COMPSCI 688: Probabilistic Graphical Models

Lecture 8: Undirected Graphical Models: Inference

Dan Sheldon

Manning College of Information and Computer Sciences
University of Massachusetts Amherst

Partially based on materials by Benjamin M. Marlin (marlin@cs.umass.edu) and Justin Domke (domke@cs.umass.edu)

Review

Markov Random Fields

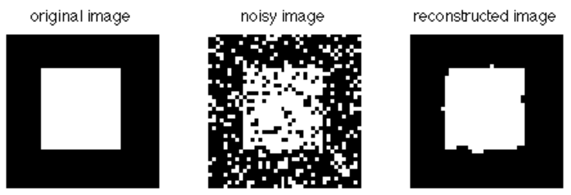
- Markov random field
$$p(\mathbf{x}) = \frac{1}{Z}\prod_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c)$$
- *Dependence graph* $\mathcal{G}$: where nodes i and j are connected by an edge if they appear together in some factor
- Ising Model: grid-structured graph, unary/pairwise potentials express local preferences for values of x_i or (x_i, x_j) pairs

$$p(\mathbf{x}) = \frac{1}{Z}\prod_{i} \beta_i(x_i) \prod_{(i,j)\in E} \beta_{ij}(x_i, x_j)$$

Examples

Example: Statistical Image Models

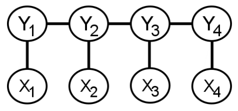
The Ising model with pairwise potentials encourages smoothness and can be used as a model for images for denoising:

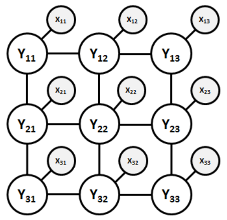


Example: Image Denoising

Conditional Random Fields

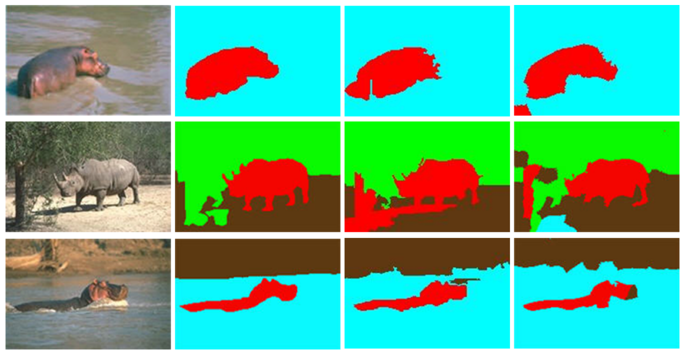
The image denoising model is one example of a **conditional random fields** (CRFs), a very important model class in machine learning. A CRF is essentially a Markov network where one set of nodes is always conditioned on.



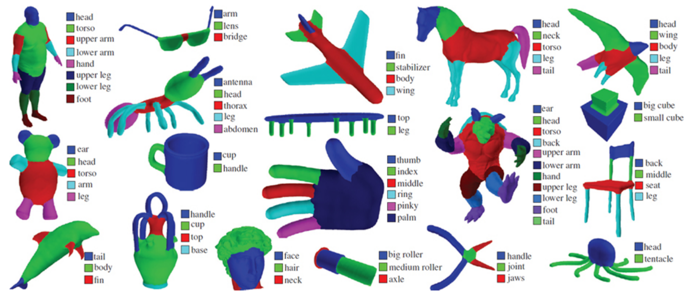


The $\mathbf{y}$ nodes are *labels*, and the $\mathbf{x}$ nodes are *features*.

Example: Image Segmentation



Example: 3D Mesh Segmentation



Example: Bayes Nets as MRFs

Example: Bayes Nets as MRFs

Some structure is lost in this transformation. When we replace $p(a|b,c)$ by $\phi(a,b,c)$, we "forget" that a Bayes net is **locally normalized**

$$\sum_a \phi(a,b,c) = 1 \quad \forall b, c.$$

This is a special property of Bayes nets and is central to V-structures, explaining away, and D-separation. It occurs "internally" to the factor $\phi(a,b,c)$ and is not represented in the MRF graph structure.

