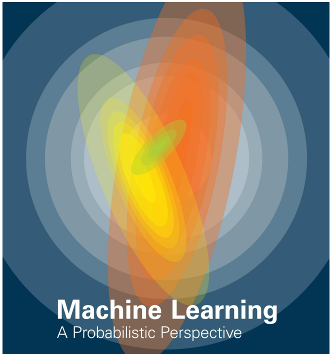
Tutorial on Probabilistic Graphical Models
ML Summer School
UC Santa Cruz

Kevin P. Murphy
kpmurphy@google.com
Research Scientist, Google, Mtn View, California
Formerly Assoc. Prof., UBC, Canada

July 2012

Overview of tutorial

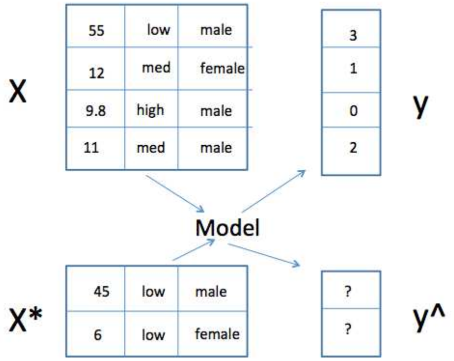
- Why probabilistic models?
- Directed graphical models
- Undirected graphical models
- Exact inference
- Variational inference
- Monte Carlo inference
- Based on chapters 10, 11, 17, 19-24 of my forthcoming textbook



Outline

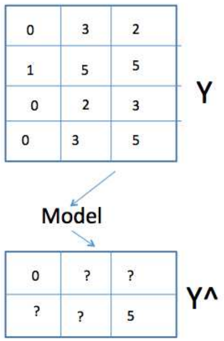
Why probabilistic models?

Supervised learning



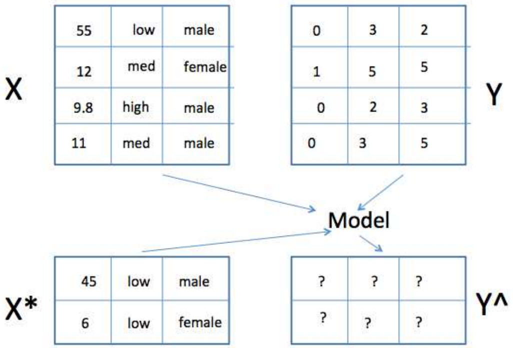
- Often posed as function esimation, $y = f(\mathbf{x})$.
- If output is ambiguous, better to compute $p(y|\mathbf{x})$.
- Supervised learning is univariate conditional density estimation.

Unsupervised learning



- Goal 1: predict y_i given y_{-i}
- Goal 2: outlier detection, density estimation
- Goal 3: knowledge discovery
- Need **joint probability models** $p(\mathbf{y})$.

Multi-output supervised learning



55	low	male
12	med	female
9.8	high	male
11	med	male

0	3	2
1	5	5
0	2	3
0	3	5

45	low	male
6	low	female

?	?	?
?	?	?

- Goal 1: predict y_i given $\mathbf{x}$ (and maybe y_{-i})
- Goal 2: outlier detection, conditional density estimation
- Goal 3: knowledge discovery
- Need **conditional joint probability models** $p(\mathbf{y}|\mathbf{x})$.

Defining joint probability distributions

- By the **chain rule** of probability,

$$\begin{aligned} p(x_{1:V}) &= p(x_1)p(x_2|x_1)p(x_3|x_2,x_1) \\ &\quad p(x_4|x_1,x_2,x_3)\dots p(x_V|x_{1:V-1}) \end{aligned}$$

- $p(x_t|\mathbf{x}_{1:t-1})$ needs $O(K^t)$ parameters if K states per variable.
- We will make **conditional independence** assumptions to simplify things.

Conditional independence

- Definition

$$X \perp Y|Z \iff p(X, Y|Z) = p(X|Z)p(Y|Z)$$

- Example: first order Markov chain: “the future is independent of the past given the present”:

$$y_{t+1} \perp \mathbf{y}_{1:t-1}|y_t$$

- Joint distribution

$$p(y_{1:T}) = p(y_1) \prod_{t=2}^{T} p(y_t|y_{1:t-1}) \overset{*}{=} p(y_1) \prod_{t=2}^{T} p(y_t|y_{t-1})$$

This is characterized by an initial distribution over states, $p(y_1 = i)$, plus a **state transition matrix** $p(y_t = j|y_{t-1} = i)$.

Graphical models

- Nodes represent random variables
- Edges represent conditional independence.
- Details of CI depends on whether the graph is directed ("Bayes net") or undirected ("Markov random field")
- Structure of graph brings statistical and computational efficiencies (less data, less time).

Tutorial on Probabilistic Graphical Models
2: Directed graphical models

Kevin P. Murphy
kpmurphy@google.com

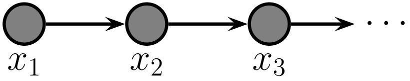
July 2012

Outline

Example: Markov chains

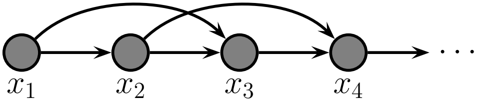
- First order Markov chain

$$p(x_{1:T}) = p(x_1) \prod_{t=2}^{T} p(x_t | x_{t-1})$$



- Second order Markov chain

$$p(x_{1:T}) = p(x_1, x_2) \prod_{t=3}^{T} p(x_t | x_{t-1}, x_{t-2})$$



General DAGs

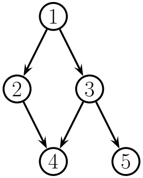
- If the graph is a DAG (directed acyclic graph), we can order nodes such that parents come before children. This is called a topological ordering.
- **Ordered Markov property** is the assumption that a node only depends on its immediate parents, not on all predecessors in the ordering, i.e.,

$$x_s \perp \mathbf{x}_{\mathrm{pred}(s) \backslash \mathrm{pa}(s)} | \mathbf{x}_{\mathrm{pa}(s)}$$

  where $\mathrm{pa}(s)$ are the parents of node s, and $\mathrm{pred}(s)$ are the predecessors of node s in the ordering. This is a natural generalization of the first-order Markov property from chains to general DAGs.
- Each node has a CPD $p(x_t | \mathbf{x}_{\mathrm{pa}(t)})$

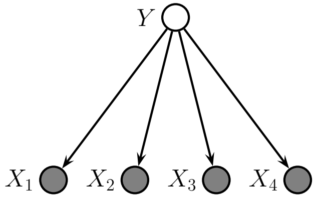
$$p(\mathbf{x}_{1:V} | G) = \prod_{t=1}^{V} p(x_t | \mathbf{x}_{\mathrm{pa}(t)})$$

Example



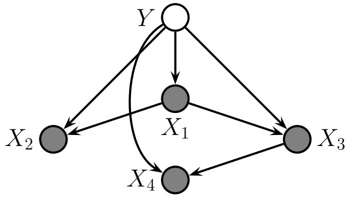
$$\begin{aligned} p(\mathbf{x}_{1:5}) &= p(x_1)p(x_2|x_1)p(x_3|x_1,\cancel{x_2})p(x_4|\cancel{x_1},x_2,x_3)p(x_5|\cancel{x_1},\cancel{x_2},x_3,\cancel{x_4}) \\ &= p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2,x_3)p(x_5|x_3) \end{aligned}$$

Naive Bayes classifiers



$$p(y, \mathbf{x}) = p(y) \prod_{j=1}^{D} p(x_j | y)$$

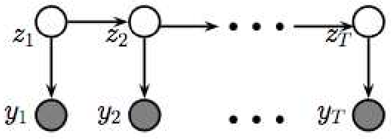
Tree augmented Naive Bayes classifiers



$$p(y, \mathbf{x}) = p(y) \prod_{j=1}^{D} p(x_j | x_{\mathrm{pa}(j)}, y)$$

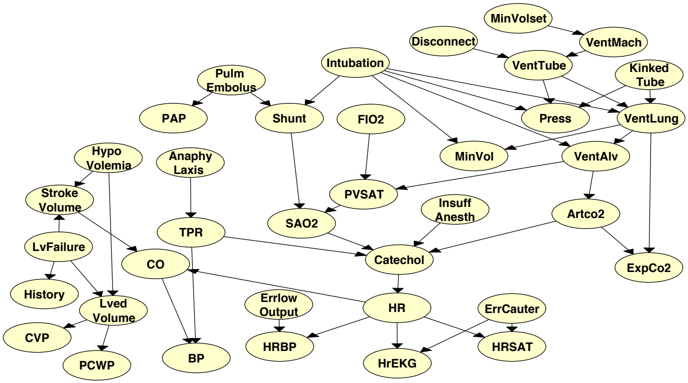
Hidden Markov Models (HMMs)

$$\begin{aligned} p(y_{1:T}, z_{1:T}) &= p(\mathbf{z}_{1:T}) p(\mathbf{y}_{1:T} | \mathbf{z}_{1:T}) \\ &= \left[p(z_1) \prod_{t=2}^{T} p(z_t | z_{t-1}) \right] \left[\prod_{t=1}^{T} p(y_t | z_y) \right] \end{aligned}$$



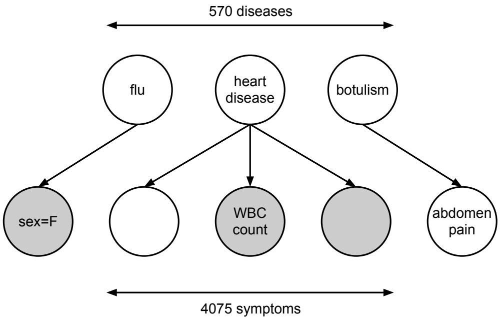
- Transition model $p(z_t = j | z_{t-1} = i) = A(i, j)$
- Discrete emission model $p(y_t = j | z_t = i) = B(i, j)$
- Gaussian emission model $p(\mathbf{y}_t | z_t = k) = \mathcal{N}(\mathbf{y}_t | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$

Alarm network



- This model has 37 variables and 504 parameters, created by hand using knowledge elicitation (probabilistic expert system)

Quick Medical Reference (QMR)



- Joint

$$p(\mathbf{v}, \mathbf{h}) = \prod_s p(h_s) \prod_t p(v_t | \mathbf{h}_{\mathrm{pa}(t)})$$

where h_s represent the **hidden nodes** (diseases), and v_t represent the **visible nodes** (symptoms).

Sigmoid Bayes nets

- The CPD for the root nodes are just Bernoulli distributions, representing the prior probability of that disease, $p(h_s = 1)$.
- Representing the CPDs for the leaves (symptoms) using tables (CPT) would require too many parameters, because the **fan-in** (number of parents) of many leaf nodes is very high.
- A natural alternative is to use logistic regression to model the CPD, $p(v_t = 1|\mathbf{h}_{\mathrm{pa}(t)}) = \mathrm{sigm}(\mathbf{w}_t^T \mathbf{h}_{\mathrm{pa}(t)})$.
- A DGM in which the CPDs are logistic regression distributions is known as a **sigmoid belief net**.

Noisy-OR CPDs

- The noisy-OR model assumes that if a parent is on, then the child will usually also be on (since it is an or-gate), but occasionally the "links" from parents to child may fail, independently at random. In this case, even if the parent is on, the child may be off.
- To model this more precisely, let θ_{st} be the probability that the $s \to t$ link fails, so $1 - \theta_{st} = p(v_t = 1 | h_s = 1, \mathbf{h}_{-s} = 0)$ is the probability that s can activate t on its own (its "causal power"). The only way for the child to be off is if all the links from all parents that are on fail independently at random. Thus
$$p(v_t = 0|\mathbf{h}) = \prod_{s \in \mathrm{pa}(t)} \theta_{st}^{\mathbb{I}(h_s=1)}$$
Obviously, $p(v_t = 1|\mathbf{h}) = 1 - p(v_t = 0|\mathbf{h})$.

Leak nodes

- If we observe that $v_t = 1$ but all its parents are off, then this contradicts the model. Such a data case would get probability zero under the model, which is problematic, because it is possible that someone exhibits a symptom but does not have any of the specified diseases. To handle this, we add a dummy **leak node** h_0, which is always on; this represents "all other causes". The modified CPD becomes $p(v_t = 0|\mathbf{h}) = \theta_{0t} \prod_{s \in \mathrm{pa}(t)} \theta_{st}^{h_s}$.

h_0	h_1	h_2	$P(v = 0\|h_1, h_2)$	$P(v = 1\|h_1, h_2)$
1	0	0	θ_0	$1 - \theta_0$
1	1	0	$\theta_0\theta_1$	$1 - \theta_0\theta_1$
1	0	1	$\theta_0\theta_2$	$1 - \theta_0\theta_2$
1	1	1	$\theta_0\theta_1\theta_2$	$1 - \theta_0\theta_1\theta_2$

Table: Noisy-OR CPD for 2 parents augmented with leak node. We have omitted the t subscript for brevity.

Outline

I-maps

- At the heart of any graphical model is a set of conditional indepence (CI) assumptions. We write $\mathbf{x}_A \perp_G \mathbf{x}_B | \mathbf{x}_C$ if A is independent of B given C in the graph G, using the semantics to be defined below. Let $I(G)$ be the set of all such CI statements encoded by the graph.
- We say that G is an **I-map** (independence map) for p, or that p is **Markov** wrt G, iff $I(G) \subseteq I(p)$, where $I(p)$ is the set of all CI statements that hold for distribution p. In other words, the graph is an I-map if it does not make any assertions of CI that are not true of the distribution.
- This allows us to use the graph as a safe proxy for p when reasoning about p's CI properties. This is helpful for designing algorithms that work for large classes of distributions, regardless of their specific numerical parameters $\boldsymbol{\theta}$.

Minimal I-maps

- Note that the fully connected graph is an I-map of all distributions, since it makes no CI assertions at all (since it is not missing any edges). We therefore say G is a **minimal I-map** of p if G is an I-map of p, and if there is no $G' \subseteq G$ which is an I-map of p.

d-separation

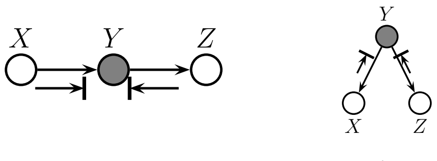
- We say an *undirected path* P is **d-separated** by a set of nodes E (containing the evidence) iff at least one of the following conditions hold:
  1. P contains a chain, $s \to m \to t$ or $s \leftarrow m \leftarrow t$, where $m \in E$
  2. P contains a tent or fork, $s \swarrow^{m} \searrow t$, where $m \in E$
  3. P contains a **collider** or **v-structure**, $s \searrow_{m} \swarrow t$, where m is not in E and nor is any descendant of m.
- Next, we say that a *set of nodes* A is d-separated from a different set of nodes B given a third observed set E iff each undirected path from every node $a \in A$ to every node $b \in B$ is d-separated by E.
- Finally, we define the CI properties of a DAG as follows:

$$\mathbf{x}_A \perp_G \mathbf{x}_B | \mathbf{x}_E \iff \text{A is d-separated from B given E}$$

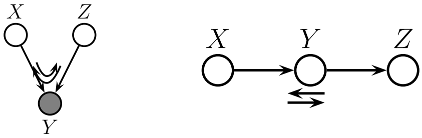
Bayes Ball algorithm

- The **Bayes ball algorithm** Shachter98 is a simple way to see if A is d-separated from B given E, based on the above definition. The idea is this. We "shade" all nodes in E, indicating that they are observed. We then place "balls" at each node in A, let them "bounce around" according to some rules, and then ask if any of the balls reach any of the nodes in B. The three main rules are shown on the next page.
- Notice that balls can travel opposite to edge directions. We see that a ball can pass through a chain, but not if it is shaded in the middle. Similarly, a ball can pass through a fork, but not if it is shaded in the middle. However, a ball cannot pass through a v-structure, unless it is shaded in the middle.

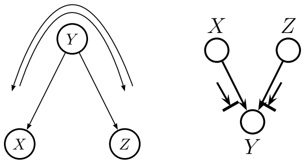
Bayes Ball algorithm



(a) (b)



(c) (d)



(e) (f)

Deriving the rules of Bayes Ball 1

First consider a chain structure $X \to Y \to Z$, which encodes

$$p(x,y,z) = p(x)p(y|x)p(z|y)$$

When we condition on y, are x and z independent? We have

$$p(x,z|y) \quad = \quad \frac{p(x)p(y|x)p(z|y)}{p(y)} = \frac{p(x,y)p(z|y)}{p(y)} = p(x|y)p(z|y)$$

and therefore $x \perp z|y$. So observing the middle node of chain breaks it in two (as in a Markov chain).

Deriving the rules of Bayes Ball 2

Now consider the tent structure $X \leftarrow Y \rightarrow Z$. The joint is

$$p(x, y, z) = p(y)p(x|y)p(z|y)$$

When we condition on y, are x and z independent? We have

$$p(x, z|y) = \frac{p(x, y, z)}{p(y)} = \frac{p(y)p(x|y)p(z|y)}{p(y)} = p(x|y)p(z|y)$$

and therefore $x \perp z|y$. So observing a root node separates its children

Deriving the rules of Bayes Ball 3 (explaining away)

Finally consider a v-structure $X \to Y \leftarrow Z$. The joint is

$$p(x, y, z) = p(x)p(z)p(y|x, z)$$

When we condition on y, are x and z independent? We have

$$p(x, z|y) = \frac{p(x)p(z)p(y|x, z)}{p(y)}$$

so $x \not\perp z|y$. However, in the unconditional distribution, we have

$$p(x, z) = p(x)p(z)$$

so we see that x and z are marginally independent. So we see that conditioning on a common child at the bottom of a v-structure makes its parents become dependent. This important effect is called **explaining away**, **inter-causal reasoning**, or **Berkson's paradox**.

Explaining away example

As an example of explaining away, suppose we toss two coins, representing the binary numbers 0 and 1, and we observe the "sum" of their values. A priori, the coins are independent, but once we observe their sum, they become coupled (e.g., if the sum is 1, and the first coin is 0, then we know the second coin is 1).

Markov blankets

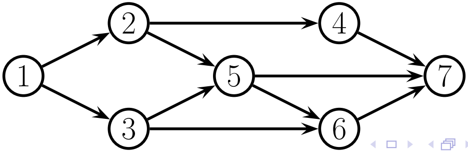
- The set of nodes that renders a node t conditionally independent of all the other nodes in the graph is called t's **Markov blanket**; we will denote this by $\mathrm{mb}(t)$. One can show that the Markov blanket of a node in a DGM is equal to the parents, the children, and the **co-parents**, i.e., other nodes who are also parents of its children:

$$\mathrm{mb}(t) \triangleq \mathrm{ch}(t) \cup \mathrm{pa}(t) \cup \mathrm{copa}(t)$$

- For example, in the figure below, we have

$$\mathrm{mb}(5) = \{6, 7\} \cup \{2, 3\} \cup \{4\} = \{2, 3, 4, 6, 7\}$$

where 4 is a co-parent of 5 because they share a common child, namely 7.



