# Notes of PRML

Siyu Wang

August 2018

# Chapter 1

# Mathematical Foundations

# Chapter 2

# Probabilistic Models

## 2.1 Graphical Models

We shall find it highly advantageous to augment the analysis using diagrammatic representations of probability distributions, called *probabilistic graphical models*. These offer several useful properties:

1. They provide a simple way to visualize the structure of a probabilistic model and can be used to design and motivate new models.
2. Insights into the properties of the model, including conditional independence properties, can be obtained by inspection of the graph.
3. Complex computations, required to perform inference and learning in sophisticated models, can be expressed in terms of graphical manipulations, in which underlying mathematical expressions are carried along implicitly.

*nodes*: random variables. *links*: probabilistic relationships between the variables.
*Bayesan networks (directed graphical models)*, *undirected graphical models*, *factor graph*.

### 2.1.1 Bayesian Networks

Decomposition of a joint probability distribution:

$$p(a, b, c) = p(c|a, b)p(b|a)p(a) \tag{8.1}$$

and the corresponding graphical model can be seen in figure2.1.
**Rule**: for each conditional distribution we add directed links (arrows) to the graph from the nodes corresponding to the variables on which the distribution is conditioned.
we can extend the decomposition to $K$ variables:

$$p(x_1, ..., x_K) = p(x_K|x_1, ..., x_{K-1})...p(x_2|x_1)p(x_1). \tag{2.1}$$

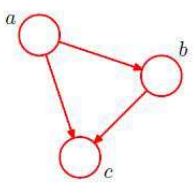


Figure 2.1:

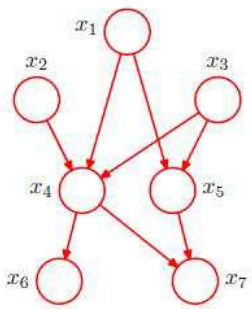


Figure 2.2:

And the corresponding directed graph is called *fully connected* because there is a link between every pair of nodes.
**Case** Given a directed graph as seen in 2.2, we can write the corresponding probability product:

$$p(x_1)p(x_2)p(x_3)p(x_4|x_1,x_2,x_3)p(x_5|x_2,x_3)p(x_6|x_4)p(x_7|x_4,x_5) \tag{2.2}$$

**Rule**:

$$p(\mathbf{x}) = \Pi_{k=1}^{K} p(x_k|\mathrm{pa}_k) \tag{2.3}$$

where $\mathrm{pa}_k$ denotes the set of parents of $x_k$, and $\mathbf{x} = \{x_1, .., x_K\}$.
The directed graphs that we are considering are subject to an important restriction namely that there must be no *directed cycles*, in other words there are no closed paths within the graph such that we can move from node to node along links following the direction of the arrows and end up back at the starting node. Such graphs are also called *directed acyclic graphs*, or *DAGs*. This is equivalent to the statement that there exists an ordering of the nodes such that there are no links that go from any node to any lower numbered node.

### Example: polynomial regression

For more complex models, we shall adopt the convention that random variables will be denoted by open circles, and deterministic parameters will be denoted by smaller solid circles. some concepts: *observed variables, hidden variables, deterministic parameters.*

### Generative models

*sampling*: given a joint distribution, we want to draw a sample $\hat{x}_1, \hat{x}_2, ..., \hat{x}_K$ from it. **ancestral sampling**
We start with the lowest-numbered node and draw a sample from the distribution $p(x_1)$, which we all $\hat{x}_1$. Then we work through each of the nodes in order, so that for node n we draw a sample from the conditional distribution p(xn—pan) in which the parent variables have been set to their sampled values. Note that at each stage, these parent values will always be available because they correspond to lower-numbered nodes that have already been sampled. The graphical model captures the *causal* process by which the observed data was generated. For this reason, such models are often called *generative* models. But the regression problem is not generative because there is no probability distribution associated with the input variable $x$, and it is not possible to generate synthetic data points from this model. But we can make it generative by introducing a suitable prior distribution $p(x)$, at the expense of a more complex model.