Similarly, when we replace $\prod_i p(x_i|\mathbf{x}_{\mathsf{pa}(i)})$ by $\frac{1}{Z}\prod_{c\in\mathcal{C}}\phi_c(\mathbf{x}_c)$, we "forget" that a Bayes net is **globally normalized**:

$$\sum_x \prod_{c\in\mathcal{C}} \phi_c(\mathbf{x}_c) = 1 \implies Z = 1.$$

This is another special property of Bayes nets that makes learning easy.

Inference: Conditioning

Inference in Markov Networks

- Given a Markov network, the main task is *probabilistic inference*, which means answering probability queries of the form

$$p(\mathbf{x}_Q|\mathbf{x}_E) = \sum_{\mathbf{x}_U} p(\mathbf{x}_Q, \mathbf{x}_U|\mathbf{x}_E)$$

  - condition on *evidence variables* $\mathbf{x}_E$
  - marginalize *unobserved variables* $\mathbf{x}_U$
  - compute the joint distribution over *query variables* $\mathbf{x}_Q$
- . . . often by transforming Markov network into one with fewer or simpler factors
  - Conditioning is easy
  - Marginalization is hard!

Conditioning: Single Factor

Suppose we have a single-factor MRF $p(x_1, x_2) = \frac{1}{Z}\phi(x_1, x_2)$ for two binary variables. We are given a fixed value for x_2, and want an MRF for $p(x_1|x_2)$, i.e.:

$$p(x_1|x_2) = \frac{1}{Z'}\phi'(x_1)$$

Observe

$$p(x_1|x_2) = \frac{p(x_1, x_2)}{p(x_2)} = \underbrace{\frac{1}{p(x_2)} \cdot \frac{1}{Z}}_{1/Z'}\phi(x_1, x_2)$$

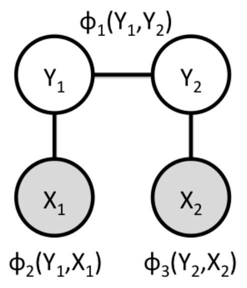
For fixed x_2, the conditional $p(x_1|x_2)$ is *proportional* to the joint $p(x_1, x_2)$. We can use the same factor, but hard-code x_2 so that only x_1 is a free variable:

$$\phi'(x_1) = \phi(x_1, x_2), \quad Z' = p(x_2)Z$$

Conditioning: General Case

For a general MRF, we can apply the same reasoning to *reduce* every factor by hard-coding the evidence variables

Factor Reduction: Example



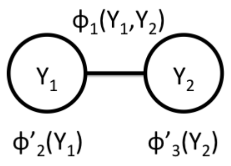
$\phi_1(Y_1,Y_2)$	$Y_2=0$	$Y_2=1$
$Y_1=0$	1	2
$Y_1=1$	7	2

$\phi_2(Y_1,X_1)$	$X_1=0$	$X_1=1$
$Y_1=0$	3	9
$Y_1=1$	4	1

$\phi_3(Y_2,X_2)$	$X_2=0$	$X_2=1$
$Y_2=0$	6	2
$Y_2=1$	2	7

Query: $P(Y_1,Y_2 | X_1=0, X_2=1)$

Factor Reduction: Step 1



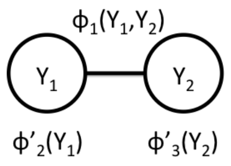
$\phi_1(Y_1,Y_2)$	$Y_2=0$	$Y_2=1$
$Y_1=0$	1	2
$Y_1=1$	7	2

$\phi_2(Y_1,X_1)$	$X_1=0$	$X_1=1$
$Y_1=0$	3	9
$Y_1=1$	4	1

$\phi_3(Y_2,X_2)$	$X_2=0$	$X_2=1$
$Y_2=0$	6	2
$Y_2=1$	2	7

Query: $P(Y_1,Y_2 | X_1=0, X_2=1)$

Factor Reduction: Step 2



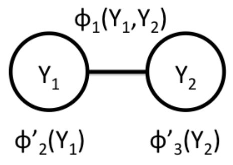
$\phi_1(Y_1,Y_2)$	$Y_2=0$	$Y_2=1$
$Y_1=0$	1	2
$Y_1=1$	7	2

