Machine Learning

Topic 14

- Structuring Probability Functions for Storage
- Structuring Probability Functions for Inference
- Basic Graphical Models
- Graphical Models
- Parameters as Nodes

Structuring PDFs for Storage

- Probability tables quickly grow if p has many variables

$$p(x) = p(flu?, headache?, \ldots, temperature?)$$

- For D true/false "medical" variables $table\ size = 2^D$
- Exponential blow-up of storage size for the probability
- Example: 8x8 binary images of digits
- If multinomial with M choices, probabilities are how big?
- As in Naïve Bayes or Multivariate Bernoulli, if words were independent things are much more efficient

$$p(x) = p(flu?)\,p(headache?)\ldots p(temperature?)$$

0.73	0.27	0.2	0.8	0.54	0.46

- For D true/false "medical" variables (really even less than that…) $table\ size = 2 \times D$

Structuring PDFs for Inference

- Inference: goal is to predict some variables given others

x1: flu
x2: fever
x3: sinus infection
x4: temperature
x5: sinus swelling
x6: headache

Patient claims headache and high temperature. Does he have a flu?

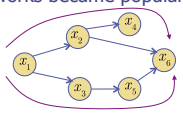
Given findings variables X_f and unknown variables X_u predict queried variables X_q

- Classical approach: truth tables (slow) or logic networks
- Modern approach: probability tables (slow) or Bayesian networks (fast belief propagation, junction tree algorithm)

From Logic Nets to Bayes Nets

- 1980's expert systems & logic networks became popular

x1	x2	x1 v x2	x1^x2	x1 -> x2
T	T	T	T	T
T	F	T	F	F
F	T	T	F	T
F	F	F	F	T



- Problem: inconsistency, 2 paths can give different answers
- Problem: rules are hard, instead use soft probability tables

x3 = x1 ^ x2 p(x3|x1,x2)

x3=0	x2=0	x2=1
x1=0	1.0	1.0
x1=1	1.0	0.0

x3=1	x2=0	x2=1
x1=0	0.0	0.0
x1=1	0.0	1.0

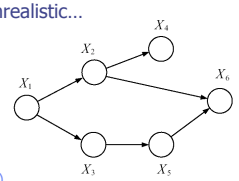
x3=0	x2=0	x2=1
x1=0	0.8	0.7
x1=1	0.7	0.1

x3=1	x2=0	x2=1
x1=0	0.2	0.3
x1=1	0.3	0.9

- These directed graphs are called Bayesian Networks

Graphical Models & Bayes Nets

- Independence assumptions make probability tables smaller
- But real events in the world not completely independent!
- Complete independence is unrealistic…
- Graphical models use a graph to describe more subtle dependencies and independencies:
  …namely: conditional independencies
  (like causality but not exactly…)



- Directed Graphical Model, also called Bayesian Network use a directed acylic graph (DAG).
- Neural Network = Graphical Function Representation
- Bayesian Network = Graphical Probability Representation

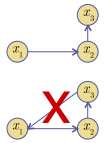
Graphical Models & Bayes Nets

- Node: a random variable (discrete or continuous) 
- Independent: no link  $p(x,y) = p(x)p(y)$
- Dependent: link  $p(x,y) = p(y \mid x)p(x)$
- Arrow: from parent to child (like causality, not exactly)
- Child: destination of arrow, response
- Parent: root of arrow, trigger $parents\ of\ child\ i = pa_i = \pi_i$
- Graph: dependence/independence
- Graph: shows factorization of joint

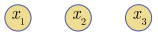
joint = products of conditionals

$$p(x_1,\ldots,x_n) = \prod_{i=1}^{n} p(x_i \mid pa_i) = \prod_{i=1}^{n} p(x_i \mid \pi_i)$$

- DAG: directed acyclic graph



Basic Graphical Models

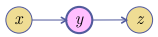
- Independence: all nodes are unlinked 
- Shading: variable is 'observed', condition on it moves to the right of the bar in the pdf 
- Examples of simplest conditional independence situations…

$$p(x_1,\ldots,x_n) = \prod_{i=1}^{n} p(x_i \mid pa_i) = \prod_{i=1}^{n} p(x_i \mid \pi_i)$$

1) Markov chain: 

$$p(x,y,z) = p(x)\,p(y \mid x)\,p(z \mid y)$$

Example binary events:
x = president says war
y = general orders attack
z = soldier shoots gun



$x \perp\!\!\!\perp z \mid y$

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

Basic Graphical Models

2) 1 Cause, 2 effects: $p(x,y,z) = p(y)\,p(x \mid y)\,p(z \mid y)$

y = flu
x = sore throat
z = temperature

$x \perp\!\!\!\perp z \mid y$

3) 2 Causes, 1 effect: $p(x,y,z) = p(x)\,p(z)\,p(y \mid x,z)$

x = rain
y = wet driveway
z = car oil leak

Explaining away…

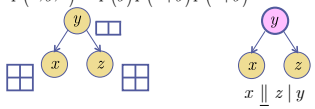
$x \perp\!\!\!\perp z$ $x \not\perp\!\!\!\perp z \mid y$

- Each conditional is a mini-table (Multinomial or Bernoulli conditioned on parents)

Basic Graphical Models

2) 1 Cause, 2 effects: $p(x,y,z) = p(y)p(x \mid y)p(z \mid y)$

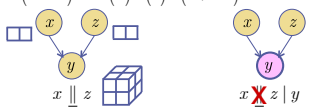
y = flu
x = sore throat
z = temperature



$x \perp z \mid y$

3) 2 Causes, 1 effect: $p(x,y,z) = p(x)p(z)p(y \mid x,z)$

x = dad is diabetic
y = child is diabetic
z = mom is diabetic



$x \perp z$ $\quad x \not\perp z \mid y$

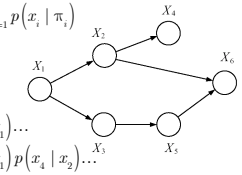
Explaining away...

- Each conditional is a mini-table
  (Multinomial or Bernoulli conditioned on parents)

Graphical Models

- Example: factorization of the following system of variables

$$p(x_1,\ldots,x_n) = \prod_{i=1}^{n} p(x_i \mid pa_i) = \prod_{i=1}^{n} p(x_i \mid \pi_i)$$



$$\begin{aligned} p(x_1,\ldots,x_6) &= p(x_1)\ldots \\ &= p(x_1)p(x_2 \mid x_1)\ldots \\ &= p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)\ldots \\ &= p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)\ldots \\ &= p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)p(x_5 \mid x_3)\ldots \\ &= p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)p(x_5 \mid x_3)p(x_6 \mid x_2,x_5) \end{aligned}$$

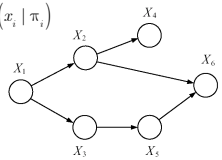
$2^6 \quad 2^1 \quad 2^2 \quad 2^2 \quad 2^2 \quad 2^2 \quad 2^3$

- How big are these tables (if binary variables)?

Graphical Models

- Example: factorization of the following system of variables

$$p(x_1,\ldots,x_n) = \prod_{i=1}^{n} p(x_i \mid pa_i) = \prod_{i=1}^{n} p(x_i \mid \pi_i)$$



- Interpretation:
  1: flu
  2: fever
  3: sinus infection
  4: temperature
  5: sinus swelling
  6: headache

$$p(x_1,\ldots,x_6) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)p(x_5 \mid x_3)p(x_6 \mid x_2,x_5)$$

$2^6 \quad 2^1 \quad 2^2 \quad 2^2 \quad 2^2 \quad 2^2 \quad 2^3$

Graphical Models

- Normalizing probability tables. Joint distributions sum to 1.
- BUT, conditionals sum to 1 for *each* setting of parents.

p(x) **2-1**

$\sum_{x=0}^{1} p(x) = 1$

p(x,y) **4-1**

$\sum_{x,y} p(x,y) = 1$

p(x|y) **4-2**

$\sum_{x} p(x \mid y=0) = 1$
$\sum_{x} p(x \mid y=1) = 1$

p(x,y,z) **8-1**

$\sum_{x,y,z} p(x,y,z) = 1$

p(x|y,z) **8-4**

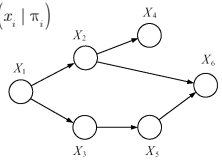
$\sum_{x} p(x \mid y=0, z=0) = 1$
$\sum_{x} p(x \mid y=1, z=0) = 1$
$\sum_{x} p(x \mid y=0, z=1) = 1$
$\sum_{x} p(x \mid y=1, z=1) = 1$

Graphical Models

- Example: factorization of the following system of variables

$$p(x_1,\ldots,x_n) = \prod_{i=1}^{n} p(x_i \mid pa_i) = \prod_{i=1}^{n} p(x_i \mid \pi_i)$$

- Interpretation
  1: flu
  2: fever
  3: sinus infection
  4: temperature
  5: sinus swelling
  6: headache



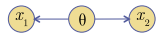
$$p(x_1,\ldots,x_6) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)p(x_5 \mid x_3)p(x_6 \mid x_2,x_5)$$

$2^6-1 \quad 2^1-1 \quad 2^2-2 \quad 2^2-2 \quad 2^2-2 \quad 2^2-2 \quad 2^3-4$

63 vs. 13 degrees of freedom

Parameters as Nodes

- Consider the model variable θ ALSO as a random variable



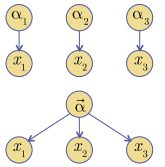
- But would need a prior distribution P(θ)… ignore for now
- Recall: Naïve Bayes, word probabilities are independent

- Text: Multivariate Bernoulli

$$p(x \mid \vec{\alpha}) = \prod_{d=1}^{50000} \alpha_d^{x_d} \left(1-\alpha_d\right)^{\left(1-x_d\right)}$$

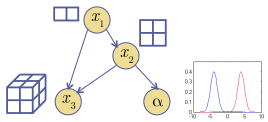
- Text: Multinomial

$$p(X \mid \vec{\alpha}) = \frac{\left\{\sum_{m=1}^{M} X_m\right\}!}{\prod_{m=1}^{M} X_m!} \prod_{m=1}^{M} \alpha_m^{X_m}$$



Continuous Conditional Models

- In previous slide, θ and α were a random variable in graph
- But, θ and α are continuous
- Network can have both discrete & continuous nodes
- Joint factorizes into conditionals that are either:
  1) discrete conditional probability tables
  2) continuous conditional probability distributions

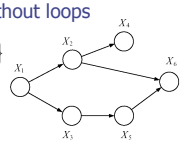


- Most popular continuous distribution = Gaussian

Graphical Models

- In EM, we saw how to handle nodes that are: observed (shaded), hidden variables (E), parameters (M)
- But, only considered simple iid, single parent, structures
- More generally, have arbitrary DAG without loops
- Notation:

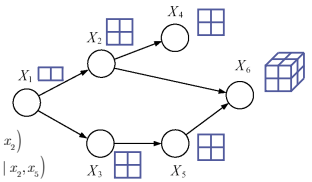
$$G = \{X,E\} = \{\texttt{nodes/randomvars,edges}\}$$
$$X = \{x_1,\ldots,x_M\}$$
$$E = \{(x_i,x_j) : i \neq j\}$$
$$X_s = \{x_1,x_3,x_4\} = subset$$



- Want to do 4 things with these graphical models:
  1) Learn Parameters (to fit to data)
  2) Query independence/dependence
  3) Perform Inference (get marginals/max a posteriori)
  4) Compute Likelihood (e.g. for classification)

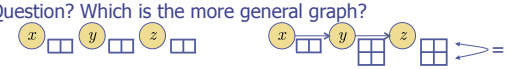
Graphical Models

- Graph factorizes probability: $p(x_1,\ldots,x_n) = \prod_{i=1}^{n} p(x_i \mid \pi_i)$
- Topological graph: nodes are in order so that parents π come before children



$$p(x_1,\ldots,x_6) = p(x_1)p(x_2 \mid x_1) \times p(x_3 \mid x_1)p(x_4 \mid x_2) \times p(x_5 \mid x_3)p(x_6 \mid x_2,x_5)$$

- Question? Which is the more general graph?



