Outline

- Directed graphical models.
- Undirected graphical models.
- Message passing and belief propagation.

Reminder: The two fundamental rules for manipulating probabilities

- Practically all mathematical derivations involving probability distributions are based on two basic operations

1. **Sum rule:**
   \[ p(X) = \sum_Y p(X,Y) \]

2. **Product rule:**
   \[ p(X,Y) = p(Y|X)p(X) \]

Some examples of applying the two rules

\[ p(a,b,c,d,e) = p(a,c,d|b,e)p(b,e) \]
\[ p(a,b|c) = p(a|b,c)p(b|c) \]
\[ p(a) = \sum_b p(a,b) \]
\[ p(a|c) = \sum_b p(a,b|c) \]
Probabilistic Graphical Models

- A graph is a collection of nodes (or vertices) and edges.
- Graphs turn out to be highly useful for representing probability distributions.
- They allow us to translate complicated mathematical operations on distributions into simple and intuitive manipulations on graphs.
- A distribution that is represented as a graph is referred to as a (probabilistic) graphical model.

Two types of graphical model

- There are two basic types of graphical model:
  1. Directed graphical models (aka Bayesian networks)
  2. Undirected graphical models
- There is a third, known as factor graph, that can represent both with a single formalism.

Directed graphical models

- We can write a joint distribution over any number of variables as a product, after choosing an ordering for the variables
  \[ p(x_1, \ldots, x_K) = p(x_K | x_1, \ldots, x_{K-1}) \cdots p(x_2 | x_1) p(x_1) \]
- And likewise, we can always represent this distribution as a graph.
- Of course, we may need a lot of parameters to represent all conditionals if \( K \) is large.
- The main idea behind graphical models is to leave out some variables in the conditionals. This will imply that we make conditional independence assumptions.
An example graph

Example: This graph

represents the distribution

\[ p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5) \]

Directed models as DAG

More generally, we can write

\[ p(x) = \prod_{k=1}^{K} p(x_k|p_{a_k}) \]

where \( p_{a_k} \) denotes the set of parent nodes for node \( k \).

The only restriction on the graph for this to work: There must be no directed cycles – The graph must be a directed acyclic graph (DAG).

One can show that this product will always be a normalized distribution.

Markov models and HMMs

Markov models:

![Markov model diagram]

and Hidden Markov models:

![Hidden Markov model diagram]

are examples of graphical models.

- **Shading** is a common way to distinguish observed variables from unobserved (latent) variables.
- Note that in these two models, parameters are usually tied, so all conditional distributions are the same.

More examples

- Inference in the Naive Bayes model (for a single data-point) may be represented graphically as follows:

- Another example: Being Bayesian amounts to letting model parameters be random variables. The problem of inferring the mean of a Gaussian may in that case be represented graphically as follows:
Plates

- It is common to have many copies of the same kind of variable with the same conditional distribution.
- One can depict these using a graphical short-cut known as plate, which is a box with a number in the corner indicating the number of copies of the variable.
- The Bayesian estimation of the mean of a Gaussian can be represented using plates like this:

![Plate Diagram]

Conditional independence

- Random variables $a$ and $b$ are conditionally independent given $c$ if
  \[ p(a|b,c) = p(a|c) \]
- As a result, the conditional joint over $a$ and $b$ factorizes:
  \[ p(a,b|c) = p(a|b,c)p(b|c) = p(a|c)p(b|c) \]
- Conditional independencies are useful for deriving inference and learning methods. A recent example: Hidden Markov Models.
- We can identify the conditional independence properties of a joint distribution by looking at its graph.
- Formally, this approach is known as d-separation, or the “Bayes Ball” algorithm, which can be derived by considering the following three building blocks of a graph:

Three fundamental graphs

Graph 1

\[ p(a,b|c) = \frac{p(a,b,c)}{p(c)} = \frac{p(a|c)p(b|c)}{p(c)} \]
Here, $a$ and $b$ are conditionally independent given $c$. 

Graph 2

\[ p(a,b|c) = \frac{p(a,b,c)}{p(c)} = \frac{p(a)p(c|a)p(b|c)}{p(c)} = \frac{p(a|c)p(b|c)}{p(c)} \]
Here, $a$ and $b$ are conditionally independent given $c$. 

Three fundamental graphs

Graph 3

\[ p(a, b | c) = \frac{p(a, b, c)}{p(c)} = \frac{p(a)p(b)p(c | a, b)}{p(c)} \]

Here, \( a \) and \( b \) are not conditionally independent given \( c \). (They may still be conditionally independent if the distributions turn out to have exactly the right numbers, but there are distributions consistent with this graph, where they are not conditionally independent.)

