Topics for today

- Inference in Junction trees
- Factor graph message passing
- Max-product
- Hidden Markov Models (inference, learning)
- Conditional Random Fields (inference, learning)

Inference in a directed graph – Bayes Net

We have exact algorithms for inference on an undirected tree.

By agglomerating variables into cliques we can transform a Bayes Net (a directed model) into an tree-structured MRF (undirected model).

The transformation process consists of several steps

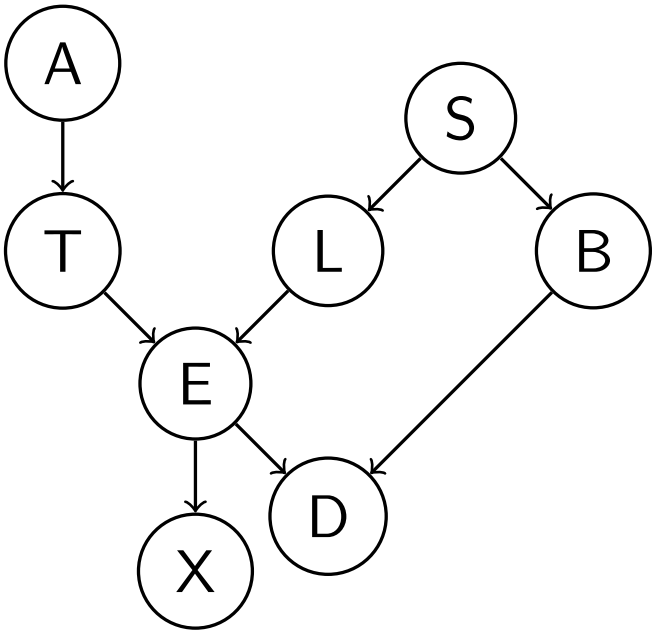
- moralization
- triangulation
- construction of a clique tree

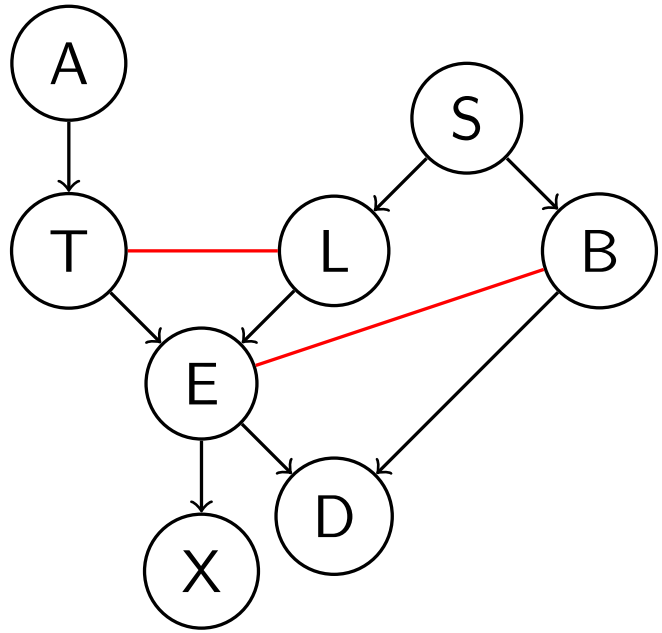
Once we have a clique tree all that remains is to

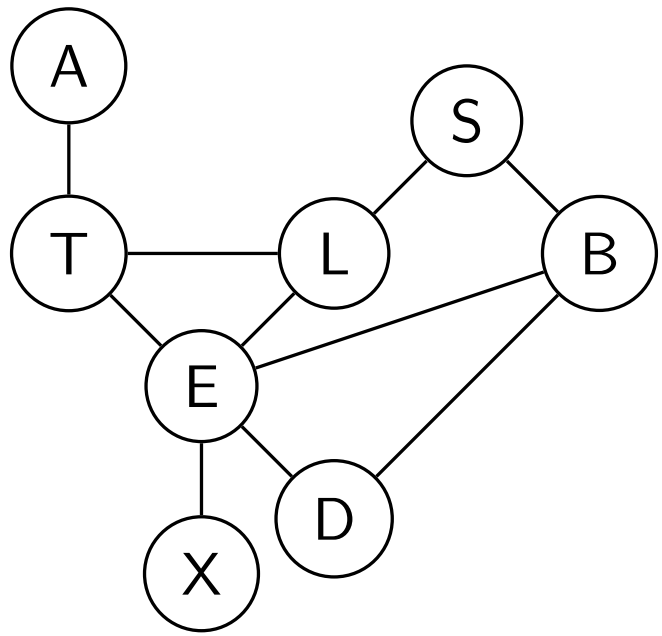
- incorporate observations (transform each ϕ to ϕ^E)
- run message passing algorithm to get marginals of interest