Markov blankets

- To see why the co-parents are in the Markov blanket, note that when we derive $p(x_t|\mathbf{x}_{-t}) = p(x_t, \mathbf{x}_{-t})/p(\mathbf{x}_{-t})$, all the terms that do not involve x_t will cancel out between numerator and denominator, so we are left with a product of CPDs which contain x_t in their **scope**. Hence the full conditional is

$$p(x_t|\mathbf{x}_{-t}) \quad \propto \quad p(x_t|\mathbf{x}_{\mathrm{pa}(t)}) \prod_{s \in \mathrm{ch}(t)} p(x_s|\mathbf{x}_{\mathrm{pa}(s)})$$

For example, in Figure **??** we have

$$p(x_5|\mathbf{x}_{-5}) \propto p(x_5|x_2, x_3)p(x_6|x_3, x_5)p(x_7|x_4, x_5, x_6)$$

Outline

Inference

- We want to infer the probability of the hidden variables given the visible ones:

$$p(\mathbf{x}_h|\mathbf{x}_v, \boldsymbol{\theta}) = \frac{p(\mathbf{x}_h, \mathbf{x}_v|\boldsymbol{\theta})}{p(\mathbf{x}_v|\boldsymbol{\theta})} = \frac{p(\mathbf{x}_h, \mathbf{x}_v|\boldsymbol{\theta})}{\sum_{\mathbf{x}'_h} p(\mathbf{x}'_h, \mathbf{x}_v|\boldsymbol{\theta})}$$

- Essentially we are **conditioning** on the data by **clamping** the visible variables to their observed values, $\mathbf{x}_v$, and then normalizing, to go from $p(\mathbf{x}_h, \mathbf{x}_v)$ to $p(\mathbf{x}_h|\mathbf{x}_v)$.
- The normalization constant $p(\mathbf{x}_v|\boldsymbol{\theta})$ is the likelihood of the data, also called the **probability of the evidence**.
- Let us partition the hidden variables into **query variables**, $\mathbf{x}_q$, whose value we wish to know, and the remaining **nuisance variables**, $\mathbf{x}_n$, which we are not interested in (**marginalizing out**):

$$p(\mathbf{x}_q|\mathbf{x}_v, \boldsymbol{\theta}) = \sum_{\mathbf{x}_n} p(\mathbf{x}_q, \mathbf{x}_n|\mathbf{x}_v, \boldsymbol{\theta})$$

Computational complexity

- Suppose we have V discrete random variables with say K states each.
- If the joint distribution is represented as a multi-dimensional table, we can always perform exact inferencee in $O(K^V)$ time.
- Later we explain how to exploit the factorization encoded by the GM to perform these operations in $O(VK^{w+1})$ time, where w is a quantity known as the **treewidth** of the graph.
- This measures how "tree-like" the graph is. If the graph is a tree (or a chain), we have $w = 1$, so for these models, inference takes $O(VK^2)$ time (see later).
- Unfortunately, for more general graphs, w can be large. We will therefore examine various approximate inference schemes.

Outline

Inference vs Learning

- Inference means computing (functions of) $p(\mathbf{x}_h|\mathbf{x}_v, \boldsymbol{\theta})$, where v are the visible nodes, h are the hidden nodes, and $\boldsymbol{\theta}$ are the parameters of the model, assumed to be known.
- Learning usually means computing a MAP estimate of the parameters given data:

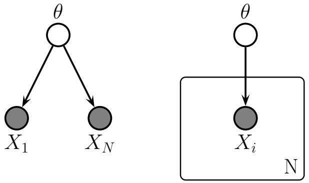
$$\hat{\boldsymbol{\theta}} = \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log p(\mathbf{x}_{i,v}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})$$

  where $\mathbf{x}_{i,v}$ are the visible variables in case i. If we have a uniform prior, $p(\boldsymbol{\theta}) \propto 1$, this reduces to the MLE, as usual.
- If we adopt a Bayesian view, the parameters are unknown variables, so learning means computing $p(\boldsymbol{\theta}|\mathcal{D})$. Thus to a Bayesian, the only distinction between inference and learning is which nodes we are inferring.

Plate notation

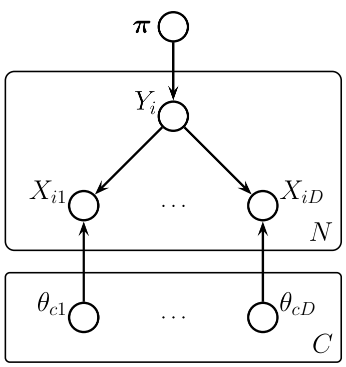
- When inferring parameters from data, we often assume the data is iid. We can represent this assumption explicitly using a graphical model.
- To avoid visual clutter, it is common to use a form of **syntactic sugar** called **plates**: we simply draw a little box around the repeated variables, with the convention that nodes within the box will get repeated when the model is **unrolled**.

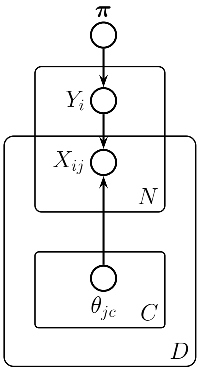
$$p(\boldsymbol{\theta}, \mathcal{D}) = p(\boldsymbol{\theta}) \left[\prod_{i=1}^{N} p(\mathbf{x}_i | \boldsymbol{\theta}) \right]$$



Nested Plates

- Nodes with multiple indices must be inside the scope of the corresponding plate.
- What is not clear from the figure is that θ_{jc} is used to generate x_{ij} iff $y_i = c$, otherwise it is ignored. This is an example of **context specific independence**, since the CI relationship $x_{ij} \perp \theta_{jc}$ only holds if $y_i \neq c$.





MLE for DGMs with fully observed data

- If all the variables are fully observed in each case, so there is no missing data and there are no hidden variables, we say the data is **complete**. For a DGM with complete data, the likelihood is given by

$$\begin{aligned} p(\mathcal{D}|\boldsymbol{\theta}) &= \prod_{i=1}^{N} p(\mathbf{x}_i|\boldsymbol{\theta}) = \prod_{i=1}^{N}\prod_{t=1}^{V} p(x_{it}|\mathbf{x}_{i,\mathrm{pa}(t)}, \boldsymbol{\theta}_t) \\ &= \prod_{t=1}^{V} p(\mathcal{D}_t|\boldsymbol{\theta}_t) \end{aligned}$$

  where $\mathcal{D}_t$ is the data associated with node t and its parents, i.e., the t'th family. This is a product of terms, one per CPD. We say that the likelihood **decomposes** according to the graph structure.

MLE for Tabular CPDs

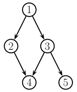
- If all CPDs are tables, we have

$$
\begin{aligned}
p(\mathcal{D}|\boldsymbol{\theta}) &= \prod_{i=1}^{N}\prod_{t=1}^{V}\prod_{c=1}^{C_t}\prod_{k=1}^{K_t} p(x_{it} = k|\mathbf{x}_{i,\mathrm{pa}(t)} = c)^{\mathbb{I}(x_{it}=k,\mathbf{x}_{i,\mathrm{pa}(t)}=c)} \\
\log p(\mathcal{D}|\boldsymbol{\theta}) &= \sum_{t,c,k}\left[\sum_{i}\mathbb{I}(x_{it} = k, \mathbf{x}_{i,\mathrm{pa}(t)} = c)\right] p(x_{it} = k|\mathbf{x}_{i,\mathrm{pa}(t)} = \\
&= \sum_{t,c,k} N_{tck}\theta_{tck}
\end{aligned}
$$

- MLE can be derived using calculus (and Lagrange multipliers to enforce $\sum_k \theta_{tck} = 1$) to give

$$
\hat{\theta}_{tck} = \frac{N_{tck}}{\sum_{k'} N_{tck'}}
$$

Worked example (for node $t = 4$)



x_1	x_2	x_3	x_4	x_5
0	0	1	0	0
0	1	1	1	1
1	1	0	1	0
0	1	1	0	0
0	1	1	1	0

x_2	x_3	$N_{tck=1}$	$N_{tck=0}$	$\hat{\theta}_{tck=1}$	$\hat{\theta}_{tck=0}$
0	0	0	0	0.0	0.0
1	0	1	0	1.0	0.0
0	1	0	1	0.0	1.0
1	1	2	1	2/3	1/3

Bayesian inference for CPD parameters

- The likelihood factorizes,

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{t=1}^{V} p(\mathcal{D}_t|\boldsymbol{\theta}_t)$$

- Now suppose that the prior factorizes as well:

$$p(\boldsymbol{\theta}) = \prod_{t=1}^{V} p(\boldsymbol{\theta}_t)$$

- Then clearly the posterior also factorizes:

$$p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}) = \prod_{t=1}^{V} p(\mathcal{D}_t|\boldsymbol{\theta}_t)p(\boldsymbol{\theta}_t)$$

- This means we can compute the posterior of each CPD independently. In other words, "factored prior plus factored likelihood implies factored posterior".

Bayesian inference for tabular CPDs

- Let us put a separate Dirichlet prior on each row of each CPT, i.e., $\boldsymbol{\theta}_{tc} \sim \mathrm{Dir}(\boldsymbol{\alpha}_{tc})$.
- Factorized posterior

$$p(\boldsymbol{\theta}|\mathcal{D}) = \prod_t \prod_c \mathrm{Dir}(\boldsymbol{\theta}_{tc}|\boldsymbol{\alpha}_{tc} + \mathbf{N}_{tc})$$

- Posterior mean estimate:

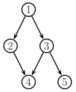
$$\overline{\theta}_{tck} = \frac{N_{tck} + \alpha_{tck}}{\sum_{k'}(N_{tck'} + \alpha_{tck'})}$$

- MAP estimate:

$$\hat{\theta}_{tck} = \frac{N_{tck} + \alpha_{tck} - 1}{\sum_{k'}(N_{tck'} + \alpha_{tck'}) - 1}$$

- MLE: just set $\alpha = 1$ (uniform prior):

Worked example (for node $t = 4$, prior $\alpha_{tck} = 1$)



x_1	x_2	x_3	x_4	x_5
0	0	1	0	0
0	1	1	1	1
1	1	0	1	0
0	1	1	0	0
0	1	1	1	0

x_2	x_3	$N_{tck=1}$	$N_{tck=0}$	$\overline{\theta}_{tck=1}$	$\overline{\theta}_{tck=0}$
0	0	0	0	$1/2$	$1/2$
1	0	1	0	$2/3$	$1/3$
0	1	0	1	$1/3$	$2/3$
1	1	2	1	$3/5$	$2/5$

Learning other kinds of CPDs

- Recall that the likelihood factorizes

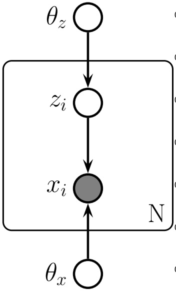
$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{t=1}^{V} p(\mathcal{D}_t|\boldsymbol{\theta}_t)$$

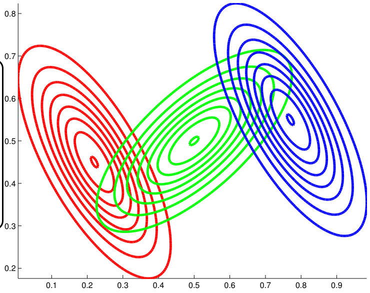
- We can optimize each $\boldsymbol{\theta}_t$ separately.
- eg Sigmoid belief net: use IRLS to fit each logistic regression node.
- Can also compute $p(\boldsymbol{\theta}_t|\mathcal{D}_t)$ using (approximate) Bayesian inference.

Outline

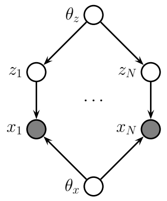
Mixture of Gaussians

$$p(\mathbf{x}_i|\boldsymbol{\theta}) = \sum_{k=1}^{K} p(z = k|\boldsymbol{\pi})\mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$





Multimodal parameter likelihood/ posterior



Posterior is a sum of $O(K^N)$ components.

$$\begin{aligned} p(\boldsymbol{\theta}|\mathcal{D}) &\propto p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta}) \\ p(\mathcal{D}|\boldsymbol{\theta}) &= \prod_i \sum_{z_i=1}^{K} p(z_i|\boldsymbol{\theta})p(\mathbf{x}_i|z_i,\boldsymbol{\theta}) \end{aligned}$$

Finding global maximum is NP-hard. Finding full posterior also intractable. Aim to find a “good” local maximum.

The Expectation Maximization (EM) algorithm

- Goal: maximize

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p(\mathbf{x}_i|\boldsymbol{\theta}) = \sum_{i=1}^{N} \log \left[\sum_{\mathbf{z}_i} p(\mathbf{x}_i, \mathbf{z}_i|\boldsymbol{\theta}) \right]$$

  Log cannot be pushed inside the sum.
- EM gets around this problem as follows. Define **complete data log likelihood**:

$$\ell_c(\boldsymbol{\theta}) \triangleq \sum_{i=1}^{N} \log p(\mathbf{x}_i, \mathbf{z}_i|\boldsymbol{\theta})$$

  This cannot be computed, since $\mathbf{z}_i$ is unknown.
- Define **expected complete data log likelihood**:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1}) \triangleq \mathbb{E}\left[\ell_c(\boldsymbol{\theta}) \middle| \mathcal{D}, \boldsymbol{\theta}^{t-1}\right]$$

  where t is the current iteration number. Q is called the **auxiliary function**. The expectation is taken wrt the old parameters, $\boldsymbol{\theta}^{t-1}$, and the observed data $\mathcal{D}$.

The Expectation Maximization (EM) algorithm

- The goal of the **E step** is to compute $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1})$, or rather, the terms inside of it which the MLE depends on; these are known as the **expected sufficient statistics** or ESS.
- In the **M step**, we optimize the Q function wrt $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}^t = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1})$$

  To perform MAP estimation, we modify the M step as follows:

$$\boldsymbol{\theta}^t = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1}) + \log p(\boldsymbol{\theta})$$

  The E step remains unchanged.

Example: EM for GMMs

The expected complete data log likelihood is given by

$$\begin{aligned}
Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t-1)}) &= \mathbb{E}\left[\sum_i \log p(\mathbf{x}_i, z_i|\boldsymbol{\theta})\right] \\
&= \sum_i \mathbb{E}\left[\log\left[\prod_{k=1}^{K} (\pi_k p(\mathbf{x}_i|\boldsymbol{\theta}_k))^{\mathbb{I}(z_i=k)}\right]\right] \\
&= \sum_i \sum_k \mathbb{E}\left[\mathbb{I}(z_i = k)\right] \log[\pi_k p(\mathbf{x}_i|\boldsymbol{\theta}_k)] \\
&= \sum_i \sum_k p(z_i = k|\mathbf{x}_i, \boldsymbol{\theta}^{t-1}) \log[\pi_k p(\mathbf{x}_i|\boldsymbol{\theta}_k)] \\
&= \sum_i \sum_k r_{ik} \log \pi_k + \sum_i \sum_k r_{ik} \log p(\mathbf{x}_i|\boldsymbol{\theta}_k)
\end{aligned}$$

where the responsibility is computed in the E step

$$r_{ik} = \frac{\pi_k p(\mathbf{x}_i|\boldsymbol{\theta}_k^{(t-1)})}{\sum_{k'} \pi_{k'} p(\mathbf{x}_i|\boldsymbol{\theta}_{k'}^{(t-1)})}$$

Example: EM for GMMs

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t-1)}) = \sum_i \sum_k r_{ik} \log \pi_k + \sum_i \sum_k r_{ik} \log p(\mathbf{x}_i | \boldsymbol{\theta}_k)$$

M step:

$$\begin{aligned}
\pi_k &= \frac{1}{N} \sum_i r_{ik} \\
\boldsymbol{\mu}_k &= \frac{\sum_i r_{ik} \mathbf{x}_i}{r_k} \\
\boldsymbol{\Sigma}_k &= \frac{\sum_i r_{ik} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T}{r_k} \\
r_k &\triangleq \sum_i r_{ik}
\end{aligned}$$

Example: EM for HMMs

- E step: compute $N_{t,i,j} = p(z_t = i, z_{t+1} = j|\mathbf{y})$ using forwards backwards algorithm (see later)
- M step for transition matrix:

$$\hat{A}(i,j) = \frac{\sum_t N_{t,i,j}}{\sum_t \sum_{j'} N_{t,i,j'}}$$

- M step for observation matrix: same as GMM case.

EM for general DGMs

- Let us assume tabular CPDs for notational simplicity.

$$p(x_{it}|\mathbf{x}_{i,\mathrm{pa}(t)}, \boldsymbol{\theta}_t) = \prod_{c=1}^{K_{\mathrm{pa}(t)}} \prod_{k=1}^{K_t} \theta_{tck}^{\mathbb{I}(x_{it}=i, \mathbf{x}_{i,\mathrm{pa}(t)}=c)}$$

- The log-likelihood of the complete data is given by

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = \sum_{t=1}^{V} \sum_{c=1}^{K_{\mathrm{pa}(t)}} \sum_{k=1}^{K_t} N_{tck} \log \theta_{tck}$$

  where $N_{tck} = \sum_{i=1}^{N} \mathbb{I}(x_{it} = i, \mathbf{x}_{i,\mathrm{pa}(t)} = c)$ are the empirical counts.

- Hence the expected complete data log-likelihood has the form

$$\mathbb{E}\left[\log p(\mathcal{D}|\boldsymbol{\theta})\right] \quad = \quad \sum_{t} \sum_{c} \sum_{k} \overline{N}_{tck} \log \theta_{tck}$$

  where

$$\overline{N}_{tck} = \sum_{i=1}^{N} \mathbb{E}\left[\mathbb{I}(x_{it} = i, \mathbf{x}_{i,\mathrm{pa}(t)} = c)\right]$$

EM for general DGMs

- The expected complete data log-likelihood has the form

$$\mathbb{E}\left[\log p(\mathcal{D}|\boldsymbol{\theta})\right] = \sum_t \sum_c \sum_k \overline{N}_{tck} \log \theta_{tck}$$

  where

$$\overline{N}_{tck} = \sum_i p(x_{it} = k, \mathbf{x}_{i,\mathrm{pa}(t)} = c|\mathcal{D}_i)$$

  where $\mathcal{D}_i$ are all the visible variables in case i.
- The quantity $p(x_{it}, \mathbf{x}_{i,\mathrm{pa}(t)}|\mathcal{D}_i, \boldsymbol{\theta})$ is known as a **family marginal**, and can be computed using any GM inference algorithm. The $\overline{N}_{tjk}$ are the expected sufficient statistics, and constitute the output of the E step.
- Given these ESS, the M step has the simple form

$$\hat{\theta}_{tck} = \frac{\overline{N}_{tck}}{\sum_{k'} \overline{N}_{tjk'}}$$

Tutorial on Probabilistic Graphical Models
3: Undirected graphical models

Kevin P. Murphy
kpmurphy@google.com

July 2012

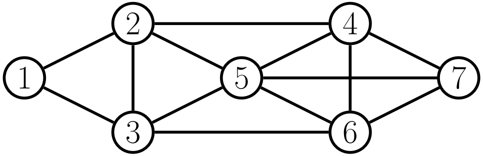
Outline

Why UGMs/ MRFs?