- Conditional probability tables can be chosen to make 'busier' graph look like simpler graph

Machine Learning

Topic 15

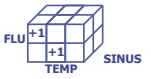
- Graphical Models
- Maximum Likelihood for Graphical Models
- Testing for Conditional Independence & D-Separation
- Bayes Ball

Learning Fully Observed Models

- Easiest scenario: we have observed all the nodes
- Want to learn the probability tables from data…
- Have N iid patients:

PATIENT	FLU	FEVER	SINUS	TEMP	SWELL	HEAD
1	Y	Y	N	L	Y	Y
2	N	N	N	M	N	Y
3	Y	N	Y	H	Y	N
4	Y	N	Y	M	N	N

- Simplest case: least general graph 
  handle each dim individually as Bernoulli/Multinomial
- 2nd Simplest case: most general, count each entry in pdf

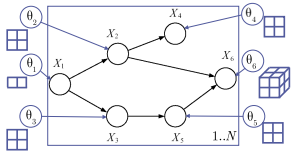
Divide by total count
Since $\sum_{x_1}\cdots\sum_{x_6} p(x) = 1$

- What about learning graphs in between?

Maximum Likelihood CPTs

- Each conditional probability table θ_i part of our parameters
- Given table, have pdf $p(X_U \mid \theta) = \prod_{i=1}^{M} p(x_i \mid \pi_i, \theta_i)$
- Have M variables: $X_U = \{x_1, \ldots, x_M\}$
- Have N x M dataset: $\mathcal{D} = \{X_{U,1}, \ldots, X_{U,N}\}$
- Maximum likelihood:

$$\begin{aligned}\theta^* &= \arg\max_\theta \log p(\mathcal{D} \mid \theta) \\ &= \arg\max_\theta \sum_{n=1}^{N} \log p(X_{U,n} \mid \theta) \\ &= \arg\max_\theta \sum_{n=1}^{N} \log \prod_{i=1}^{M} p(x_{i,n} \mid \pi_{i,n}, \theta_i) \\ &= \arg\max_\theta \sum_{n=1}^{N} \sum_{i=1}^{M} \log p(x_{i,n} \mid \pi_{i,n}, \theta_i)\end{aligned}$$



each θ_i appears independently, can do ML for each CPT alone! efficient storage & efficient learning

Maximum Likelihood CPTs

$$\delta(X_{U,n}, X_{U,m}) = \begin{cases} 1 & \text{if } X_{U,n} = X_{U,m} \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} m(x_i) &= \sum_{n=1}^{N} \delta(x_i, x_{i,n}) \\ m(X_U) &= \sum_{n=1}^{N} \delta(X_U, X_{U,n}) \\ m(X_C) &= \sum_{X_{U \setminus C}} m(X_U) \end{aligned}$$

Counts: # of times what's in the bracket appeared in data, for example:

$$N = \sum_{x_1} m(x_1) = \sum_{x_1}\left(\sum_{x_2} m(x_1, x_2)\right) = \sum_{x_1}\left(\sum_{x_2}\left(\sum_{x_3} m(x_1, x_2, x_3)\right)\right)$$

- So…

$$\begin{aligned} l(\theta) &= \sum_{n=1}^{N} \log p(X_{U,n} \mid \theta) \\ &= \sum_{n=1}^{N} \log \prod_{X_U} p(X_U \mid \theta)^{\delta(X_U, X_{U,n})} \\ &= \sum_{n=1}^{N} \sum_{X_U} \delta(X_U, X_{U,n}) \log p(X_U \mid \theta) \\ &= \sum_{X_U} m(X_U) \log p(X_U \mid \theta) = \sum_{X_U} m(X_U) \log \prod_{i=1}^{M} p(x_i \mid \pi_i, \theta_i) \\ &= \sum_{X_U} \sum_{i=1}^{M} m(X_U) \log p(x_i \mid \pi_i, \theta_i) \end{aligned}$$

Maximum Likelihood CPTs

- Continuing:

$$\begin{aligned} l(\theta) &= \sum_{X_U} \sum_{i=1}^{M} m(X_U) \log p(x_i \mid \pi_i, \theta_i) \\ &= \sum_{i=1}^{M} \sum_{x_i, \pi_i} \sum_{X_{U \setminus x_i, \pi_i}} m(X_U) \log p(x_i \mid \pi_i, \theta_i) \\ &= \sum_{i=1}^{M} \sum_{x_i, \pi_i} m(x_i, \pi_i) \log p(x_i \mid \pi_i, \theta_i) \end{aligned}$$

- Define: $\theta(x_i, \pi_i) = p(x_i \mid \pi_i, \theta_i)$ Constraint: $\sum_{x_i} \theta(x_i, \pi_i) = 1$
- Now have above with Lagrange multipliers:

$$l(\theta) = \sum_{i=1}^{M} \sum_{x_i} \sum_{\pi_i} m(x_i, \pi_i) \log \theta(x_i, \pi_i) - \sum_{i=1}^{M} \sum_{\pi_i} \lambda_{\pi_i} \left(\sum_{x_i} \theta(x_i, \pi_i) - 1\right)$$

$$\frac{\partial l(\theta)}{\partial \theta(x_i, \pi_i)} = \frac{m(x_i, \pi_i)}{\theta(x_i, \pi_i)} - \lambda_{\pi_i} = 0 \;\rightarrow\; \theta(x_i, \pi_i) = \frac{m(x_i, \pi_i)}{\lambda_{\pi_i}}$$

- Plug constraint: $\sum_{x_i} \frac{m(x_i, \pi_i)}{\lambda_{\pi_i}} = 1 \;\rightarrow\; \lambda_{\pi_i} = \sum_{x_i} m(x_i, \pi_i) = m(\pi_i)$
- Final solution (trivial!): $\theta(x_i, \pi_i) = \frac{m(x_i, \pi_i)}{m(\pi_i)}$

Maximum Likelihood CPTs

- Continuing:

$$\begin{aligned} l(\theta) &= \sum_{X_U} \sum_{i=1}^{M} m(X_U) \log p(x_i \mid \pi_i, \theta_i) \\ &= \sum_{i=1}^{M} \sum_{x_i, \pi_i} \sum_{X_{U \setminus x_i, \pi_i}} m(X_U) \log p(x_i \mid \pi_i, \theta_i) \\ &= \sum_{i=1}^{M} \sum_{x_i, \pi_i} m(x_i, \pi_i) \log p(x_i \mid \pi_i, \theta_i) \end{aligned}$$

- Define: $\theta(x_i, \pi_i) = p(x_i \mid \pi_i, \theta_i)$ Constraint: $\sum_{x_i} \theta(x_i, \pi_i) = 1$
- Now have above with Lagrange multipliers:

$$l(\theta) = \sum_{i=1}^{M} \sum_{x_i} \sum_{\pi_i} m(x_i, \pi_i) \log \theta(x_i, \pi_i) - \sum_{i=1}^{M} \sum_{\pi_i} \lambda_{\pi_i} \left(\sum_{x_i} \theta(x_i, \pi_i) - 1\right)$$

$$\frac{\partial l(\theta)}{\partial \theta(x_i, \pi_i)} = \frac{m(x_i, \pi_i)}{\theta(x_i, \pi_i)} - \lambda_{\pi_i} = 0 \;\rightarrow\; \theta(x_i, \pi_i) = \frac{m(x_i, \pi_i)}{\lambda_{\pi_i}}$$

- Plug constraint: $\sum_{x_i} \frac{m(x_i, \pi_i)}{\lambda_{\pi_i}} = 1 \;\rightarrow\; \lambda_{\pi_i} = \sum_{x_i} m(x_i, \pi_i) = m(\pi_i)$
- Final solution (trivial!): $\theta(x_i, \pi_i) = \frac{m(x_i, \pi_i) + \varepsilon}{m(\pi_i) + \varepsilon |x_i|}$ **MAP VERSION**

Maximum Likelihood CPTs

- Let's try an example:
- Compute the cpt $p(x_3 \mid x_1)$

PATIENT	FLU	FEVER	SINUS	TEMP	SWELL	HEAD
1	Y	Y	N	L	Y	Y
2	N	N	N	M	N	Y
3	Y	N	Y	H	Y	N
4	Y	N	Y	M	N	N

- Using the formula: $\theta(x_i, \pi_i) = \frac{m(x_i, \pi_i)}{m(\pi_i)}$ **Note, here 0/0 = prior constant**

x_3 \ x_1	1	0
1	1	2
0		

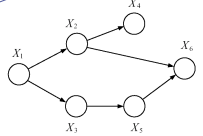
$m(x_3, x_1)$

1	3

$m(x_1)$

1	1/3
0	2/3

$p(x_3 \mid x_1)$



Efficient, only count over subset of variables in $p(X_B \mid X_A)$ Not all $p(x_1, \ldots, x_M)$

Conditional Dependence Tests

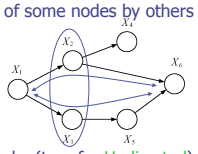
- Another thing we would like to do with a graphical model: Check conditional independencies…
  "Is Temperature Indep. of Flu Given Fever?"
  "Is Temperature Indep. of Sinus Infection Given Fever?"
- Try computing & simplify marginals of p(x)

$$\begin{aligned} p(X) &= p(x_1) p(x_2 \mid x_1) p(x_3 \mid x_1) p(x_4 \mid x_2) p(x_5 \mid x_3) p(x_6 \mid x_2, x_5) \\ p(x_4 \mid x_1, x_2, x_3) &= \frac{p(x_1, x_2, x_3, x_4)}{p(x_1, x_2, x_3)} = \frac{\sum_{x_5}\sum_{x_6} p(X)}{\sum_{x_4}\sum_{x_5}\sum_{x_6} p(X)} \\ &= \frac{p(x_1) p(x_2 \mid x_1) p(x_3 \mid x_1) p(x_4 \mid x_2)}{p(x_1) p(x_2 \mid x_1) p(x_3 \mid x_1)} \\ &= p(x_4 \mid x_2) \end{aligned} \quad \longrightarrow \quad x_4 \perp\!\!\!\perp x_1, x_3 \mid x_2$$

- In this case it was easy, what if checking: $x_1 \perp\!\!\!\perp x_6 \mid x_2, x_3$
- Hard to compute $p(x_1 \mid x_2, x_3, x_6)$ want efficient algorithm…

