (a) and (b) are topologically equivalent graphs, and represent the MRF in Figure 4(left), with one potential per maximal clique. (c) and (d) are topologically equivalent graphs, and represent the MRF in Figure 4(right), with one potential per edge.

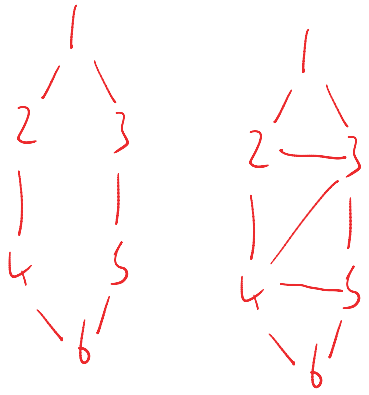

Figure 6: Left: a non triangulated graph. Right: one possible triangulated version.

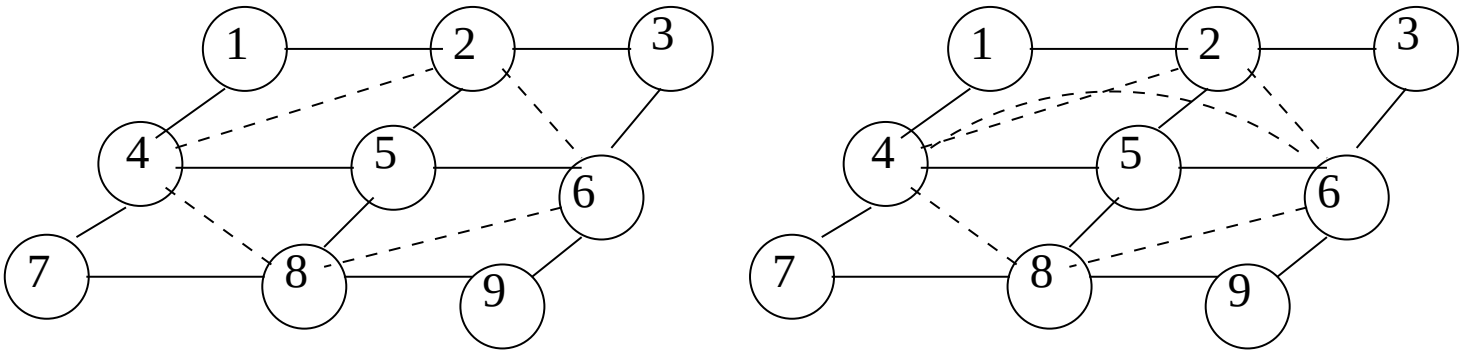

Figure 7: Left: a non triangulated grid. Note the dotted 2-4-6-8 loop is a chordless 4-cycle. Right: one possible triangulation.

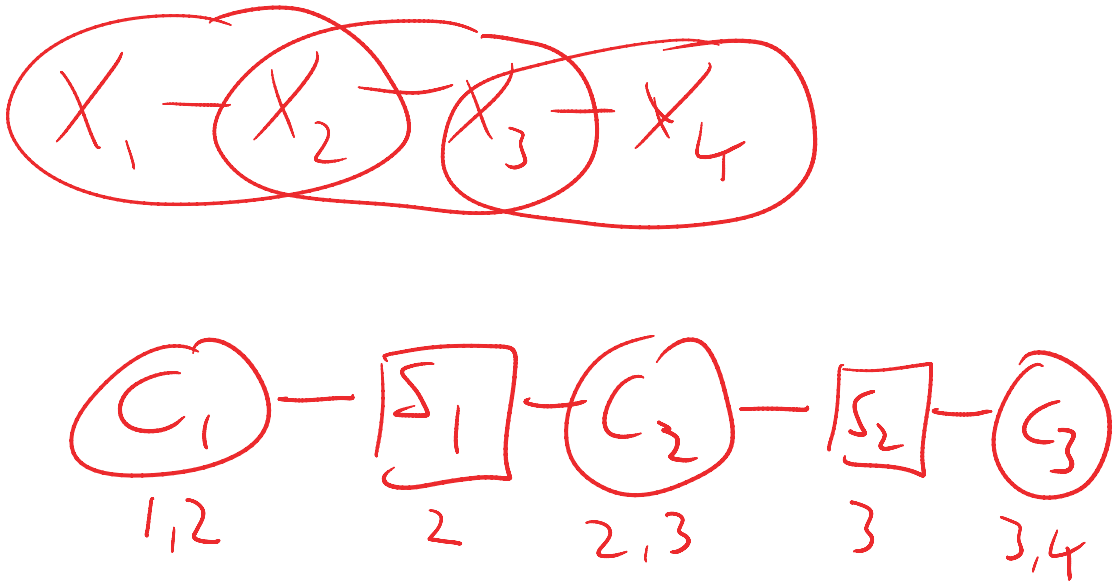

Figure 8: Cliques (ovals) and separators (squares) for a chain structured MRF.