Moralization

Moralization joins parents of nodes and removes edge directions

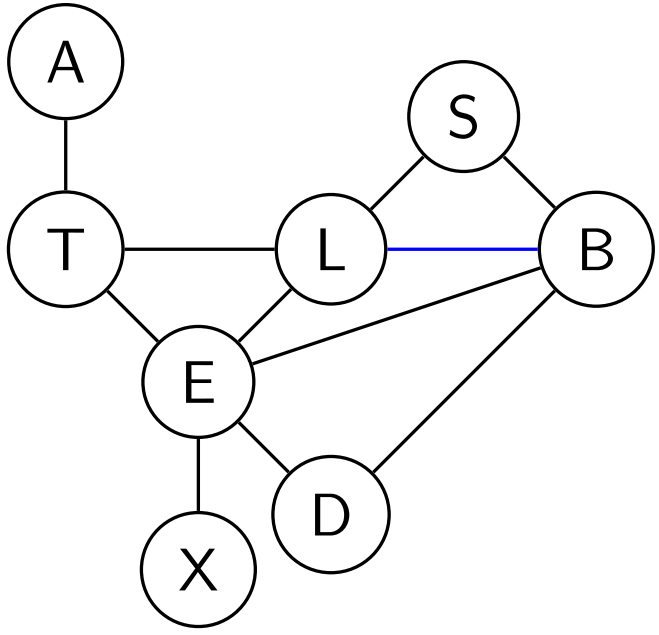






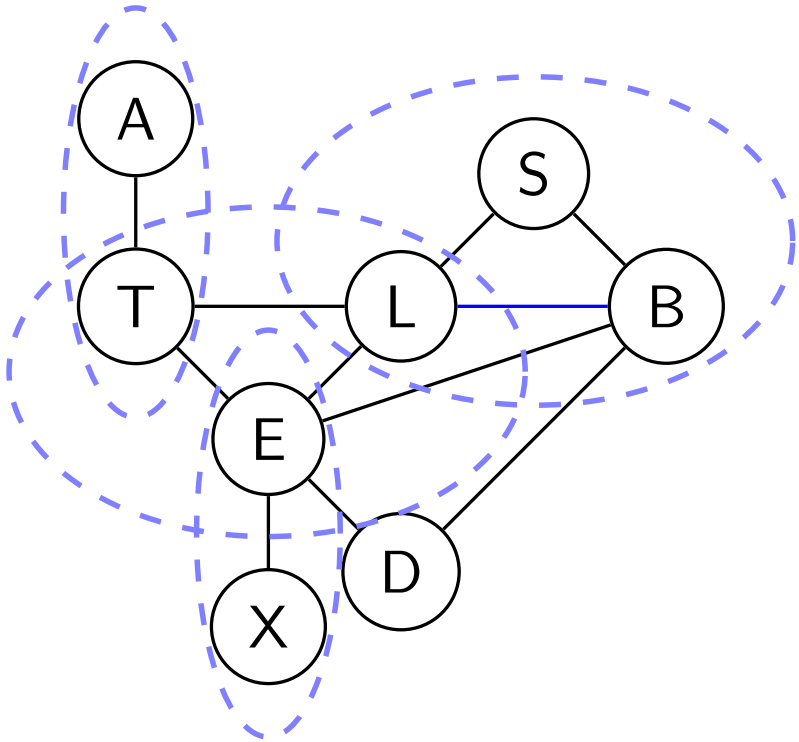
Triangulation

Every cycle of length ≥ 4 has to have a chord.



Listing maximal cliques

From the moralized and triangulated graph we now read out maximal cliques

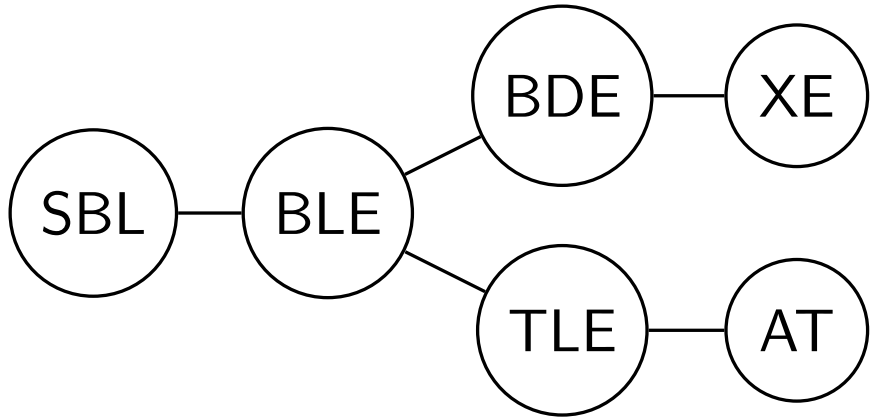


In this case AT,TLE,EX,BDE,BLE,SLB

Running intersection property

We are going to construct a clique tree on the nodes representing each of the cliques (AT,TLE,EX,BDE,BLE,SLB).