- UGMs define joint distributions in terms of undirected graphs (eg., 2d lattice/ grid).
- Advantages over DGMs: (1) UGMs are symmetric and therefore more "natural" for some domains such as spatial statistics or relational data; (2) UGMs handle conditioning on features to give $p(\mathbf{y}|\mathbf{x})$ in a more desirable way (conditional random fields or CRFs)
- Disdvantages over DGMs: (1) parameter learning in UGMs is more computationally expensive; (2) UGMs are not "modular", cannot plug-in off-the-shelf CPDs
- The inference problem is (basically) the same in DGMs and UGMs.

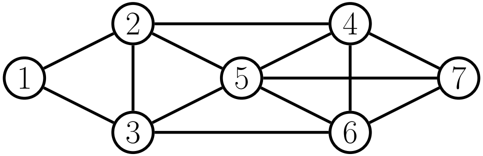
Outline

Global Markov property



- For sets of nodes A, B, and C, we say $\mathbf{x}_A \perp_G \mathbf{x}_B | \mathbf{x}_C$ iff C separates A from B in the graph G. This means that, when we remove all the nodes in C, if there are no paths connecting any node in A to any node in B, then the CI property holds.
- For example, we have that $\{1, 2\} \perp \{6, 7\} | \{3, 4, 5\}$.

Local Markov property



- The set of nodes that renders a node t conditionally independent of all the other nodes in the graph is called t's **Markov blanket**; we will denote this by $\mathrm{mb}(t)$. Formally, the Markov blanket satisfies the following property:

$$t \perp \mathcal{V} \setminus \mathrm{cl}(t) | \mathrm{mb}(t)$$

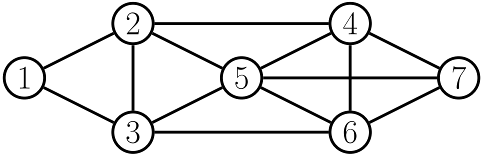
  where $\mathrm{cl}(t) \triangleq \mathrm{mb}(t) \cup \{t\}$ is the **closure** of node t. One can show that, in a UGM, a node's Markov blanket is its set of immediate neighbors. This is called the **local Markov property**.
- For example, we have $\mathrm{mb}(5) = \{2, 3, 4, 6\}$.

Pairwise Markov property

- From the local Markov property, we can easily see that two nodes are conditionally independent given the rest if there is no direct edge between them. This is called the **pairwise Markov property**. In symbols, this is written as

$$s \perp t | \mathcal{V} \setminus \{s, t\} \iff G_{st} = 0$$

Example of Markov properties of UGMs



- **Pairwise** $1 \perp 7|\text{rest}$
- **Local** $1 \perp \text{rest}|2, 3$
- **Global** $1, 2 \perp 6, 7|3, 4, 5$

Relationship of Markov properties of UGMs

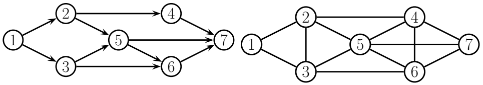
$$\begin{tikzcd}
G \arrow[r, Rightarrow] & L \arrow[r, Rightarrow] & P \arrow[ll, Rightarrow, bend left, "p(x) > 0"]
\end{tikzcd}$$

- It is obvious that global Markov implies local Markov which implies pairwise Markov. What is less obvious, but nevertheless true (assuming $p(\mathbf{x}) > 0$ for all $\mathbf{x}$, i.e., that p is a positive density), is that pairwise implies global, and hence that all these Markov properties are the same.
- The importance of this result is that it is usually easier to empirically assess pairwise conditional independence; such pairwise CI statements can be used to construct a graph from which global CI statements can be extracted.

An undirected alternative to d-separation

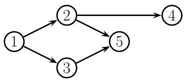
- We have seen that determinining CI relationships in UGMs is much easier than in DGMs, because we do not have to worry about the directionality of the edges. We now show how to determine CI relationships for a DGM using a UGM.
- It is tempting to simply convert the DGM to a UGM by dropping the orientation of the edges, but this is clearly incorrect, since a v-structure $A \rightarrow B \leftarrow C$ has quite different CI properties than the corresponding undirected chain $A - B - C$. The latter graph incorrectly states that $A \perp C | B$.
- To avoid such incorrect CI statements, we can add edges between the "unmarried" parents A and C, and then drop the arrows from the edges, forming (in this case) a fully connected undirected graph. This process is called **moralization**.

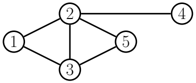
Moralization



- Figure b gives a larger example of moralization: we interconnect 2 and 3, since they have a common child 5, and we interconnect 4, 5 and 6, since they have a common child 7.
- Unfortunately, moralization loses some CI information, and therefore we cannot use the moralized UGM to determine CI properties of the DGM. For example, in Fig a, using d-separation, we see that $4 \perp 5|2$. Adding a moralization arc $4 - 5$ would lose this fact.

Ancestral graph





- Notice that the 4-5 moralization edge, due to the common child 7, is not needed if we do not observe 7 or any of its descendants.
- This suggests the following approach to determining if $A \perp B|C$. First we form the **ancestral graph** of DAG G with respect to $U = A \cup B \cup C$. This means we remove all nodes from G that are not in U or are not ancestors of U. We then moralize this ancestral graph, and apply the simple graph separation rules for UGMs.
- For example, in Fig (a) we show the ancestral graph using $U = \{2, 4, 5\}$. In Fig (b), we show the moralized version of this graph. It is clear that we now correctly conclude that $4 \perp 5|2$.

Outline

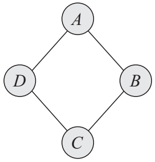
Relative expressive power

- Which model has more "expressive power", a DGM or a UGM? To formalize this question, recall that we say that G is an I-map of a distribution p if $I(G) \subseteq I(p)$. Now define G to be **perfect map** of p if $I(G) = I(p)$, in other words, the graph can represent all (and only) the CI properties of the distribution.
- It turns out that DGMs and UGMs are perfect maps for different sets of distributions. In this sense, neither is more powerful than the other as a representation language.

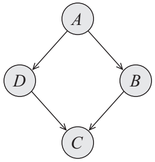
Distributions which can only be perfectly modeled by a DGM

- As an example of some CI relationships that can be perfectly modeled by a DGM but not a UGM, consider a v-structure $A \to C \leftarrow B$. This asserts that $A \perp B$, and $A \not\perp B|C$. If we drop the arrows, we get $A - C - B$, which asserts $A \perp B|C$ and $A \not\perp B$, which is incorrect. In fact, there is no UGM that can precisely represent all and only the two CI statements encoded by a v-structure.
- In general, CI properties in UGMs are monotonic, in the following sense: if $A \perp B|C$, then $A \perp B|(C \cup D)$. But in DGMs, CI properties can be non-monotonic, since conditioning on extra variables can eliminate conditional independencies due to explaining away.

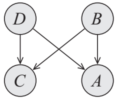
Distributions which can only be perfectly modeled by a UGM



(a)



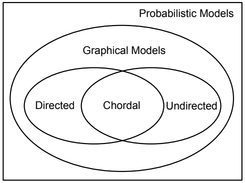
(b)



(c)

- As an example of some CI relationships that can be perfectly modeled by a UGM but not a DGM, consider the 4-cycle shown in (a). One attempt to model this with a DGM is shown in (b). This correctly asserts that $A \perp C | B, D$. However, it incorrectly asserts that $B \perp D | A$. Figure (c) is another incorrect DGM: it correctly encodes $A \perp C | B, D$, but incorrectly encodes $B \perp D$. In fact there is no DGM that can precisely represent all and only the CI statements encoded by this UGM.

Distributions which can be perfectly modeled by a DGM or UGM



- Some distributions can be perfectly modeled by either a DGM or a UGM; the resulting graphs are called **decomposable** or **chordal**. Roughly speaking, this means the following: if we collapse together all the variables in each maximal clique, to make "mega-variables", the resulting graph will be a tree. Of course, if the graph is already a tree (which includes chains as a special case), it will be chordal.

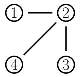
Example: directed tree

$$\begin{tikzcd}
1 \arrow[r] & 2 \arrow[dl] \arrow[d] & 1 & 2 \arrow[l] \arrow[dl] \arrow[d] \\
4 & 3 & 4 & 3
\end{tikzcd}$$

- Two Markov equivalent DGMs:

$$\begin{aligned}
p(x_1, x_2, x_3, x_4 | T) &= p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_2) \\
&= p(x_2)p(x_1|x_2)p(x_3|x_2)p(x_4|x_2)
\end{aligned}$$

Example: undirected tree



- For any undirected tree, we can write

$$p(\mathbf{x}|T) = \prod_{t\in V} p(x_t) \prod_{(s,t)\in E} \frac{p(x_s, x_t)}{p(x_s)p(x_t)}$$

  where $p(x_s, x_t)$ is an edge marginal and $p(x_t)$ is a node marginal.
- In this example we have

$$\begin{aligned} p &= p(x_1)p(x_2)p(x_3)p(x_4)\frac{p(x_1,x_2)p(x_2,x_3)p(x_2,x_4)}{p(x_1)p(x_2)p(x_2)p(x_3)p(x_2)p(x_4)} \\ &= p(x_1,x_2)\frac{p(x_2,x_3)}{p(x_2)}\frac{p(x_2,x_4)}{p(x_2)} \\ &= p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_2) \\ &= p(x_2)p(x_1|x_2)p(x_3|x_2)p(x_4|x_2) \end{aligned}$$

Outline

Parameterizing UGMs

- Since there is no topological ordering associated with an undirected graph, we can't use the chain rule to represent $p(\mathbf{y})$.
- So instead of associating CPDs with each node, we associate **potential functions** or **factors** with each maximal clique in the graph. We will denote the potential function for clique c by $\psi_c(\mathbf{y}_c|\boldsymbol{\theta}_c)$.
- A potential function can be any non-negative function of its arguments.
- The joint distribution is then defined to be proportional to the product of clique potentials.
- Rather surprisingly, one can show that any positive distribution whose CI properties can be represented by a UGM can be represented in this way.

Hammersley Clifford Theorem

Theorem (Hammersley-Clifford)

A positive distribution $p(\mathbf{y}) > 0$ satisfies the CI properties of an undirected graph G iff p can be represented as a product of factors, one per maximal clique, i.e.,

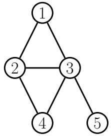
$$p(\mathbf{y}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{c \in \mathcal{C}} \psi_c(\mathbf{y}_c|\boldsymbol{\theta}_c)$$

where $\mathcal{C}$ is the set of all the (maximal) cliques of G, and $Z(\boldsymbol{\theta})$ is the **partition function** *given by*

$$Z(\boldsymbol{\theta}) \triangleq \sum_{\mathbf{x}} \prod_{c \in \mathcal{C}} \psi_c(\mathbf{y}_c|\boldsymbol{\theta}_c)$$

Note that the partition function is what ensures the overall distribution sums to 1.

Example



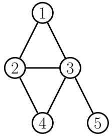
- If p satisfies the CI properties of this graph then we can write p as follows:

$$p(\mathbf{y}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})}\psi_{123}(y_1, y_2, y_3)\psi_{234}(y_2, y_3, y_4)\psi_{35}(y_3, y_5)$$

where

$$Z = \sum_{\mathbf{y}} \psi_{123}(y_1, y_2, y_3)\psi_{234}(y_2, y_3, y_4)\psi_{35}(y_3, y_5)$$

Pairwise UGMs



- We are free to restrict the parameterization to the edges of the graph, rather than the maximal cliques. This is called a **pairwise MRF**:

$$\begin{aligned} p(\mathbf{y}|\boldsymbol{\theta}) &\propto \psi_{12}(y_1, y_2)\psi_{13}(y_1, y_3)\psi_{23}(y_2, y_3)\psi_{24}(y_2, y_4)\psi_{34}(y_3, y_4)\psi_3 \\ &\propto \prod_{s \sim t} \psi_{st}(y_s, y_t) \end{aligned}$$

UGMs and Energy-based models

- Let every clique potential be associated with a energy function $E_c(\mathbf{y}_c) \geq 0$:
$$\psi_c(\mathbf{y}_c|\boldsymbol{\theta}_c) = \exp(-E(\mathbf{y}_c|\boldsymbol{\theta}_c))$$
- The corresponding joint is known as the Gibbs distribution:
$$p(\mathbf{y}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp(-\sum_c E(\mathbf{y}_c|\boldsymbol{\theta}_c))$$
- We see that high probability states correspond to low energy configurations.

Outline

UGMs and log-linear models

- Let every clique potential be a log-linear function

$$\log \psi_c(\mathbf{y}_c) \triangleq \phi_c(\mathbf{y}_c)^T \boldsymbol{\theta}_c$$

  where $\phi_c(\mathbf{x}_c)$ is a feature vector derived from the values of the variables $\mathbf{y}_c$.

- The resulting log joint has the form

$$\log p(\mathbf{y}|\boldsymbol{\theta}) = \sum_c \phi_c(\mathbf{y}_c)^T \boldsymbol{\theta}_c - Z(\boldsymbol{\theta})$$

  This is also known as a **maximum entropy** or a **log-linear** model.

Feature-based representation

- Consider a pairwise UGM, where for each edge, we associate a feature vector of length K^2 as follows:

$$\phi_{st}(y_s, y_t) = [\ldots, \mathbb{I}(y_s = j, y_t = k), \ldots]$$

- If we have a weight for each feature, we can convert this into a $K \times K$ potential function as follows:

$$\psi_{st}(y_s = j, y_t = k) = \exp([\boldsymbol{\theta}_{st}^T \phi_{st}]_{jk}) = \exp(\theta_{st}(j, k))$$

So we see that we can easily represent tabular potentials using a log-linear form. But the log-linear form is more general because we can choose (or learn) the features (basis vectors).

Probabilistic model of English spelling

- For trigram features, a table has $26^3 = 17,576$ parameters in it. However, most of these triples will never occur.
- An alternative approach is to define indicator functions that look for certain "special" triples, such as "ing", "qu-", etc. Then we can define the potential on each trigram as follows:

$$\psi(y_{t-1}, y_t, y_{t+1}) = \exp(\sum_k \theta_k \phi_k(y_{t-1}, y_t, y_{t+1}))$$

  where k indexes the different features, corresponding to "ing", "qu-", etc., and ϕ_k is the corresponding binary **feature function**.
- By tying the parameters across locations, we can define the probability of a word of any length using

$$p(\mathbf{y}|\boldsymbol{\theta}) \propto \exp(\sum_t \sum_k \theta_k \phi_k(y_{t-1}, y_t, y_{t+1}))$$

Feature induction for UGMs

- Consider the following method for learning features for a log-linear model of English spelling

  `"Inducing features of random fields",`
  `S. Della Pietra and V. Della Pietra and J. Lafferty`
  `PAMI 1997`

- Greedily add feature which bests improves likelihood. Features are atomic or conjunctions of previous features.
- Initially the model has no features, which represents the uniform distribution.
- The algorithm starts by choosing to add the feature

$$\phi_1(\mathbf{y}) = \sum_t \mathbb{I}(y_t \in \{a, \ldots, z\})$$

  which checks if any letter is lower case or not.

First feature

- After the feature is added, the parameters are (re)-fit by maximum likelihood (see later). For this feature, it turns out that $\hat{\theta}_1 = 1.944$, which means that a word with a lowercase letter in any position is about $e^{1.944} \approx 7$ times more likely than the same word without a lowercase letter in that position.
- Some samples from this model, generated using (annealed) Gibbs sampling (see later) are shown below.

```
m, r, xevo, ijjiir, b, to, jz, gsr, wq, vf, x, ga, msm
pcp, d, oziVlal, hzagh, yzop, io, advzmxnv,
emx, kayerf, mlj, rawzyb, jp, ag, ctdnnnbg, wgdw, t, kg
cy, spxcq, uzflbbf, dxtkkn, cxwx, jpd, ztzh, lv, zhpkvr
zcngotcnx, igcump, zjcjs, lqpWiqu, cefmfhc, o, lb, fdcY
yopxmvk, by, fz,, t, govyccm, ijyiduwfzo, 6xr, duh, ejv
pjw, l, fl, w
```

Next few features

- The second feature added by the algorithm checks if two adjacent characters are lower case:

$$\phi_2(\mathbf{y}) = \sum_{s \sim t} \mathbb{I}(y_s \in \{a, \ldots, z\}, y_t \in \{a, \ldots, z\})$$

  Now the model has the form

$$p(\mathbf{y}) = \frac{1}{Z} \exp(\theta_1 \phi_1(\mathbf{y}) + \theta_2 \phi_2(\mathbf{y}))$$

- Continuing in this way, the algorithm adds features for the strings `s>` and `ing>`, where $>$ represents the end of word, and for various regular expressions such as `[0-9]`, etc.

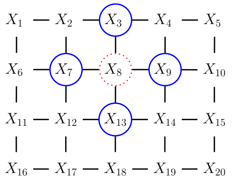
After 1000 features

- Some samples from the model with 1000 features, generated using (annealed) Gibbs sampling, are shown below.

```
was, reaser, in, there, to, will, ,, was, by, homes, th
be, reloverated, ther, which, conists, at, fores, andit
Mr., proveral, the, ,, ***, on't, prolling, prothere, ,
at, yaou, 1, chestraing, for, have, to, intrally, of, q
best, compers, ***, cluseliment, uster, of, is, deveral
thise, of, offect, inatever, thifer, constranded, state
in, thase, in, youse, menttering, and, ., of, in, verat
```

Outline

Ising model



- Consider modeling the correlation of atomic spins, $y_s \in \{-1, +1\}$, using a pairwise lattice UGM:

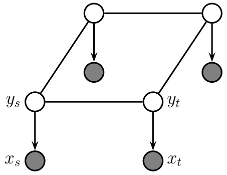
$$\psi_{st}(y_s, y_t) = \begin{pmatrix} e^{w_{st}} & e^{-w_{st}} \\ e^{-w_{st}} & e^{w_{st}} \end{pmatrix}$$

where w_{st} is the coupling strength for edge $s - t$.

