Undirected graphical models

- Semantics of probabilistic models over undirected graphs
- Parameters of undirected models
- Example applications

Undirected graphical models

- So far we have used directed graphs as the underlying structure of a Bayes net
- Why not use *undirected* graphs as well?

  E.g., variables might not be in a "causality" relation, but they can still be correlated, like the pixels in a neighborhood in an image
- An undirected graph over a set of random variables $\{X_1, \ldots X_n\}$ is called a **undirected graphical model** or **Markov random field (MRF)** or **Markov network**

Conditional independence

- We need to be able to specify, for a given graph, if $X \perp\!\!\!\perp Z|Y$, for any disjoint subsets of nodes X, Y, Z.
- In directed graphs, we did this using the Bayes Ball algorithm
- In undirected graphs, independence can be established simply by graph separation: if every path from a node in X to a node in Z goes through a node in Y, we conclude that $X \perp\!\!\!\perp Z|Y$
- Hence, independence can be established by removing the nodes in the conditioning set then doing reachability analysis on the remaining graph.
- What is the Markov blanket of a node in an undirected model?

How expressive are undirected models?

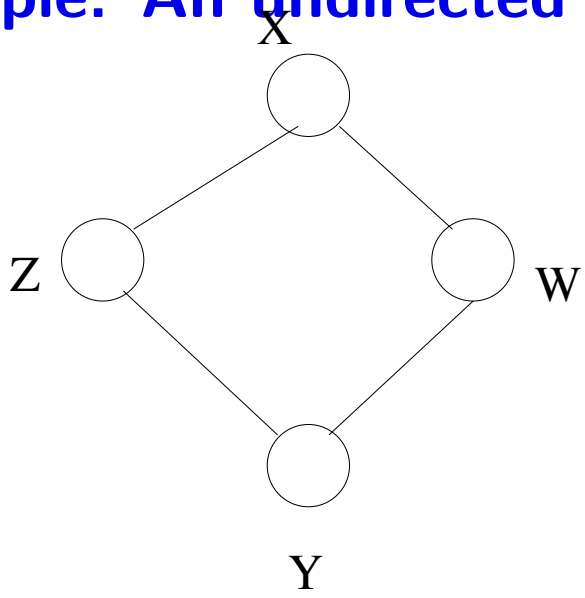
- Are undirected models more expressive than directed models?

  I.e. for any directed model, can we find an undirected model that satisfies exactly the same conditional independence relations?

- Are undirected models less expressive?

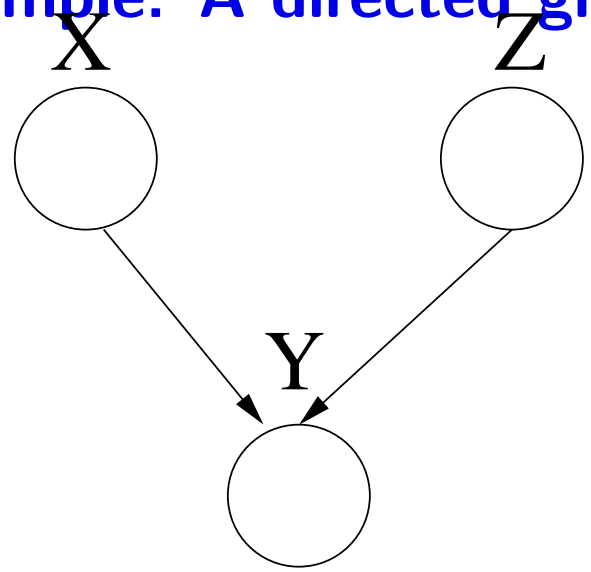
  I.e. for any undirected model, can we find a directed model that satisfies exactly the same conditional independencies?

Example: An undirected graph



Can we find a directed graph that satisfies the same independence relations?

Example: A directed graph



Can we find an undirected graph that satisfies the same independence relations?

Expressiveness of undirected models

- Undirected models are neither more nor less expressive than directed models; they are simply different
- The semantics of an undirected model naturally capture *correlation* of r.v.s, not causation
- If you ever want, in an application, to write a Bayes net with cycles, it is a sign that the right model is undirected.