D-Separation & Bayes Ball

- There is a graph algorithm for checking independence
- Intuition: separation or blocking of some nodes by others
- Example:



if nodes x_2, x_3 "block" path from x_1 to x_6 we might say that

$$x_1 \perp\!\!\!\perp x_6 \mid x_2, x_3$$

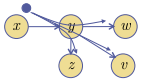
- This is not exact for directed graphs (true for Undirected)
- We need more than just simple Separation
- Need D-Separation (directed separation)
- D-Separation is computed via the Bayes Ball algorithm
- Use to prove general statements over subsets of vars:

$$X_A \perp\!\!\!\perp X_B \mid X_C$$

Bayes Ball Algorithm

- The algorithm: $X_A \perp\!\!\!\perp X_B \mid X_C$
  1) Shade nodes X_C
  2) Place a ball at each node in X_A
  3) Bounce balls around graph according to some *rules*
  4) If no balls reach X_B, then $X_A \perp\!\!\!\perp X_B \mid X_C$ is true (else false)

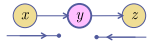
Balls can travel along/against arrows
Pick any incoming & outgoing path
Test each to see if ball goes through or bounces back



Look at canonical sub-graphs & leaf cases for rules…

Bayes Ball Algorithm

1) Markov Chain:



Rule depends only on shading of middle node

Ball stops $x \perp\!\!\!\perp z \mid y$

Go Through $x \not\perp\!\!\!\perp z$

2) Two Effects:

Rule depends only on shading of middle node

Ball stops $x \perp\!\!\!\perp z \mid y$

Go Through $x \not\perp\!\!\!\perp z$

3) Two Causes (V):

Rule depends only on shading of middle node

Go Through $x \not\perp\!\!\!\perp z \mid y$

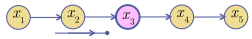
Ball stops $x \perp\!\!\!\perp z$

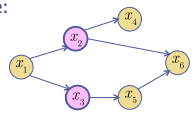
Bayes Ball Algorithm

- Also need to look at special 'leaf' cases:

Bounces back   Ball is stopped

Ball is stopped   Bounces back

- Example:  $x_1 \perp\!\!\!\perp x_5 \mid x_3$
- Example:



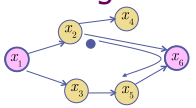
X_1, X_2, X_4 Stopped
X_1, X_2, X_6 Stopped
X_1, X_3, X_5 Stopped

$\therefore\ x_1 \perp\!\!\!\perp x_6 \mid \{x_2, x_3\}$

Flu is independent of headache given fever & sinus infection!

Bayes Ball Algorithm

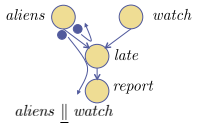
- Example:



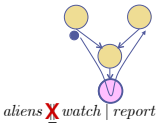
X_2, X_6, X_5 Goes Through Because of V-structure

$\therefore\ x_2 \not\perp\!\!\!\perp x_3 \mid \{x_1, x_6\}$

- Example:



$aliens \perp\!\!\!\perp watch$



$aliens \not\perp\!\!\!\perp watch \mid report$

Ball bounces back from report leaf and goes to right if report is shaded. Bob is waiting for Alice but can't know if she is late. Instead a security guard says if she is. She can be late if aliens abduct her or Bob's watch is ahead (daylight savings time). Guard reports she is late. If watch is ahead, p(alien=true) goes down, they are dependent.

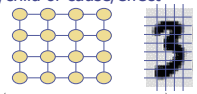
Machine Learning

Topic 16

- Undirected Graphs
- Undirected Separation
- Inferring Marginals & Conditionals
- Moralization
- Junction Trees
- Triangulation

Undirected Graphs

- Separation is *much easier* for undirected graphs
- But, what are undirected graphs and why use them?
- Might be hard to call vars parent/child or cause/effect
- Example: Image pixels
- Each pixel is Bernoulli = {0,1}
- Where 0=dark, 1=bright

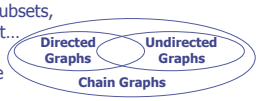


- Have probability over all pixels $p\left(x_{11},\ldots,x_{1M},\ldots,x_{M1},\ldots,x_{MM}\right)$
- Bright pixels have Bright neighbors
- Nearby pixels dependent, so connect with links
- Get a graphical model that looks like a grid
- But who is parent? No parents really, just probability
- Grid models are called Markov Random Fields
- Used in vision, physics (lattice, spin, or Ising models), etc.

Tony Jebara, Columbia University

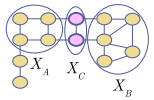
Undirected Graphs

- Undirected & directed not subsets,
- Chain Graphs are a superset…
- Some distributions behave as undirected graphs, some as directed, some as both

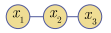


- Undirected graphs use the standard definition of separation:

an undirected graph says that $p\left(x_1,\ldots,x_M\right)$ satisfies any statement $X_A \perp\!\!\!\perp X_B \mid X_C$ if no paths can go from X_A to X_B unless they go through X_C



- Thus, undirected graphs obey the general Markov property
- Recall the simple Markov property



$$x_1 \perp\!\!\!\perp x_3 \mid x_2 \Rightarrow p\left(x_1 \mid x_2, x_3\right) = p\left(x_1 \mid x_2\right)$$

Hammersley Clifford Theorem

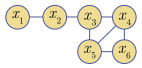
Theorem[HC]: any distribution that obeys the Markov property

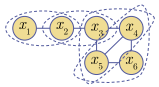
$$p\left(x_i \mid X_{U\setminus i}\right) = p\left(x_i \mid X_{Ne(i)}\right) \quad \forall\, i \in U$$

can be written as a product of terms over all maximal cliques

$$p\left(X_U\right) = p\left(x_1,\ldots,x_M\right) = \frac{1}{Z}\prod_{c\in C}\psi_c\left(X_c\right)$$

Clique: a subset of nodes that are all pair-wise adjacent
Maximal clique: cannot add more variables and still be a clique
Each c is a maximal clique of variables X_c in the graph
C is the set of all maximal cliques





Undirected Graph Functions

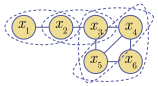
- Probability for undirected factorizes as a product of small non-negative Potential Functions over cliques in the graph

$$p\left(X\right) = p\left(x_1,\ldots,x_M\right) = \frac{1}{Z}\prod_{c\in C}\psi_c\left(X_c\right)$$

- Normalizing term $Z = \sum_X \prod_{c\in C}\psi_c\left(X_c\right)$ makes p(X) sum to 1
- Potentials ψ are non-negative un-normalized functions over cliques (subgroups of fully inter-connected variables)

- Use only maximal cliques since small ψ absorb into larger ψ

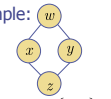
$$\psi\left(x_2,x_3\right)\psi\left(x_2\right) \rightarrow \psi\left(x_2,x_3\right) = \begin{bmatrix} 1 & 2 \\ 5 & 0 \end{bmatrix}$$



$$p\left(X\right) = \frac{1}{Z}\psi\left(x_1,x_2\right)\psi\left(x_2,x_3\right)\psi\left(x_3,x_4,x_5\right)\psi\left(x_4,x_5,x_6\right)$$

Undirected Separation Examples

- Example:



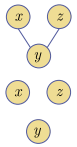
$$x \perp\!\!\!\perp y \mid \{w,z\}$$
$$w \perp\!\!\!\perp z \mid \{x,y\}$$

$$x \perp\!\!\!\perp y \mid \{w\}$$
$$x \not\perp\!\!\!\perp y \mid \{w,z\}$$

Directed can't do it!
Must be acyclic
Will have at least one V structure and ball goes through

- Example:

$$x \perp\!\!\!\perp z$$
$$x \not\perp\!\!\!\perp z \mid y$$

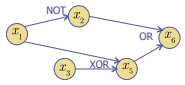


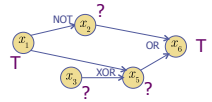
Undirected can't do it!

$$x \perp\!\!\!\perp z \mid y$$
$$x \perp\!\!\!\perp z$$

Logical Inference

- Classic logic network: nodes are binary
- Arrows represent AND, OR, XOR, NAND, NOR, NOT etc.
- Inference: given observed binary variables, predict others

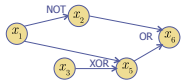




- Problems: uncertainty, conflicts and inconsistency
- Could get x_3=T and x_3=F following two different paths
- We need a way to enforce consistency and combine conflicting statements via probabilities and Bayes rule!

Probabilistic Inference

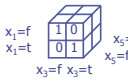
- Replace logic network with Bayesian network
- Tables represent AND, OR, XOR, NAND, NOR, NOT etc.
- Probabilistic Inference: given observed binary variables, predict marginals over others



NOT

	x_3=f	x_3=t
x_1=f	0	1
x_1=t	1	0

XOR



- Can also have soft versions of the functions

soft NOT

	x_3=f	x_3=t
x_1=f	.1	.9
x_1=t	.9	.1

Probabilistic Inference

- Two types of inference with a probability distribution:

$p(X) = p(x_1,\ldots,x_M)$ *with queries* $X_F \subseteq X$ *given evidence* $X_E \subseteq X$

- Marginal Inference:

$$p(X_F \mid X_E) = \frac{p(X_F, X_E)}{p(X_E)} = \frac{\sum_{X \setminus X_F \cup X_E} p(X)}{\sum_{X \setminus X_E} p(X)}$$

or... $p(x_i \mid X_E) \;\forall\; x_i \in X_F$

- Maximum a posteriori (MAP) inference:

$$\arg\max_{X_F} p(X_F \mid X_E)$$

...for now we focus on marginal inference

Traditional Marginal Inference

- Marginal inference problem: given graph and probability function $p(X) = p(x_1,\ldots,x_M)$ for any subsets of variables find

$$p(X_F \mid X_E) = \frac{p(X_F, X_E)}{p(X_E)}$$

- So, we basically compute both marginals and divide
- But finding marginals can take exponential work!
- A problem for both directed & undirected graphs:

$$p(x_j, x_k) = \sum_{x_1}\sum_{x_2}\cdots\sum_{x_M}\prod_{i=1}^{M} p(x_i \mid \pi_i)$$
$$p(x_j, x_k) = \sum_{x_1}\sum_{x_2}\cdots\sum_{x_M}\frac{1}{Z}\prod_{c\in C}\psi_c(X_c)$$

- Graphs gave efficient storage, learning, Bayes Ball...
- Graphs can also be used to perform efficient inference!
- Junction Tree Algorithm: method to efficiently find marginals

Traditional Marginal Inference

- Example: brute force inference on a directed graph...
- Given a directed graph structure & *filled-in* CPTs
- We would like to efficiently compute arbitrary marginals
- Or we would like to compute arbitrary conditionals

$p(X) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)p(x_5 \mid x_3)p(x_6 \mid x_2, x_5)$
$p(x_1, x_3) = p(x_1)p(x_3 \mid x_1)$
$p(x_1, x_6) = \sum_{x_2, x_3, x_4, x_5} p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)p(x_5 \mid x_3)p(x_6 \mid x_2, x_5)$
$p(x_1 \mid x_6) = \frac{\sum_{x_2 x_3 x_4 x_5} p(X)}{\sum_{x_1 x_2 x_3 x_4 x_5} p(X)}$