Associative Markov networks

- If two nodes are not connected in the graph, we set $w_{st} = 0$. We assume that the weight matrix $\mathbf{W}$ is symmetric, so $w_{st} = w_{ts}$. Often we assume all edges have the same strength, so $w_{st} = J$ (assuming $w_{st} \neq 0$).
- If all the weights are positive, $J > 0$, then neighboring spins are likely to be in the same state; this can be used to model ferromagnets, and is an example of an **associative Markov network**.
- If the weights are sufficiently strong, the corresponding probability distribution will have two modes, corresponding to the all $+1$'s state and the all -1's state. These are called the **ground states** of the system.

Associative Markov networks as priors



- We can easily extend the Ising model to multiple states, resulting in a **Potts model**.
- It is common to use this as a prior for image labeling problems:

$$\begin{aligned} p(\mathbf{y}, \mathbf{x}|\boldsymbol{\theta}) &= p(\mathbf{y}|J) \prod_t p(x_t|y_t, \boldsymbol{\theta}) \\ &= \left[\frac{1}{Z(J)} \prod_{s \sim t} \psi(y_s, y_t; J) \right] \prod_t p(x_t|y_t, \boldsymbol{\theta}) \end{aligned}$$

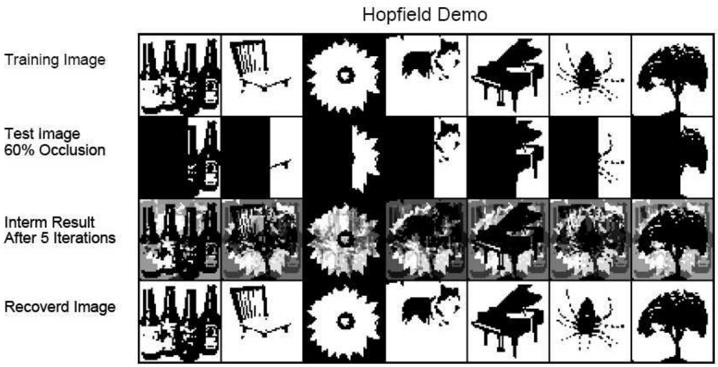
Frustrated systems

- If all of the weights are negative, $J < 0$, then the spins want to be different from their neighbors; this can be used to model an anti-ferromagnet, and results in a **frustrated system**, in which not all the constraints can be satisfied at the same time. The corresponding probability distribution will have multiple modes.
- Interestingly, computing the partition function $Z(J)$ can be done in polynomial time for associative Markov networks, but is NP-hard for general Ising models.

Hopfield networks

- A **Hopfield network** is a fully connected Ising model with a symmetric weight matrix, $\mathbf{W} = \mathbf{W}^T$.
- These weights, plus the bias terms $\mathbf{b}$, can be learned from training data using (approximate) maximum likelihood, although it is more common to use other (inferior) heuristics.
- The main application of Hopfield networks is as an **associative memory** or **content addressable memory**. The idea is this: suppose we train on a set of fully observed bit vectors, corresponding to patterns we want to memorize. Then, at test time, we present a partial pattern to the network. We would like to estimate the missing variables; this is called **pattern completion**. This can be thought of as retrieving an example from memory based on a piece of the example itself, hence the term "associative memory".

Hopfield networks



Hopfield networks

- Since exact inference is intractable in this model, it is standard to use a coordinate descent algorithm known as **iterative conditional modes** (ICM), which just sets each node to its most likely (lowest energy) state, given all its neighbors.
- The full conditional can be shown to be

$$p(y_s = 1|\mathbf{y}_{-s}, \boldsymbol{\theta}) = \mathrm{sigm}(\mathbf{w}_{s,:}^T \mathbf{y}_{-s} + b_s)$$

  Picking the most probable state amounts to using the rule $y_s^* = 1$ if $\sum_t w_{st} y_t > b_s$ and using $y_s^* = 0$ otherwise.
- Since inference is deterministic, it is also possible to interpret this model as a **recurrent neural network**.

Gaussian MRFs

- A GMRF has the form

$$
\begin{aligned}
p(\mathbf{y}|\boldsymbol{\theta}) &\propto \prod_{s\sim t} \psi_{st}(y_s, y_t) \prod_t \psi_t(y_t) \\
\psi_{st}(y_s, y_t) &= \exp(-\frac{1}{2} y_s \Lambda_{st} y_t) \\
\psi_t(y_t) &= \exp(-\frac{1}{2} \Lambda_{tt} y_t^2 + \eta_t y_t)
\end{aligned}
$$

- The joint distribution can be written as follows:

$$
p(\mathbf{y}|\boldsymbol{\theta}) \propto \exp[\boldsymbol{\eta}^T \mathbf{y} - \frac{1}{2} \mathbf{y}^T \boldsymbol{\Lambda} \mathbf{y}]
$$

  We recognize this as a multivariate Gaussian written in **information form** where $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ and $\boldsymbol{\eta} = \boldsymbol{\Lambda}\boldsymbol{\mu}$.

Gaussian MRFs

- The joint distribution can be written as follows:

$$p(\mathbf{y}|\boldsymbol{\theta}) \quad \propto \quad \exp[\boldsymbol{\eta}^T \mathbf{y} - \frac{1}{2}\mathbf{y}^T \boldsymbol{\Lambda} \mathbf{y}]$$

- If $\Lambda_{st} = 0$, then there is no pairwise term connecting s and t, so by the factorization theorem,we conclude that

$$y_s \perp y_t | \mathbf{y}_{-(st)} \iff \Lambda_{st} = 0$$

  The zero entries in $\boldsymbol{\Lambda}$ are called **structural zeros**, since they represent the absent edges in the graph. Thus undirected GGMs correspond to sparse precision matrices.

Outline

MLE for the parameters of a log-linear model

- Consider an MRF in log-linear form:

$$p(\mathbf{y}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(\sum_c \boldsymbol{\theta}_c^T \phi_c(\mathbf{y})\right)$$

  where c indexes the cliques.

- The scaled log-likelihood is given by

$$\ell(\boldsymbol{\theta}) \triangleq \frac{1}{N}\sum_i \log p(\mathbf{y}_i|\boldsymbol{\theta}) = \frac{1}{N}\sum_i \left[\sum_c \boldsymbol{\theta}_c^T \phi_c(\mathbf{y}_i) - \log Z(\boldsymbol{\theta})\right]$$

- Since MRFs are in the exponential family, we know that this function is convex in $\boldsymbol{\theta}$, so it has a unique global maximum which we can find using gradient-based optimizers.

Gradient descent

- In particular, the derivative for the weights of a particular clique, c, is given by
$$\frac{\partial \ell}{\partial \boldsymbol{\theta}_c} \quad = \quad \frac{1}{N}\sum_i \left[\phi_c(\mathbf{y}_i) - \frac{\partial}{\partial \boldsymbol{\theta}_c}\log Z(\boldsymbol{\theta})\right]$$
- One can show that the derivative of the log partition function wrt $\boldsymbol{\theta}_c$ is the expectation of the c'th feature under the model, i.e.,
$$\frac{\partial \log Z(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_c} \quad = \quad \mathbb{E}\left[\phi_c(\mathbf{y})|\boldsymbol{\theta}\right] = \sum_{\mathbf{y}} \phi_c(\mathbf{y}) p(\mathbf{y}|\boldsymbol{\theta})$$
where
$$\frac{\partial \ell}{\partial \boldsymbol{\theta}_c} = \left[\frac{1}{N}\sum_i \phi_c(\mathbf{y}_i)\right] - \mathbb{E}\left[\phi_c(\mathbf{y})\right]$$
- In the first term, we fix $\mathbf{y}$ to its observed values; this is sometimes called the **clamped term**. In the second term, $\mathbf{y}$ is free; this is sometimes called the **unclamped term** or **contrastive term**

Moment matching

- The gradient of the log likelihood can be rewritten as the expected feature vector according to the empirical distribution minus the model's expectation of the feature vector:

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}_c} = \mathbb{E}_{p_{\mathrm{emp}}}[\phi_c(\mathbf{y})] - \mathbb{E}_{p(\cdot|\boldsymbol{\theta})}[\phi_c(\mathbf{y})]$$

- At the optimum, the gradient will be zero, so the empirical distribution of the features will match the model's predictions:

$$\mathbb{E}_{p_{\mathrm{emp}}}[\phi_c(\mathbf{y})] = \mathbb{E}_{p(\cdot|\boldsymbol{\theta})}[\phi_c(\mathbf{y})]$$

This is called **moment matching**.

Training partially observed log-linear models

- Suppose we have missing data and/or hidden variables in our model. In general, we can represent such models as follows:

$$p(\mathbf{y}, \mathbf{h}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp(\sum_c \boldsymbol{\theta}_c^T \phi_c(\mathbf{h}, \mathbf{y}))$$

- The log likelihood has the form

$$\begin{aligned} \ell(\boldsymbol{\theta}) &= \frac{1}{N} \sum_i \log \left(\sum_{\mathbf{h}_i} p(\mathbf{y}_i, \mathbf{h}_i|\boldsymbol{\theta}) \right) \\ &= \frac{1}{N} \sum_i \log \left(\frac{1}{Z(\boldsymbol{\theta})} \sum_{\mathbf{h}_i} \tilde{p}(\mathbf{y}_i, \mathbf{h}_i|\boldsymbol{\theta}) \right) \end{aligned}$$

where

$$\tilde{p}(\mathbf{y}, \mathbf{h}|\boldsymbol{\theta}) \triangleq \exp \left(\sum_c \boldsymbol{\theta}_c^T \phi_c(\mathbf{h}, \mathbf{y}) \right)$$

is the unnormalized distribution.

Training partially observed log-linear models

- The term $\sum_{\mathbf{h}_i} \tilde{p}(\mathbf{y}_i, \mathbf{h}_i|\boldsymbol{\theta})$ is the same as the partition function for the whole model, except that $\mathbf{y}$ is fixed at $\mathbf{y}_i$. Hence the gradient is just the expected features where we clamp $\mathbf{y}_i$, but average over $\mathbf{h}$:

$$\frac{\partial}{\partial \boldsymbol{\theta}_c} \log \left(\sum_{\mathbf{h}_i} \tilde{p}(\mathbf{y}_i, \mathbf{h}_i|\boldsymbol{\theta}) \right) = \mathbb{E}\left[\phi_c(\mathbf{h}, \mathbf{y}_i)|\boldsymbol{\theta}\right]$$

- So the overall gradient is given by

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}_c} = \frac{1}{N} \sum_i \left\{ \mathbb{E}\left[\phi_c(\mathbf{h}, \mathbf{y}_i)|\boldsymbol{\theta}\right] - \mathbb{E}\left[\phi_c(\mathbf{h}, \mathbf{y})|\boldsymbol{\theta}\right] \right\}$$

- The first set of expectations are computed by "clamping" the visible nodes to their observed values, and the second set are computed by letting the visible nodes be free. In both cases, we marginalize over $\mathbf{h}_i$.
- An alternative approach is to use generalized EM, where we use gradient methods in the M step.

Outline

Approximating the MLE

- Although the log-likelihood is convex (in the fully observed case), computing it can be slow since it requires inference at each gradient step.
- Consequently various approximations have been proposed such as
  - Pseudo-likelihood (no hidden variables)
  - Stochastic ML (exact up to MC error)
  - Contrastive divergence (approximate version of SML)
  - Max-margin training (CRFs only)
  - Ratio matching, min. prob. flow, ...
  - etc
- We will briefly discuss a few of these.

Pseudo-likelihood

- One alternative to MLE is to maximize the **pseudo likelihood**, defined as follows:

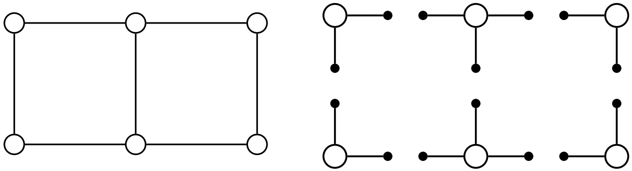
$$\begin{aligned}\ell_{PL}(\boldsymbol{\theta}) &\triangleq \sum_{\mathbf{y}} \sum_{d=1}^{D} p_{\text{emp}}(\mathbf{y}) \log p(y_d|\mathbf{y}_{-d}) \\ &= \frac{1}{N} \sum_{i=1}^{N} \sum_{d=1}^{D} \log p(y_{id}|\mathbf{y}_{i,-d}, \boldsymbol{\theta})\end{aligned}$$

  That is, we optimize the product of the full conditionals, also known as the **composite likelihood**.

- Compare this to the objective for maximum likelihood:

$$\ell_{ML}(\boldsymbol{\theta}) = \sum_{\mathbf{y},\mathbf{x}} p_{\text{emp}}(\mathbf{y}) \log p(\mathbf{y}|\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p(\mathbf{y}_i|\boldsymbol{\theta})$$

Pseudo-likelihood



- The PL approach is illustrated for a 2d grid. We learn to predict each node, given all of its neighbors. This objective is generally fast to compute since each full conditional $p(y_{id}|\mathbf{y}_{i,-d}, \boldsymbol{\theta})$ only requires summing over the states of a single node, y_{id}, in order to compute the local normalization constant.

Pseudo-likelihood

- The PL approach is similar to fitting each full conditional separately (which is the method used to train dependency networks), except that the parameters are tied between adjacent nodes.
- In the case of Gaussian MRFs, PL is equivalent to ML, but this is not true in general.
- One problem with PL is that it is hard to apply to models with hidden variables.
- Another more subtle problem is that each node assumes that its neighbors have known values. If node $t \in \mathrm{nbr}(s)$ is a perfect predictor for node s, then s will learn to rely completely on node t, even at the expense of ignoring other potentially useful information, such as its local evidence.
- PL can work well, and is much faster than exact ML.

SGD plus MC

- Recall that the gradient of the log-likelihood for a fully observed MRF is given by

$$\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) = \frac{1}{N} \sum_i \left[\phi(\mathbf{y}_i) - \mathbb{E}\left[\phi(\mathbf{y})\right]\right]$$

  The gradient for a partially observed MRF is similar.
- In both cases, we can approximate the model expectations using Monte Carlo sampling. We can combine this with stochastic gradient descent, which takes samples from the empirical distribution.

SGD plus MC

Algorithm 19.1: Stochastic maximum likelihood for fitting an MRF

```
1  Initialize weights $\boldsymbol{\theta}$ randomly;
2  $k = 0, \eta = 1$ ;
3  for each epoch do
4      for each minibatch of size $B$ do
5          for each sample $s = 1 : S$ do
6              Sample $\mathbf{y}^{s,k} \sim p(\mathbf{y}|\boldsymbol{\theta}_k)$ ;
7          $\hat{E}(\boldsymbol{\phi}(\mathbf{y})) = \frac{1}{S}\sum_{s=1}^{S} \boldsymbol{\phi}(\mathbf{y}^{s,k})$;
8          for each training case $i$ in minibatch do
9              $\mathbf{g}_{ik} = \boldsymbol{\phi}(\mathbf{y}_i) - \hat{E}(\boldsymbol{\phi}(\mathbf{y}))$ ;
10         $\mathbf{g}_k = \frac{1}{B}\sum_{i \in B} \mathbf{g}_{ik}$;
11         $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \mathbf{g}_k$;
12         $k = k + 1$;
13         Decrease step size $\eta$;
```

SGD plus MCMC (Contrastive divergence)

- We can use MCMC to generate the samples, but running MCMC to convergence at each step of the inner loop would be extremely slow.
- Fortunately, it was shown by Younes (1989) that we can start the MCMC chain at its previous value, and just take a few steps. In otherwords, we sample $\mathbf{y}^{s,k}$ by initializing the MCMC chain at $\mathbf{y}^{s,k-1}$, and then run for a few iterations.
- This is valid since $p(\mathbf{y}|\boldsymbol{\theta}^k)$ is likely to be close to $p(\mathbf{y}|\boldsymbol{\theta}^{k-1})$, since we only changed the parameters a small amount.
- This was independently discovered by Tieleman in 2008, who called it persistent contrastive divergence.
- In regular contrastive divergence, we restart the Markov chain at the data rather than at the previous state. This will not converge to the MLE.

Outline

Conditional random fields (CRFs)

- A CRF is just a version of an MRF where all the clique potentials are conditioned on input features:

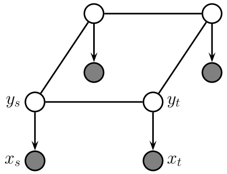
$$p(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \frac{1}{Z(\mathbf{x}, \mathbf{w})} \prod_c \psi_c(\mathbf{y}_c|\mathbf{x}, \mathbf{w})$$

- A CRF can be thought of as a **structured output** extension of logistic regression.
- We will usually assume a log-linear representation of the potentials:

$$\psi_c(\mathbf{y}_c|\mathbf{x}, \mathbf{w}) = \exp(\mathbf{w}_c^T \phi(\mathbf{x}, \mathbf{y}_c))$$

  where $\phi(\mathbf{x}, \mathbf{y}_c)$ is a feature vector derived from the global inputs $\mathbf{x}$ and the local set of labels $\mathbf{y}_c$.

MRFs and CRFs for images



- An MRF is a generative model of the form

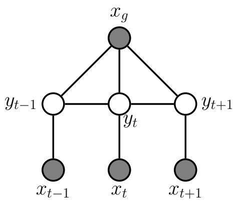
$$p(\mathbf{y}, \mathbf{x}) = p(\mathbf{y}) \prod_t p(x_t|y_t) = \left[\frac{1}{Z} \prod_{s \sim t} \psi(y_s, y_t) \right] \prod_t p(x_t|y_t, \boldsymbol{\theta})$$

- A CRF is a discriminative model of the form

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{s \sim t} \psi(y_s, y_t|\mathbf{x}) \prod_t p(y_t|\mathbf{x})$$

- We can make the edge-potentials be data-dependent, e.g., we can not enforce label smoothness if there is an observed edge between two nodes.