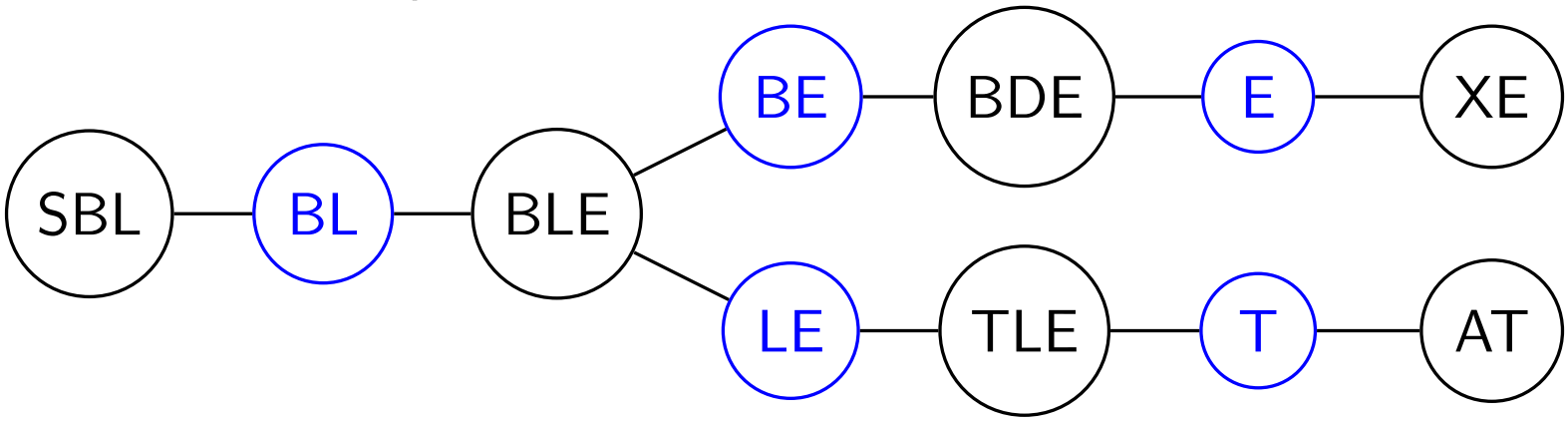
The running intersection property: if two nodes share a variable then all nodes in the path between them contain that variable.



A graph with the running intersection property can be obtained by running maximum spanning tree on the adjacency matrix with weights being the number of shared variables.

Adding separators

Add nodes corresponding to variables shared between the neighboring nodes of the clique tree.

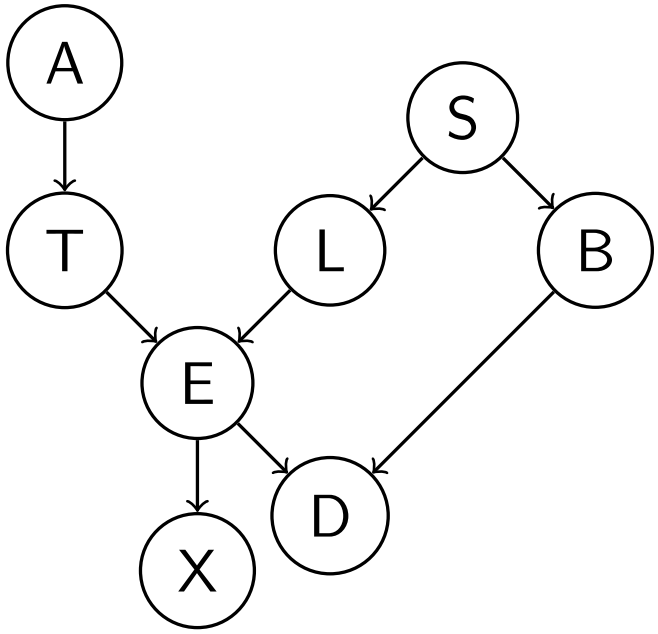


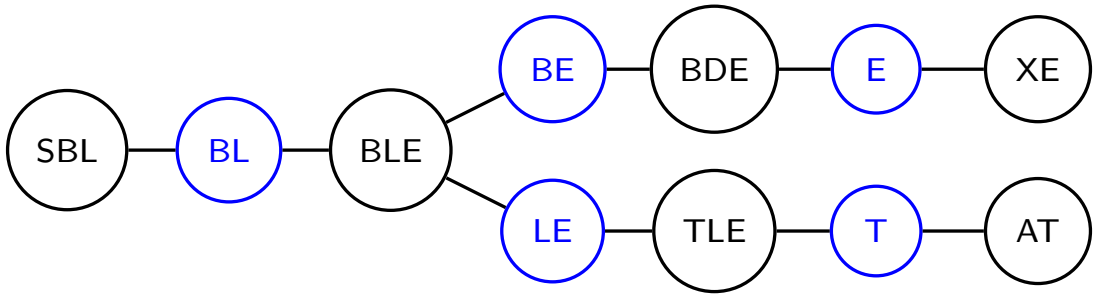
We note that each of these new variables in a clique tree is an agglomeration of multiple variables of the original Bayes net.

For example, SBL can assume any configuration of values that S,B and L can assume (hence exponential in size), e.g. if S,B,L are binary then SBL can assume $2^3 = 8$ different values.