- For example, we may have some evidence, i.e. x_6=TRUE

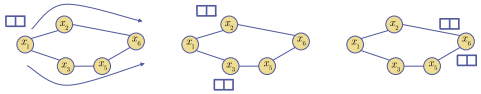
$$p(x_1 \mid x_6 = TRUE) = \frac{\sum_{x_2 x_3 x_4 x_5} p(X_{V\setminus 6}, x_6 = TRUE)}{\sum_{x_1 x_2 x_3 x_4 x_5} p(X_{V\setminus 6}, x_6 = TRUE)}$$

- This is tedious & does not exploit the graph's efficiency

Tony Jebara, Columbia University

Efficient Marginals & Inference

- Another idea is to use some efficient graph algorithm
- Try sending messages (small tables) around the graph



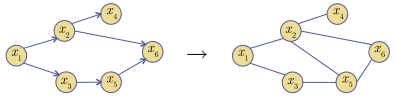
- Hopefully these somehow settle down and equal marginals

$$\hat{p}(x_1, x_6) = \sum_{x_2, x_3, x_4, x_5} p(X)$$

- AND marginals are self-consistent $\sum_{x_1}\hat{p}(x_1, x_6) = \sum_{x_2}\hat{p}(x_2, x_6)$
- Note: can't just return conditionals since they can be inconsistent $\sum_{x_1}\hat{p}(x_6 \mid x_1) \neq \sum_{x_2}\hat{p}(x_2 \mid x_6)$
- Junction Tree Algorithm must find consistent marginals

Junction Tree Algorithm

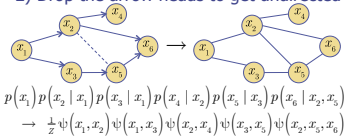
- An algorithm that achieves fast inference, by doing message passing on undirected graphs.
- We first convert a directed graph to an undirected one



- Then apply the efficient Junction Tree Algorithm:
  1) Moralization
  2) Introduce Evidence
  3) Triangulate
  4) Construct Junction Tree
  5) Propagate Probabilities (Junction Tree Algorithm)

Moralization

- Converts directed graph into undirected graph
- By moralization, marrying the parents:
  1) Connect nodes that have common children
  2) Drop the arrow heads to get undirected



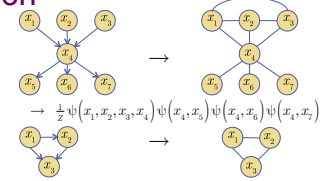
$p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)p(x_5 \mid x_3)p(x_6 \mid x_2, x_5)$
$\rightarrow \frac{1}{Z}\psi(x_1, x_2)\psi(x_1, x_3)\psi(x_2, x_4)\psi(x_3, x_5)\psi(x_2, x_5, x_6)$

$p(x_1)p(x_2 \mid x_1) \rightarrow \psi(x_1, x_2)$
$p(x_4 \mid x_2) \rightarrow \psi(x_2, x_4)$
$Z \rightarrow 1$

- Note: moralization resolves *coupling* due to marginalizing
- moral graph is more general (loses some independencies)

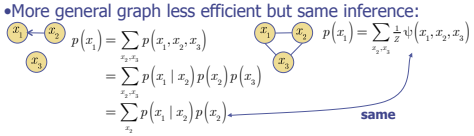
Moralization

- More examples:



$\rightarrow \frac{1}{Z}\psi(x_1, x_2, x_3, x_4)\psi(x_4, x_5)\psi(x_4, x_6)\psi(x_4, x_7)$

- More general graph less efficient but same inference:



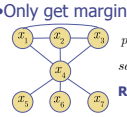
$p(x_1) = \sum_{x_2, x_3} p(x_1, x_2, x_3)$
$= \sum_{x_2, x_3} p(x_1 \mid x_2)p(x_2)p(x_3)$
$= \sum_{x_2} p(x_1 \mid x_2)p(x_2)$

$p(x_1) = \sum_{x_2, x_3}\frac{1}{Z}\psi(x_1, x_2, x_3)$

same

Introducing Evidence

- Given moral graph, note what is observed $X_E \rightarrow \bar{X}_E$
  $p(X_F \mid X_E = \bar{X}_E) \equiv p(X_F \mid \bar{X}_E)$
- If we know this is *always* observed at $X_E \rightarrow \bar{X}_E$, simplify...
- Reduce the probability function since those X_E fixed
- Only keep probability function over remaining nodes X_F
- Only get marginals and conditionals with subsets of X_F



$p(X) = \frac{1}{Z}\psi(x_1, x_2, x_3, x_4)\psi(x_4, x_5)\psi(x_4, x_6)\psi(x_4, x_7)$
say $X_E = \{x_3, x_4\} \rightarrow \bar{X}_E = \{\bar{x}_3, \bar{x}_4\}$

Replace potential functions with slices

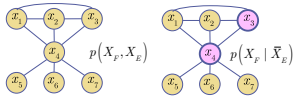
0.3	0.13
0.12	0.1

$p(X_F \mid \bar{X}_E) \propto \frac{1}{Z}\psi(x_1, x_2, x_3 = \bar{x}_3, x_4 = \bar{x}_4)\psi(x_4 = \bar{x}_4, x_5)\psi(x_4 = \bar{x}_4, x_6)\psi(x_4 = \bar{x}_4, x_7)$
$\propto \frac{1}{Z}\tilde{\psi}(x_1, x_2)\tilde{\psi}(x_5)\tilde{\psi}(x_6)\tilde{\psi}(x_7)$

But, need to recompute different normalization Z...

Introducing Evidence

- Recall undirected separation, observing X_E separates others



- But, need to recompute new normalization ...

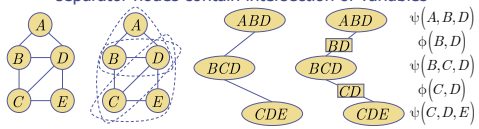
$p(X_F \mid \bar{X}_E) \propto \frac{1}{Z}\tilde{\psi}(x_1, x_2)\tilde{\psi}(x_5)\tilde{\psi}(x_6)\tilde{\psi}(x_7)$
$\rightarrow \tilde{p}(X_F) = \frac{1}{Z}\tilde{\psi}(x_1, x_2)\tilde{\psi}(x_5)\tilde{\psi}(x_6)\tilde{\psi}(x_7)$

- Just avoid Z & normalize at the end when we are querying individual marginals and conditionals as subsets of X_F

$$\tilde{p}(x_2) = \frac{\sum_{x_1, x_5, x_6, x_7}\tilde{\psi}(x_1, x_2)\tilde{\psi}(x_5)\tilde{\psi}(x_6)\tilde{\psi}(x_7)}{\sum_{x_2}\sum_{x_1, x_5, x_6, x_7}\tilde{\psi}(x_1, x_2)\tilde{\psi}(x_5)\tilde{\psi}(x_6)\tilde{\psi}(x_7)}$$

Junction Trees

- Given moral graph want to build Junction Tree:
  - each node is a clique (ψ) of variables in moral graph
  - edges connect cliques of the potential functions
  - unique path between nodes & root node (tree)
  - between adjacent clique nodes, create separators (ϕ)
  - separator nodes contain intersection of variables

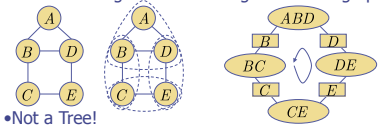


undirected **cliques** **clique tree** **junction tree**

$$p(X) = \frac{1}{Z}\psi(A,B,D)\psi(B,C,D)\psi(C,D,E)$$

Triangulation

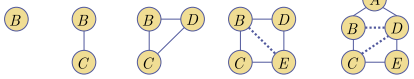
- Problem: imagine the following undirected graph



- Not a Tree!
- To ensure Junction Tree is a tree (no loops, etc.)
  - before forming it must first Triangulate moral graph
  - before finding the cliques…
- Triangulating gives more general graph (like moralization)
- Adds links to get rid of cycles or loops
- Triangulation: Connect nodes in moral graph until no chordless cycle of 4 or more nodes remains in the graph

Triangulation

- Triangulation: Connect nodes in moral graph such that no cycle of 4 or more nodes remains in graph

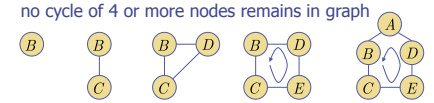


1-cycle OK **2-cycle OK** **3-cycle OK** **3-cycle OK** **3-cycle OK**

- So, *add links*, but many possible choices…
- HINT: Try to keep largest clique size small (makes junction tree algorithm more efficient)
- Sub-optimal triangulations of moral graph are Polynomial
- Triangulation that minimizes largest clique size is NP
- But, OK to use a suboptimal triangulation (slower JTA…)

Triangulation

- Triangulation: Connect nodes in moral graph such that no cycle of 4 or more nodes remains in graph



1-cycle OK **2-cycle OK** **3-cycle OK** **4-cycle BAD** **5-cycle BAD**

- So, *add links*, but many possible choices…
- HINT: Try to keep largest clique size small (makes junction tree algorithm more efficient)
- Sub-optimal triangulations of moral graph are Polynomial
- Triangulation that minimizes largest clique size is NP
- But, OK to use a suboptimal triangulation (slower JTA…)

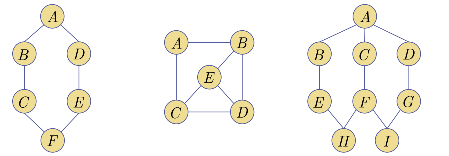
Machine Learning

4771

Topic 17

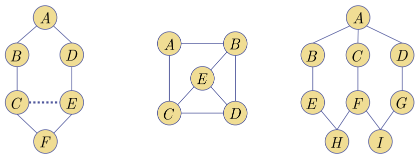
- •Triangulation Examples
- •Running Intersection Property
- •Building a Junction Tree
- •The Junction Tree Algorithm

Triangulation Examples

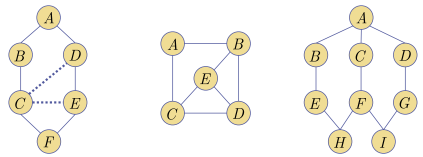


•Cycle: A closed (simple) path, with no repeated vertices other than the starting and ending vertices
•Chordless Cycle: a cycle where no two non-adjacent vertices on the cycle are joined by an edge.
•Triangulated Graph: a graph that contains no chordless cycle of four or more vertices (aka a Chordal Graph).