CRFs for sequences

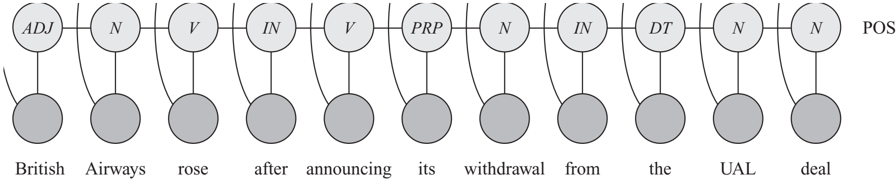


- 1d chain structured CRF:

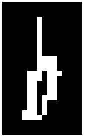
$$p(\mathbf{y}|\mathbf{x},\mathbf{w}) = \frac{1}{Z(\mathbf{x},\mathbf{w})} \prod_{t=1}^{T} \psi(y_t|\mathbf{x},\mathbf{w}) \prod_{t=1}^{T-1} \psi(y_t, y_{t+1}|\mathbf{x},\mathbf{w})$$

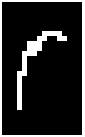
- Widely used for sequence labeling tasks (examples later).

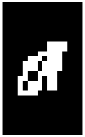
CRFs for part of speech (POS) tagging

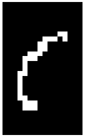


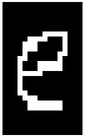
CRFs for optical character recognition (OCR)


(a)

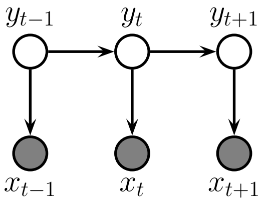

(b)


(c)


(d)


(e)

HMMs for sequences

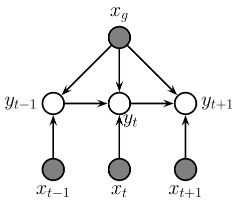


- HMM is a generative model of sequences

$$p(\mathbf{x},\mathbf{y}) = \prod_{t=1}^{T} p(y_t|y_{t-1})p(\mathbf{x}_t|y_t)$$

- We cannot easily use global features $\mathbf{x}$ (e.g., is the text ALL CAPS), since such features would be "caused" by all the y_t, leading to loss of the Markov property.
- In addition, even fitting local generative feature models $p(\mathbf{x}_t|y_t)$ can be hard.

MEMM for sequences

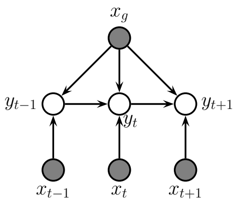


- A maximum entropy Markov model (MEMM) is a discriminative directed model of the form

$$p(\mathbf{y}|\mathbf{x}) = \prod_t p(y_t|y_{t-1}, \mathbf{x})$$

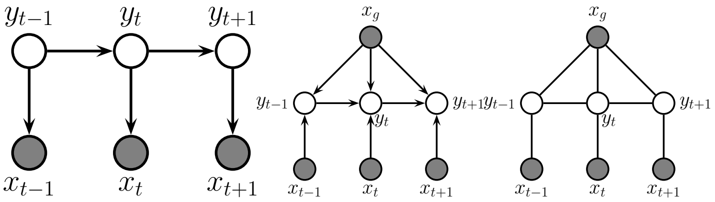
- This is easy to train (given labeled data): we just fit a logistic regression (say) model $p(y_t = k|y_{t-1} = j, \mathbf{x})$ separarately for each k.
- However, it suffers from a subtle problem known as “label bias”.

The label bias problem



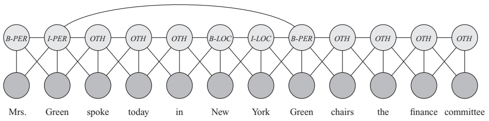
- The label bias problem is that local features at time t do not influence states prior to time t. This follows by examining the DAG, which shows that $\mathbf{x}_t$ is d-separated from y_{t-1} (and all earlier time points) by the v-structure at y_t, which is a hidden child, thus blocking the information flow.

The label bias problem



- To understand what this means in practice, consider the part of speech (POS) tagging task. Suppose we see the word "banks"; this could be a verb (as in "he banks at BoA"), or a noun (as in "the river banks were overflowing"). Locally the POS tag for the word is ambiguous. However, suppose that later in the sentence, we see the word "fishing"; this gives us enough context to infer that the sense of "banks" is "river banks". However, in an MEMM (unlike in an HMM and CRF), the "fishing" evidence will not flow backwards, so we will not be able to disambiguate "banks".

Skip-chain CRFs for named-entity recognition

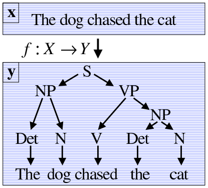


KEY

B-PER	Begin person name	*I-LOC*	Within location name
I-PER	Within person name	*OTH*	Not an entitiy
B-LOC	Begin location name		

- The label space is BIO (begin, inside, other) crossed with Person or Location.
- Textually identical strings are likely to correspond to the same state, so we add long-range correlation arcs.
- This makes inference more complicated (see later).

Discriminative PCFGs



$$\Psi(\mathbf{x},\mathbf{y}) = \left(\begin{array}{c|c} 1 & S \to NP\ VP \\ 0 & S \to NP \\ 2 & NP \to Det\ N \\ 1 & VP \to V\ NP \\ \vdots & \\ 0 & Det \to dog \\ 2 & Det \to the \\ 1 & N \to dog \\ 1 & V \to chased \\ 1 & N \to cat \end{array}\right)$$

- We can define a model of the form

$$p(\mathbf{y}|\mathbf{x}) \propto \exp(\mathbf{w}^T \boldsymbol{\Psi}(\mathbf{x},\mathbf{y}))$$

  where $\mathbf{x}$ is a sentence, $\mathbf{y}$ is a labeled parse tree, and $\boldsymbol{\Psi}(\mathbf{x},\mathbf{y})$ is a feature vector.
- If $\boldsymbol{\Psi}(\mathbf{x},\mathbf{y})$ factorizes over the parse tree, and the grammar is a probabilistic context free grammar (PCFG), we can compute $\operatorname{argmax} p(\mathbf{y}|\mathbf{x})$ in $O(T^3)$ time using dynamic programming.

Outline

MLE for CRF

- The scaled log-likelihood is

$$\begin{aligned}\ell(\mathbf{w}) &\triangleq \frac{1}{N}\sum_i \log p(\mathbf{y}_i|\mathbf{x}_i,\mathbf{w}) \\ &= \frac{1}{N}\sum_i \left[\sum_c \mathbf{w}_c^T \phi_c(\mathbf{y}_i,\mathbf{x}_i) - \log Z(\mathbf{w},\mathbf{x}_i)\right]\end{aligned}$$

- The gradient is

$$\begin{aligned}\frac{\partial \ell}{\partial \mathbf{w}_c} &= \frac{1}{N}\sum_i \left[\phi_c(\mathbf{y}_i,\mathbf{x}_i) - \frac{\partial}{\partial \mathbf{w}_c}\log Z(\mathbf{w},\mathbf{x}_i)\right] \\ &= \frac{1}{N}\sum_i \left[\phi_c(\mathbf{y}_i,\mathbf{x}_i) - \mathbb{E}\left[\phi_c(\mathbf{y},\mathbf{x}_i)\right]\right]\end{aligned}$$

- Note that we now have to perform inference for every single training case inside each gradient step, which is $O(N)$ times slower than the MRF case. This is because the partition function depends on the inputs $\mathbf{x}_i$.

Parameter tying

- In most applications of CRFs (and some applications of MRFs), the size of the graph structure can vary. Hence we need to use parameter tying to ensure we can define a distribution of arbitrary size.
- In the pairwise case, we can write the model as follows:

$$p(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \frac{1}{Z(\mathbf{w}, \mathbf{x})} \exp\left(\mathbf{w}^T \phi(\mathbf{y}, \mathbf{x})\right)$$

where $\mathbf{w} = [\mathbf{w}_n, \mathbf{w}_e]$ are the node and edge parameters, and

$$\phi(\mathbf{y}, \mathbf{x}) \triangleq [\sum_t \phi_t(y_t, \mathbf{x}), \sum_{s \sim t} \phi_{st}(y_s, y_t, \mathbf{x})]$$

are the summed node and edge features (these are the sufficient statistics). The gradient expression is easily modified to handle this case.

Regularization

- In practice, it is important to use a prior/ regularization to prevent overfitting. If we use a Gaussian prior, the new objective becomes

$$\ell'(\mathbf{w}) \triangleq \frac{1}{N}\sum_i \log p(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w}) - \lambda||\mathbf{w}||_2^2$$

  It is simple to modify the gradient expression.
- Alternatively, we can use ℓ_1 regularization. For example, we could use ℓ_1 for the edge weights $\mathbf{w}_e$ to learn a sparse graph structure, and ℓ_2 for the node weights $\mathbf{w}_n$. In other words, the objective becomes

$$\ell'(\mathbf{w}) \triangleq \frac{1}{N}\sum_i \log p(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w}) - \lambda_1||\mathbf{w}_e||_1 - \lambda_2||\mathbf{w}_n||_2^2$$

  Unfortunately, the optimization algorithms are more complicated when we use ℓ_1 (see Vishy's tutorial), although the problem is still convex.

Outline

Structural SVMs, max-margin Markov nets, and all that

- SSVMs, M3nets, etc are essentially identical to CRFs The main difference is in the training procedure.
- For CRFs, we usually use MAP estimation, i.e., we minimize

$$R_{MAP}(\mathbf{w}) = -\log p(\mathbf{w}) - \sum_{i=1}^{N} \log p(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w})$$

- However, at test time, we pick the label so as to minimize the posterior expected loss:

$$\hat{\mathbf{y}}(\mathbf{x}|\mathbf{w}) = \operatorname*{argmin}_{\hat{\mathbf{y}}} \sum_{\mathbf{y}} L(\mathbf{y}, \hat{\mathbf{y}}) p(\mathbf{y}|\mathbf{x}, \mathbf{w})$$

  where $L(\mathbf{y}^*, \hat{\mathbf{y}})$ is the loss we incur when we estimate $\hat{\mathbf{y}}$ but the truth is $\mathbf{y}^*$.
- It therefore seems reasonable to take the loss function into account when performing parameter estimation.

A new objective

- Suppose we minimize the (regularized) posterior expected loss on the training set:

$$R_{EL}(\mathbf{w}) \triangleq -\log p(\mathbf{w}) + \sum_{i=1}^{N} \log \left[\sum_{\mathbf{y}} L(\mathbf{y}_i, \mathbf{y}) p(\mathbf{y}|\mathbf{x}_i, \mathbf{w}) \right]$$

  In the special case of 0-1 loss, $L(\mathbf{y}_i, \mathbf{y}) = 1 - \delta_{\mathbf{y},\mathbf{y}_i}$, this reduces to R_{MAP}.

- Assume that we can write our model in the following form:

$$p(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}))}{Z(\mathbf{x}, \mathbf{w})}, p(\mathbf{w}) = \frac{\exp(-E(\mathbf{w}))}{Z}$$

  where $Z(\mathbf{x}, \mathbf{w}) = \sum_{\mathbf{y}} \exp(\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}))$.

- Let $L(\mathbf{y}_i, \mathbf{y}) = \exp \tilde{L}(\mathbf{y}_i, \mathbf{y})$.

Substituting in

$$\begin{aligned}
R_{EL}(\mathbf{w}) &\triangleq -\log p(\mathbf{w}) + \sum_{i=1}^{N} \log \left[\sum_{\mathbf{y}} L(\mathbf{y}_i, \mathbf{y}) p(\mathbf{y}|\mathbf{x}_i, \mathbf{w}) \right] \\
&= -\log p(\mathbf{w}) + \sum_{i} \log \left[\sum_{\mathbf{y}} \exp \tilde{L}(\mathbf{y}_i, \mathbf{y}) \frac{\exp(\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}))}{Z(\mathbf{x}_i, \mathbf{w})} \right] \\
&= E(\mathbf{w}) + \sum_{i} -\log Z(\mathbf{x}_i, \mathbf{w}) \\
&\quad + \log \sum_{\mathbf{y}} \exp \left(\tilde{L}(\mathbf{y}_i, \mathbf{y}) + \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y}) \right)
\end{aligned}$$

A convex upper bound

- Using some simple bounds on the log-sum-exp function (see Yuille and He, "Probabilistic models of vision and max-margin methods", *Frontiers of Electrical and Electronic Engineering* 2011) we can show

$$R_{EL}(\mathbf{w}) \le E(\mathbf{w}) + \sum_{i=1}^{N} \left[\max_{\mathbf{y}} \left\{ \tilde{L}(\mathbf{y}_i, \mathbf{y}) + \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y}) \right\} - \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y}_i) \right]$$

- If we set $E(\mathbf{w}) = -\frac{1}{2C}||\mathbf{w}||_2^2$,

$$R_{SSVM}(\mathbf{w}) \triangleq \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^{N} \left[\max_{\mathbf{y}} \left\{ \tilde{L}(\mathbf{y}_i, \mathbf{y}) + \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y}) \right\} - \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y}_i) \right]$$

using loss-augmented decoding.

The SVM objective

- From before

$$R_{SSVM}(\mathbf{w}) \triangleq \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^{N} \left[\max_{\mathbf{y}} \left\{ \tilde{L}(\mathbf{y}_i, \mathbf{y}) + \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y}) \right\} - \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y}_i) \right]$$

- In the special case that $\mathcal{Y} = \{-1, +1\}$ $L(y^*, y) = 1 - \delta_{y,y^*}$, and $\phi(\mathbf{x}, y) = \frac{1}{2} y \mathbf{x}$, this criterion reduces to the following (by considering the two cases that $y = y_i$ and $y \neq y_i$):

$$R_{SVM}(\mathbf{w}) \triangleq \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^{N} \left[\max\{0, 1 - y_i \mathbf{w}^T \mathbf{x}_i\} \right]$$

which is the standard binary SVM objective.

SSVMs vs CRFs

- CRF training and SSVM training are both convex problems (assuming no latent variables).
- However, CRF training requires computing marginals $p(\mathbf{y}_c|\mathbf{x})$ in the inner loop, whereas SSVM requires computing MAP estimates $\operatorname{argmax} p(\mathbf{y}|\mathbf{x})$. The latter is provably faster in some cases (see later).
- In addtion, SSVM training is “loss aware”.
- In both cases, the graph structure plays a crucial role in defining the scope of allowable features and enabling efficient inference/ decoding.

Tutorial on Probabilistic Graphical Models
4: Exact inference

Kevin P. Murphy
kpmurphy@google.com

July 2012

Outline

Inferential goals

- Marginals:

$$p(\mathbf{x}_h|\mathbf{x}_v, \boldsymbol{\theta}) = \frac{p(\mathbf{x}_h, \mathbf{x}_v|\boldsymbol{\theta})}{p(\mathbf{x}_v|\boldsymbol{\theta})}$$

- Probability of the evidence:

$$p(\mathbf{x}_v|\boldsymbol{\theta}) = Z(\mathbf{x}_v, \boldsymbol{\theta}) = \sum_{\mathbf{x}'_h} p(\mathbf{x}'_h, \mathbf{x}_v|\boldsymbol{\theta})$$

- MAP estimation

$$\hat{\mathbf{x}}_h = \operatorname{argmax} p(\mathbf{x}_h, \mathbf{x}_v|\boldsymbol{\theta})$$

- Posterior samples $\mathbf{x}_h^s \sim p(\mathbf{x}_h|\mathbf{x}_v, \boldsymbol{\theta})$
- Inference (often) needed as subroutine for learning $\hat{\boldsymbol{\theta}} = \operatorname{argmax} p(\boldsymbol{\theta}|\mathcal{D})$
- Can also compute posterior over parameters, $p(\boldsymbol{\theta}|\mathcal{D})$ — Bayesian learning, not covered

Outline of algorithms

- Exact, deterministic
- Approximate, deterministic
- Approximate, stochastic

Outline

Forwards-backwards algorithm

- Consider a 1d chain representing an HMM

$$p(\mathbf{z}, \mathbf{x}) = \prod_{t=1}^{T} p(z_t | z_{t-1}) p(\mathbf{x}_t | z_t)$$

- We first compute the filtered marginals in the forwards pass

$$\alpha_t(j) \triangleq p(z_t = j | \mathbf{x}_{1:t})$$

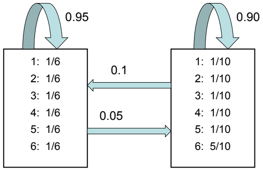
- We then compute the smoothed marginals in the backwards pass

$$\gamma_t(j) \triangleq p(z_t = j | \mathbf{x}_{1:T})$$

- As we explain below, this forwards-backwards algorithm takes $O(TK^2)$ time. It can also be applied to chain CRFs representing $p(\mathbf{z}_{1:T} | \mathbf{x}_{1:T})$.
- A simple modification yields the Viterbi algorithm which computes the most probable sequence of states (not the same as the sequence of most probable states):

$$\mathbf{z}^* = \operatorname{argmax} p(\mathbf{z} | \mathbf{x}_{1:T})$$

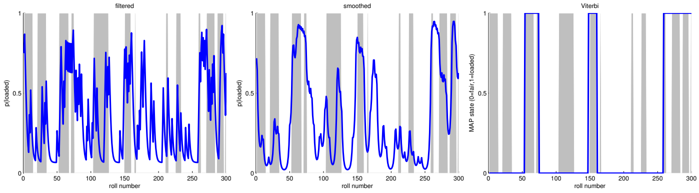
The occasionally dishonest casino



- To illustrate the difference between filtering and smoothing, consider the following example from Durbin et al, "Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids".
- $x_t \in \{1, 2, \ldots, 6\}$ represents which dice face shows up, and z_t represents the identity of the dice that is being used. Most of the time the casino uses a fair dice, $z = 1$, but occasionally it switches to a loaded dice, $z = 2$, for a short period.

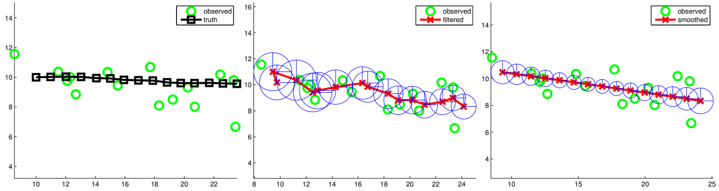
```
Roll: 66415321616211523465321435663426165523423231514246415
Dice: LLLLLLLLLLLLLLFFFFFFLLLLLLLLLLLLLLFFFFFFFFFFFFFFFFFF
```

Filtering, smoothing, MAP in the occasionally dishonest casino



- If we threshold the estimates at 0.5 and compare to the true sequence, we find that the filtered method makes 71 errors out of 300, the smoothed method makes 49/300, and the MAP path makes 60/300 errors.
- It is not surprising that smoothing makes fewer errors than Viterbi, since the optimal way to minimize bit-error rate is to threshold the posterior marginals.
- Nevertheless, for some applications, we may prefer the Viterbi decoding, to enforce consistency.