Setting potentials

For clique nodes the potentials are given by the Bayes Net conditional probabilities on variables in the clique with a constraint that each conditional probability occurs in only one potential.





For example

$$\phi_{SBL,BL}(SBL, BL) = \begin{cases} p(B|S)p(L|S)p(S) & \text{BL and SBL agree on B,L} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_{BLE,BL}(BLE, BL) = \begin{cases} p(E|L) & \text{BL and BLE agree on B,L} \\ 0 & \text{otherwise} \end{cases}$$

Junction graph

The resulting MRF is called a junction graph and is amenable to inference using the message passing algorithms we introduced earlier.

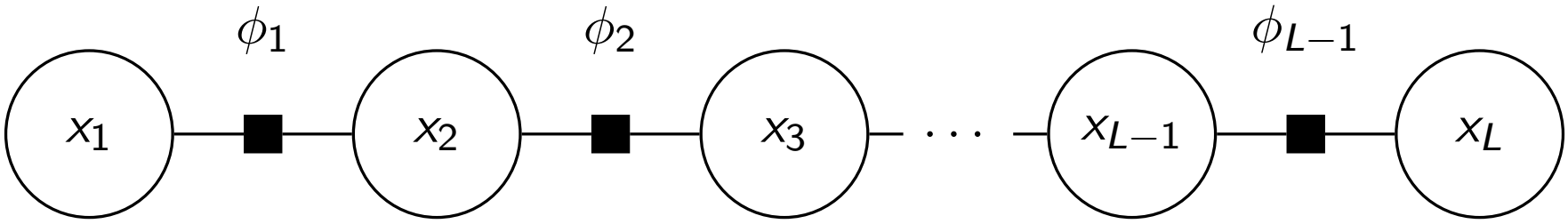
The marginals computed in this graph are the marginals defined by the Bayes Net.

All we did is aggregate variables to get to a tree structure.

The resulting message passing algorithm is called Shafer-Shenoy propagation.

For HW3, you will implement junction tree algorithm from scratch.

Factor graph view



Factor graphs use two types of messages: from variables to factors and vice versa.

$$\mu_{\phi_k \to x_i}(v) = \sum_{x_{C_k}, x_i = v} \phi_k(x_{C_k}) \prod_{j \in C_k, j \neq i} \mu_{x_j \to \phi_k}(x_j)$$

$$\mu_{x_i \to \phi_k}(v) = \prod_{\phi_l \in n(x_i), k \neq l} \mu_{\phi_l \to x_i}(v)$$

In the case of chains and trees all of the message passing algorithms we described so far are exact.

Sum-product and max-product

Last time we looked at message passing algorithms for a model specified by pairwise clique potentials

$$p(\mathbf{x}) = \frac{1}{C} \prod_I \phi_I(\mathbf{x}_{C_I})$$

we derived message passing algorithms that iterate

$$m_{x_j,x_k}(v) = \sum_{x_j} \phi_{jk}(x_j, x_k = v) \prod_{x_l \in n(x_j), l \neq k} m_{x_l,x_j}(x_j)$$

and can be used to compute marginal probabilities

$$p(x_k = v) = \frac{1}{Z} \prod_{x_j \in n(x_k)} m_{x_j,x_k}(v)$$

Sum-product and max-product

The key property we used in deriving message passing algorithms on a chain/tree is the distributivity of multiplication (over a sum)

$$\sum_x \sum_y f(x)g(y,x) = \sum_x f(x) \sum_y g(y,x)$$

max is also an operation that multiplication can distribute over

$$\max_x \max_y f(x)g(y,x) = \max_x f(x) \max_y g(y,x)$$

Finding most probable assignment to hidden variables

Taking just the first step towards deriving the dynamic program on a chain

$$\max_{\mathbf{x}} p(\mathbf{x}) = \max_{\mathbf{x}} \frac{1}{Z} \prod_{l=1}^{L-1} \phi_l(x_l, x_{l+1})$$

$$= \frac{1}{Z} \times$$

$$\max_{x_{k-1}} \phi_{k-1}(x_{k-1}, x_k) \max_{x_{k-2}} \phi_{k-2}(x_{k-2}, x_{k-1}) \ldots \max_{x_2} \phi_2(x_2, x_3) \max_{x_1} \phi_1(x_1, x_2)$$

$$\times \max_{x_{k+1}} \phi_k(x_k, x_{k+1}) \ldots \max_{x_{L-1}} \phi_{L-1}(x_{L-2}, x_{L-1}) \max_{x_L} \phi_{L-1}(x_{L-1}, x_L)$$

Following the same trajectory as with forward-backward (αs and βs) last time we can obtain a different message passing algorithm

Max product message passing

Max product message passing on a tree

$$m_{x_j,x_k}(v) = \max_{x_j} \phi_{jk}(x_j, x_k = v) \prod_{x_l \in n(x_j), l \neq k} m_{x_l,x_j}(x_j)$$

and we can obtain the most probable assignment

$$x_k^* = \operatorname{argmax}_v \frac{1}{Z} \prod_{x_j \in n(x_k)} m_{x_j,x_k}(v)$$

Note that $\mathbf{x}^* = \operatorname{argmax}_{\mathbf{v}} p(\mathbf{x} = \mathbf{v})$

Max-product algorithm on a chain is also called Viterbi algorithm.