For example, in the case of the chain, we have

$$p(X_{1:4}) = \frac{p(X_1, X_2)p(X_2, X_3)p(X_3, X_4)}{p(X_2)p(X_3)} \tag{44}$$

$$= p(X_1, X_2)p(X_3|X_2)p(X_4|X_3) \tag{45}$$

$$= p(X_1|X_2)p(X_2|X_3)p(X_3, X_4) \tag{46}$$

which corresponds to the joint distribution of a Markov chain going forwards or backwards in time. (Undirected graphs are acausal, and have no notion of directionality.) It turns out that Equation 43 applies to any decomposable graph, not just chains. The reason is that by dividing by the separator potentials, we convert marginals into conditionals, and then the chain rule applies.

6 Applications of MRFs

6.1 Image denoising

One important application of MRFs is to low level vision. Consider the image in Figure 9 (top left). Let each "true" pixel be $x_i \in \{-1, +1\}$. Suppose this image gets sent over some channel and gets corrupted by noise. What we

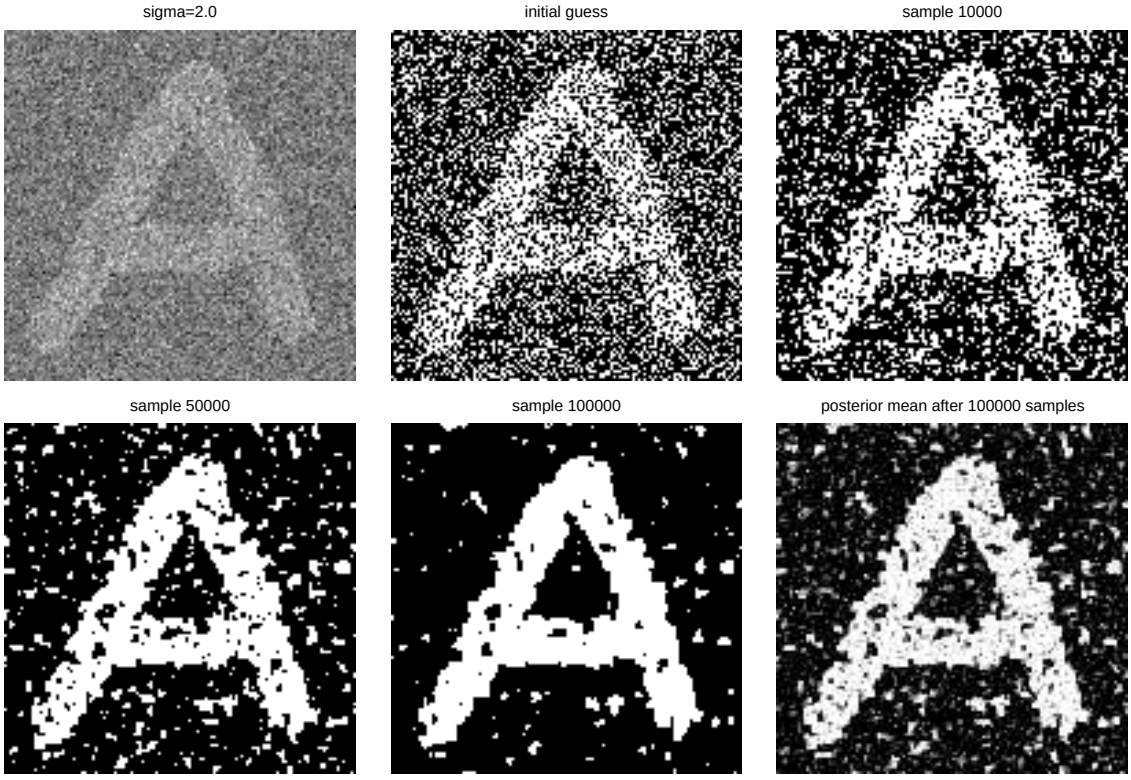

Figure 9: Example of image denoising using Gibbs sampling. We use an Ising prior with $J = 1$ and a Gaussian noise model with $\sigma = 2$.

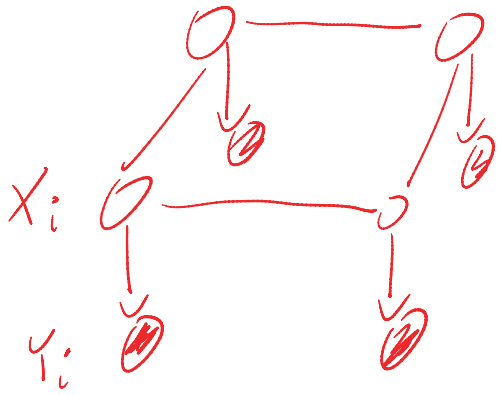

Figure 10: Grid-structured MRF. The shaded nodes y_i are observed, the unshaded nodes x_i are hidden and need to be estimated.

receive is the image in Figure 9 (top middle). Let each observed observed pixel be $y_i \in \mathbb{R}$. Our goal is to *estimate the underlying image from the noisy signal*. There are two formulations of this. The first is to find the single best interpretation:

$$x^* = \arg\max_x p(x_{1:D}|y_{1:D}, \theta) \tag{47}$$

This is called the **maximum a posteriori (MAP) estimate**. An alternative is to find the single most probable guess for each pixel locally:

$$x_i^* = \arg\max_{x_i} p(x_i|y_{1:D}, \theta) \tag{48}$$

These are called the **max marginals**. Both of these are examples of **state estimation (inference)** which we will discuss later.