Kalman filtering and smoothing



- Suppose $p(\mathbf{z}, \mathbf{x})$ is a linear dynamical system subject to Gaussian noise, where $\mathbf{z} \in \mathbb{R}^K$ and $\mathbf{x} \in \mathbb{R}^D$.
- The analog of forwards-backwards is known as Kalman filtering/ smoothing.

The forwards algorithm

- We just apply Bayes rule sequentially to recursively update the belief state:

$$\begin{aligned}
\alpha_t(j) &\triangleq p(z_t = j|\mathbf{x}_{1:t}) = p(z_t = j|\mathbf{x}_t, \mathbf{x}_{1:t-1}) \\
&\propto p(\mathbf{x}_t|z_t = j, \cancel{\mathbf{x}_{1:t-1}})p(z_t = j|\mathbf{x}_{1:t-1})
\end{aligned}$$

  where $p(\mathbf{x}_t|z_t = j)$ is the likelihood and the prior is the one-step-ahead predictive density

$$p(z_t = j|\mathbf{x}_{1:t-1}) = \sum_i p(z_t = j|z_{t-1} = i)\alpha_{t-1}(i)$$

- This is called the predict-update cycle.
- In matrix-vector notation, we can write the update in the following simple form:

$$\boldsymbol{\alpha}_t \propto \boldsymbol{\psi}_t \odot (\boldsymbol{\Psi}^T \boldsymbol{\alpha}_{t-1})$$

- The base case is $\boldsymbol{\alpha}_1 \propto \boldsymbol{\psi}_1 \odot \boldsymbol{\pi}$

The backwards algorithm

- The key decomposition relies on the fact that we can break the chain into two parts, the past and the future, by conditioning on z_t:

$$\begin{aligned} p(z_t = j|\mathbf{x}_{1:T}) &\propto p(z_t = j, \mathbf{x}_{t+1:T}|\mathbf{x}_{1:t}) \propto p(z_t = j|\mathbf{x}_{1:t})p(\mathbf{x}_{t+1:T}|z \\ \gamma_t(j) &\propto \alpha_t(j)\beta_t(j) \end{aligned}$$

  where we have defined

$$\beta_t(j) \triangleq p(\mathbf{x}_{t+1:T}|z_t = j)$$

  as the conditional likelihood of future evidence given that the hidden state at time t is j.
- We now derive a recursive update for β_t. (It is also possible to directly update γ_t.)

The backwards algorithm

- Working right to left, we have

$$\begin{aligned}
\beta_{t-1}(i) &= p(\mathbf{x}_{t:T} | z_{t-1} = i) \\
&= \sum_j p(z_t = j, \mathbf{x}_t, \mathbf{x}_{t+1:T} | z_{t-1} = i) \\
&= \sum_j p(\mathbf{x}_{t+1:T} | z_t = j, \cancel{\mathbf{x}_t}, \cancel{z_{t-1} = i}) p(z_t = j, \mathbf{x}_t | z_{t-1} = i) \\
&= \sum_j p(\mathbf{x}_{t+1:T} | z_t = j) p(\mathbf{x}_t | z_t = j, \cancel{z_{t-1} = i}) p(z_t = j | z_{t-1} \\
&= \sum_j \beta_t(j) \psi_t(j) \psi(i, j)
\end{aligned}$$

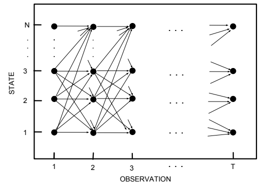
- We can write the resulting equation in matrix-vector form as

$$\boldsymbol{\beta}_{t-1} = \boldsymbol{\Psi}(\boldsymbol{\psi}_t \odot \boldsymbol{\beta}_t)$$

- The base case is

$$\beta_T(i) = p(\mathbf{x}_{T+1:T} | z_T = i) = p(\emptyset | z_T = i) = 1$$

The Viterbi algorithm



- The **Viterbi** algorithm can be used to compute the most probable sequence of states in a chain-structured graphical model, i.e., it can compute

$$\mathbf{z}^* = \arg\max_{\mathbf{z}_{1:T}} p(\mathbf{z}_{1:T} | \mathbf{x}_{1:T})$$

- This is equivalent to computing a shortest path through the **trellis diagram**, where the nodes are possible states at each time step, and the node and edge weights are log probabilities. That is, the weight of a path $z_1, z_2, \ldots, z_T$ is given by

$$\log \pi_1(z_1) + \log \phi_1(z_1) + \sum^{T} [\log \psi(z_{t-1}, z_t) + \log \phi_t(z_t)]$$

MAP vs MPM

- In Viterbi, when we estimate z_t, we "max out" the other variables:

$$z_t^* = \underset{z_t}{\operatorname{argmax}} \max_{\mathbf{z}_{1:t-1}, \mathbf{z}_{t+1:T}} p(\mathbf{z}_{1:t-1}, z_t, \mathbf{z}_{t+1:T} | \mathbf{x}_{1:T})$$

  whereas we when we use forwards-backwards, we sum out the other variables to get a maximum posterior marginal (MPM):

$$\hat{z}_t = \underset{z_t}{\operatorname{argmax}} \sum_{\mathbf{z}_{1:t-1}, \mathbf{z}_{t+1:T}} p(\mathbf{z}_{1:t-1}, z_t, \mathbf{z}_{t+1:T} | \mathbf{x}_{1:T})$$

- The sequence of MPMs may not be equal to the joint MAP sequence, e.g., "wreck a nice beach" might be the MPM interpretation of a series of acoustic waveforms, but the MAP sequence is more likely to be "recognize speech".

The Viterbi algorithm: forwards pass

- Define the probability of ending up in state j at time t, given that we take the most probable path:

$$\delta_t(j) \triangleq \max_{z_1,\dots,z_{t-1}} p(\mathbf{z}_{1:t-1}, z_t = j | \mathbf{x}_{1:t})$$

- The most probable path to state j at time t must consist of the most probable path to some other state i at time $t-1$, followed by a transition from i to j.

$$\delta_t(j) = \max_i \delta_{t-1}(i)\psi(i,j)\phi_t(j)$$

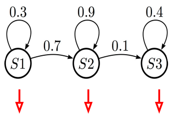
- We also keep track of the most likely previous state, for each possible state that we end up in:

$$a_t(j) = \operatorname*{argmax}_i \delta_{t-1}(i)\psi(i,j)\phi_t(j)$$

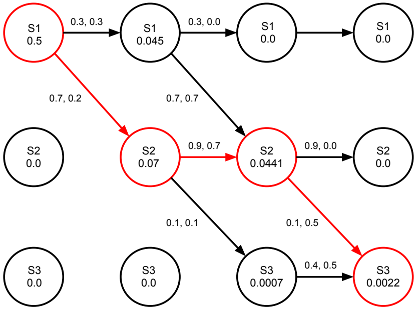
- We initialize by setting

$$\delta_1(j) = \pi_j \phi_1(j)$$

The Viterbi algorithm: traceback pass (data = C1, C3, C4, C6)



C1	0.5	0	0
C2	0.3	0	0
C3	0.2	0.2	0
C4	0	0.7	0.1
C5	0	0.1	0
C6	0	0	0.5
C7	0	0	0.4



- We start by computing the most probable final state z_T^*:

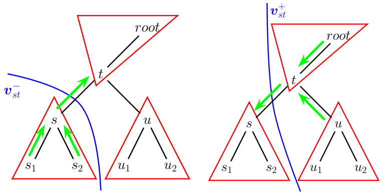
$$z_T^* = \arg\max_i \delta_T(i)$$

- We can then compute the most probable sequence of states using **traceback**:

$$z_t^* = a_{t+1}(z_{t+1}^*)$$

Outline

Belief propagation (BP) for trees

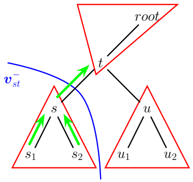


- Let us assume we have a pairwise MRF or CRF

$$p(\mathbf{x}|\mathbf{v}) = \frac{1}{Z(\mathbf{v})} \prod_{s \in \mathcal{V}} \psi_s(x_s) \prod_{(s,t) \in \mathcal{E}} \psi_{s,t}(x_s, x_t)$$

- We can send messages from the leaves up to the root and back down to compute posterior marginals $p(x_t|\mathbf{v})$ for each node in $O(VK^2)$ time.
- This is a natural generalization of forwards-backwards from chains to trees.
- A similar algorithm can be used to compute the MAP estimate or posterior samples.

Bottom-up pass



- Goal: compute $\mathrm{bel}_t^-(x_t) \triangleq p(x_t|\mathbf{v}_t^-)$,
- Suppose, by induction, that we have computed "messages" from t's children: $m_{s\to t}^-(x_t) = p(x_t|\mathbf{v}_{st}^-)$.
- Absorb incoming messages from children:

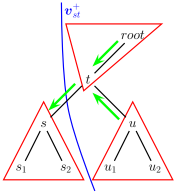
$$\mathrm{bel}_t^-(x_t) \triangleq p(x_t|\mathbf{v}_t^-) = \frac{1}{Z_t}\psi_t(x_t)\prod_{c\in\mathrm{ch}(t)} m_{c\to t}^-(x_t)$$

- Compute message from node s to its parent t:

$$m_{s\to t}^-(x_t) = \sum_{x_s}\psi_{st}(x_s,x_t)\mathrm{bel}_s^-(x_s)$$

- Analog of forwards pass.

Top-down pass



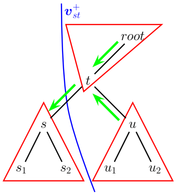
- Assume we have top-down messages:

$$m_{t\to s}^{+}(x_s) \triangleq p(x_t|\mathbf{v}_{st}^{+})$$

- Smoothed posterior belief: absorb all incoming top-down messages

$$\mathrm{bel}_s(x_s) \triangleq p(x_s|\mathbf{v}) \propto \mathrm{bel}_s^{-}(x_s) \prod_{t\in\mathrm{pa}(s)} m_{t\to s}^{+}(x_t)$$

Top-down pass



- Top-down message (belief updating version):

$$m_{t\to s}^+(x_s) \triangleq p(x_s|\mathbf{v}_{st}^+) = \sum_{x_t} \psi_{st}(x_s, x_t) \frac{\text{bel}_t(x_t)}{m_{s\to t}^-(x_t)}$$

- Top-down message (sum-product version):

$$m_{t\to s}^+(x_s) = \sum_{x_t} \psi_{st}(x_s, x_t)\psi_t(x_t) \prod_{c\in \text{ch}(t), c\neq s} m_{c\to t}^-(x_t) \prod_{p\in \text{pa}(t)} m_{p\to t}^+(x$$

- In the case of a chain, t only has one child s and one parent p, so the above simplifies to

$$m_{t\to s}^+(x_s) = \sum \psi_{st}(x_s, x_t)\psi_t(x_t) m_{p\to t}^+(x_t)$$

Parallel, distributed version

- We can define a parallel version as follows
- Absorb from all neighbors

$$\mathrm{bel}_s(x_s) \propto \psi_s(x_s) \prod_{t \in \mathrm{nbr}_s} m_{t \to s}(x_s)$$

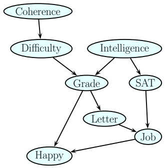
- Send messages to each of its neighbors:

$$m_{s \to t}(x_t) \quad = \quad \sum_{x_s} \left(\psi_s(x_s) \psi_{st}(x_s, x_t) \prod_{u \in \mathrm{nbr}_s \setminus t} m_{u \to s}(x_s) \right)$$

- cf systolic array
- Takes V steps to converge for a tree.
- For a loopy graph, may not converge, and even if it does, may not be correct (see later).

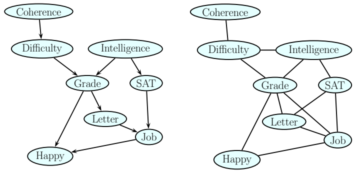
Outline

Exact inference for general graphs



- What if the graph is not a chain or tree?
- We will see that exact inference is exponential in the treewidth of the graph, $O(VK^{w+1})$.
- For a chain or tree, treewidth is 1, so inference is $O(VK^2)$.
- Some graphs have low treewidth, others (eg grids) have large treewidth.

Example



- Consider the DGM

$$
\begin{aligned}
&P(C, D, I, G, S, L, J, H) \\
&\quad = \quad P(C)P(D|C)P(I)P(G|I, D)P(S|I)P(L|G)P(J|L, S)P(H|G, J)
\end{aligned}
$$

- To convert the DGM to a UGM, we simply define a potential or factor for every CPD, yielding

$$
\begin{aligned}
&p(C, D, I, G, S, L, J, H) \\
&\quad = \quad \psi_C(C)\psi_D(D, C)\psi_I(I)\psi_G(G, I, D)\psi_S(S, I)\psi_L(L, G)\psi_J(J, L, S)\psi
\end{aligned}
$$

- Suppose we want to compute the prior marginal $p(J)$

Variable elimination

- Naively

$$p(J) = \sum_L \sum_S \sum_G \sum_H \sum_I \sum_D \sum_C p(C, D, I, G, S, L, J, H)$$

- We can be smarter by **pushing sums inside products** (or maxes over sums: generalized distributive law)

Variable elimination

$$\sum_L \sum_S \psi_J(J,L,S) \sum_G \psi_L(L,G) \sum_H \psi_H(H,G,J) \sum_I \psi_S(S,I)\psi_I(I) \sum_D \psi_G(G,I,D) \underbrace{\sum_C \psi_C(C)\psi_D(D,C)}_{\tau_1(D)}$$

$$\sum_L \sum_S \psi_J(J,L,S) \sum_G \psi_L(L,G) \sum_H \psi_H(H,G,J) \sum_I \psi_S(S,I)\psi_I(I) \underbrace{\sum_D \psi_G(G,I,D)\tau_1(D)}_{\tau_2(G,I)}$$

$$\sum_L \sum_S \psi_J(J,L,S) \sum_G \psi_L(L,G) \sum_H \psi_H(H,G,J) \underbrace{\sum_I \psi_S(S,I)\psi_I(I)\tau_2(G,I)}_{\tau_3(G,S)}$$

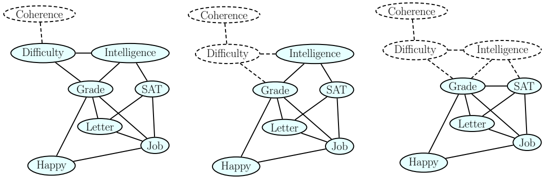
$$\sum_L \sum_S \psi_J(J,L,S) \sum_G \psi_L(L,G) \underbrace{\sum_H \psi_H(H,G,J)}_{\tau_4(G,J)} \tau_3(G,S)$$

$$\sum_L \sum_S \psi_J(J,L,S) \underbrace{\sum_G \psi_L(L,G)\tau_4(G,J)\tau_3(G,S)}_{\tau_5(J,L,S)}$$

$$\sum_L \underbrace{\sum_S \psi_J(J,L,S)\tau_5(J,L,S)}_{\tau_6(J,L)}$$

$$\underbrace{\sum_L \tau_6(J,L)}_{\tau_7(J)}$$

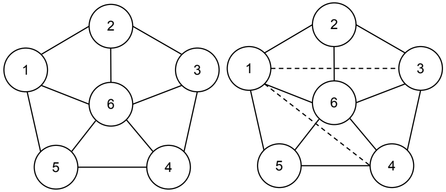
Graphical representation of Variable elimination



- Every time we eliminate a variable, we add edges to its uneliminated neighbors.
- The resulting maximal cliques correspond to the factors created by VE.

$$\{C, D\}, \{D, I, G\}, \{G, L, S, J\}, \{G, J, H\}, \{G, I, S\}$$

Chordal graphs



- The fill-in edges added during VE mean the resulting graph is chordal/ triangulated / chordal.
- This means that every undirected 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.

Treewidth

- It is clear that the time complexity of VE is
$$\prod_{c \in \mathcal{C}(G(\prec))} K^{|c|}$$
where $\mathcal{C}$ are the cliques that are created, $|c|$ is the size of the clique c, and we assume for notational simplicity that all the variables have K states each.
- Let us define the **induced width** of a graph given elimination ordering $\prec$, denoted $w(\prec)$, as the size of the largest factor (i.e., the largest clique in the induced graph) minus 1.
- Then it is easy to see that the complexity of VE with ordering $\prec$ is $O(K^{w(\prec)+1})$.
- Let us define the **treewidth** of a graph as the minimal induced width.
$$w \triangleq \min_{\prec} \max_{c \in G(\prec)} |c| - 1$$
Then clearly the best possible running time for VE is $O(DK^{w+1})$.

Minimizing treewidth

- Unfortunately, one can show that for arbitrary graphs, finding an elimination ordering $\prec$ that minimizes $w(\prec)$ is NP-hard.
- In practice greedy search techniques are used to find reasonable orderings, although people have tried other heuristic methods for discrete optimization, such as genetic algorithms.
- It is also possible to derive approximate algorithms with provable performance guarantees (Amir10).

Illustration of a bad elimination order

$$\sum_D \sum_C \psi_D(D,C) \sum_H \sum_L \sum_S \psi_J(J,L,S) \sum_I \psi_I(I)\psi_S(S,I) \underbrace{\sum_G \psi_G(G,I,D)\psi_L(L,)\psi_H(H,G,J)}_{\tau_1(I,D,L,J,H)}$$

$$\sum_D \sum_C \psi_D(D,C) \sum_H \sum_L \sum_S \psi_J(J,L,S) \underbrace{\sum_I \psi_I(I)\psi_S(S,I)\tau_1(I,D,L,J,H)}_{\tau_2(D,L,S,J,H)}$$

$$\sum_D \sum_C \psi_D(D,C) \sum_H \sum_L \underbrace{\sum_S \psi_J(J,L,S)\tau_2(D,L,S,J,H)}_{\tau_3(D,L,J,H)}$$

$$\sum_D \sum_C \psi_D(D,C) \sum_H \underbrace{\sum_L \tau_3(D,L,J,H)}_{\tau_4(D,J,H)}$$

$$\sum_D \sum_C \psi_D(D,C) \underbrace{\sum_H \tau_4(D,J,H)}_{\tau_5(D,J)}$$

$$\sum_D \underbrace{\sum_C \psi_D(D,C)\tau_5(D,J)}_{\tau_6(D,J)}$$

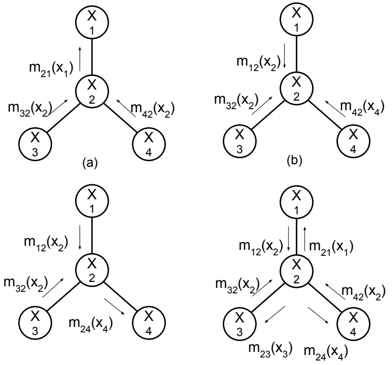
$$\underbrace{\sum_D \tau_6(D,J)}_{\tau_7(J)}$$

Outline

A weakness of VE

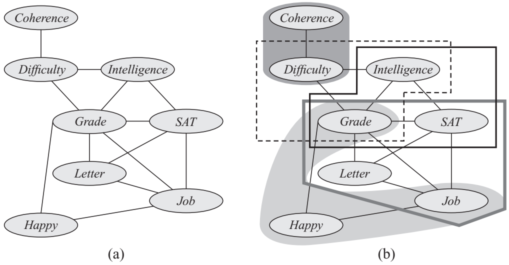
- Consider computing all the marginals in a chain-structured graphical model such as an HMM.
- We can easily compute the final marginal $p(x_T|\mathbf{v})$ by eliminating all the nodes x_1 to x_{T-1} in order.
- This is equivalent to the forwards algorithm, and takes $O(K^2 T)$ time.
- But now suppose we want to compute $p(x_{T-1}|\mathbf{v})$. We have to run VE again, at a cost of $O(K^2 T)$ time. So the total cost to compute all the marginals is $O(K^2 T^2)$.
- However, we know that we can solve this problem in $O(K^2 T)$ using forwards-backwards.
- The difference is that FB caches the messages computed on the forwards pass, so it can reuse them later.