### Discrete variables

*parent-child pair in a directed graph. Two cases are particularly worthy of note, namely when the parent and child node each correspond to discrete variables and when they each correspond to Gaussian variables, because in these two cases the relationship can be extended hierarchically to construct arbitrarily complex directed acyclic graphs.*
**Case**: discrete variable $\mathbf{x}$ having $K$ possible states is given by

$$p(\mathbf{x}|\mathbf{mu}) = \Pi_{k=1}^{K} \mu_k^{x_k} \tag{2.4}$$

governed by $\mu$. Suppose we have to discrete variables $\mathbf{x}_1, \mathbf{x}_2$, joint distribution can be written as

$$p(\mathbf{x}_1, \mathbf{x}_2|\mu) = \Pi_{k=1}^{K} \Pi_{k=1}^{K} \mu_{kl}^{x_{1k}x_{2l}}. \tag{2.5}$$

$x_{1k}$ denotes the $k^{th}$ component of $\mathbf{x}_1$.
$K^2 - 1$ parameters. If we have $M$ variables, there is $K^M - 1$ parameters, exponential with the num $M$.
Suppose $\mathbf{x}_1$ and $\mathbf{x}_2$ are independent. Each variable is then described by a separate multinomial distribution, and total number of parameters will be $2(K-1)$. If extend to $M$ variables, there will be $M(K-1)$ variables, linear growth.
**Generally case**: more links than independent variables but less links than a fully connected graph.
We can turn a graph over discrete variables into a Bayesian model by introducing Dirichlet priors

for the parameters.
Another way of controlling the exponential growth in the number of parameters in models of discrete variables is to use parameterized models for the conditional distributions instead of complete tables of conditional probability values.

**Linear-Gaussian models**

### 2.1.2 Conditional Independence

If

$$p(a|b,c) = p(a|c), \tag{2.6}$$

we say that $a$ is conditionally independent of $b$ given $c$, and we denote this by

$$a \perp\!\!\!\perp b|c.$$

And we will have

$$p(a,b|c) = p(a|c)p(b|c). \tag{2.7}$$

**Examples**

Three examples related to conditional independence: *tail-to-tail, head-to-tail, head-to-head.*

**D-separation**

We wish to ascertain whether a particular conditional independence statement

$$A \perp\!\!\!\perp B|C$$

is implied by a given directed acyclic graph.
i.i.d data points. Given $\mu$, we can say the data points $x_1, x_2, ..., x_N$ are conditionally independent, but we can not say the data points are independent. because given $x_1$, the probability of $\mu$ will be affected and then $x_2$ will be affected.
**naive Bayes Model** Observation of $\mathbf{z}$ will block the path between $x_i$ and $x_j$ for $j \neq i$. So, if we are given a training set will, comprising inputs $\{x_1, ..., x_N\}$ together with their labels, then we can fit the naive Bayes model to the training data using maximum likelihood assuming that the data are drawn independently from the model.
We can view a directed graph as a filter and all probability distribution $p(\mathbf{x})$ that will be allowed through can make a set $\mathcal{DF}$.
**Markov blanket**.

$$p(\mathbf{x}_i|\mathbf{x}_{\{j\neq i\}}) = \frac{p(\mathbf{x}_1,...\mathbf{x}_D)}{\int p(\mathbf{x}_1,...,\mathbf{x}_D)\mathrm{d}\mathbf{x}_\mathrm{i}} = \frac{\prod_k p(\mathbf{x}_k|\mathrm{pa}_\mathrm{k})}{\int \prod_k p(\mathbf{x}_k|\mathrm{pa}_\mathrm{k})\mathrm{d}\mathbf{x}_\mathrm{i}} \tag{2.8}$$

Some factors in $\prod_k p(\mathbf{x}_k|\mathrm{pa}_\mathrm{k})$ will disappear if they are not relative to $\mathbf{x}_i$.

### 2.1.3 Markov Random Fields

A *Markov random field*, also known as a *Markov network* or an *undirected graphical model* , has a set of nodes each of which corresponds to a variable or group of variables, as well as a set of links each of which connects a pair of nodes. The links are undirected, that is they do not carry arrows.

**Conditional independence properties**

Properties in an undirected graph is much simpler than those in a directed graph because the nodes linked in a pair is symmetric, and there is no head-to-head situation. So if all paths that connect nodes in $A$ to nodes in $B$ pass through one or more nodes in the set $C$, then all such paths are 'blocked' and so the conditional independence property hold, aka.

$$A \perp\!\!\!\perp B|C$$

.