Max prod solution $\neq$ max of sum product

Let $\mathbf{x}^*$ be the solution computed by max product algorithm

$$\mathbf{x}^* = \operatorname*{argmax}_{\mathbf{v}} p(\mathbf{x} = \mathbf{v})$$

and hence the $\mathbf{x}^*$ is an assignment[1] that achieves maximum probability.

Let $\hat{\mathbf{x}}$ be defined as

$$\hat{x}_k = \operatorname*{argmax}_{x_k} p(x_k) = \operatorname*{argmax}_{x_k} \sum_{x_1, x_2, \ldots, x_{k-1}, x_{k+1}, \ldots} p(\mathbf{x})$$

so each variable in $\hat{\mathbf{x}}$ is set to the maximum of its marginal.
By definition

$$p(\mathbf{x}^*) \geq p(\hat{\mathbf{x}})$$

[1] there can be more than one

Suboptimality of max of marginals

$p(x_1, x_2)$	$x_1 = 0$	$x_1 = 1$	$p(x_2)$
$x_2 = 0$	**0.35**	0.3	**0.65**
$x_2 = 1$	0.05	0.3	0.35
$p(x_1)$	0.4	**0.6**	

Most likely state is $(x_1 = 0, x_2 = 0)$ with prob 0.35.

If we went by the maximum of marginals we would obtain $(x_1 = 1, x_2 = 0)$ and that state has prob $0.3 < 0.35$

Max product finds the most likely assignment.

Sum product computes the marginals, don't mix and match.

Max-product in a factor graph

Just swap out the sum and swap in a max

$$\mu_{\phi_k \to x_i}(v) = \max_{x_{C_k}, x_i = v} \phi_k(x_{C_k}) \prod_{j \in C_k, j \neq i} \mu_{x_j \to \phi_k}(x_j)$$

$$\mu_{x_i \to \phi_k}(v) = \prod_{\phi_l \in n(x_i), k \neq l} \mu_{\phi_l \to x_i}(v)$$

And the assignment is given by

$$x_k^* = \operatorname*{argmax}_v \prod_{\phi_j \in n(x_k)} \mu_{\phi_j, x_k}(v)$$

Clarifications – expectations and marginals

Suppose we need to compute an expectation of some function f with respect to some distribution $q(h_1, \ldots, h_L)$.

Let us assume that f works on h_A where $A \subset \{1, \ldots, L\}$

I want to compute

$$E_q[f(h_A)] = \sum_{\mathbf{h}} q(\mathbf{h}) f(h_A)$$

and my claim is that

$$E_q[f(h_A)] = \sum_{\mathbf{h}} q(\mathbf{h}) f(h_A) = \sum_{h_A} q(h_A) f(h_A)$$

and hence I only need marginal $q(h_A)$ to compute this expectation.

Clarifications – expectations and marginals

My goal: $\sum_{\mathbf{h}} q(\mathbf{h}) f(h_A) = \sum_{h_A} q(h_A) f(h_A)$

Introduce B such that $A \cup B = \{1, \dots L\}$ and $A \cap B = \emptyset$.

$$
\begin{aligned}
E_q[f(h_A)] &= \sum_{\mathbf{h}} q(\mathbf{h}) f(h_A) \\
&= \sum_{h_A} \sum_{h_B} q(h_A, h_B) f(h_A) \\
&= \sum_{h_A} \sum_{h_B} q(h_B|h_A) q(h_A) f(h_A) \\
&= \sum_{h_A} q(h_A) f(h_A) \sum_{h_B} q(h_B|h_A) \\
&= \sum_{h_A} q(h_A) f(h_A) \underbrace{\sum_{h_B} q(h_B|h_A)}_{=1} \\
&= \sum_{h_A} q(h_A) f(h_A)
\end{aligned}
$$