The simplest model would be to assume that pixels are independent and have a uniform prior (i.e., +1 and -1 are equally likely). For the likelihood, let us assume a Gaussian noise model

$$p(y_i|x_i) = \mathcal{N}(y_i|x_i, \sigma) \tag{49}$$

If σ is very small, then if $x_i = -1$ then $y_i \approx -1$, and similarly if $x_i = +1$ then $y_i \approx +1$, so we will can easily estimate x from y by thresholding each value separately. But if the noise level is high, we need to use some prior to help.

A simple prior is to assume that nearby pixels "like" to be in the same state. In other words, if x_i is "on", its neighbors are more likely to be on; and vice versa if x_i is off. This is called a **smoothness prior**. More precisely, we can write the probability model as

$$\begin{aligned} p(x, y) &= p(x)p(y|x) &(50)\\ &= \left[\frac{1}{Z}\prod_{<ij>}\psi_{ij}(x_i, x_j)\right]\left[\prod_i p(y_i|x_i)\right] &(51)\end{aligned}$$

where $\phi_i(x_i) = p(y_i|x_i)$ is called the **local evidence potential** for node i, and ψ_{ij} is called the **edge potential** for edge $i - j$. (We write $\phi_i(x_i)$ as a function of x_i only, since the y_i's are observed constants.) The notation $\prod_{<ij>}$ means "product over all edges in the graph". We define the edge potential as

$$\psi_{ij}(x_i, x_j) = \exp[J_{ij}x_i x_j] = \begin{pmatrix} e^{J_{ij}} & e^{-J_{ij}} \\ e^{-J_{ij}} & e^{J_{ij}} \end{pmatrix} \tag{52}$$

where $J_{ij} = 0$ if $i - j$ are not neighbors in the graph. If $i - j$ are neighbors, we set $J_{ij} = J > 0$ which gives higher probability (lower energy) to configurations in which $x_i = x_j$ (since $x_i x_j = 1$ if $x_i = x_j$, and $x_i x_j = -1$ if $x_i \neq x_j$). J is the **strength** of our smoothness prior.

Using this model, we can then perform inference to estimate $p(x|y, \theta)$, where $\theta = (J, \sigma)$ are the parameters of the prior and likelihood. In Figure 9, we show the result of using **Gibbs sampling** to draw samples from $p(x|y, \theta)$. We also show an estimate of the posterior mean, $E[X|y, \theta]$. We will explain this and other algorithms for inference later.

6.2 Ising models

The image denoising model above is closely related to the **Ising model**, which is used in statistical physics for modelling magnets. In particular, $x_i \in \{-1, +\}$ represents the atomic spin of a particle, either spin down or up. In an Ising model we assume that the **coupling strength** $J_{ij} = J$ is the same for all edges, and that the **external field** $h_i = H$ is the same for all nodes. In this case, the model becomes

$$\begin{aligned} p(x) &= \frac{1}{Z(\theta)}\exp[-\beta E(x)] &(53)\\ E(x) &= -[\sum_{<ij>} J x_i x_j + \sum_i H x_i] &(54)\end{aligned}$$

where $E(x)$ is the energy of the system, and $\beta = 1/(kT)$, where T is the temperature of the system, and k is Boltzmann's constant. Here $\theta = (J, H, \beta)$ are the parameters of the system. Often we take $H = 0$.

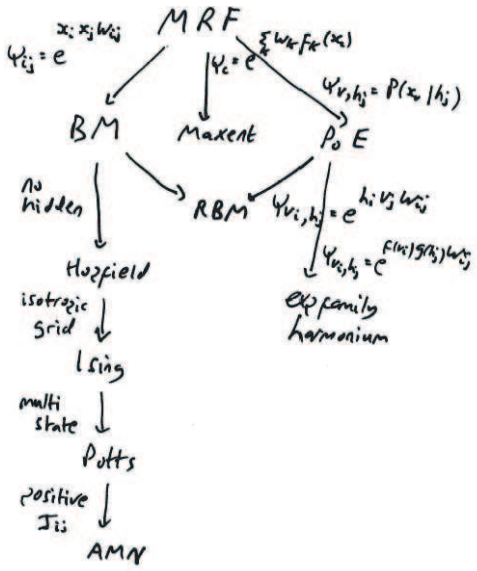

Figure 11: Markov random fields and some variants. BM = Boltzmann machine. RBM = Restricted Boltzmann machine (bipartite graph). PoE = product of experts. AMN = Associative Markov network.