$\phi_2(Y_1,X_1)$	$X_1=0$	$X_1=1$
$Y_1=0$	3	9
$Y_1=1$	4	1

$\phi_3(Y_2,X_2)$	$X_2=0$	$X_2=1$
$Y_2=0$	6	2
$Y_2=1$	2	7

Query: $P(Y_1,Y_2 | X_1=0, X_2=1)$

Factor Reduction: Step 2



$\phi_1(Y_1,Y_2)$	$Y_2=0$	$Y_2=1$
$Y_1=0$	1	2
$Y_1=1$	7	2

	$\phi'_2(Y_1)$
$Y_1=0$	3
$Y_1=1$	4

	$\phi'_3(Y_2)$
$Y_2=0$	2
$Y_2=1$	7

Query: $P(Y_1,Y_2 \mid X_1=0, X_2=1) \propto \phi_1(Y_1,Y_2)\, \phi'_2(Y_1)\, \phi'_3(Y_2)$

Factor Reduction: General Algorithm

Suppose $p(\mathbf{x}) = \frac{1}{Z}\prod_{c\in\mathcal{C}} \phi_c(\mathbf{x}_c)$ and we observe $X_i = x_i$ for a single node i

We obtain a new MRF for $p(\mathbf{x}_{-i}|x_i)$ by the following procedure:

For each factor ϕ_c such that $i \in c$

- Replace $\phi_c(\mathbf{x}_c)$ by $\phi'_{c\setminus i}(\mathbf{x}_{c\setminus i}) := \phi_c(\mathbf{x}_{c\setminus i}, x_i)$
- The $\mathbf{x}_{c\setminus i}$ variables remain "free", and x_i is hard-coded

To condition on many variables, we can repeat this procedure. Since order doesn't matter, we can hard-code all evidence variables in each factor at the same time.

Inference: Marginalization

Marginalization

Marginalization is the process of summing over some of the variables to get the marginal distribution of the remaining variables, or the partition function.

For example, the partition function is

$$Z = \sum_{x_1}\sum_{x_2}\cdots\sum_{x_n}\prod_{c\in\mathcal{C}}\phi_c(x_c)$$

Naively, this takes exponential time, but we can sometimes use the factorization structure to speed it up.

Example: Variable Elimination on a Chain

Consider the following MRF on a four-node "chain" graph:

$$p(x_1,x_2,x_3,x_4) = \frac{1}{Z}\phi_1(x_1)\phi_2(x_2)\phi_3(x_3)\phi_4(x_4)\phi_{12}(x_1,x_2)\phi_{23}(x_2,x_3)\phi_{34}(x_3,x_4)$$

x_i	$\phi_i(x_i)$
0	1
1	2

x_i	x_j	$\phi_{ij}(x_i,x_j)$
0	0	2
0	1	1
1	0	1
1	1	2

Let's compute Z:

$$\begin{aligned}
Z &= \sum_{x_1}\sum_{x_2}\sum_{x_3}\sum_{x_4}\phi_1(x_1)\phi_2(x_2)\phi_3(x_3)\phi_4(x_4)\phi_{1,2}(x_1,x_2)\phi_{2,3}(x_2,x_3)\phi_{3,4}(x_3,x_4)\\
&= \sum_{x_1}\phi_1(x_1)\sum_{x_2}\phi_2(x_2)\phi_{1,2}(x_1,x_2)\sum_{x_3}\phi_3(x_3)\phi_{2,3}(x_2,x_3)\underbrace{\sum_{x_4}\phi_4(x_4)\phi_{3,4}(x_3,x_4)}_{m_{4\to3}(x_3)}\\
&= \sum_{x_1}\phi_1(x_1)\sum_{x_2}\phi_2(x_2)\phi_{1,2}(x_1,x_2)\underbrace{\sum_{x_3}\phi_3(x_3)\phi_{2,3}(x_2,x_3)m_{4\to3}(x_3)}_{m_{3\to2}(x_2)}\\
&= \sum_{x_1}\phi_1(x_1)\underbrace{\sum_{x_2}\phi_2(x_2)\phi_{1,2}(x_1,x_2)m_{3\to2}(x_2)}_{m_{2\to1}(x_1)}\\
&= \sum_{x_1}\phi_1(x_1)m_{2\to1}(x_1).
\end{aligned}$$