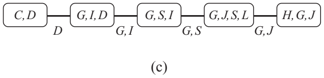
BP is more efficient



- VE on a tree is like sending messages to a single root.
- If we re-run it, we are repeating work.
- BP is more efficient.

Junction trees





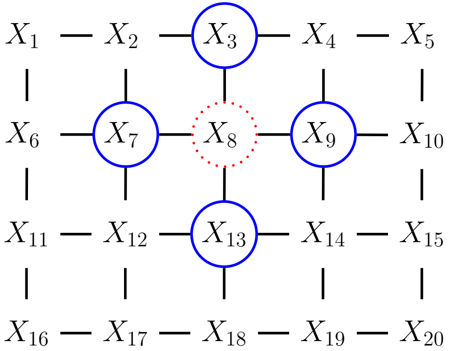
- A junction tree is a tree created from the max cliques of the chordal graph (using some elimination order).
- A jtree enjoys the **running intersection property** (RIP), which means that any subset of nodes containing a given variable forms a connected component.

Message passing on a junction tree

- BP on a junction tree will compute the exact values of $p(\mathbf{x}_c|\mathbf{v})$ for each node c in the tree (i.e., clique in the induced graph).
- From this, we can easily extract the node and edge marginals, $p(x_t|\mathbf{v})$ and $p(x_s, x_t|\mathbf{v})$ from the original model.
- The cost is $O(|\mathcal{C}|K^{w+1})$, where w is the treewidth.
- This generalizes forwards-backwards / BP to arbitrary graphs.
- Can also compute MAP estimate and posterior samples.

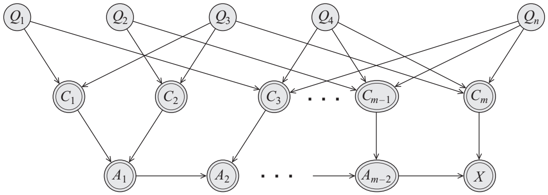
Outline

Treewidth can be large



- The treewidth for an $n \times n$ grid is $O(n)$ (Arnborg87).
- For binary nodes in an 1000×1000 image, exact inference would take $O(2^{1000})$ time!

Inference is computationally hard



- Inference using JTA is exponential in treewidth, but maybe there are better algorithms?
- Unlikely since exact inference is #P-hard (reduction to 3-SAT).
- In fact even approximate inference is hard.

Stochasticity to the rescue

- The hardness results focus on structural (graphical) properties of the model.
- Often the edge strengths are weak, so the components are only loosely dependent.
- We can exploit this for fast approximate inference.

Tutorial on Probabilistic Graphical Models 4: Variational inference

Kevin P. Murphy
kpmurphy@google.com

July 2012

Outline

Variational inference

- "A variational principle is a scientific principle ... which develops general methods for finding functions which minimize or maximize the value of quantities that depend upon those functions" — Wikipedia.
- Here, the goal is to pick a function from some family, $q \in \mathcal{Q}$, so as to make q as close as possible to the target distribution p^*.
- Often p^* is a posterior, $p^*(\mathbf{x}) = p(\mathbf{x}|\mathcal{D})$.
- We will measure "closeness" using KL divergence:

$$\mathbb{KL}\left(p^* || q\right) = \sum_{\mathbf{x}} p^*(\mathbf{x}) \log \frac{p^*(\mathbf{x})}{q(\mathbf{x})}$$

Unnormalized posterior

- Recall the objective

$$\mathbb{KL}\left(p^* || q\right) = \sum_{\mathbf{x}} p^*(\mathbf{x}) \log \frac{p^*(\mathbf{x})}{q(\mathbf{x})}$$

- This is intractable to compute, since evaluating $p^*(\mathbf{x}) = p(\mathbf{x}|\mathcal{D})$ requires computing $Z = p(\mathcal{D})$.
- However the unnormalized distribution is tractable

$$\tilde{p}(\mathbf{x}) \triangleq p(\mathbf{x}, \mathcal{D}) = p^*(\mathbf{x}) Z$$

- So we define our objective as

$$J(q) \quad \triangleq \quad \mathbb{KL}\left(q || \tilde{p}\right)$$

where we are slightly abusing notation, since $\tilde{p}$ is not a normalized distribution.

Rewriting the objective

- Plugging in the definition of KL, we get

$$\begin{aligned} J(q) &= \sum_{\mathbf{x}} q(\mathbf{x}) \log \frac{q(\mathbf{x})}{\tilde{p}(\mathbf{x})} \\ &= \sum_{\mathbf{x}} q(\mathbf{x}) \log \frac{q(\mathbf{x})}{Z p^*(\mathbf{x})} \\ &= \sum_{\mathbf{x}} q(\mathbf{x}) \log \frac{q(\mathbf{x})}{p^*(\mathbf{x})} - \log Z \\ &= \mathbb{KL}\left(q \| p^*\right) - \log Z \end{aligned}$$

  Since Z is a constant, by minimizing $J(q)$, we will force q to become close to p^*.
- Since KL ≥ 0, we are minimizing an upper bound on NLL:

$$J(q) = \mathbb{KL}\left(q \| p^*\right) - \log Z \geq -\log Z = -\log p(\mathcal{D})$$

- Equivalently we are maximizing a lower bound of LL:

$$L(q) \triangleq -J(q) = -\mathbb{KL}\left(q \| p^*\right) + \log Z \leq \log Z = \log p(\mathcal{D})$$

Variational free energy

- Let $E(\mathbf{x}) = -\log \tilde{p}(\mathbf{x})$ be the energy.
- We define the variational free energy or Helmholtz free energy as

$$\begin{aligned} J(q) &= \mathbb{E}_q \left[\log q(\mathbf{x})\right] + \mathbb{E}_q \left[-\log \tilde{p}(\mathbf{x})\right] \\ &= -\mathbb{H}\left(q\right) + \mathbb{E}_q \left[E(\mathbf{x})\right] \end{aligned}$$

- We are minimizing the expected energy while maximizing the entropy.

Outline

The mean field method

- In mean field, we assume the posterior is a fully factorized approximation of the form

$$q(\mathbf{x}) = \prod_{i=1}^{D} q_i(\mathbf{x}_i)$$

- Our goal is to solve this optimization problem:

$$\min_{q_1,\ldots,q_D} \mathbb{KL}\left(q||p\right)$$

  where we optimize over the parameters of each marginal distribution q_i.

Coordinate descent

- If we use coordinate descent method for solving this problem, each update has the form

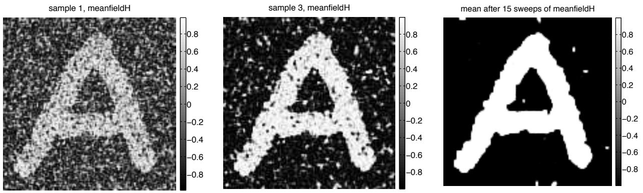
$$\log q_j(\mathbf{x}_j) = \mathbb{E}_{-q_j}\left[\log \tilde{p}(\mathbf{x})\right] + \mathrm{const}$$

  where $\mathbb{E}_{-q_j}[f(\mathbf{x})]$ means to take the expectation over $f(\mathbf{x})$ with respect to all the variables except for x_j eg

$$\mathbb{E}_{-q_2}[f(\mathbf{x})] = \sum_{x_1}\sum_{x_3} q(x_1)q_3(x_3)f(x_1, x_2, x_3)$$

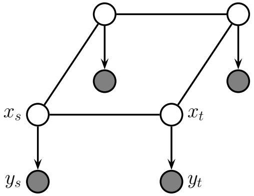
- Here we only need to incude terms on the RHS which are part of x_j's Markov blanket.
- This is a deterministic version of Gibbs sampling.
- Other optimization algorithms can also be used, but are less popular.

Mean field for the Ising model



- Suppose $x_i \in \{-1, +1\}$ are the hidden pixel values of the "clean" image. We have a joint model of the form

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y}|\mathbf{x})$$



Mean field for the Ising model

- The prior has the form

$$
\begin{aligned}
p(\mathbf{x}) &= \frac{1}{Z_0} \exp(-E_0(\mathbf{x})) \\
E_0(\mathbf{x}) &= -\sum_{i=1}^{D} \sum_{j \in \mathrm{nbr}_i} W_{ij} x_i x_j
\end{aligned}
$$

  and the likelihood has the form

$$
p(\mathbf{y}|\mathbf{x}) = \prod_i p(\mathbf{y}_i|x_i) = \sum_i \exp(-L_i(x_i))
$$

  Therefore the posterior has the form

$$
\begin{aligned}
p(\mathbf{x}|\mathbf{y}) &= \frac{1}{Z} \exp(-E(\mathbf{x})) \\
E(\mathbf{x}) &= E_0(\mathbf{x}) - \sum_i L_i(x_i)
\end{aligned}
$$

Mean field updates

- Let us assume
$$q(\mathbf{x}) = \prod_i q(x_i, \mu_i)$$
where μ_i is the mean value of node i.
- We have
$$\log \tilde{p}(\mathbf{x}) = x_i \sum_{j \in \mathrm{nbr}_i} W_{ij} x_j + L_i(x_i) + \mathrm{const}$$
This only depends on the states of the neighboring nodes.
- Now we take expectations of this wrt $\prod_{j \neq i} q_j(x_j)$ to get
$$q_i(x_i) \propto \exp\left(x_i \sum_{j \in \mathrm{nbr}_i} W_{ij} \mu_j + L_i(x_i) \right)$$
Thus we replace the states of the neighbors by their average values.

Mean field updates

- Let
$$m_i = \sum_{j \in \mathrm{nbr}_i} W_{ij} \mu_j$$
be the mean field influence on node i.
- Also, let $L_i^+ \triangleq L_i(+1)$ and $L_i^- \triangleq L_i(-1)$.
- We have
$$\begin{aligned} q_i(x_i = 1) &= \frac{e^{m_i + L_i^+}}{e^{m_i + L_i^+} + e^{-m_i + L_i^-}} = \frac{1}{1 + e^{-2m_i + L_i^- - L_i^+}} = \mathrm{sigm}(2a_i \\ a_i &\triangleq m_i + 0.5(L_i^+ - L_i^-) \end{aligned}$$
- Hence
$$\mu_i = \mathbb{E}_{q_i}[x_i] = \tanh(a_i)$$
- Damped udpate for $0 < \lambda < 1$
$$\mu_i^t = (1-\lambda)\mu_i^{t-1} + \lambda \tanh\left(\sum_{j \in \mathrm{nbr}_i} W_{ij} \mu_j^{t-1} + 0.5(L_i^+ - L_i^-) \right)$$

Variational Bayes

- VB means applying the mean field method to parameter inference, e.g., computing

$$p(\theta_1, \theta_2 | \mathcal{D}) \approx q(\theta_1) q(\theta_2)$$

- Example application: Bayesian alternative to cross validation for linear models:

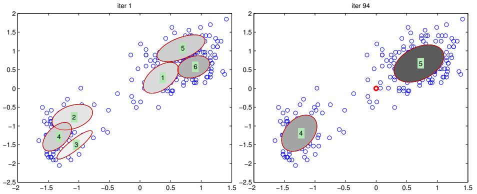
$$\boldsymbol{\alpha} \rightarrow \mathbf{w} \rightarrow \mathcal{D}$$

Variational Bayes EM

- If we have missing data, and we make the factorization

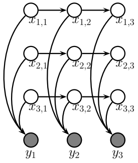
$$p(\mathbf{h}, \boldsymbol{\theta}|\mathcal{D}) \approx q(\mathbf{h})q(\boldsymbol{\theta})$$

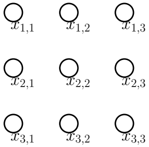
  we can alternate between updating $q(\mathbf{h})$ and $q(\boldsymbol{\theta})$ — this is known as variational Bayes EM.
- Example application: fitting GMM and automatically inferring $p(K|\mathcal{D})$ by "killing off" unnecessary mixture components

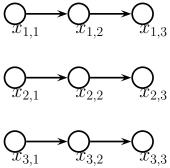


Structured mean field

- Exploit tractable substructure by factorizing into a product of chains







Outline

Exponential family

- The exponential family captures many commonly used distributions, e.g., Gaussian, multinomial, Poisson

$$p(\mathbf{x}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} h(\mathbf{x}) \exp[\boldsymbol{\theta}^T \phi(\mathbf{x})]$$

where

$$Z(\boldsymbol{\theta}) = \int_{\mathcal{X}^m} h(\mathbf{x}) \exp[\boldsymbol{\theta}^T \phi(\mathbf{x})] d\mathbf{x}$$

Exponential family for UGMs

- Pairwise MRF/ CRF

$$p(\mathbf{x}|\boldsymbol{\theta}, G) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left\{ \sum_{s \in \mathcal{V}} \theta_s(x_s) + \sum_{(s,t) \in \mathcal{E}} \theta_{st}(x_s, x_t) \right\}$$

- In expfam form

$$\begin{aligned} p(\mathbf{x}|\boldsymbol{\theta}) &= \frac{1}{Z(\boldsymbol{\theta})} \exp(-E(\mathbf{x})) \\ E(\mathbf{x}) &\triangleq -\boldsymbol{\theta}^T \phi(\mathbf{x}) \end{aligned}$$

where $\boldsymbol{\theta} = (\{\theta_{s;j}\}, \{\theta_{s,t;j,k}\})$ are all the node and edge parameters (the canonical parameters), and $\phi(\mathbf{x}) = (\{\mathbb{I}(x_s = j)\}, \{\mathbb{I}(x_s = j, x_t = k)\})$ are all the node and edge indicator functions (the sufficient statistics). Note: we use $s, t \in \mathcal{V}$ to index nodes and $j, k \in \mathcal{X}$ to index states.

The marginal polytope

- The mean of the sufficient statistics are known as the mean parameters of the model, and are given by

$$\begin{aligned}\boldsymbol{\mu} &= \mathbb{E}[\phi(\mathbf{x})] = (\{p(x_s = j)\}_s, \{p(x_s = j, x_t = k)\}_{s \neq t}) \\ &= (\{\mu_{s;j}\}_s, \{\mu_{st;jk}\}_{s \neq t})\end{aligned}$$

  This is a vector of length $d = |\mathcal{X}||V| + |\mathcal{X}|^2|E|$, containing the node and edge marginals.

- The space of allowable $\boldsymbol{\mu}$ vectors is called the **marginal polytope**, and is denoted $\mathbb{M}(G)$, where G is the structure of the graph defining the UGM. This is defined to be the set of all mean parameters for the given model that can be generated from a valid probability distribution:

$$\mathbb{M}(G) \triangleq \{\boldsymbol{\mu} \in \mathbb{R}^d : \exists p \quad \text{s.t.} \quad \boldsymbol{\mu} = \sum_{\mathbf{x}} \phi(\mathbf{x}) p(\mathbf{x}) \text{ for some } p(\mathbf{x}) \geq 0, \sum$$

The marginal polytope for 2 node Ising model

- Consider an Ising model $X_1 - X_2$. One can show that we have the following minimal set of constraints: $0 \leq \mu_{12}$, $0 \leq \mu_{12} \leq \mu_1$, $0 \leq \mu_{12} \leq \mu_2$, and $1 + \mu_{12} - \mu_1 - \mu_2 \geq 0$.
- Convex hull of the feature set

$$\begin{aligned} \mathbb{M}(G) &= \operatorname{conv}\{\phi_1(\mathbf{x}), \ldots, \phi_d(\mathbf{x})\} \\ \mathbb{M}(G) &= \operatorname{conv}\{(0,0,0), (1,0,0), (0,1,0), (1,1,1)\} \end{aligned}$$

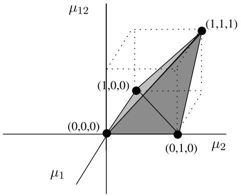


Figure: Michael Jordan

Outline

Exact inference as a variational problem

- We want to maximize the **energy functional**

$$L(q) = -\mathbb{KL}\left(q||p\right) + \log Z = \mathbb{E}_q\left[\log \tilde{p}(\mathbf{x})\right] + \mathbb{H}\left(q\right) \leq \log Z$$

  where $\tilde{p}(\mathbf{x}) = Zp(\mathbf{x})$ is the unnormalized posterior.

- If we write $\log \tilde{p}(\mathbf{x}) = \boldsymbol{\theta}^T \phi(\mathbf{x})$, and we let $q = p$, then the exact energy functional becomes

$$\max_{\boldsymbol{\mu} \in \mathbb{M}(G)} \boldsymbol{\theta}^T \boldsymbol{\mu} + \mathbb{H}\left(\boldsymbol{\mu}\right)$$

  where $\boldsymbol{\mu} = \mathbb{E}_p\left[\phi(\mathbf{x})\right]$ is a joint distribution over all state configurations $\mathbf{x}$ (so it is valid to write $\mathbb{H}\left(\boldsymbol{\mu}\right)$).

- Since the KL divergence is zero when $p = q$, we know that

$$\max_{\boldsymbol{\mu} \in \mathbb{M}(G)} \boldsymbol{\theta}^T \boldsymbol{\mu} + \mathbb{H}\left(\boldsymbol{\mu}\right) = \log Z(\boldsymbol{\theta})$$

- Maximizing concave objective over a convex set.