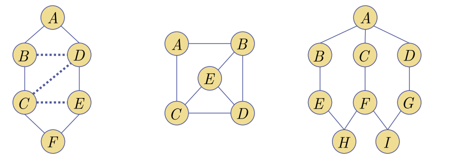
Triangulation Examples



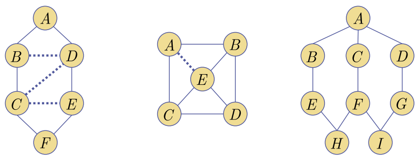
Triangulation Examples



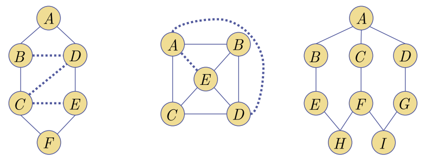
Triangulation Examples



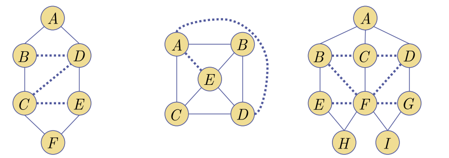
Triangulation Examples



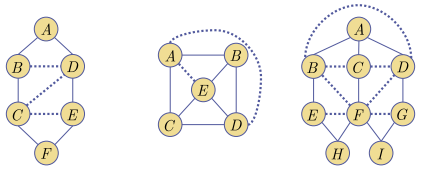
Triangulation Examples



Triangulation Examples

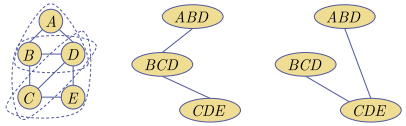


Triangulation Examples



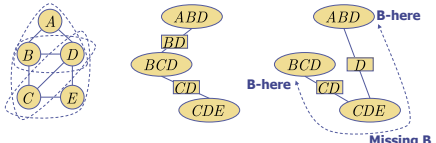
Running Intersection Property

- Junction Tree must satisfy Running Intersection Property
- RIP: On unique path connecting clique V to clique W, all other cliques share nodes in $V \cap W$



Running Intersection Property

- Junction Tree must satisfy Running Intersection Property
- RIP: On unique path connecting clique V to clique W, all other cliques share nodes in $V \cap W$



HINT: Junction Tree has largest total separator cardinality

$$|\Phi| = |\phi(B,D)| + |\phi(C,D)| = 2+2$$

$$|\Phi| = |\phi(C,D)| + |\phi(D)| = 2+1$$

Forming the Junction Tree

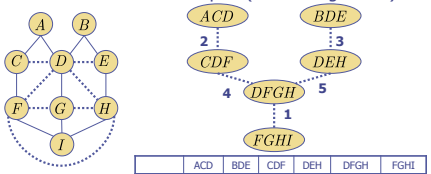
- Goal: connect k cliques into a tree… k^{k-2} possibilities!
- For each, check Running Intersection Property, too slow…
- Theorem: a valid (RIP) Junction Tree connection is one that maximizes the cardinality of the separators

$$JT^* = \arg\max_{TREE\ STRUCTURES} |\Phi| = \arg\max_{TREE\ STRUCTURES} \sum_s |\phi(X_s)|$$

- Use very fast Kruskal algorithm:
  1) Init Tree with all cliques unconnected (no edges)
  2) Compute size of separators between all pairs
  3) Connect the two cliques with the biggest separator cardinality which doesn't create a loop in current Tree (maintains Tree structure)
  4) Stop when all nodes are connected, else goto 3

Kruskal Example

- Start with unconnected cliques (after triangulation)



	ACD	BDE	CDF	DEH	DFGH	FGHI
ACD	-	1	2	1	1	0
BDE		-	1	2	1	0
CDF			-	1	2	1
DEH				-	2	1
DFGH					-	3
FGHI						-

Junction Tree Probabilities

- We now have a valid Junction Tree!
- What does that mean?
- Recall probability for undirected graphs:

$$p(X) = p(x_1,\ldots,x_M) = \frac{1}{Z}\prod_C \psi(X_C)$$

- Can write junction tree as potentials of its cliques:

$$p(X) = \frac{1}{Z}\prod_C \tilde{\psi}(X_C)$$

- Alternatively: clique potentials over separator potentials:

$$p(X) = \frac{1}{Z}\frac{\prod_C \psi(X_C)}{\prod_S \phi(X_S)}$$

- This doesn't change/do anything! Just less compact…
- Like *de-absorbing* smaller cliques from maximal cliques:

$$\tilde{\psi}(A,B,D) = \frac{\psi(A,B,D)}{\phi(B,D)} \quad \phi(B,D) \triangleq 1$$

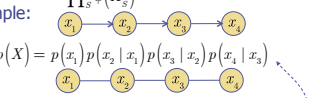
…gives back original formula if

Junction Tree Probabilities

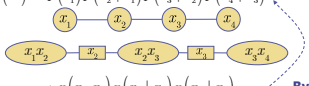
- Can quickly converted directed graph into this form:

$$p(X) = \frac{1}{Z}\frac{\prod_C \psi(X_C)}{\prod_S \phi(X_S)}$$

- Example:



$$p(X) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_2)p(x_4 \mid x_3)$$



$$p(X) = \frac{1}{1}\frac{p(x_1,x_2)p(x_3 \mid x_2)p(x_4 \mid x_3)}{1 \times 1} = \frac{1}{Z}\frac{\psi(x_1,x_2)\psi(x_2,x_3)\psi(x_3,x_4)}{\phi(x_2)\phi(x_3)}$$

By inspection, can just cut & paste CPTs as clique and separator potential functions

Junction Tree Algorithm

- Running the JTA converts clique potentials & separator potentials into marginals over their variables … and does not change p(X)

$$\psi(A,B,D) \to p(A,B,D)$$
$$\phi(B,D) \to p(B,D)$$
$$\psi(B,C,D) \to p(B,C,D)$$

- Don't want just normalization!

$$\frac{\psi(A,B,D)}{\sum_{A,B,D}\psi(A,B,D)} \neq p(A,B,D)$$

- These marginals should all agree & be consistent

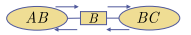
$$\psi(A,B,D) \to p(A,B,D) \to \sum_A p(A,B,D) = \tilde{p}(B,D)$$
$$\phi(B,D) \to p(B,D) \to p(B,D)$$
$$\psi(B,C,D) \to p(B,C,D) \to \sum_C p(B,C,D) = \tilde{p}(B,D)$$

ALL EQUAL

- Consistency: all distributions agree on submarginals
- JTA sends messages between cliques & separators dividing each by the others marginals until consistency…

Junction Tree Algorithm

- Send message from each clique *to* its separators of what it thinks the submarginal on the separator is.
- Normalize each clique by incoming message *from* its separators so it agrees with them



$V = \{A,B\}$ $S = \{B\}$ $W = \{B,C\}$

If agree: $\sum_{V\setminus S}\psi_V = \phi_S = p(S) = \phi_S = \sum_{W\setminus S}\psi_W$ **…Done!**

Else: Send message From V to W…

$$\phi_S^* = \sum_{V\setminus S}\psi_V$$
$$\psi_W^* = \frac{\phi_S^*}{\phi_S}\psi_W$$
$$\psi_V^* = \psi_V$$

Send message From W to V…

$$\phi_S^{**} = \sum_{W\setminus S}\psi_W^*$$
$$\psi_V^{**} = \frac{\phi_S^{**}}{\phi_S^*}\psi_V^*$$
$$\psi_W^{**} = \psi_W^*$$

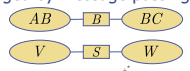
Now they Agree…Done!

$$\sum_{V\setminus S}\psi_V^{**} = \sum_{V\setminus S}\frac{\phi_S^{**}}{\phi_S^*}\psi_V^* = \frac{\phi_S^{**}}{\phi_S^*}\sum_{V\setminus S}\psi_V^* = \phi_S^{**} = \sum_{W\setminus S}\psi_W^{**}$$

Junction Tree Algorithm

- When "Done", all clique potentials are marginals and all separator potentials are submarginals!
- Note that p(X) is unchanged by message passing step:

$$\phi_S^* = \sum_{V \setminus S} \psi_V$$
$$\psi_W^* = \frac{\phi_S^*}{\phi_S} \psi_W$$
$$\psi_V^* = \psi_V$$



$$p(X) = \frac{1}{Z} \frac{\psi_V^* \psi_W^*}{\phi_S^*} = \frac{1}{Z} \frac{\psi_V \frac{\phi_S^*}{\phi_S} \psi_W}{\phi_S^*} = \frac{1}{Z} \frac{\psi_V \psi_W}{\phi_S}$$

- Potentials set to conditionals (or slices) become marginals!

$$\psi_{AB} = p(B \mid A) p(A) = p(A,B) \longrightarrow \phi_B^* = \sum_A \psi_{AB} = \sum_A p(A,B) = p(B)$$

$$\psi_{BC} = p(C \mid B) \longrightarrow \psi_{BC}^* = \frac{\phi_S^*}{\phi_S} \psi_{BC} = \frac{p(B)}{1} p(C \mid B) = p(B,C)$$

$$\phi_B = 1$$

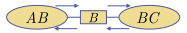
Machine Learning

Topic 18

- The Junction Tree Algorithm
- Collect & Distribute
- Algorithmic Complexity
- ArgMax Junction Tree Algorithm

Review: Junction Tree Algorithm

- Send message from each clique *to* its separators of what it thinks the submarginal on the separator is.
- Normalize each clique by incoming message *from* its separators so it agrees with them



$V=\{A,B\}$ $S=\{B\}$ $W=\{B,C\}$

If agree: $\sum_{V\setminus S}\psi_V=\phi_S=p(S)=\phi_S=\sum_{W\setminus S}\psi_W$ **...Done!**

Else:

Send message From V to W...

$$\phi_S^*=\sum_{V\setminus S}\psi_V$$
$$\psi_W^*=\frac{\phi_S^*}{\phi_S}\psi_W$$
$$\psi_V^*=\psi_V$$

Send message From W to V...

$$\phi_S^{**}=\sum_{W\setminus S}\psi_W^*$$
$$\psi_V^{**}=\frac{\phi_S^{**}}{\phi_S^*}\psi_V^*$$
$$\psi_W^{**}=\psi_W^*$$

Now they Agree...Done!

$$\sum_{V\setminus S}\psi_V^{**}=\sum_{V\setminus S}\frac{\phi_S^{**}}{\phi_S^*}\psi_V^*$$
$$=\frac{\phi_S^{**}}{\phi_S^*}\sum_{V\setminus S}\psi_V^*$$
$$=\phi_S^{**}=\sum_{W\setminus S}\psi_W^{**}$$

JTA with many cliques

- Problem: what if we have more than two cliques?

1) Update AB & BC

2) Update BC & CD

- Problem: AB has not heard about CD!
  After BC updates, it will be inconsistent for AB

- Need to iterate the pairwise updates many times
- This will eventually converge to consistent marginals
- But, inefficient… can we do better?