Local parameterization

- In directed models, we had local probability models (CPDs) attached to every node, giving the conditional probability of the corresponding random variable given its parents
- The joint probability distribution expressed by a directed model factorizes over the graph
- This means that the joint can be written as a product of "local" factors, which depend on subsets of the variables.
- We want a similar property for directed models.
- But what should the local factors be?

Local parameterizations

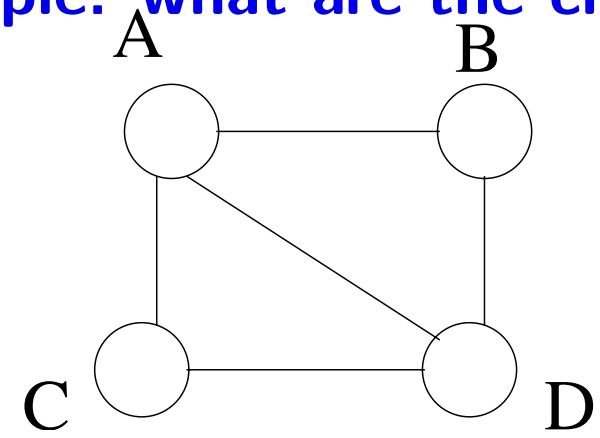
- Consider a pair of nodes X and Y that are not directly connected through an arc
- According to the conditional independence interpretation, X and Y are independent given all the other nodes in the graph

$$X \perp\!\!\!\perp Y | \{X_1, \ldots X_n\} - X - Y$$

- Hence, there must be a factorization in which they do not appear in the same factor
- This suggests that we should define factors on **cliques**

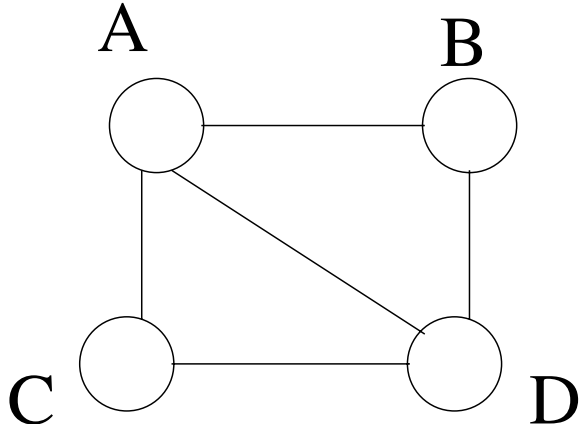
  Recall that a clique is a fully connected subset of nodes (i.e., there is an arc between every pair of nodes)

Example: what are the cliques?



Defining parameters on cliques

The main idea is that if variables do not have an arc between them, they are conditionally independent given the rest of the graph, and hence should not be in the same local model.



Important result (for strictly positive distributions)

- Consider the family of probability distributions that respect all the conditional independencies implied by an undirected graph G. These are the distributions that satisfy the *global Markov properties* of the graph
- Consider the family of probability distributions defined by ranging over all allowed maximal clique potential functions. These are the distributions that *factorize* on the graph G.
- The **Hammersley-Clifford theorem** shows that these two families are identical.

Clique potentials

- We will represent the joint distribution as a **product of clique potentials**:

$$p(X_1 = x_1, \ldots X_n = x_n) = \frac{1}{Z} \prod_{\text{cliques } C} \psi_C(\mathbf{x}_C)$$

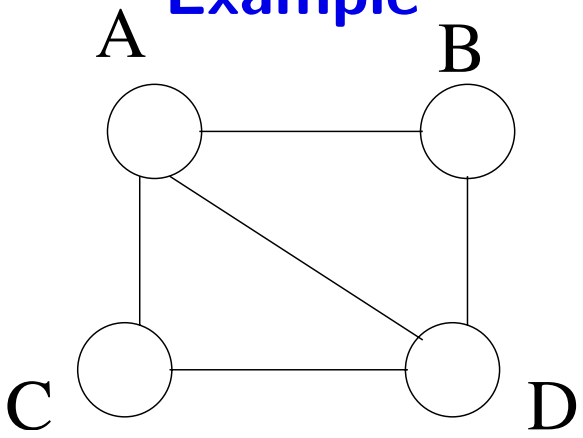
  where $\mathbf{x}_C$ are the values for the variables that participate in clique C and Z is a normalization constant, to make probabilities sum to 1:

$$Z = \sum_{\mathbf{x}} \prod_{\text{cliques } C} \psi_C(\mathbf{x}_C)$$

- Without loss of generality, we can consider only *maximal cliques*

  These are the cliques that cannot be extended with other nodes without losing the fully connected property