Explaining away

- In examples 1 and 2, we can think of conditioning on \( c \) as blocking the way from \( a \) to \( b \).
- In example 3, conditioning on \( c \) does not block the way.
- Rather the opposite: By marginalizing both sides of \( p(a, b, c) = p(a)p(b)p(c | a, b) \) it follows that \( p(a, b) = p(a)p(b) \). So without conditioning on \( c \), variables \( a \) and \( b \) are independent.
- The phenomenon that conditioning on \( c \) can make \( a \) and \( b \) dependent is known as Explaining Away.
- Example: Let \( a \) and \( b \) be the outcomes of flipping two coins. They are independent. Let \( c \) be 1 if both coins come up the same and 0 otherwise. Now, when \( c \) is given, \( a \) and \( b \) become dependent!

d-separation

- One can show that explaining away ("unblocking by conditioning") occurs if any of the descendents of node \( c \) is observed not just \( c \) itself.
- Variables \( a \) and \( b \) are called \( d \)-separated by node \( c \) if all paths from \( a \) to \( b \) via \( c \) are blocked (so \( c \) is neither part of, nor a descendent of, an explaining away-subgraph).
- \( d \)-separation in the graph is equivalent to conditional independence. This allows us to determine conditional independencies by inspecting the graph:

- The "Bayes Ball" algorithm: Consider three groups of random variables \( A, B, C \), which are subsets of nodes in a directed graphical model. \( A \) and \( B \) are conditionally independent given \( C \), if any path from a node in \( A \) to a node in \( B \) via a node in \( C \) is blocked.
- Example: Are \( x_1 \) and \( x_6 \) conditionally independent, given \( x_2 \) and \( x_3 \)?

\[ x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \]
Undirected graphical models

- We now turn to a completely different way of using graphs to represent distributions, known as undirected graphical models.
- Undirected graphs are graphs whose edges do not carry any arrows.
- A repercussion will be that there is no explaining away and reading off independence properties is easier.
- For undirected models, the following definitions will be important: A clique is a set of nodes in a graph, which are fully connected.
- A maximal clique is a clique which ceases to be a clique if we include another node from the graph.

Potential functions

- It is common to let $x_C$ denote the vector containing all variables which are part of clique $C$.
- Undirected graphical models are defined by using positive potential functions $\psi_C(x_C)$ over the joint instantiations of all the variables within the maximal cliques of the graph.
- Potential functions are like unnormalized probabilities, in that they allow us to express which joint configurations of the variables are likely (large value for $\psi(x_C)$) and which are unlikely (small value for $\psi(x_C)$).
- The normalized product of all the potential functions then defines the distribution.

Undirected graphical model definition

Let $x_C, C = 1, \ldots, M^C,$ be the maximal cliques in the graph. An undirected graphical model defines the distribution

$$p(x) = \frac{1}{Z} \prod_C \psi_C(x_C)$$

$$Z = \sum x \prod_C \psi_C(x_C)$$

- The normalizing constant, $Z$, is called partition function.
- This seems like a fairly arbitrary definition. But it leads to conditional independence properties which are much more intuitive than in the case of directed models:
Conditional independence

Conditional independence in undirected models
In an undirected model, two subsets of variables $A$ and $B$ are conditionally independent, given a subset of variables $C$, if any path from a node in $A$ to a node in $B$ goes through a node in $C$.

How to define the potential functions

- The potential functions should reflect which joint variable instantiations we want to be likely and which ones we want to be unlikely.
- To define the functions in practice, it is common to use the “exp-trick” in order to ensure positivity:

$$
\psi(x_C) = \exp(-E(x_C))
$$

where $-E(x_C)$ is a real-valued “energy function” which is small for configurations we want to be likely and large for configurations we want to be unlikely.

- The potential functions may contain parameters to be learned from data.

Markov Random Fields

- Example: An MRF is a latent variable model that models pixels $y_i$ in an image using latent variables $x_i$.
- There is one latent variable per pixel and the latent variables are arranged on a grid.
- This allows us to encode the fact that nearby pixels tend to take on similar values.