An inner approximation to the marginal polytope

- The marginal polytope $\mathbb{M}(G)$ has exponentially many facets.
- We will now consider a smaller but non-convex set $\mathbb{M}_F(G) \subseteq \mathbb{M}(G)$.
- To define this set, first consider the canonical parameter space for the submodel as follows:

$$\Omega(F) \triangleq \{\boldsymbol{\theta} \in \Omega : \boldsymbol{\theta}_\alpha = 0 \ \forall \alpha \in \mathcal{I} \setminus \mathcal{I}(F)\}$$

  In other words, we require that the natural parameters associated with the sufficient statistics α outside of our chosen class to be zero.
- For example, in the case of a fully factorized approximation, F_0, we remove all edges from the graph, giving

$$\Omega(F_0) \triangleq \{\boldsymbol{\theta} \in \Omega : \boldsymbol{\theta}_{st} = 0 \ \forall (s,t) \in E\}$$

- Finally define

$$\mathbb{M}_F(G) \triangleq \{\boldsymbol{\mu} \in \mathbb{R}^d : \boldsymbol{\mu} = \mathbb{E}_{\boldsymbol{\theta}}[\phi(\mathbf{x})] \ \text{ for some } \boldsymbol{\theta} \in \Omega(F)\}$$

An inner approximation to the marginal polytope

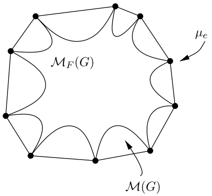


Figure: Michael Jordan

Mean field uses an inner approximation to the marginal polytope

- We define the **mean field energy functional** optimization problem as follows:

$$\max_{\boldsymbol{\mu} \in \mathbb{M}_F(G)} \boldsymbol{\theta}^T \boldsymbol{\mu} + \mathbb{H}(\boldsymbol{\mu}) \leq \log Z(\boldsymbol{\theta})$$

- In the case of the fully factorized mean field approximation for pairwise UGMs, we can write this objective as follows:

$$\max_{\boldsymbol{\mu} \in \mathcal{P}^d} \sum_{s \in V} \sum_{x_s} \theta_s(x_s)\mu_s(x_s) + \sum_{(s,t) \in E} \sum_{x_s, x_t} \theta_{st}(x_s, x_t)\mu_s(x_s)\mu_t(x_t) + \sum_{s \in V} \mathbb{H}(\boldsymbol{\mu}_s)$$

where $\boldsymbol{\mu}_s \in \mathcal{P}$, and $\mathcal{P}$ is the probability simplex over $\mathcal{X}$.

Mean field optimization

- Mean field involves a concave objective being maximized over a non-convex set. It is typically optimized using coordinate ascent, since it is easy to optimize a scalar concave function over $\mathcal{P}$ for each μ_s.
- For example, for a pairwise UGM we get

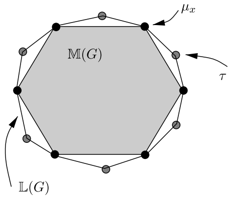
$$\mu_s(x_s) \propto \exp(\theta_s(x_s)) \exp\left(\sum_{t \in \mathrm{nbr}(s)} \sum_{x_t} \mu_t(x_t) \theta_{st}(x_s, x_t) \right)$$

Variational EM

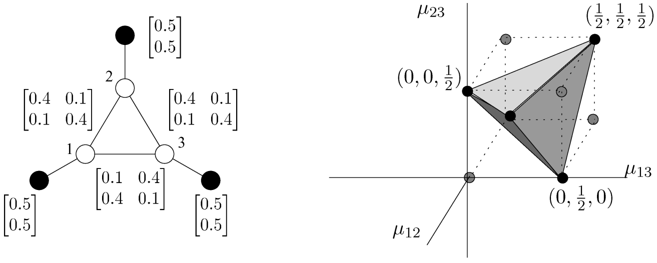
- Because mean field optimizes a lower bound on $\log Z$, it can be used inside EM to yield an approximate E step.
- The overall algorithm is called variational EM and will monotonically converge to a local maximum of the log likelihood (or log posterior).
- Some of the other approximations we will look at later do not optimize lower bounds on $\log Z$, and therefore do not behave as well when combined with parameter learning.

An outer approximation to the marginal polytope

- Suppose we allow $q = \boldsymbol{\tau}$ to come from a larger set $\boldsymbol{\tau} \in \mathbb{L}(G) \supseteq \mathbb{M}(G)$, where $\mathbb{L}(G)$ must only satisfy local consistency
$$\sum_{x_t} \tau_{st}(x_s, x_t) = \tau_s(x_s)$$
and behave locally like a probability distribution
$$0 \leq \tau_s(x_s) \leq 1, \quad \sum_{x_s} \tau_s(x_s) = 1$$
- The set $\mathbb{L}(G)$ is also a polytope, but it only has $O(|V| + |E|)$ constraints. It is a convex **outer approximation** on $\mathbb{M}(G)$.



Pseudo marginals



$\begin{bmatrix}0.5\\0.5\end{bmatrix}$
2
$\begin{bmatrix}0.4 & 0.1\\0.1 & 0.4\end{bmatrix}$
$\begin{bmatrix}0.4 & 0.1\\0.1 & 0.4\end{bmatrix}$
1
3
$\begin{bmatrix}0.1 & 0.4\\0.4 & 0.1\end{bmatrix}$
$\begin{bmatrix}0.5\\0.5\end{bmatrix}$
$\begin{bmatrix}0.5\\0.5\end{bmatrix}$
μ_{23}
$(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$
$(0, 0, \frac{1}{2})$
μ_{13}
$(0, \frac{1}{2}, 0)$
μ_{12}

(a) (b)

- We call the terms $\tau_s, \tau_{st} \in \mathbb{L}(G)$ **pseudo marginals**,
- Not lobally consistent: τ_{12} implies $p(X_1 = X_2) = 0.8$, τ_{23} implies $p(X_2 = X_3) = 0.8$, but τ_{13} implies $p(X_1 = X_3) = 0.2$,
- Fig b shows that $\mathbb{L}(G)$ contains points that are not in $\mathbb{M}(G)$.

Figure: Michael Jordan

Loopy belief propagation (LBP)

- LBP is simply running the BP algorithm on graphs with loops.
- In practice it often works well.
- Surprisingly, it can be shown that this locally minimizes the following Bethe free energy objective

$$\min_{\boldsymbol{\tau} \in \mathbb{L}(G)} F_{\text{Bethe}}(\boldsymbol{\tau}) = \max_{\boldsymbol{\tau} \in \mathbb{L}(G)} \boldsymbol{\theta}^T \boldsymbol{\tau} + \mathbb{H}_{\text{Bethe}}(\boldsymbol{\tau})$$

  where $\mathbb{H}_{\text{Bethe}}(\boldsymbol{\tau})$ is the entropy of $\boldsymbol{\tau}$ assuming the distribution can be represented as a tree:

$$\mathbb{H}_{\text{Bethe}}(\boldsymbol{\tau}) \quad = \quad \sum_{s \in V} H_s(\tau_s) - \sum_{(s,t) \in E} I_{st}(\tau_{st})$$

- The messages that are sent between nodes turn out to be Lagrange multipliers enforcing the pairwise constraints in the optimization.

Tree-reweighted BP (TRW)

- LBP optimizes a non concave objective over a convex set.
- We can define a concave objective (details omitted).
- Tree-reweighted BP (TRW) is a message passing algorithm that will find the global optimum of this function.
- This method is useful as a subroutine for learning algorithms.

Outline

MAP state estimation in UGMs

- Our goal is

$$\begin{aligned}\mathbf{x}^* &= \arg\max_{\mathbf{x}\in\mathcal{X}^m} p(\mathbf{x}|\boldsymbol{\theta}) \\ &= \arg\max_{\mathbf{x}\in\mathcal{X}^m} \sum_{i\in V} \theta_i(x_i) + \sum_{f\in F} \theta_f(\mathbf{x}_f) \\ &= \arg\max_{\mathbf{x}\in\mathcal{X}^m} \boldsymbol{\theta}^T \phi(\mathbf{x})\end{aligned}$$

- We can solve this exactly using dynamic programming (junction tree algorithm).
- But this takes time exponential in treewidth.
- We now discuss faster approximate methods.

Linear programming relaxation

- Exact objective

$$\arg\max_{\mathbf{x}\in\mathcal{X}^m} \boldsymbol{\theta}^T\phi(\mathbf{x}) = \arg\max_{\boldsymbol{\mu}\in\mathbb{M}(G)} \boldsymbol{\theta}^T\boldsymbol{\mu}$$

- Relaxed objective

$$\max_{\mathbf{x}\in\mathcal{X}^m} \boldsymbol{\theta}^T\phi(\mathbf{x}) = \max_{\boldsymbol{\mu}\in\mathbb{M}(G)} \boldsymbol{\theta}^T\boldsymbol{\mu} \leq \max_{\boldsymbol{\tau}\in\mathbb{L}(G)} \boldsymbol{\theta}^T\boldsymbol{\tau}$$

- Max-product version of TRW can be used to solve this (much faster than CPLEX!)

Graphcuts

- An alternative optimization method, that often works faster, is known as graphcuts.
- Suppose initially all nodes are binary and potentials are submodular.
- We say that $f : 2^S \to R$ is submodular if for any $A \subset B \subset S$ and $x \in S$, we have

$$f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B).$$

- This means the energy must satisfy

$$E_{uv}(1,1) + E_{uv}(0,0) \leq E_{uv}(1,0) + E_{uv}(0,1)$$

Attractive / associate MRF.

Maxflow mincut

- We can construct a graph and use max-flow min-cut to find the minimal cost $s - t$ cut (details omitted).
- If x_i is on the s side, then $x_i^* = 0$, else $x_i^* = 1$. This is the global optimum!
- Mincut algorithms take $O(EV \log V)$ time. Their speed depends on the density of the graph, not its treewidth.

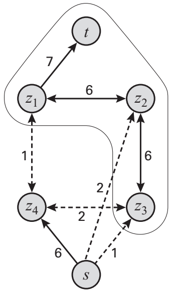


Figure: Daphne Koller

Non-binary case

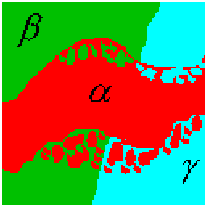
- For non-binary nodes, we require the energy I to be a metric.
- A function f is said to be a **metric** if it satisfies the following three properties: Reflexivity: $f(a, b) = 0$ iff $a = b$; Symmetry: $f(a, b) = f(b, a)$; and Triangle inequality: $f(a, b) + f(b, c) \geq f(a, c)$. If f satisfies only the first two properties, it is called a **semi-metric**.
- If the the states must have a natural ordering (eg. discretization), we can define the following metric potential

$$E(x_s, x_t) = \min(\delta, ||x_s - x_t||)$$

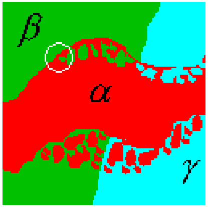
This is widely used in computer vision.

Non-binary case

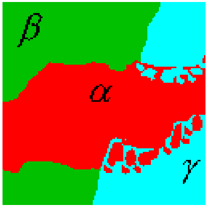
- There are two ways to convert a non-binary problem with metric potentials into a binary problem with submodular potentials: alpha expansion or alpha-beta swap.
- Both produce strong local maxima.



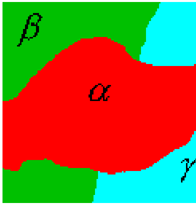
(a) initial labeling



(b) standard move



(c) α-β-swap



(d) α-expansion

Figure: Ramin Zabih

Example: depth from stereo

- Estimate discretized depth of each pixel given two observed images.
- We will use a 2d grid CRF.

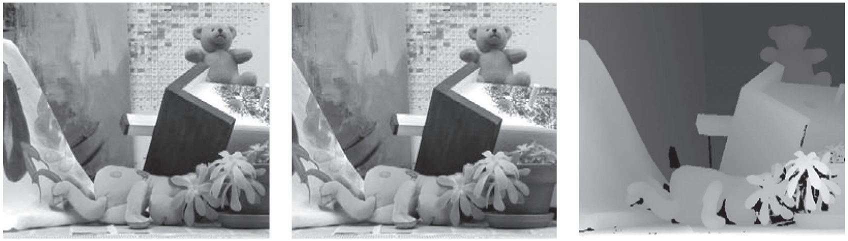


Figure: Daphne Koller

Energy vs time

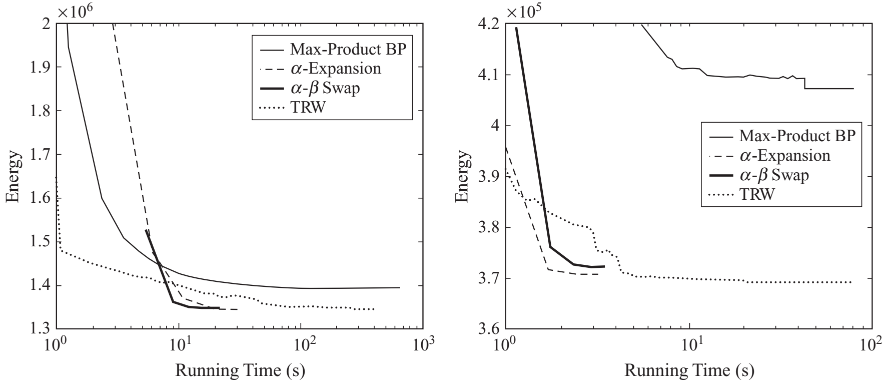


Figure: Daphne Koller

Qualitative comparison

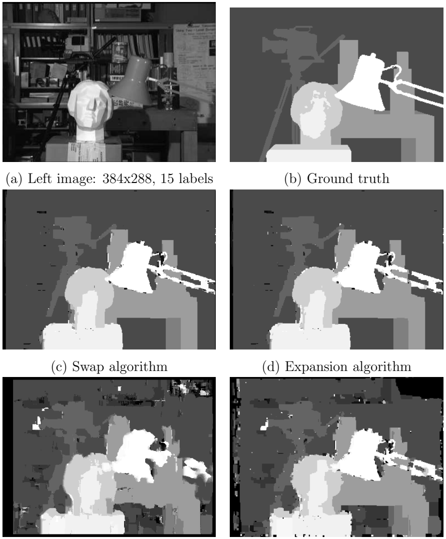

(a) Left image: 384x288, 15 labels
(b) Ground truth
(c) Swap algorithm
(d) Expansion algorithm
(e) Normalized correlation
(f) Simulated annealing

Figure: Ramin Zabih

Example: Image stitching

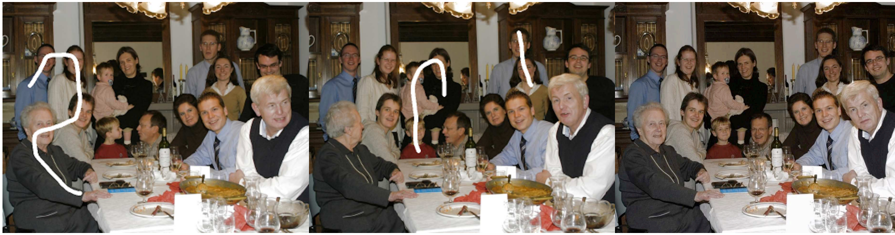


Figure: Rick Szeliski.

Tree-reweighted max product (TRW)

Figure: Rick Szeliski.

Max product BP

Figure: Rick Szeliski.

Alpha beta swap

Figure: Rick Szeliski.

Iterative conditional modes (ICM)

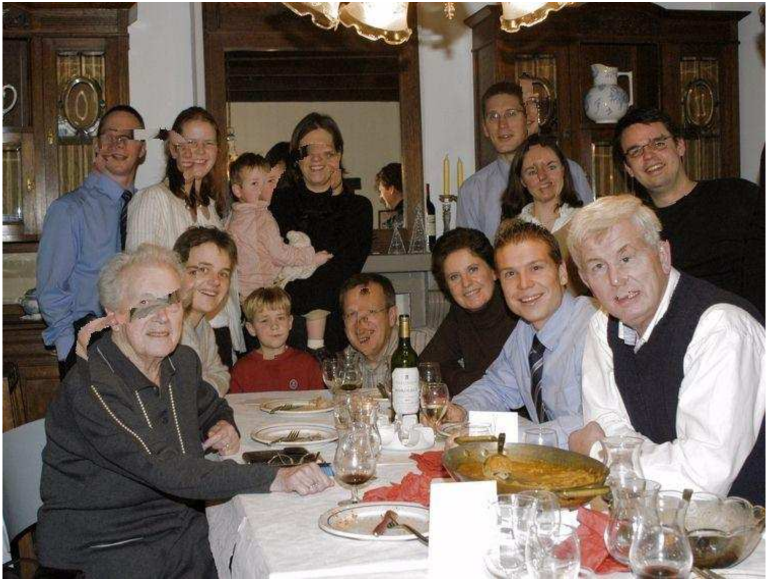
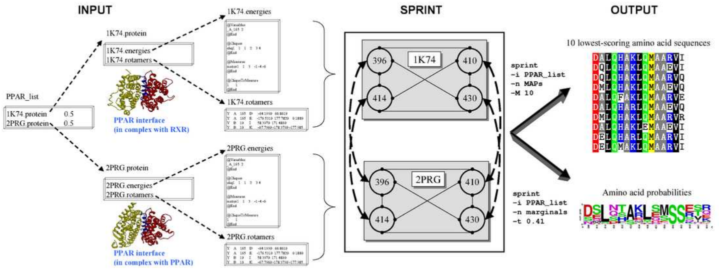


Figure: Rick Szeliski.

Example: protein side chain prediction

MAP estimation equivalent to energy minimization



Menachem Fromer, Chen Yanover, Amir Harel, Ori Shachar, Yair Weiss, and Michal Linial. SPRINT: side-chain prediction inference toolbox for multistate protein design. Bioinformatics, 26(19):2466-2467, 2010.

Outline

Computing MLE for CRFs

- Gradient descent methods for fitting MRFs and CRFs, as well as DGMs with hidden variables, requires computing all node marginals $p(x_s|\mathbf{v})$ and edge marginals $p(x_s, x_t|\mathbf{v})$.
- This can be approximated by sum-product BP.
- Training the parameters of the side-chain prediction model gave much better results that using the standard physics-based parameters (due to model inaccuracies) — Yannover.

Structural SVM training

- Training SSVMs requires MAP state estimation.
- This can be approximated by max-product BP or graphcuts.

Approximate inference at training and test time

- It has been observed experimentally that it is best to use the same approximate method for training as will be used for test time prediction.
- There is also some theory for this, e.g., 'Estimating the "Wrong" Graphical Model: Benefits in the Computation-Limited Setting', Wainwright JMLR 2006

Outline

Conclusions

- Graphical models allow us to easily define high-dimensional probability distributions
- Supports efficient inference
- Supports efficient learning
- Widely used in many applications