Hidden Markov Models

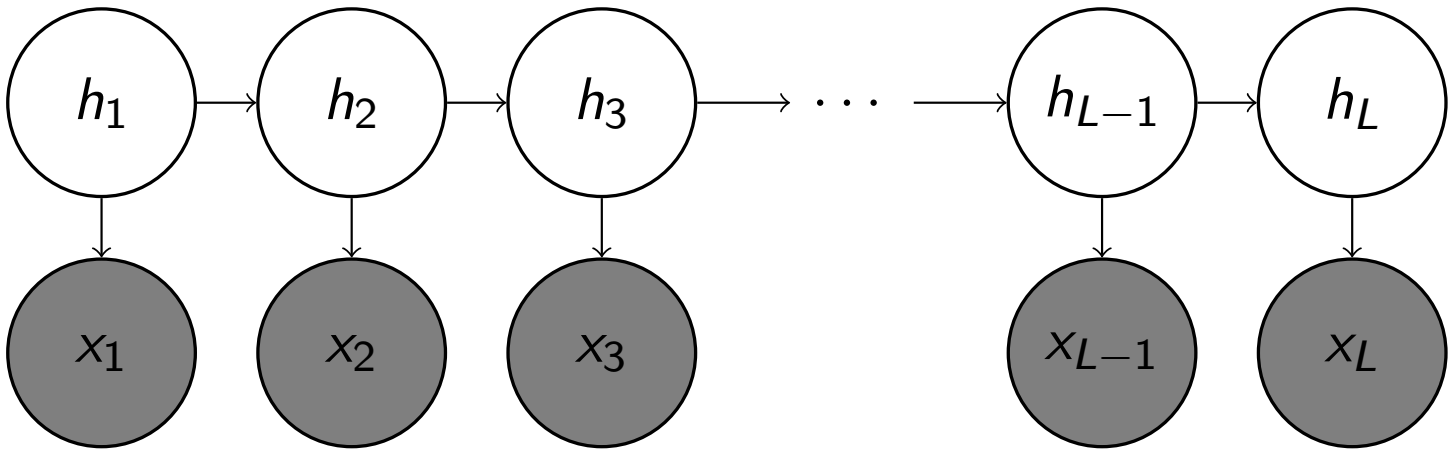
We are going to run through a complete EM for a basic HMM to fortify our understanding of EM.

... and look at some code.

Hidden Markov Models

A family of models on two sets of variables $(h_1, \ldots h_L, x_1 \ldots x_L)$ that honor a particular set of conditional independencies.

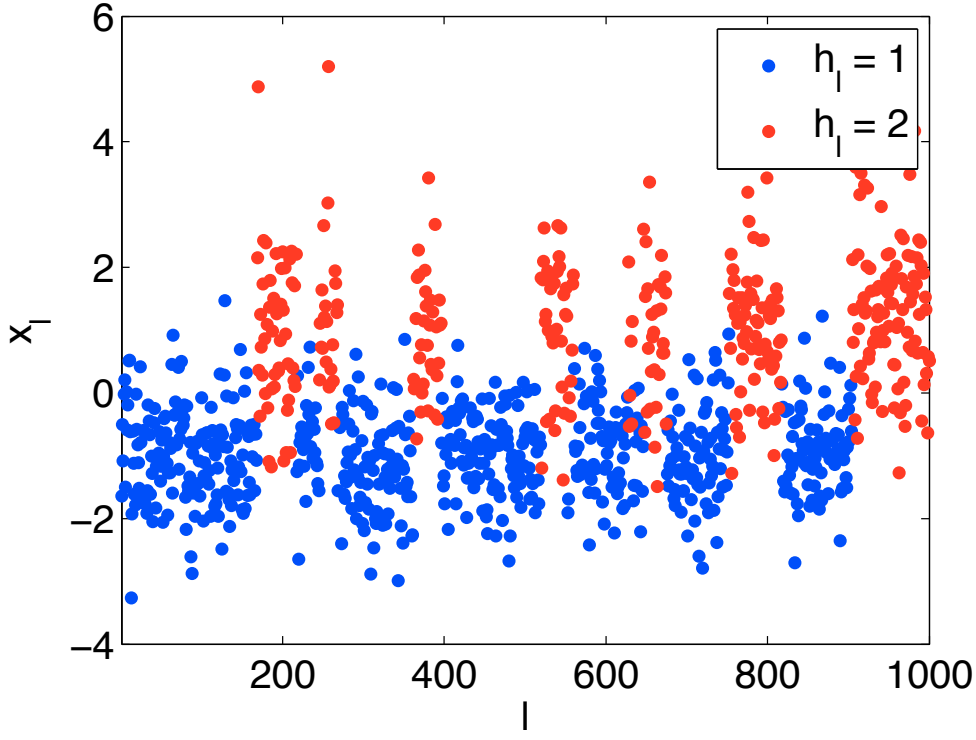
$$h_i \perp h_j | h_{i-1}, h_{i+1} \qquad (j \notin \{i-1, i, i+1\})$$
$$h_i \perp x_j | h_{i-1}, h_{i+1} \qquad (j \notin \{i-1, i, i+1\})$$
$$x_i \perp h_j | h_i \qquad (j \neq i)$$
$$x_i \perp x_j | h_i \qquad (j \neq i)$$



HMM - conditional probabilities

$$
\begin{aligned}
p(h_1 = m|\pi) &= \pi_m \\
p(h_i|h_{i-1}, T) &= T(h_i|h_{i-1}) \\
p(x_i|h_i, \nu) &= g(x_i; \nu_{h_i})
\end{aligned}
$$