In a **ferromagnet**, neighboring spins want to be the same, so $J > 0$. In this case, the lowest energy (most probable) states are all -1s or all +1s. In an **anti-ferromagnet**, neighboring spins want to be different, so $J < 0$. In this case, the lowest energy (most probable) states are alternating checkerboards of +1/-1; these are called the **ground states** of the system. However, the system can only enter this state at temperature $T = 0$ (so $\beta = \infty$). At higher temperatures, differences in energy matter less, so the system will fluctuate around, and will spend time in other states (this is the motivation behind simulated annealing).

If the J_{ij}'s and h_i's are not constant, the result is called a **spin glass** in physics, or a **Boltzmann machine** or **Hopfield network** in machine learning. If the J_{ij}'s have mixed sign, then resulting system might exhibit **frustration**, since some states will want to be +1, and some -1. We discuss methods to learn the parameters J_{ij} below.

If we generalize the Ising model from binary to multiple states, we get a **Potts model**. For example, for 3 state variables, the edge potentials look like this

$$\psi_{ij}(x_i, x_j) = \exp[J \times I(x_i = x_j)] = \begin{pmatrix} e^J & e^{-J} & e^{-J} \\ e^{-J} & e^J & e^{-J} \\ e^{-J} & e^{-J} & e^J \end{pmatrix} \tag{55}$$

Obviously we can make J_{ij} be non-constant in this case, too. If all $J_{ij} > 0$, this is called an **associative Markov network**. See Figure 11 for some other model variants.

7 Parameter estimation*

In this section, we will assume that all the variables are discrete, and that there is one parameter for each possible assignment to a clique potential, i.e., $\psi_c(x_c)$ can be represented as a table of $K^{|c|}$ numbers, where $|c|$ is the size of clique c. (The results in this section also apply to Gaussian MRFs.) In Section 7.3, we consider a more general parameterization in which ψ_c is defined in terms of a set of "features". This can require fewer than $K^{|c|}$ parameters, which is useful if K or $|c|$ is large. We will focus on maximum likelihood estimation, although Bayesian approaches can also be used (see e.g., [MG04]).

The log-likelihood of a set of N iid datasets x_n is

$$\ell(\theta) = \log \prod_n \frac{1}{Z} \prod_c \psi_c(x_{n,c}|\theta_c) \tag{56}$$

$$= \sum_n \sum_c \log \psi_c(x_{nc}|\theta_c) - N \log Z \tag{57}$$

$$= \sum_c \sum_{x_c} N(x_c) \log \psi_c(x_c) - N \log Z \tag{58}$$

where $N(x_c) = \sum_n I(x_c, x_{nc})$ is the number of times clique c is in configuration x_c in the data. For example, consider a chain $X_1 - X_2 - X_3 - X_4$. Then

$$\ell(\theta) = \sum_{x_1,x_2} N(x_1, x_2) \log \psi_{12}(x_1, x_2) + \sum_{x_2,x_3} N(x_2, x_3) \log \psi_{23}(x_2, x_3) \tag{59}$$

$$+ \sum_{x_3,x_4} N(x_3, x_4) \log \psi_{34}(x_3, x_4) - N \log Z \tag{60}$$

So the derivative wrt one of the clique potentials, say ψ_{12}, is

$$\frac{\partial \ell}{\partial \psi_{12}(x_1', x_2')} = \frac{N(x_1', x_2')}{\psi_{12}(x_1', x_2')} - N \frac{\partial}{\partial \psi_{12}(x_1', x_2')} \log Z \tag{61}$$

The key question is: what are the derivatives wrt the log partition function? In this example we find

$$\frac{\partial \log Z}{\partial \psi_{12}(x_1', x_2')} = \frac{1}{Z} \frac{\partial}{\partial \psi_{12}(x_1', x_2')} \sum_{x_1,x_2,x_3,x_4} \psi_{12}(x_1, x_2)\psi_{23}(x_2, x_3)\psi_{34}(x_3, x_4) \tag{62}$$

$$= \frac{1}{Z} \sum_{x_3,x_4} \frac{\partial}{\partial \psi_{12}(x_1', x_2')} \psi_{12}(x_1', x_2')\psi_{23}(x_2', x_3)\psi_{34}(x_3, x_4) \tag{63}$$

$$= \frac{1}{Z} \sum_{x_3,x_4} \psi_{23}(x_2', x_3)\psi_{34}(x_3, x_4) \tag{64}$$

$$= \sum_{x_3,x_4} \frac{1}{Z} \frac{\psi_{12}(x_1', x_2')\psi_{23}(x_2', x_3)\psi_{34}(x_3, x_4)}{\psi_{12}(x_1', x_2')} \tag{65}$$

$$= \sum_{x_3,x_4} \frac{p(x_1', x_2', x_3, x_4)}{\psi_{12}(x_1', x_2')} \tag{66}$$

$$= \frac{p(x_1', x_2')}{\psi_{12}(x_1', x_2')} \tag{67}$$

And in general we have

$$\frac{\partial \log Z}{\partial \psi_c(x_c)} = \frac{p(x_c)}{\psi_c(x_c)} \tag{68}$$

so the derivative of the log-likelihood is

$$\frac{\partial \ell}{\partial \psi_c(x_c)} = \frac{N(x_c)}{\psi_c(x_c)} - N \frac{p(x_c))}{\psi_c(x_c)} \tag{69}$$

Hence $\frac{\partial \ell}{\partial \psi_c(x_c)} = 0$ implies the following constraint must hold at the MLE:

$$p^{ML}(x_c) = \frac{N(x_c)}{N} \stackrel{\text{def}}{=} \tilde{p}(x_c) \tag{70}$$

where $\tilde{p}(x_c)$ is the empirical distribution on clique c. In other words, at the maximum likelihood setting of the parameters, for each clique, *the model marginals must be equal to the observed marginals* (empirical counts).