JTA: Collect & Distribute

- Use tree recursion rather than iterate messages mindlessly!

initialize(DAG){ Pick root
Set all variables as: $\psi_{C_i}=p(x_i\mid\pi_i),\phi_S=1$ **}**

collectEvidence(node) {
for each child of node {
update1(node,collectEvidence(child)); }
return(node); }

distributeEvidence(node) {
for each child of node {
update2(child,node);
distributeEvidence(child); } }

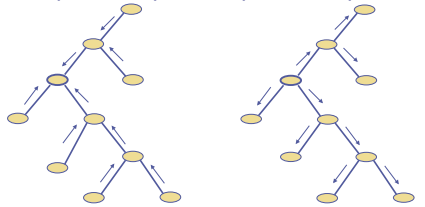
update1(node w,node v) { $\phi^*_{V\cap W}=\sum_{V\setminus(V\cap W)}\psi_V,\ \psi_W=\frac{\phi^*_{V\cap W}}{\phi_{V\cap W}}\psi_W$ **}**

update2(node w,node v) { $\phi^{**}_{V\cap W}=\sum_{V\setminus(V\cap W)}\psi_V,\ \psi_W=\frac{\phi^{**}_{V\cap W}}{\phi^*_{V\cap W}}\psi_W$ **}**

normalize() { $p(X_C)=\frac{1}{\sum_C\psi^{**}_C}\psi^{**}_C\ \forall C,\ p(X_S)=\frac{1}{\sum_S\phi^{**}_S}\phi^{**}_S\ \forall S$ **}**

Junction Tree Algorithm

- JTA: 1)*Initialize* 2)*Collect* 3)*Distribute* 4)*Normalize*



- Note: leaves do not change their ψ during *collect*
- Note: the first cliques *collect* changes are parents of leaves
- Note: root does not change its ψ during *distribute*

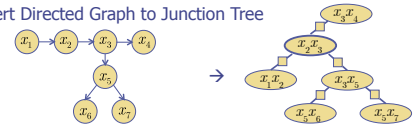
Algorithmic Complexity

- The 5 steps of JTA are all efficient:

1) Moralization
   Polynomial in # of nodes
2) Introduce Evidence (fixed or constant)
   Polynomial in # of nodes (convert pdf to slices)
3) Triangulate (Tarjan & Yannakakis 1984)
   Suboptimal=Polynomial, Optimal=NP
4) Construct Junction Tree (Kruskal)
   Polynomial in # of cliques
5) Junction Tree Algorithm (Init,Collect,Distribute,Normalize)
   Polynomial (linear) in # of cliques, *Exponential* in Clique Cardinality

Junction Tree Algorithm

- Convert Directed Graph to Junction Tree



- *Initialize* separators to 1 (and Z=1) and set clique tables to the CPTs in the Directed Graph

$$p(X)=p(x_1)p(x_2\mid x_1)p(x_3\mid x_2)p(x_4\mid x_3)p(x_5\mid x_3)p(x_6\mid x_5)p(x_7\mid x_5)$$

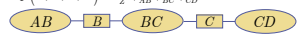
$$p(X)=\frac{1}{Z}\frac{\prod_C\psi(X_C)}{\prod_S\phi(X_S)}=\frac{1}{1}\frac{p(x_1,x_2)p(x_3\mid x_2)p(x_4\mid x_3)p(x_5\mid x_3)p(x_6\mid x_5)p(x_7\mid x_5)}{1\times1\times1\times1\times1}$$

- Run *Collect, Distribute, Normalize*
- Get valid marginals from all ψ,ϕ tables

JTA with Extra Evidence

- If extra *evidence* is observed, must slice tables accordingly
- Example: $p(A,B,C,D)=\frac{1}{Z}\psi_{AB}\psi_{BC}\psi_{CD}$

$Z=1$



$$\psi_{AB}=\begin{bmatrix}8 & 4\\ 3 & 1\end{bmatrix}^{A=0}_{A=1}\quad \psi_{BC}=\begin{bmatrix}2 & 3\\ 1 & 1\end{bmatrix}^{B=0}_{B=1}\quad \psi_{CD}=\begin{bmatrix}1 & 4\\ 1 & 1\end{bmatrix}^{C=0}_{C=1}$$

- You are given *evidence*: A=0. Replace table with slices…

$$\psi_{AB}\to\begin{bmatrix}8 & 4\end{bmatrix}\quad \psi_{BC}=\begin{bmatrix}2 & 3\\ 1 & 1\end{bmatrix}\quad \psi_{CD}=\begin{bmatrix}1 & 4\\ 1 & 1\end{bmatrix}$$

- JTA now gives ψ,ϕ as marginals *conditioned* on evidence

$$p(B\mid A=0)=\frac{\psi^{**}_{AB}}{\sum_B\psi^{**}_{AB}}\quad p(B,C\mid A=0)=\frac{\psi^{**}_{BC}}{\sum_{B,C}\psi^{**}_{BC}}\quad p(C,D\mid A=0)=\frac{\psi^{**}_{CD}}{\sum_{C,D}\psi^{**}_{CD}}$$

- All denominators equal the new normalizer Z'

$$Z'=p(EVIDENCE)=\sum_B\psi^{**}_{AB}=\sum_{B,C}\psi^{**}_{BC}=\sum_{C,D}\psi^{**}_{CD}$$

ArgMax Junction Tree Algorithm

- We can also use JTA for finding the max not the sum over the joint to get argmax of marginals & conditionals
- Say have some evidence: $p\left(X_E, \bar{X}_E\right) = p\left(x_1,\ldots,x_n,\bar{x}_{n+1},\ldots,\bar{x}_N\right)$
- Most likely (highest p) X_E? $X_E^* = \arg\max_{X_E} p\left(X_E, \bar{X}_E\right)$
- What is most likely state of patient with fever & headache?

$$\begin{aligned} p_F^* &= \max_{x_2,x_3,x_4,x_5} p\left(x_1=1,x_2,x_3,x_4,x_5,x_6=1\right) \\ &= \max_{x_2} p\left(x_2 \mid x_1=1\right) p\left(x_1=1\right) \max_{x_3} p\left(x_3 \mid x_1=1\right) \\ &\quad \max_{x_4} p\left(x_4 \mid x_2\right) \max_{x_5} p\left(x_5 \mid x_3\right) p\left(x_6=1 \mid x_2,x_5\right) \end{aligned}$$

- Solution: replace sum with max inside JTA update code

$$\phi^*_{V\cap W} = \max_{V\setminus(V\cap W)} \psi_V, \quad \psi_W = \frac{\phi^*_{V\cap W}}{\phi_{V\cap W}} \psi_W \qquad \phi^{**}_{V\cap W} = \max_{V\setminus(V\cap W)} \psi_V, \quad \psi_W = \frac{\phi^{**}_{V\cap W}}{\phi^*_{V\cap W}} \psi_W$$

- Final potentials are *max marginals*: $\psi^{**}\left(X_C\right) = \max_{V\setminus C} p\left(X\right)$
- Highest value in potential is most likely: $X_C^* = \arg\max_C \psi^{**}\left(X_C\right)$

ArgMax Junction Tree Algorithm

- Why do I need the ArgMax junction tree algorithm?
- Can't I just compute marginals using the Sum algorithm and then find the highest value in each marginal???
- No!! Here's a counter-example:

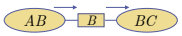
$$p\left(x_1,x_2\right) = \begin{array}{c} \\ x_2=0 \\ x_2=1 \end{array} \begin{array}{c} \begin{array}{ccc} x_1 & x_1 & x_1 \\ A & B & C \end{array} \\ \begin{bmatrix} .14 & .05 & .27 \\ .24 & .20 & .10 \end{bmatrix} \end{array}$$

- Most likely is x_1*=C and x_2*=0
- But the sub-marginals p(x_1) and p(x_2) do not reveal this…

$$p\left(x_1\right) = \begin{array}{cccc} A & B & C \\ \end{array} \\ \begin{bmatrix} 0.38 & 0.25 & 0.37 \end{bmatrix} \qquad p\left(x_2\right) = \begin{array}{cc} x_2=0 & \\ x_2=1 & \end{array} \begin{bmatrix} .46 \\ .54 \end{bmatrix}$$

- The marginals would *falsely* imply that is x_1*=A and x_2*=1

Example



- Note that products are element-wise
- Let us send a regular JTA message from AB to BC

$$\psi_{AB} = \begin{bmatrix} 8 & 4 \\ 3 & 1 \end{bmatrix}^{A=0}_{A=1} \qquad \phi_B = \begin{bmatrix} 1 \\ 1 \end{bmatrix}^{B=0}_{B=1} \qquad \psi_{BC} = \begin{bmatrix} 2 & 3 \\ 1 & 1 \end{bmatrix}^{B=0}_{B=1}$$

$$\phi^*_B = \sum_A \psi_{AB} = \sum_A \begin{bmatrix} 8 & 4 \\ 3 & 1 \end{bmatrix} = \begin{bmatrix} 11 & 5 \end{bmatrix} = \begin{bmatrix} 11 \\ 5 \end{bmatrix}^{B=0}_{B=1}$$

$$\psi^*_{BC} = \frac{\phi^*_B}{\phi_B} \psi_{BC} = \frac{\begin{bmatrix} 11 \\ 5 \end{bmatrix}}{\begin{bmatrix} 1 \\ 1 \end{bmatrix}} \begin{bmatrix} 2 & 3 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 22 & 33 \\ 5 & 5 \end{bmatrix}^{B=0}_{B=1}$$

- If argmax JTA, just change the separator update to:

$$\phi^*_B = \max_A \psi_{AB} = \max_A \begin{bmatrix} 8 & 4 \\ 3 & 1 \end{bmatrix} = \begin{bmatrix} 8 & 4 \end{bmatrix} = \begin{bmatrix} 8 \\ 4 \end{bmatrix}^{B=0}_{B=1}$$

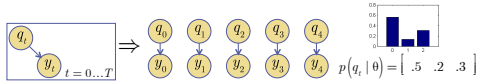
Machine Learning

Topic 19

- Hidden Markov Models
- HMMs as State Machines & Applications
- HMMs Basic Operations
- HMMs via the Junction Tree Algorithm

Hidden Markov Models

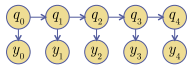
- A great application of Junction Tree Algorithm and EM
- Recall mixture of Gaussians model on IID data



- Example: location data of a single parent as a mixture of Gaussians
- Parent has 3 internal states:
  q={home,daycare,work}
- Based on q, sample from appropriate Gaussian mean and covariance to get
  y=(latitude,longitude)

Hidden Markov Models

- Parent drops child at daycare before & after work. Not IID!



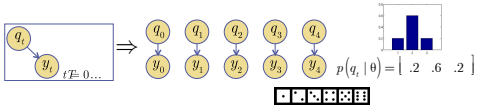
q={1=home,2=daycare,3=work}

$$p(q_t \mid q_{t-1}) = \begin{bmatrix} 0.8 & 0.2 & 0 \\ 0.1 & 0.8 & 0.1 \\ 0 & 0.2 & 0.8 \end{bmatrix} \begin{matrix} q_{t-1}=1 \\ q_{t-1}=2 \\ q_{t-1}=3 \end{matrix}$$
$$\begin{matrix} q_t=1 & q_t=2 & q_t=3 \end{matrix}$$

- Have dependence on previous state
- Can't go straight from home to work!
- Now, order of $y_0,\ldots,y_T$ matters (in IID order doesn't matter)

Hidden Markov Models

- Consider mixture of multinomials (dice) y={1,2,3,4,5,6}



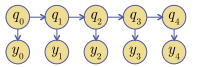
$p(q_t \mid \theta) = [\ .2\ \ .6\ \ .2\ ]$

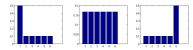
- Example: a crooked casino croupier using mixture of dice.
- You win if he rolls 1,2,3. You lose he rolls 4,5,6.
- Croupier has 3 internal states q={helpful,fair,adversarial}
- Based on q, sample different 'dice' multinomial

1=helpful 2=fair 3=adversarial

Hidden Markov Models

- But if the dealer has a memory or mood? Not IID!
  5646166166 4321534161414341634 1113114121





p(y|q)

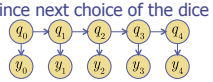
q={1=helpful,2=fair,3=adversarial}
y={1,2,3,4,5,6}

$$p(q_t \mid q_{t-1}) = \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.8 & 0.1 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} \begin{matrix} q_{t-1}=1 \\ q_{t-1}=2 \\ q_{t-1}=3 \end{matrix}$$
$$\begin{matrix} q_t=1 & q_t=2 & q_t=3 \end{matrix}$$

- If you tip, dealer starts to like you and rolls the helpful die
- Dealer has a memory of his mood and last type of die q_{t-1}
- Will often use same die for q^t as was rolled before…
- Now, order of $y_0,\ldots,y_T$ matters (if IID order doesn't matter)

Hidden Markov Models

- Since next choice of the dice depends on previous one…



Order of $y_0,\ldots,y_T$ matters
Temporal or sequence model!