Example



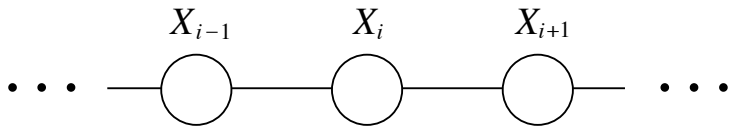
Normalizing constant

- The normalizing constant Z can be ugly to compute, since we have to sum over all possible assignments of values to variables
- Depending on the shape of the graph, the summation could be done efficiently
- However, if we are interested in conditional probabilities, we do not even need to compute it! (why?)

Interpretation of clique potentials

- Potentials are *NOT* probabilities (conditional or marginal)
- But they do have a natural interpretation as "agreement" or "energy"
- Example: spin glass model

1D spin glass



(a)

x_{i-1} \ x_i	-1	1
-1	1.5	0.2
1	0.2	1.5

x_i \ x_{i+1}	-1	1
-1	1.5	0.2
1	0.2	1.5

(b)

More on spin glasses

- In general, a spin glass is a collection of magnetic moment (spins) whose low temperature state is disordered.
- They have been studied a lot in statistical physics and they can model many practical materials
- These models have two important features:
  - There is *competition* among the interactions between moments, so there is no configuration of spins that is favored by all interactions; this is called **frustration**
  - Interactions are at least partially random
- There are many states whose energy is locally optimal (low)
- Finding such a state can be done by probabilistic inference.

Boltzmann (Gibbs) distribution

- The fact that potentials must be non-negative is annoying
- We can escape from that by using the exponential function, which is non-negative:

$$\psi_C(\mathbf{x}_C) = e^{-H_C(\mathbf{x}_C)}$$

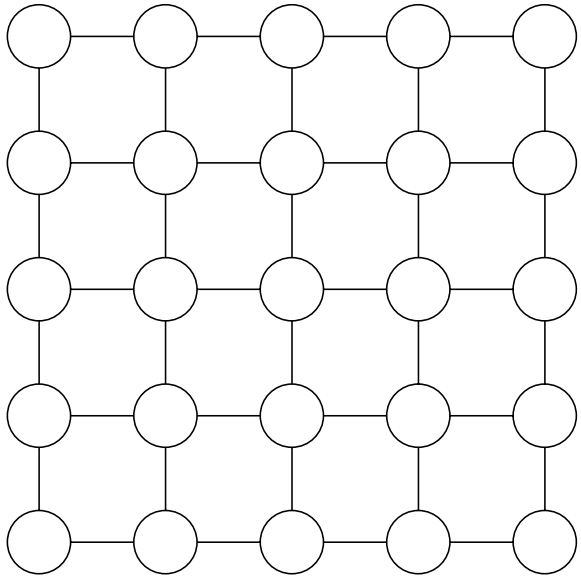
- Now we have to define $H_C(\mathbf{x}_C)$, which can be anything!
- Moreover, the joint also has a nice form:

$$p(\mathbf{x}) = \frac{1}{Z}\prod_C e^{-H_C(\mathbf{x}_C)} = \frac{1}{Z}e^{-\sum_C H_C(\mathbf{x}_C)} = \frac{1}{Z}e^{-H(\mathbf{x})}$$

  where $H(\mathbf{x}) = \sum_C H_C(\mathbf{x}_C)$ is the "free energy"
- Hence, p is represented using a *Boltzmann distribution*

Special case: Ising Model

- All r.v.s are binary and nodes are arranged in a regular fashion and connected only to geometric neighbors.
- E.g., Spin glass in 2D:



- Energy has the form:

$$H(\mathbf{x}) = \sum_{i,j} \beta_{ij} x_i x_j + \sum_i \alpha_i x_i$$

Applications of the Ising model

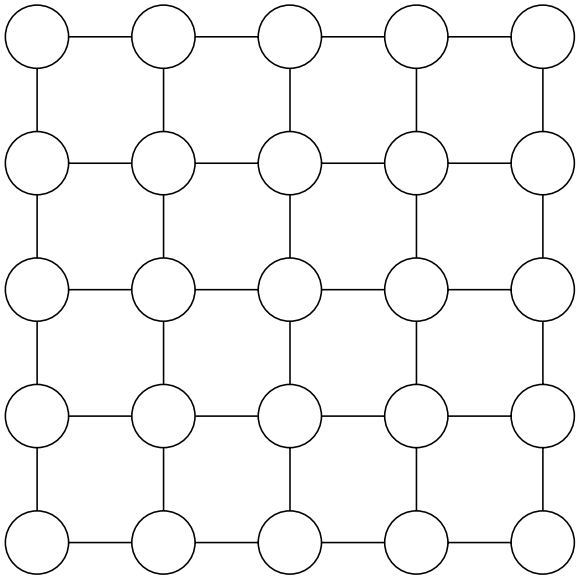
- Very popular for explaining the effect of "society" or "environment" on a "component" or "individual"
  - Flocking behavior
  - Behavior of neural networks
  - Sociology studies
- In all these cases, the effort is both to find, from the data, what the model should be, as well as to use inference in order to determine what will be the next state of minimum energy to which the model "settles"

Choosing the parameters of a Markov network

- These nets often have a regular structures, and parameters may be similar (or identical) in all cliques of a given type
- As a result, optimization or learning are often the preferred ways of coming up with parameters

A real example: Texture synthesis

- You are given a small patch of texture and want to produce a "similar" larger patch
- We can define a Markov random field over pixels, e.g:

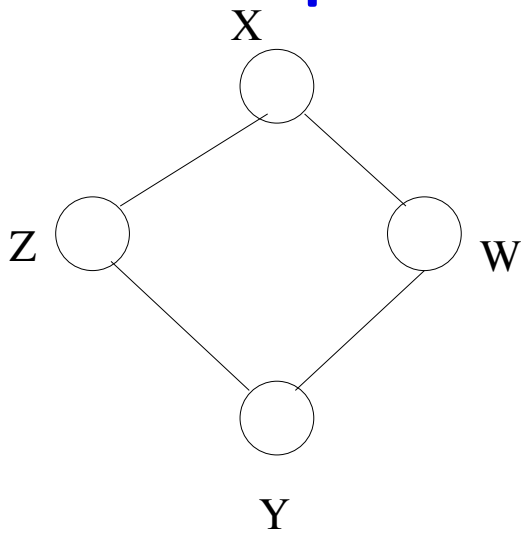


- The "potentials" favor certain configurations of pixels over others
- We get the texture by doing inference (and sometimes learning) for this model

More applications

- MRFs are used extensively in computer vision, e.g for labeling tasks
- Labeling can be low-level, like labeling edges or other pixel configurations, or high-level, like labeling objects in an image
- Images often obey constraints (e.g. smoothness of surfaces, texture) which can be captured easily as a MRF structure
- Labeling then becomes a search for a pattern of minimum energy, which is often solved by optimization
- Learning can help establish the parameters of the model.

Example



$$
\begin{aligned}
P(X,Y,W,Z) &\propto \psi(W,X)\psi(W,Y)\psi(Y,Z)\psi(Z,X) \\
&\propto e^{-(E(W,X)+E(W,Y)+E(Y,Z)+E(Z,X))}
\end{aligned}
$$

- In general, potentials are defined over *cliques* of nodes
- A clique is a set of nodes where all pairs are interconnected
- In such a case, all nodes have to agree on the value

Belief propagation updates

- Node X_j computes the following message for node X_i

$$m_{ji}(x_i) = \sum_{x_j} \left(\psi^E(x_j)\psi(x_i, x_j) \prod_{k \in \text{neighbors}(x_j) - \{x_i\}} m_{kj}(x_j) \right)$$

  Note that it aggregates the evidence of X_j itself, the evidence received from all neighbors except X_i, and "modulates" it by the potential between X_j and X_i.