This doesn't tell us how to get the ML parameters, it just gives us a condition that must be satisfied when we have them. The method we have to use depends on whether the graph is decomposable or not, and whether the potentials are defined on maximal cliques or on sub cliques, and on whether the potentials are "fully parameterized" (one parameter per clique configuration, as in a tabular representation), or whether they are parameterized more generally. We summarize the options below.

Decomposable?	Max. Cliques	Potential	Method
Yes	Yes	Tabular	Closed form
-	-	Tabular	IPF
-	-	-	Gradient descent

7.1 MLE for decomposable graphs with tabular potentials on the max cliques

If the graph is decomposable, and the clique potentials are defined on maximal cliques, and the potentials are tabular, then we can just write down the MLEs in terms of the empirical counts, since these simultaneously satisfy all the above constraints. Recall that for a decomposable MRF

$$p(x_{1:d}) = \frac{\prod_c \psi_c(x_c)}{\prod_s \phi_s(x_s)} \tag{71}$$

The MLE is gotten by setting $\psi_c = \tilde{p}(x_c)$ and $\psi_s = \tilde{p}(x_s)$. For example, consider a chain $X_1 - X_2 - X_3$. We set

$$\hat{\psi}_{12}^{ML}(x_1, x_2) = \tilde{p}(x_1, x_2) \tag{72}$$

$$\hat{\psi}_{23}^{ML}(x_2, x_3) = \tilde{p}(x_2, x_3), \tag{73}$$

$$\hat{\phi}_{12}^{ML}(x_2) = \tilde{p}(x_2) \tag{74}$$

7.2 Iterative proportional fitting (IPF)

Consider the graph in Figure 5(d):

$$p(x_{1:4}) = \frac{1}{Z} \prod_{<ij>} \psi_{ij}(x_{ij}) \tag{75}$$

Although this is decomposable, the potentials are not defined on the maximal cliques. So the ψ_{24} term contributes to both ψ_{124} and ψ_{234}. Hence we cannot just set the potentials to the empirical marginals. Instead, we will use an iterative scheme.

Recall that

$$\frac{\partial \ell}{\partial \psi_c(x_c)} = \frac{N(x_c)}{\psi_c(x_c)} - N\frac{p(x_c))}{\psi_c(x_c)} = 0 \tag{76}$$

From this we infer

$$\frac{N(x_c)}{N}\frac{1}{\psi_c^t(x_c)} = \frac{p^t(x_c)}{\psi_c^t(x_c)} \tag{77}$$

where the t superscript denotes the values of the parameters at iteration t. We can solve this fixed point equation iteratively

$$\psi_c^{t+1}(x_c) = \frac{\tilde{p}(x_c)}{p^t(x_c)} \tag{78}$$

This is called the **iterative proportional fitting (IPF)** algorithm. In pseudo code, we have

for each clique c
  $\tilde{p}_c =$ normalize(empirical counts(X_c))

while not converged
  for each clique c
    $\hat{p}_c^t = p(x_c|\psi^t)$ (*)
    $\psi_c^{t+1} = \psi_c^t \times \frac{\tilde{p}_c}{\hat{p}_c^t}$

The line marked (*) requires inference to compute the marginals. Note that if the graph is decomposable, IPF will terminate in one iteration.

IPF is essentially a **coordinate ascent** method, where we update a whole clique potential at each step. Below we will consider gradient-based methods that update all the parameters at each step.

We now give an example of fitting an MRF with two disconnected nodes. We use brute force enumeration for inference. Since there are only two variables, we can use a 2D matrix to represent the joint, and 1D vectors to represent the marginals. We observed some count data $C(X_1, X_2)$ and want to choose potentials ψ_1 and ψ_2 such that they match the marginals $C(X_1) = 0.4157, 0.2584, 0.3258$ and $C(X_2) = 0.3596, 0.3596, 0.2809$ respectively. We find that the potentials stop changing after the first iteration, as expected. The final answers are