An example of a data instance generated from an HMM



$$p(h_1) = \pi = [0.5\ 0.5]$$

$$p(h_i|h_{i-1}) = T = \begin{bmatrix} 0.975 & 0.025 \\ 0.025 & 0.975 \end{bmatrix}$$

$$p(x_i|h_i = 1) = f_1(x_i) = \mathcal{N}(-1, 0.25)$$

$$p(x_i|h_i = 2) = f_2(x_i) = \mathcal{N}(1, 1)$$

An example of a data instance generated from an HMM

ACTCCGTTGACTTATACCCTGATCCAAATCCGCATCCAAC

CTGGGTTGCGCGCGCGCGCGCGCGCGCGCGCGTACGACGT

ACCTCGCGCGTGAAGTTGCG

$$p(h_1) = \pi = [0.5\ 0.5\ 0]$$

$$p(h_i|h_{i-1}) = T = \begin{bmatrix} 0.95 & 0.05 & 0 \\ 0 & 0 & 1 \\ 0.1 & 0.9 & 0 \end{bmatrix}$$

$$p(x_i|h_i = 1) = f_1(x_i) = [\ 0.25 \quad 0.25 \quad 0.25 \quad 0.25\ ]$$

$$p(x_i|h_i = 2) = f_2(x_i) = [\ 0 \quad 0 \quad 1 \quad 0\ ]$$

$$p(x_i|h_i = 3) = f_3(x_i) = [\ 0 \quad 0 \quad 0 \quad 1\ ]$$

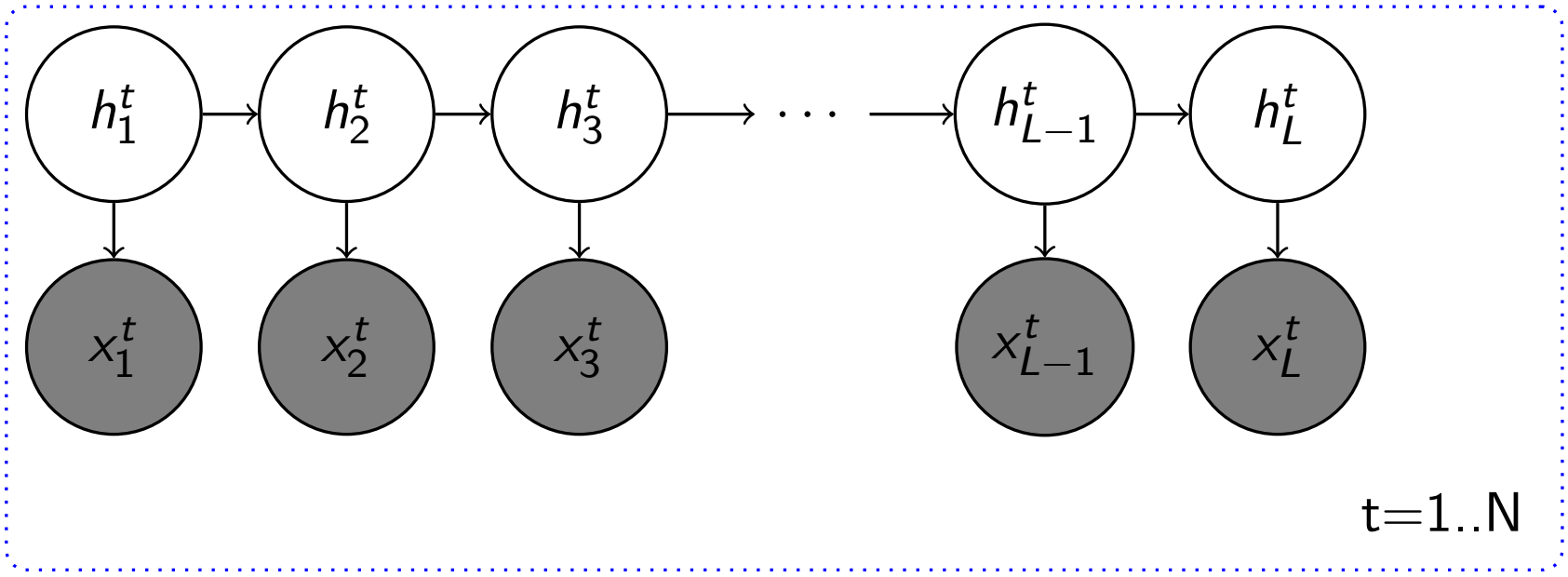
Inference problems in HMMs

Inferring most likely joint assignment for hidden variables (decoding problem) – max product.