- The desired query probability is computed as:

$$p(y|\hat{x}_E) \propto \psi^E(y) \prod_{k \in \text{neighbors}(Y)} m_{ky}(y)$$

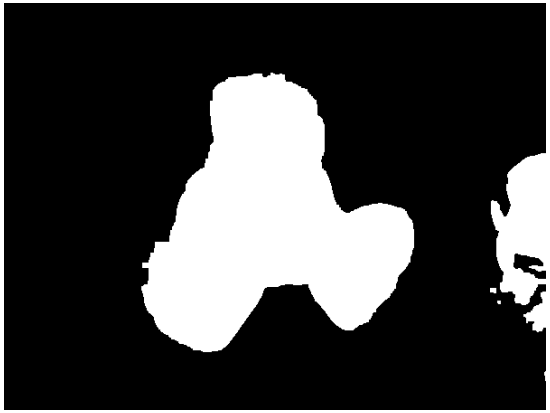
Practical considerations

- Messages do not have to be normalized, but often they are (to avoid numerical problems)
- Order of the updates is important
- In 2D lattices, people often sweep systematically (e.g. up-down-left-right)
- If there is oscillations, one can "dampen" it using momentum
- The cost per iteration is squared in the number of possible variable values
- Most likely states can be calculated using a max instead of a sum

Example: Results for image segmentation

Binary Segmentation

Szeliski et al. , 2008



Belief Propagation

Labels - {foreground, background}

Unary Potentials: -log(likelihood) using learnt fg/bg models

Pairwise Potentials: 0, if same labels

$1 - \lambda \exp(|D_a - D_b|)$, if different labels

Learning in undirected models

- Recall from the example:

$$\begin{aligned} P(X,Y,W,Z) &\propto \psi(W,X)\psi(W,Y)\psi(Y,Z)\psi(Z,X) \\ &\propto e^{-(E(W,X)+E(W,Y)+E(Y,Z)+E(Z,X))} \end{aligned}$$

- The log-likelihood has some energy terms, which can be modelled parametrically (eg. linear combination of inputs, correlations...)
- Good part: these terms have no restrictions on the parameters, so gradient-based methods would work
- Bad news: There is a normalization (lost in the proportionality above) which means that learning *cannot* be broken down locally over just nodes or cliques

Maximum likelihood approach

- Consider the discrete case, and let $N(x_C)$ be the number of times the values x_C occur in the data, and let N be the number of instances.
- The log likelihood is:

$$\begin{aligned}\log L(\psi|D) &= \sum_{i=1}^{N} \log p(x_1^i, \ldots x_n^i) = \sum_{i=1}^{N} \log \frac{1}{Z} \prod_C \psi_c(x_C) \\ &= \left(\sum_C \sum_{x_C} N(x_C) \log \psi_C(x_C) \right) - N \log Z\end{aligned}$$

- So the counts for each clique, $N(x_C)$, are sufficient statistics
- The $\log Z$ term will cause parameters estimation to be *coupled* in general.

Gradient-based approach

- Taking the derivative of the log likelihood wrt the parameters:

$$\frac{\partial \log L}{\partial \psi_C(x_C)} = \left(\sum_C \sum_{x_C} \frac{N(x_C)}{\psi_C(x_C)} \right) - N \frac{1}{Z} \frac{\partial}{\partial \psi_C(x_C)} \sum_{x_1, \ldots x_n} \prod_{C'} \psi_{C'}(x_{C'})$$

  This is now a system of *nonlinear equations*

- Working on the last term we get:

$$\frac{\partial \log Z}{\partial \psi_C(x_C)} = \frac{p(x_C)}{\psi_C(x_C)}$$

  where $p(x_C)$ is the marginal probability of x_C

- By setting the derivatives to 0 we get: $p_{ML}(x_C) = \frac{N(x_C)}{N}$

  This is nice, but how do we get parameters from here?