- Add left-right arrows. This is a hidden Markov model
- Markov: $future \perp\!\!\!\perp past \mid present$

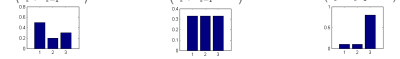
$$p(q_t \mid q_{t-1}, q_{t-2}, \ldots, q_1, q_0) = p(q_t \mid q_{t-1})$$

- From graph, have the following general pdf:

$$p(X_U) = p(q_0)\prod_{t=1}^{T} p(q_t \mid q_{t-1}) \prod_{t=0}^{T} p(y_t \mid q_t)$$

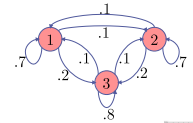
- So $p(q_t)$ depends on previous state q_{t-1} …

$p(q_t \mid q_{t-1}=1)$ $p(q_t \mid q_{t-1}=2)$ $p(q_t \mid q_{t-1}=3)$

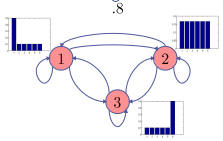


HMMs as State Machines

- HMMs have two variables: state q and emission y
- Typically, we don't know q (hidden variable 1,2,3,?)
- HMMs are like stochastic automata or finite state machines…
  next state depends on previous one…
  (helpful, fair, adversarial)

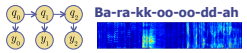


- Can't observe state q directly, just a random related emission y outcome (dice roll) so…
  doubly-stochastic automaton



HMM Applications

- Speech Recognition
  phonemes from audio cepstral vectors



Ba-ra-kk-oo-oo-dd-ah

- Language Parsing
  parts of speech from words

Noun Verb Noun
John Ate Pizza

- Genomics
  splice site from gene sequence

-Intron-|-Exon-|-Promoter-
GATTACATTATACCACCATACG

HMMs: Parameters

- We focus on HMMs with: discrete state q (of size M)
  discrete emission y (of size N)
- Input will be arbitrary length string: $y_0,\ldots,y_T$
- The pdf or (complete) likelihood is:

$$p(q,y) = p(q_0)\prod_{t=1}^{T} p(q_t \mid q_{t-1})\prod_{t=0}^{T} p(y_t \mid q_t)$$

- We don't know hidden states, the incomplete likelihood is:

$$p(y) = \sum_{q_0}\cdots\sum_{q_T} p(q,y)$$

- Assume HMM is stationary, tables are repeated: $\theta = \{\pi,\eta,\alpha\}$

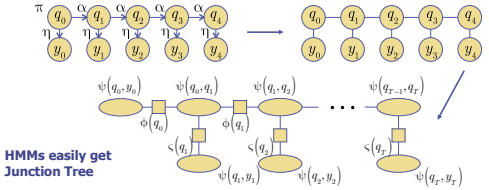
$$p(q_t \mid q_{t-1}) = \prod_{i=1}^{M}\prod_{j=1}^{M}\left[\alpha_{ij}\right]^{\#(q_{t-1}=i,\,q_t=j)} \qquad \sum_{j=1}^{M}\alpha_{ij} = 1 \qquad M \times M$$

$$p(y_t \mid q_t) = \prod_{i=1}^{M}\prod_{j=1}^{N}\left[\eta_{ij}\right]^{\#(q_t=i,\,y_t=j)} \qquad \sum_{j=1}^{N}\eta_{ij} = 1 \qquad M \times N$$

$$p(q_0) = \prod_{i=1}^{M}\left[\pi_i\right]^{\#(q_0=i)} \qquad \sum_{j=1}^{M}\pi_j = 1 \qquad M$$

HMMs: Basic Operations

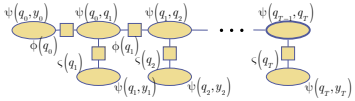
- Would like to do 3 basic things with our HMMs:
  1) Evaluate: given $y_0,\ldots,y_T$ & θ compute $p(y_1,..,y_T)$
  2) Decode: given $y_0,\ldots,y_T$ & θ find $q_0,\ldots,q_T$ or $p(q_0),\ldots,p(q_T)$
  3) Max Likelihood: given $y_0,\ldots,y_T$ learn parameters θ
- Typically use Baum-Welch (α-β algo)… JTA is more general:



HMMs easily get Junction Tree

HMMs: JTA Init & Verify

- Init: $\psi(q_0,y_0) = p(q_0)p(y_0 \mid q_0)$ $\quad \psi(q_t,q_{t+1}) = p(q_{t+1} \mid q_t) = \alpha_{q_t,q_{t+1}}$ $\quad \psi(q_t,y_t) = p(y_t \mid q_t)$ $\quad Z = 1$



$$\phi(q_t) = 1 \qquad \varsigma(q_t) = 1$$

- Collect *up* (this time it actually doesn't change the zetas)

$$\varsigma^*(q_t) = \sum_{q_0}\psi(q_t,y_t) = \sum_{q_0} p(y_t \mid q_t) = 1 \qquad \psi^*(q_{t-1},q_t) = \frac{\varsigma^*}{\varsigma}\psi(q_{t-1},q_t) = \psi(q_{t-1},q_t)$$

- Collect *left-right* via phi's: change backbone to marginals

$$\phi^*(q_0) = \sum_{q_0}\psi(q_0,y_0) = p(q_0) \qquad \psi^*(q_0,q_1) = \frac{\phi^*}{\phi}\psi(q_0,q_1) = p(q_0,q_1)$$

$$\phi^*(q_t) = \sum_{q_{t-1}}\psi^*(q_{t-1},q_t) = p(q_t) \qquad \psi^*(q_{t-1},q_t) = \frac{p(q_{t-1})}{1}p(q_t \mid q_{t-1}) = p(q_{t-1},q_t)$$

- Distribute:

$$\varsigma^{**}(q_t) = \sum_{q_{t-1}}\psi^*(q_{t-1},q_t) = \sum_{q_{t-1}} p(q_{t-1},q_t) = p(q_t)$$

$$\psi^{**}(q_t,y_t) = \frac{\varsigma^{**}}{\varsigma^*}\psi(q_t,y_t) = \frac{p(q_t)}{1}p(y_t \mid q_t) = p(y_t,q_t)$$

…done!

Machine Learning

Topic 20

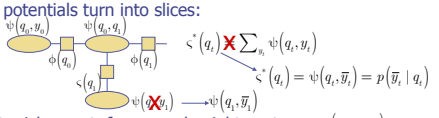
- HMMs with Evidence
- HMM Collect
- HMM Evaluate
- HMM Distribute
- HMM Decode
- HMM Parameter Learning via JTA & EM

HMMs: JTA with Evidence

- If y sequence is observed (in problems 1,2,3) get evidence:

$$p(q,\bar{y}) = p(q_0)\prod_{t=1}^{T} p(q_t \mid q_{t-1})\prod_{t=0}^{T} p(\bar{y}_t \mid q_t)$$

- The potentials turn into slices:



$$\zeta^*(q_t) = \sum_{y_t}\psi(q_t,y_t)$$

$$\zeta^*(q_t) = \psi(q_t,\bar{y}_t) = p(\bar{y}_t \mid q_t)$$

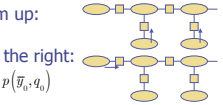
$$\psi(q_1,y_1) \rightarrow \psi(q_1,\bar{y}_1)$$

- Next, pick a root, for example *rightmost* one: $\psi(q_{T-1},q_T)$
- Collect all zeta separators bottom up:

$$\zeta^*(q_t) = \psi(q_t,\bar{y}_t) = p(\bar{y}_t \mid q_t)$$

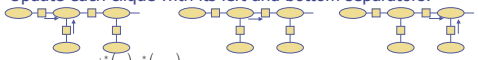
- Collect leftmost phi separator to the right:

$$\phi^*(q_0) = \sum_{q_0}\psi(q_0,\bar{y}_0)\delta(y_0-\bar{y}_0) = p(\bar{y}_0,q_0)$$



HMMs: Collect with Evidence

- Now, we will collect (*) along the backbone left to right
- Update each clique with its left and bottom separators:



$$\psi^*(q_t,q_{t+1}) = \frac{\phi^*(q_t)}{1}\frac{\zeta^*(q_{t+1})}{1}\psi(q_t,q_{t+1}) = \phi^*(q_t)\,p(\bar{y}_{t+1} \mid q_{t+1})\alpha_{q_t,q_{t+1}}$$

$$\phi^*(q_{t+1}) = \sum_{q_t}\psi^*(q_t,q_{t+1}) = \sum_{q_t}\phi^*(q_t)\,p(\bar{y}_{t+1} \mid q_{t+1})\alpha_{q_t,q_{t+1}}$$

- Keep going along chain until right most node
- Note: above formula for phi is recursive, could use as is.
- Property: recall we had $\phi^*(q_0) = p(\bar{y}_0,q_0)$

$$\phi^*(q_1) = \sum_{q_0} p(\bar{y}_0,q_0)\,p(\bar{y}_1 \mid q_1)\,p(q_1 \mid q_0) = p(\bar{y}_0,\bar{y}_1,q_1)$$

$$\phi^*(q_2) = \sum_{q_1} p(\bar{y}_0,\bar{y}_1,q_1)\,p(\bar{y}_2 \mid q_2)\,p(q_2 \mid q_1) = p(\bar{y}_0,\bar{y}_1,\bar{y}_2,q_2)$$

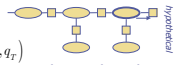
$$\phi^*(q_{t+1}) = \sum_{q_t} p(\bar{y}_0,\ldots,\bar{y}_t,q_t)\,p(\bar{y}_{t+1} \mid q_{t+1})\,p(q_{t+1} \mid q_t) = p(\bar{y}_0,\ldots,\bar{y}_{t+1},q_{t+1})$$

HMMs: Evaluate with Evidence

- Say we are solving the first HMM problem:
  1) Evaluate: given $y_0,\ldots,y_T$ & θ compute $p(y_0,\ldots,y_T|\theta)$
- If we want to compute the likelihood, we are already done!
- We really just need to do collect (not even distribute).
- From previous slide we had:

$$\phi^*(q_{t+1}) = \sum_{q_t} p(\bar{y}_0,\ldots,\bar{y}_t,q_t)\,p(\bar{y}_{t+1} \mid q_{t+1})\,p(q_{t+1} \mid q_t) = p(\bar{y}_0,\ldots,\bar{y}_{t+1},q_{t+1})$$

- Collect 'til root (rightmost node): $\psi^*(q_{T-1},q_T) = p(\bar{y}_0,\ldots,\bar{y}_T,q_{T-1},q_T)$ its normalizer is p(EVIDENCE)!