$$\psi_1 = 0.1386, 0.0861, 0.1086 \quad (79)$$

$$\psi_2 = 1.0787, 1.0787, 0.8427 \quad (80)$$

If $J(x_1, x_2) = \psi_1(x_1) \times \psi_2(x_2)$, You can check manually that $\sum_{x_1} J(x_1, x_2) = c_2(x_2)$ and $\sum_{x_2} J(x_1, x_2) = c_1(x_1)$.

```
% IPFdemo
% Approximate joint density as a product of two marginals
% i.e., fit a 2 node disconnedted MRF  X1  X2

C12 = [25 10 2;
       3  19 1;
       4 3 22];
C12 = normalise(C12);
C1 = sum(C12,2);
C2 = sum(C12, 1);

nstates = [3 3];
psi1 = ones(1,3);
psi2 = ones(1,3);

for iter=1:2
  joint = psi1(:) * psi2(:)';

  M1 = sum(joint,2);
  psi1 = psi1 .* (C1 ./ M1)'

  joint = psi1(:) * psi2(:)';
  M2 = sum(joint,1);
  psi2 = psi2 .* (C2 ./ M2)
end

joint = psi1(:) * psi2(:)';
assert(approxeq(C1, sum(joint,2)))
assert(approxeq(C2, sum(joint,1)))
```

We now give an example of fitting a loopy $1-2-3-1$ graph. Again we use brute force enumeration for inference. This takes about 10 iterations to converge. Because the joint is now on three variables, we use the tabular potential class to take care of all the bookkeeping involved in matching dimensions.

```
% IPFdemo3
% Fit loopy MRF 1-2-3-1 using iterative proportional fitting

clqs = {[1 2], [2 3], [1 3]};
NC = length(clqs);
N = 3;

% Some count data
C = reshape([53 414 11 37 0 16 4 139], [2 2 2]);
C = normalise(C);
Cpot = tabularPot(1:N, 2*ones(1,N), C);
for c=1:NC
```

```
  counts{c} = marginalizePot(Cpot, clqs{c});
end

% Initial guess is all 1's
for c=1:NC
  pots{c} = tabularPot(clqs{c}, 2*ones(1,length(clqs{c})));
end
converged = 0;
iter = 0;
thresh = 1e-3; % convergence threshold
while ~converged
  converged = 1;
  for c=1:NC
    potsOld{c} = pots{c};
  end
  iter = iter + 1;
  fprintf('iter %d\n', iter);
  for c=1:NC
    J = multiplyPots(pots{:});
    Mc = marginalizePot(J, clqs{c});
    pots{c}.T = pots{c}.T .* (counts{c}.T ./ Mc.T);
    if ~approxeq(pots{c}.T, potsOld{c}.T, thresh)
      converged = 0;
    end
    fprintf('c=%d\n', c)
    printTable(pots{c})
  end
end

J = multiplyPots(pots{:});
for c=1:NC
  Mc = marginalizePot(J, clqs{c});
  assert(approxeq(counts{c}.T, Mc.T))
end
```

7.3 Gradient based methods for maxent models

For a maxent model with general features, where $\psi_c(x_c) = \theta_c^T f_c(x_c)$, the previous derivation simplifies somewhat. The log-likelihood becomes

$$\ell(\theta) = \sum_n \sum_k \theta_k f_k(x_n) - N \log Z(\theta) \tag{81}$$

Hence the derivative is

$$\frac{\partial \ell}{\partial \theta_j} = \sum_n f_j(x_n) - N \frac{\partial}{\partial \theta_j} \log Z \tag{82}$$

Let us focus on the second term, the derivative wrt the log partition function.

$$\begin{aligned}
\frac{\partial \log Z(\theta)}{\partial \theta_j} &= \frac{1}{Z(\theta)} \frac{\partial Z(\theta)}{\partial \theta_j} && (83)\\
&= \frac{1}{Z(\theta)} \sum_x \frac{\partial}{\partial \theta_j} \exp[\sum_k \theta_k f_k(x)] && (84)\\
&= \frac{1}{Z(\theta)} \sum_x \frac{\partial}{\partial \theta_j} \prod_k \exp[\theta_k f_k(x)] && (85)\\
&= \frac{1}{Z(\theta)} \sum_x [\frac{\partial}{\partial \theta_j} \exp[\theta_j f_j(x)] \prod_{k \neq j} \exp[\theta_k f_k(x)] && (86)\\
&= \frac{1}{Z(\theta)} \sum_x [f_j(x) \exp[\theta_j f_j(x)]] \prod_{k \neq j} \exp[\theta_k f_k(x)] && (87)\\
&= \frac{1}{Z(\theta)} \sum_x f_j(x) \prod_k \exp[\theta_k f_k(x)] && (88)\\
&= \sum_x f_j(x) p(x) && (89)\\
&= E[f_j(X)] && (90)
\end{aligned}$$

Hence the gradient is

$$\frac{\partial \ell}{\partial \theta_j} = \sum_n f_j(x_n) - N E[f_j(X)] \tag{91}$$

We can find the globally optimal MLE by passing this gradient to any **gradient-based optimization algorithm**, such as conjugate gradient, quasi-Newton, etc. (In matlab, you can use `fminunc` (function minimization unconstrained) in the optimimization toolbox.) However, to compute the gradient, we have to evaluate

$$E[f_j(X)] = \sum_x p(x) f_j(x) \tag{92}$$

for every feature f_j, which might be expensive. Typically, each feature only depends on a subset of the variables. In particular, for a graphical model, each feature only depends on the variables in its clique, so what we need are ways to compute $p(x_c)$ for each clique. We discuss this later.