Revisiting the log-likelihood

- At the maximum, we have:

$$\frac{\hat{p}(x_C)}{\psi_C(x_C)} = \frac{p(x_C)}{\psi_C(x_C)}$$

- Suppose we started with a guess for the parameters, $\psi_C^0, \forall C$. This would allow us to compute *marginal under the current guess*, $p^0(x_C)$
- Next we *re-compute the parameters* so that we get something more like the equality above:

$$\psi_C^1(x_C) = \psi_C^0(x_C)\frac{\hat{p}(x_C)}{p^0(x_C)}$$

- We will continue this until the change in parameters is insignificant

Iterative proportional fitting

- Start with a guess ψ^0
- Repeat:

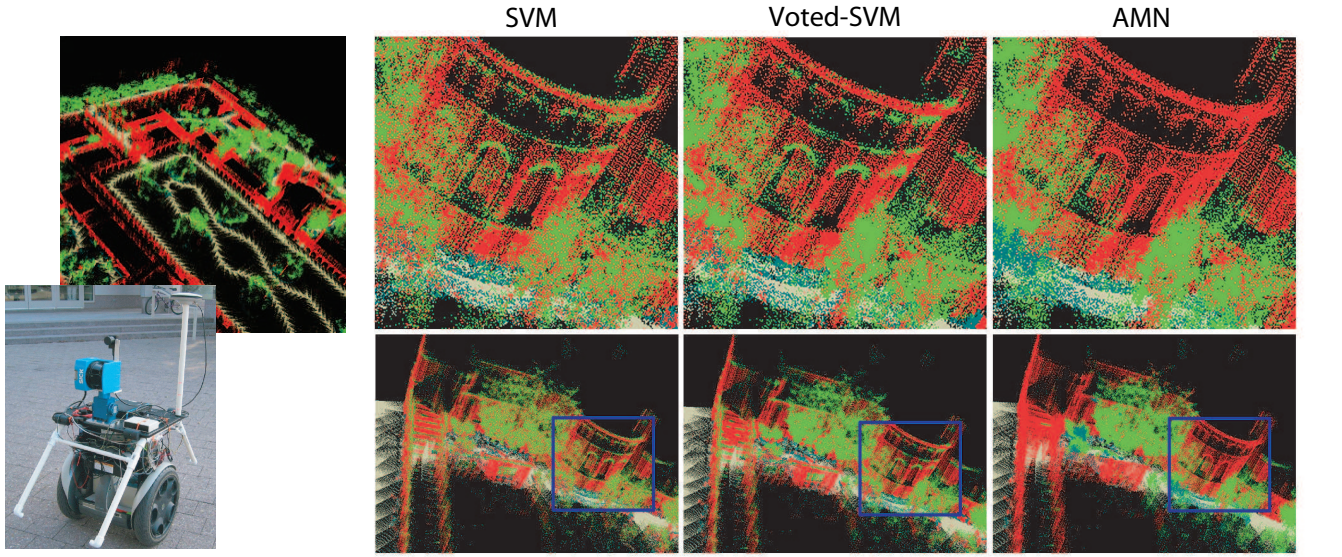
$$\psi_C^{t+1}(x_C) = \psi_C^t(x_C)\frac{\hat{p}(x_C)}{p^t(x_C)}, \forall C$$

- This is a *fixed point* algorithm and will converge in the limit
- Convergence happens in a finite number of iterations for *decomposable models*, i.e. trees and similar structures.

For larger problems

- The algorithm works the same way except instead of working with ψ, we work with the energy terms, E
- Each E is parameterized using features (e.g. as a linear function of features)
- Gradients are taken wrt the parameters of the E functions
- The algorithm has very similar convergence properties, but the features allow a bias-variance trade-off

Example: Point cloud data



a) Robot and campus map

b) Segmentation Results [1]

- 3 approaches: SVM, Voted SVM (takes majority label over the pixel and its neighbors), MRF
- Arches are much better maintained in the MRF, because it uses the whole "context" of the building

[1] Anguelov et al, CVPR 2004