Markov Random Fields

- For binary images, the joint distribution may be defined as

$$
p(x, y) = \frac{1}{Z} \exp(-h \sum_i x_i + \beta \sum_{i,j} x_i x_j + \eta \sum_i x_i y_j)
$$

where $h, \beta$ and $\eta$ are parameters.
- Several variations of this definition are common in practice.
- The maximal cliques here are pairs of variables.
- We can think of the $x_i$ as underlying “true” pixel values and the $y_i$ as the corrupted (noisy) observations that we get to see.
- Computing $\arg \max_x p(x|y)$ then allows us to recover the underlying image. This is more difficult than in an HMM, because there are cycles. As we shall see, one has to use iterative optimization methods unless the graph is a tree.
MRF Denoising example

Top: Original image, noisy version of the image.
Bottom: Two denoising results obtained by iteratively maximizing $p(x|y)$.

Inference in graphical models

- One of the main advantages of graphical models is that they allow for a general approach to performing inference, that is, summing over, or maximizing over, a subset of nodes, given another subset of nodes.
- This makes it possible to generalize the efficient inference procedure we saw in the HMM to more general types of graphs than chains.
- We first re-visit chain graphs to derive the ideas.
- In contrast to the HMM, we consider here the undirected chain
  $$p(x) = \frac{1}{Z} \prod_n \psi_n(x_{n-1}, x_n)$$
  where the $x_n$ are discrete with $K$ possible states each.

Inference as message passing

- We can now compute the marginal as
  $$p(x_n) = \frac{1}{Z} \mu_\alpha(x_n) \mu_\beta(x_n)$$
- A crucial insight is that we may interpret $\mu_\alpha(x_n)$ and $\mu_\beta(x_n)$ as messages that are being passed from node $x_{n-1}$ or $x_{n+1}$, respectively, to node $x_n$.
- Each message is a vector with $K$ entries, and the product is elementwise.
- The messages are computed recursively in $O(K^2)$.
Inference as message passing

- Suppose we want to compute $p(x_m)$ for some $m \neq n$.
- We could do so using another $O(K^2)$ computations. But this would still be wasteful, because we would re-compute a lot of the same messages.
- Since each $p(x_m) = \frac{1}{Z} \mu_\alpha(x_m) \mu_\beta(x_m)$, a better idea is to compute all $p(x_m)$ at once, using
  - A single forward pass that computes all $\mu_\alpha$.
  - A single backward pass that computes all $\mu_\beta$.
- Note that messages are local, and computing them is a local operation: They involve computations that affect only neighboring nodes.
- We can also compute pair-wise marginals $p(x_{m-1}, x_m)$ using these messages.

Inference in trees

The sum-product algorithm

1. Declare any node in the tree as root node.
2. Pass messages from leaves to the root.
3. Pass messages from root to the leaves.
4. Compute the marginals $p(x_C)$ as a normalized local products.

Conditioning

- Often, it is necessary to compute conditional marginals, such as $p(x_n | y)$.
- Conditioning can be thought of as defining a new graph over the unconditioned variables with modified local potential functions.
- So conditioning does not change the way we do inference.
Extensions

- It is possible to convert directed models to undirected models, so the sum-product algorithm works for both types of graph. In fact, the alpha- and beta-recursions we used for the HMM are examples of this.
- Instead of scalar discrete variables, nodes may represent vectors and/or continuous variables.
- When using continuous variables we need to replace sums by integrals.
- The integrals can be computed in closed form when using Gaussian local probabilities and linear dependencies between nodes.

Max-product and semi-rings

- By replacing sums by max-operations, we can compute an optimal global instantiation of the variables, in analogy to the Viterbi algorithm for HMMs.
- This is known as the max-product algorithm.
- Both the sum-product and max-product algorithm are also known as belief propagation.
- Formally, sum and multiply, as well as max and multiply, each form an algebraic structure known as a semi-ring, and belief propagation will work for any pair of operations for which this is the case.

Loopy belief propagation

- If the graph is not a tree (as, for example, in an MRF), belief propagation does not work, because there will be circular dependencies between messages.
- Under certain conditions, one can still apply message passing iteratively and converge to a meaningful solution. This is known as “Loopy belief propagation” (Loopy BP).
- This approach is fairly common in image processing tasks, like the denoising example we saw before.
- Alternatively, one can resort to other iterative optimization methods to perform inference.

Factor graphs

- Both directed and undirected models represent probability distributions as products:
  \[ p(x) = \prod f_s(x_s) \]
- In a directed model, the factors are local conditionals; in an undirected model, they are potential functions.
- Factor graphs represent both type of model by using two types of nodes: function nodes \( f_s \) and variable nodes \( x_i \).
- Belief propagation is most conveniently formalized mathematically using factor graphs. Messages are then passed from variable nodes to function nodes and vice versa.