At the optimum, $\frac{\partial \ell}{\partial \theta_k} = 0$, so the expected features according to the model should match the expected features according to the data:

$$\frac{1}{N} \sum_n f_j(x_n) = E[f_j(X)] \tag{93}$$

Hence maximum likelihood of an exponential family model gives the same results as maximum entropy, where the constraints are that the expected features match the empirical features. In the case that there is one feature for each possible clique value, then $E[f_{cj}(X_c)] = p(X_c = j)$, so this is saying the model's marginal probabilities should equal the empirical marginal probabilities, as we saw before.

7.4 Regularization

Since maximum likelihood often overfits, it is very common to find MAP estimates

$$\hat{\theta}^{MAP} = \arg\max_\theta \tilde{\ell}(\theta) = \arg\max_\theta \log p(\theta) + \log p(D|\theta) \tag{94}$$

where $\tilde{\ell}(\theta)$ is the **penalized (regularized) log likelihood**. It is common to use a Gaussian prior,

$$p(\theta) = \mathcal{N}(\theta|\mu, \Sigma) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp[-\tfrac{1}{2}(\theta - \mu)^T \Sigma^{-1} (\theta - \mu)] \tag{95}$$

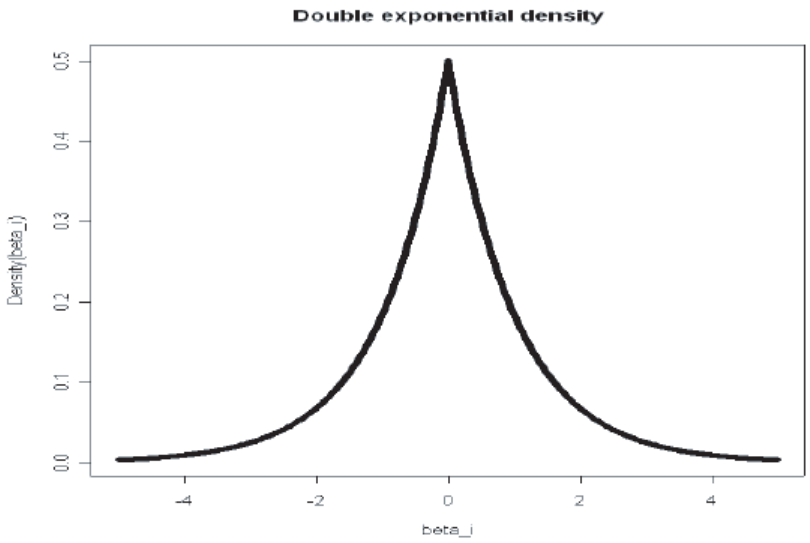

Figure 12: A Laplace prior

.

If we set $\mu = 0$ and $\Sigma = \lambda^{-1} I$, where λ is called the precision, then the log prior becomes

$$\log p(\theta) = -\lambda ||\theta||^2 + \text{const} \tag{96}$$

where $||\theta||_2^2 = \sum_j \theta_j^2$. The objective function becomes

$$\tilde{\ell}(\theta) = \log p(D|\theta) - \lambda ||\theta||_2^2 \tag{97}$$

Hence λ reflects the degree of regularization. This is called an **L2 penalty term**, since it penalizes the L_2 norm of the parameters. It is also called **weight decay**, since it encourages parameters to become small. (A priori parameters are most likely to be zero.)

The derivative of the penalized log likelihood becomes

$$\frac{\partial \tilde{\ell}}{\partial \theta_j} = [\sum_n f_j(x_n)] - NE[f_j(X)] - 2\lambda\theta_j \tag{98}$$

and can be optimized using gradient methods. The term λ is often set by cross validation.

An alternative to a Gaussian prior is to use a **Laplace (double-sided exponential)** prior:

$$p(\theta) = \prod_{j=1}^{p} \text{Laplace}(\theta_j|\lambda) = \prod_j \frac{-\lambda}{2} \exp(-\lambda|\theta_j|) = -\left(\frac{\lambda}{2}\right)^p \exp(-\lambda||\theta||_1) \tag{99}$$

See Figure 12. Hence the objective function becomes

$$\tilde{\ell}(\theta) = \log p(D|\theta) - \lambda ||\theta||_1 \tag{100}$$

This is called **L1 regularization**. Note that this function is non-differentiable if any $\theta_j = 0$, so care must be taken when optimizing this objective. The details are beyond the scope of this chapter.