Or use hypothetical $\phi^*(q_T) = p(\bar{y}_0,\ldots,\bar{y}_T,q_T)$

- Can compute the likelihood just by marginalizing this phi

$$p(\bar{y}_0,\ldots,\bar{y}_T) = \sum_{q_T} p(\bar{y}_0,\ldots,\bar{y}_T,q_T) = \sum_{q_T}\phi^*(q_T)$$

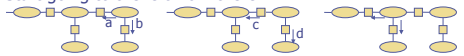
- So, adding up the entries in last ϕ^* gives us the likelihood

HMMs: Distribute with Evidence

- Back to collecting… say just finished collecting to the root with our last update formula:

$$\psi^*(q_{T-1},q_T) = \frac{\phi^*(q_{T-1})}{1}\frac{\zeta^*(q_T)}{1}\psi(q_{T-1},q_T) = \phi^*(q_{T-1})\,p(\bar{y}_T \mid q_T)\alpha_{q_{T-1},q_T}$$

- Now, we distribute (**) along the backbone right to left
- Have first ** for root (stays the same): $\psi^{**}(q_{T-1},q_T) = \psi^*(q_{T-1},q_T)$
- Start going to the left from there:



for t=T-1 to 0

a) $\phi^{**}(q_t) = \sum_{q_{t+1}}\psi^{**}(q_t,q_{t+1})$

b) $\zeta^{**}(q_{t+1}) = \sum_{q_t}\psi^{**}(q_t,q_{t+1})$

c) $\psi^{**}(q_t,q_{t+1}) = \frac{\phi^{**}(q_{t+1})}{\phi^*(q_{t+1})}\psi^*(q_t,q_{t+1})$

d) $\psi^{**}(y_t,q_t) = \frac{\zeta^{**}(q_t)}{\zeta^*(q_t)}\psi(y_t,q_t)$

HMM Example

You are given the parameters of a 2-state HMM. You observed the input sequence AB (from a 2-symbol alphabet A or B). In other words, you observe two symbols from your finite state machine, A and then B. Using the junction tree algorithm, evaluate the likelihood of this data p(y) given your HMM and its parameters. Also compute (for decoding) the individual marginals of the states after the evidence from this sequence is observed: $p(q_0|y)$ and $p(q_1|y)$. The parameters for the HMM are provided below. They are the initial state prior $p(q_0)$, the state transition matrix given by $p(q_t|\ q_{t-1})$, and the emission matrix $p(y_t|\ q_t)$, respectively.

$$\pi = p(q_0) = \begin{array}{c} \\ \end{array}\begin{array}{c} 1 \quad 2 \\ \end{array}$$

$$\begin{bmatrix} 1/3 & 2/3 \end{bmatrix}$$

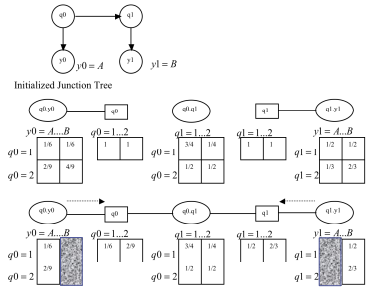
$$a^T = p(q_t \mid q_{t-1}) = \begin{array}{c} \\ 1 \\ 2 \end{array}\begin{array}{c} 1 \quad\quad 2 \\ \end{array}$$

$$\begin{bmatrix} 3/4 & 1/2 \\ 1/4 & 1/2 \end{bmatrix}$$

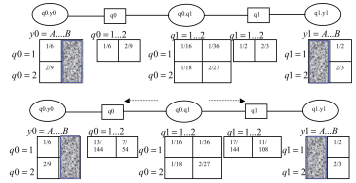
$$\eta^T = p(y_t \mid q_t) = \begin{array}{c} \\ A \\ B \end{array}\begin{array}{c} 1 \quad\quad 2 \\ \end{array}$$

$$\begin{bmatrix} 1/2 & 1/3 \\ 1/2 & 2/3 \end{bmatrix}$$

HMM Example



HMM Example



So the likelihood $p(y) = \frac{13}{144} + \frac{7}{54} = \frac{1}{16} + \frac{1}{18} + \frac{1}{36} + \frac{2}{27} = \frac{17}{144} + \frac{11}{108} = \frac{95}{432} = 0.2199$

$p(q_0 = 1 \mid y) = \frac{13/144}{13/144 + 7/54} = \frac{39}{95}$; $p(q_0 = 2 \mid y) = \frac{7/54}{13/144 + 7/54} = \frac{56}{95}$

$p(q_1 = 1 \mid y) = \frac{17/144}{17/144 + 11/108} = \frac{51}{95}$; $p(q_1 = 2 \mid y) = \frac{11/108}{17/144 + 11/108} = \frac{44}{95}$

HMMs: Marginals & MaxDecoding

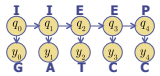
- Now that JTA is finished, we have the following:

$$\phi^{**}(q_t) \propto p(q_t \mid \bar{y}_1,\ldots,\bar{y}_T) \qquad \varsigma^{**}(q_{t+1}) \propto p(q_{t+1} \mid \bar{y}_1,\ldots,\bar{y}_T)$$
$$\psi^{**}(q_t,q_{t+1}) \propto p(q_t,q_{t+1} \mid \bar{y}_1,\ldots,\bar{y}_T)$$

- The separators define a distribution over the hidden states
- This gives the probability the DNA symbol y_t was q_t={I,E,P}
- We've done 2) Decode: given $y_0,\ldots,y_T$ & θ find $p(q_0),\ldots,p(q_T)$

- Can also do 2) Decode: given $y_0,\ldots,y_T$ & θ find $q_0,\ldots,q_T$
- We can also decode to find the most likely path $q_0 \ldots q_T$
- Here, we use the ArgMax JTA algorithm
- Run JTA but replace sums with max
- Then, find biggest entry in separators:

$$\hat{q}_t = \arg\max_{q_t} \phi^{**}(q_t) \quad \forall t = 0\ldots T$$



HMMs: EM Learning

- Finally 3) Max Likelihood: given $y_0,\ldots,y_T$ learn parameters θ
- Recall max likelihood: $\hat{\theta} = \arg\max_{\theta} \log p(\bar{y} \mid \theta)$
- If observe q, it's easy to maximize the *complete* likelihood:

$$l(\theta) = \log\left(p(q,y)\right)$$
$$= \log\left(p(q_0)\prod_{t=1}^{T} p(q_t \mid q_{t-1})\prod_{t=0}^{T} p(\bar{y}_t \mid q_t)\right)$$
$$= \log p(q_0) + \sum_{t=1}^{T} \log p(q_t \mid q_{t-1}) + \sum_{t=0}^{T} \log p(\bar{y}_t \mid q_t)$$
$$= \log \prod_{i=1}^{M} [\pi_i]^{q_0^i} + \sum_{t=1}^{T} \log \prod_{i=1}^{M}\prod_{j=1}^{M} [\alpha_{ij}]^{q_{t-1}^i q_t^j} + \sum_{t=0}^{T} \log \prod_{i=1}^{M}\prod_{j=1}^{N} [\eta_{ij}]^{q_t^i y_t^j}$$
$$= \sum_{i=1}^{M} q_0^i \log \pi_i + \sum_{t=1}^{T}\sum_{i,j=1}^{M} q_{t-1}^i q_t^j \log \alpha_{ij} + \sum_{t=0}^{T}\sum_{i=1}^{M}\sum_{j=1}^{N} q_t^i y_t^j \log \eta_{ij}$$

Introduce Lagrange & take derivatives →

$$\sum_{i=1}^{M} \pi_i = 1 \qquad \sum_{j=1}^{M} \alpha_{ij} = 1 \qquad \sum_{j=1}^{N} \eta_{ij} = 1$$

$$\hat{\pi}_i = q_0^i \qquad \hat{\alpha}_{ij} = \frac{\sum_{t=0}^{T-1} q_t^i q_{t+1}^j}{\sum_{k=1}^{M}\sum_{t=0}^{T-1} q_t^i q_{t+1}^k} \qquad \hat{\eta}_{ij} = \frac{\sum_{t=0}^{T} q_t^i y_t^j}{\sum_{k=1}^{N}\sum_{t=0}^{T} q_t^i y_t^k}$$

HMMs: EM Learning

- But, we don't observe the q's, incomplete…

$$p(\bar{y} \mid \theta) = \sum_{q} p(q,\bar{y} \mid \theta) = \sum_{q_0}\cdots\sum_{q_T} p(q_0)\prod_{t=1}^{T} p(q_t \mid q_{t-1})\prod_{t=0}^{T} p(\bar{y}_t \mid q_t)$$

- EM: Max expected complete likelihood given current p(q)

$$E\{l(\theta)\} = E_{p(q_0,\ldots,q_T \mid \bar{y})}\{\log p(q,y)\} = \sum_{q_0}\cdots\sum_{q_T} p(q \mid y)\log p(q,y)$$
$$= E\left\{\sum_{i=1}^{M} q_0^i \log \pi_i + \sum_{t=1}^{T}\sum_{i,j=1}^{M} q_{t-1}^i q_t^j \log \alpha_{ij} + \sum_{t=0}^{T}\sum_{i=1}^{M}\sum_{j=1}^{N} q_t^i y_t^j \log \eta_{ij}\right\}$$
$$= \sum_{i=1}^{M} E\{q_0^i\}\log \pi_i + \sum_{t=1}^{T}\sum_{i,j=1}^{M} E\{q_{t-1}^i q_t^j\}\log \alpha_{ij} + \sum_{t=0}^{T}\sum_{i=1}^{M}\sum_{j=1}^{N} E\{q_t^i\} y_t^j \log \eta_{ij}$$

- M-step is maximizing as before:

$$\hat{\pi}_i = E\{q_0^i\} \qquad \hat{\alpha}_{ij} = \frac{\sum_{t=0}^{T-1} E\{q_t^i q_{t+1}^j\}}{\sum_{k=1}^{M}\sum_{t=0}^{T-1} E\{q_t^i q_{t+1}^k\}} \qquad \hat{\eta}_{ij} = \frac{\sum_{t=0}^{T} E\{q_t^i\} y_t^j}{\sum_{k=1}^{N}\sum_{t=0}^{T} E\{q_t^i\} y_t^k}$$

- What are E{}'s? $E_{p(x)}\{x^i\} = \sum_x p(x) x^i = \sum_x p(x)\delta(x = x^i) = p(x^i)$
- Our JTA ψ & ϕ marginals! (JTA is the E-Step for given θ)

$$E\{q_t^i q_{t+1}^j\} = p(q_t = i, q_{t+1} = j \mid \bar{y}) = \frac{\psi^{**}(q_t = i, q_{t+1} = j)}{\sum_{ij}\psi^{**}(q_t = i, q_{t+1} = j)} \qquad E\{q_t^i\} = p(q_t = i \mid \bar{y}) = \frac{\phi^{**}(q_t = i)}{\sum_i \phi^{**}(q_t = i)}$$