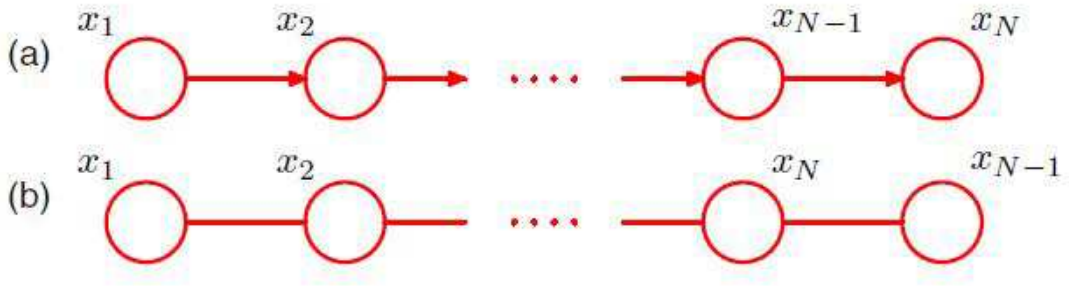


Figure 2.3:

**Factorization properties**

If two nodes are not linked directly, then we can obtain

$$p(x_i, x_j|\mathbf{x}_{\setminus\{i,j\}}) = p(x_i|\mathbf{x}_{\setminus\{i,j\}})p(x_j|\mathbf{x}_{\setminus\{i,j\}}) \tag{2.9}$$

**clique**:a subset of nodes in a graph such that there exists a link between all pairs of nodes in the subset.
**maximal clique**:a clique such that it is not possible to include any other nodes from the graph in the set without it ceasing to be a clique.
Denote a clique by $C$ and the set of variables in that clique by $\mathbf{x}_C$. Then the joint distribution is written as a product of *potential functions* $\phi_C(\mathbf{x}_C)$ over maximal cliques of the graph

$$p(\mathbf{x}) = \frac{1}{Z}\prod_C \phi_C(\mathbf{x}_C). \tag{2.10}$$

Here the quantity Z, sometimes called the partition function, is a normalization constant and is given by

$$Z = \sum_X \prod_C \phi_C(\mathbf{x}_C) \tag{2.11}$$

*energy function* $E(\mathbf{x}_C)$

$$\phi_C(\mathbf{x}_C) = \exp\{-E(\mathbf{x}_C\} \tag{2.12}$$

**Relation to directed graphs**

Given a directed graph as a chain shown in figure2.3(a).

$$p(\mathbf{x}) = p(x_1)p(x_2|x_1)p(x_3|x_2)...p(x_N|x_{N-1}) \tag{2.13}$$

Also, for undirected graph shown in figure2.3(b).

$$p(\mathbf{x}) = \frac{1}{Z}\psi_{1,2}(x_1, x_2)\psi_{2,3}(x_2, x_3)...\psi_{N-1,N}(x_{N-1}, x_N). \tag{2.14}$$

And we define

$$\psi_{1,2}(x_1, x_2) = p(x_1)p(x_2|x_1) \tag{2.15}$$
$$\psi_{2,3}(x_2, x_3) = p(x_3|x_2) \tag{2.16}$$
$$\vdots \tag{2.17}$$
$$\psi_{N-1,N}(x_{N-1}, x_N) = p(x_N|x_{N-1}) \tag{2.18}$$

# Chapter 3

# Algorithms

## 3.1 Inference Algorithms in Graphical Models

### 3.1.1 Inference on a chain

Consider the graph shown in figure2.3. The joint distribution for this graph takes the form

$$p(\mathbf{x}) = \frac{1}{Z}\psi_{1,2}(x_1, x_2)\psi_{2,3}(x_2, x_3)...\psi_{N-1,N}(x_{N-1}, x_N). \tag{3.1}$$

If we want to get marginal distribution $p(x_n)$ for a specific node, we should sum the joint distribution over all variables except $x_n$, so that

$$p(x_n) = \sum_{x_1} ... \sum_{x_{n-1}} \sum_{x_{n+1}} ... \sum_{x_N} p(\mathbf{x}) \tag{3.2}$$

But this way will cost so much computational resources, so we want to find another simpler way:

$$p(x_n) = \frac{1}{Z}[\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \dots [\sum_{x_2} \psi_{2,3}(x_2, x_3)[\sum_{x_1} \psi_{1,2}(x_1, x_2)]] \dots ] \tag{3.3}$$

$$[\sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \dots [\sum_{x_N}] \psi_{N-1,N}(x_{N-1}, x_N)] \dots ] \tag{3.4}$$

$$= \frac{1}{Z}\mu_\alpha(x_n)\mu_\beta(x_n) \tag{3.5}$$

As shown in figure3.1, we can see message passed in the chain and we can compute $\mu_\alpha(x_i)$ and $\mu_\beta(x_i)$ recursively for any $i = 1, 2, \dots, N$.