In the context of linear regression, L2 regularization is called **ridge regression** and L1 regularization is called **lasso** (least absolute shrinkage and selection operator). The key difference between them is that L2 encourages all

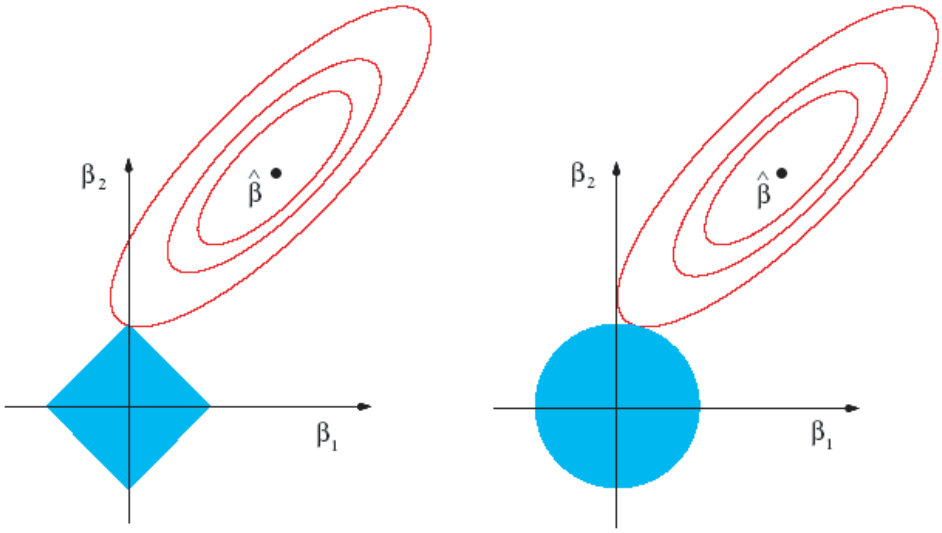

Figure 13: Illustration of L1 (left) vs L2 (right) regularization. Source: [HTF01] Figure 3.12.

parameters to become small, whereas L1 encourages some parameters to become exactly zero, and the rest to remain more or less unchanged. Thus L1 can be used to perform **feature selection**.

To see why this happens, note that an L2 penalty on the parameter vector $\theta = (1, 0)$ is the same as on $\theta = (1/\sqrt{(2)}, 1/\sqrt{(2)})$, since

$$||(1,0)||_2 = ||(1/\sqrt{(2)}, 1/\sqrt{(2)}||_2 = 1 \tag{101}$$

However, for L1, setting $\theta = (1, 0)$ is cheaper than setting $\theta = (1/\sqrt{(2)}, 1/\sqrt{(2)})$, since

$$||(1,0)||_1 = 1 < ||(1/\sqrt{(2)}, 1/\sqrt{(2)}||_1 = \sqrt{(2)} \tag{102}$$

Hence lasso prefers **sparse solutions**.

Figure 13 shows this result visually. The ellipses represent the quadratic log likelihood function. The shaded blobs represent the log prior, whose size is controlled by λ. The MAP estimate is at the intersection point. It is apparent that with the L1 prior, the solution will always be at one of the corners (since they will "touch" the likelihood surface first).

When performing regression with discrete features (factors), we need to include or exclude a block of columns for the result to be meaningful. This is called the **group lasso**. See [YL06] for details.

8 Structure learning *

Given a dataset, we might want to find a graph structure that represents it. More formally, the dataset defines an **empirical distribution**

$$\hat{p}^{emp}(x) = \frac{1}{N} \sum_n I(x = x^n) \tag{103}$$

and we want to find a graph that models the independencies in $\hat{p}^emp$. Since the fully connected graph is an I-map of all distributions, this can represent $\hat{p}^{emp}$. However, what we really want is a **minimal I-map**, i.e., the one that makes as few additional independence assumptions as necessary.

One approach to this is to apply a series of conditional independence tests to the data (e.g., using χ^2**-tests**), and then to fit a model consistent with those test results (see [Edw00] for details). Since there are $O(2^{D^2})$ possible graphs on D nodes (since each edge in the adjacency matrix can be present or absent), this can be computationally expensive. In addition to computational cost, there are fundamental statistical problems: conditional independency tests return yes/no answers (based on a threshold). When combining such results, inconsistencies can arise. An

alternative approach is to perform **Bayesian model selection**, which solves the statistical problem, at the cost of making the computational problem harder. We discuss this later.

A final approach, in the case of log-linear models, is to put a **sparsity promoting prior** on the parameters θ_c, which encourages them to go to zero, and then to perform MAP estimation. If all the weights between two nodes are zero, then the edge is effectively removed from the graph.

References

[Bis06] C. Bishop. *Pattern recognition and machine learning*. Springer, 2006. Draft version 1.21.

[Edw00] D. Edwards. *Introduction to graphical modelling*. Springer, 2000. 2nd edition.

[Fre03] B. Frey. Extending factor graphs so as to unify directed and undirected graphical models. In *UAI*, 2003.

[HTF01] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[KF06] D. Koller and N. Friedman. *Bayesian networks and beyond*. 2006. To appear.

[LT79] Richard J. Lipton and Robert E. Tarjan. A separator theorem for planar graphs. *SIAM Journal of Applied Math*, 36:177–189, 1979.

[MG04] I. Murray and Z. Ghahramani. Bayesian Learning in Undirected Graphical Models: Approximate MCMC algorithms. In *UAI*, 2004.

[PPL97] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(4), 1997.

[YL06] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *J. Royal Statistical Society, Series B*, 68(1):49–67, 2006.