Pictorially, this is how we changed the MRF

What if we want to compute the *unnormalized* marginal $\hat{p}(x_1)$?

$$\begin{aligned}
\hat{p}(x_1) &= \sum_{x_2}\sum_{x_3}\sum_{x_4}\phi_1(x_1)\phi_2(x_2)\phi_3(x_3)\phi_4(x_4)\phi_{1,2}(x_1,x_2)\phi_{2,3}(x_2,x_3)\phi_{3,4}(x_3,x_4)\\
&= \phi_1(x_1)\sum_{x_2}\phi_2(x_2)\phi_{1,2}(x_1,x_2)\sum_{x_3}\phi_3(x_3)\phi_{2,3}(x_2,x_3)\underbrace{\sum_{x_4}\phi_4(x_4)\phi_{3,4}(x_3,x_4)}_{m_{4\to3}(x_3)}\\
&= \cdots\\
&= \phi_1(x_1)m_{2\to1}(x_1).
\end{aligned}$$

The computation is the same, but we don't eliminate x_1.

What if we want to compute the *actual* marginal $p(x_1)$?

Take $\hat{p}(x_1)$ and normalize it

$$Z = \sum_{x_1} \hat{p}(x_1), \quad p(x_1) = \frac{1}{Z}\hat{p}(x_1)$$

Lesson: always normalize at the end

What if we eliminate x_3 first?

$$\begin{aligned}
Z &= \sum_{x_1}\sum_{x_2}\sum_{x_3}\sum_{x_4} \phi_1(x_1)\phi_2(x_2)\phi_3(x_3)\phi_4(x_4)\phi_{1,2}(x_1,x_2)\phi_{2,3}(x_2,x_3)\phi_{3,4}(x_3,x_4) \\
&= \sum_{x_1}\sum_{x_2}\sum_{x_4} \phi_1(x_1)\phi_2(x_2)\phi_4(x_4)\phi_{1,2}(x_1,x_2) \underbrace{\sum_{x_3} \phi_3(x_3)\phi_{2,3}(x_2,x_3)\phi_{3,4}(x_3,x_4)}_{\tau_{24}(x_2,x_4)} \\
&= \sum_{x_1}\sum_{x_2}\sum_{x_4} \phi_1(x_1)\phi_2(x_2)\phi_4(x_4)\tau_{24}(x_2,x_4) \\
&= \cdots
\end{aligned}$$

This is less efficient because we create a larger intermediate factor (with three variables instead of two, before summing out x_3), but it is still correct.

What if our graph is a star graph?

If we eliminate leaves first, it is very efficient. If we eliminate the hub node first, it creates a factor with size exponential in the number of nodes.

The Variable Elimination Algorithm

Variable elimination is an algorithm to compute any marginal distribution in any MRF

In words: pick a variable x_i to eliminate, multiply together all factors containing x_i to get an intermediate factor, then sum out x_i

- Let $F = \{\phi_c : c \in \mathcal{C}\}$ be the set of factors
- For each variable i in **some elimination order** (may not include all variables)
    - Let $A = \{\phi_c \in F : i \in c\}$ be the set of factors whose scope contains i
    - Let $\phi_a(\mathbf{x}_a) = \prod_{\phi_c \in A} \phi_c(\mathbf{x}_c)$ be the product of factors in A, with scope a equal to the union of the scopes of the individual factors
    - Let $\psi_i(\mathbf{x}_{a\setminus i}) = \sum_{x_i} \phi_a(\mathbf{x}_{a\setminus i}, x_i)$ be the result of summing out x_i
    - Let $F = F \setminus A \cup \{\psi_i\}$

The final set of factors forms an MRF for the marginal distribution of the variables that were not eliminated.

Variable Elimination Discussion

- The efficiency of variable elimination depends on the maximum size of the intermediate factors created, which depends on the elimination ordering
  - Inference in MRFs is NP-hard, so we can't always find a good elimination ordering.
  - Finding the best elimination ordering for a given MRF is also NP-hard!
- It's always efficient to eliminate leaves if present (intermediate factors are no larger than original ones)
  - $\Longrightarrow$ for trees, we can find an efficient elimination ordering
  - In fact, because the elimination ordering is predictable in trees, we can realize extra efficiencies when answering multiple queries through a dynamic programming approach known as **message passing**