$$\mu_\alpha(x_n) = \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n)\mu_\alpha(x_{n-1}). \tag{3.6}$$

and

$$\mu_\alpha(x_2) = \sum_{x_1} \psi_{1,2}(x_1, x_2). \tag{3.7}$$

Similar for $\mu_\beta(x_n)$

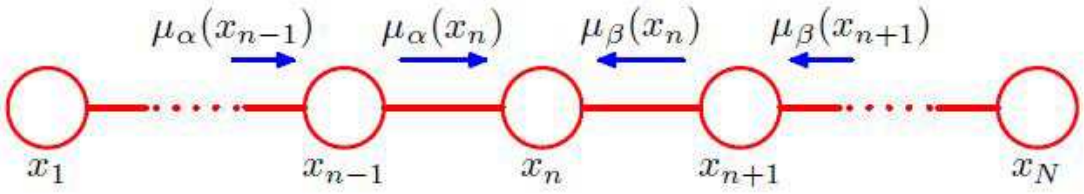


Figure 3.1:

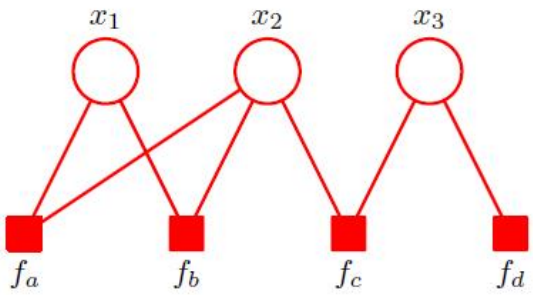


Figure 3.2: Example of a factor graph

### 3.1.2 Inference on a tree: sum-product algorithm

Tree do not have loops.

#### factor graphs

In a factor graph, there is a node (depicted as usual by **a circle**) for every variable in the distribution, as was the case for directed and undirected graphs. There are also additional nodes (depicted by **small squares**) for each factor fs(xs) in the joint distribution.
For example,

$$p(\mathbf{x}) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3) \tag{3.8}$$

can be expressed by the factor graph shown in figure3.2 Factor graphs are said to be bipartite because they consist of two distinct kinds of nodes, and all links go between nodes of opposite type.

#### sum-product algorithm

We shall now make use of the factor graph framework to derive a powerful class of efficient, exact inference algorithms that are applicable to tree-structured graphs. Here we shall focus on the problem of evaluating local marginals over nodes or subsets of nodes, which will lead us to the sum-product algorithm.
Our goal is to exploit the structure of the graph to achieve two things: (i) to obtain an efficient, exact inference algorithm for finding marginals; (ii) in situations where several marginals are required to allow computations to be shared efficiently. **finding the marginal $p(x)$ for a particular variable node $x$**

$$p(\mathbf{x}) = \prod_{s \epsilon ne(x)} F_s(x, X_s)$$

$$p(x) = \prod_{s \epsilon ne(x)} [\sum_{X_s} F_s(x, X_s)] = \prod_{s \epsilon ne(x)} \mu_{f_s \to x}(x)$$

$$F_s(x, X_s) = f_s(x, x_1, ...x_M) G_1(x, X_{s1}) ... G_M(x, X_{sM})$$

$$\mu_{x_m \to f_s}(x_m) \equiv \sum_{X_{sm}} G_m(x_m, X_{sm})$$

We have therefore introduced two distinct kinds of message, those that go from factor nodes to variable nodes denoted $\mu_{f \to x}(x)$, and those that go from variable nodes to factor nodes denoted $\mu_{x \to f}(x)$. **Illustration: example max-sum algorithm** Two other common tasks are to find a setting of the variables that has the largest probability and to find the value of that probability. We therefore seek an efficient algorithm for finding the value of x that maximizes the joint distribution p(x) and that will allow us to obtain the value of the joint distribution at its maximum.

### 3.1.3 Exact inference in a general graph

Goal: deal with graphs having loops.

**Loopy belief progpagation** One simple approach to approximate inference in graphs with loops, which builds directly on the previous discussion of exact inference in trees. The idea is simply to apply the sum-product algorithm even though there is no guarantee that it will yield good results. For some graphs, the algorithm will converge, whereas for others it will not.

We will say that a (variable or factor) node a has a message pending on its link to a node b if node a has received any message on any of its other links since the last time it send a message to b.