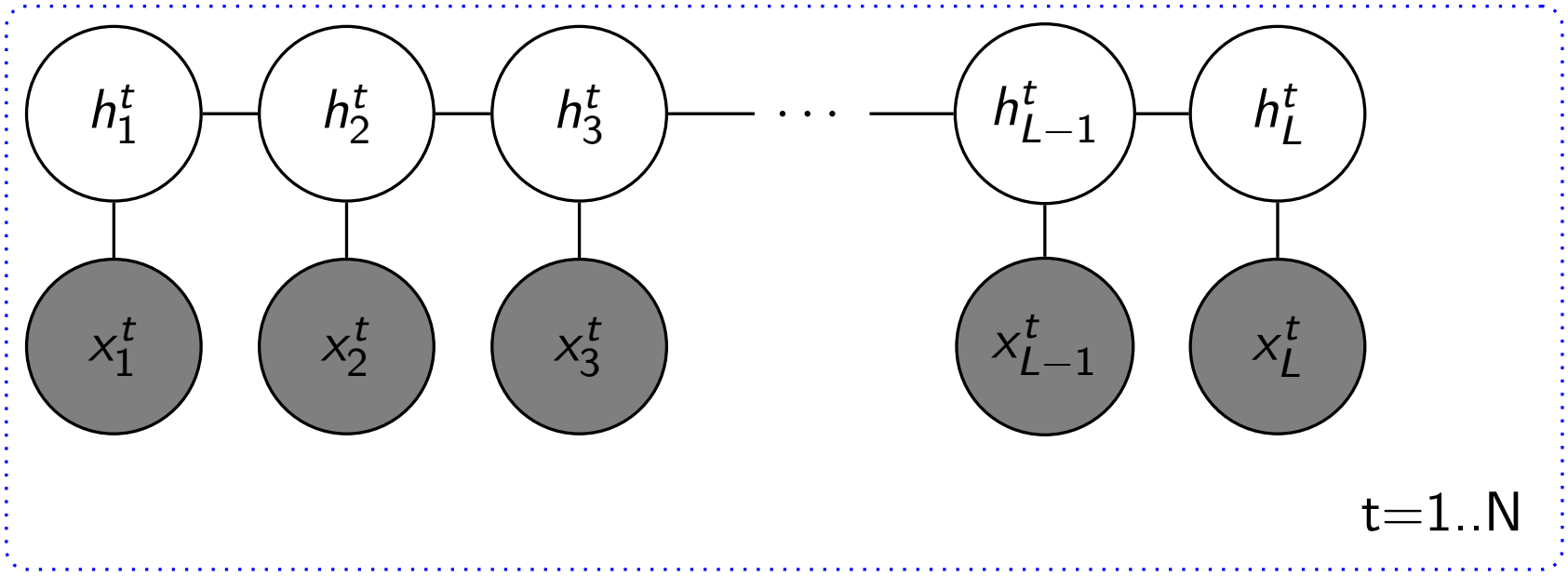
Learning parameters (transitions probabilities and state specific emission probabilities) – EM using sum-product for E-step.

Bayes Net

Note that we are again going to be explicit about data instances: (x^t) is the t^{th} data instance. To accentuate this and remind you of plate notation



Clique tree



$$q(\mathbf{h}) = \prod_t q(h^t) = \prod_t \frac{1}{Z^t} p(h_1^t) p(x_1^t | h_1) \prod_{l=2}^{L-1} p(h_l^t | h_{l-1}^t) p(x_l^t | h_l^t)$$

Max-product instantiation

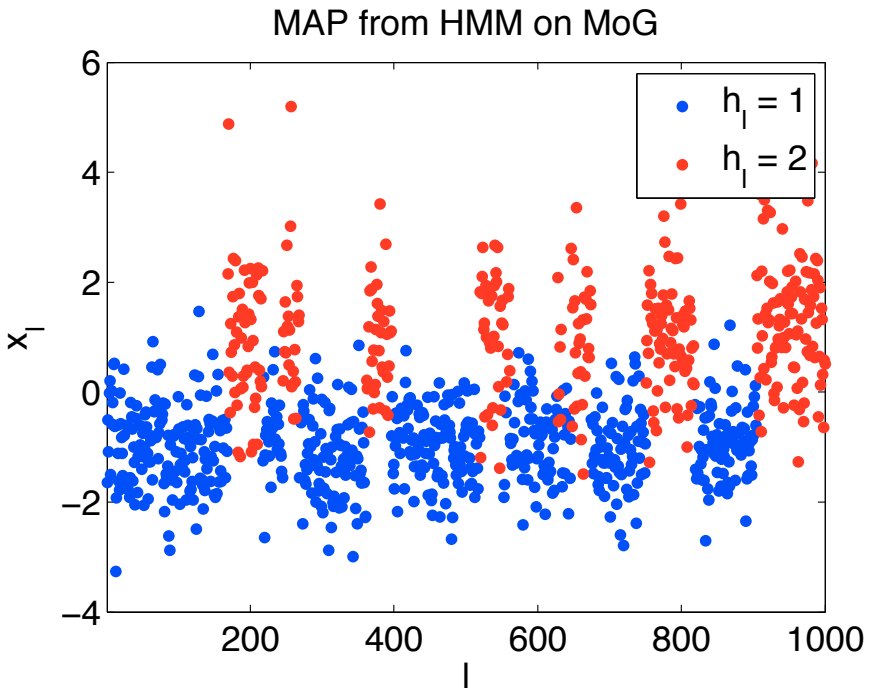
$$
\begin{aligned}
m_{h_{i-1},h_i}(v) &= \max_{h_{i-1}} p(h_i|h_{i-1})p(x_i|h_i)m_{h_{i-2},h_{i-1}}(h_{i-1}) \\
m_{h_{i+1},h_i}(v) &= \max_{h_{i+1}} p(h_{i+1}|h_i)p(x_{i+1}|h_{i+1})m_{h_{i+2},h_{i+1}}(h_{i+1})
\end{aligned}
$$

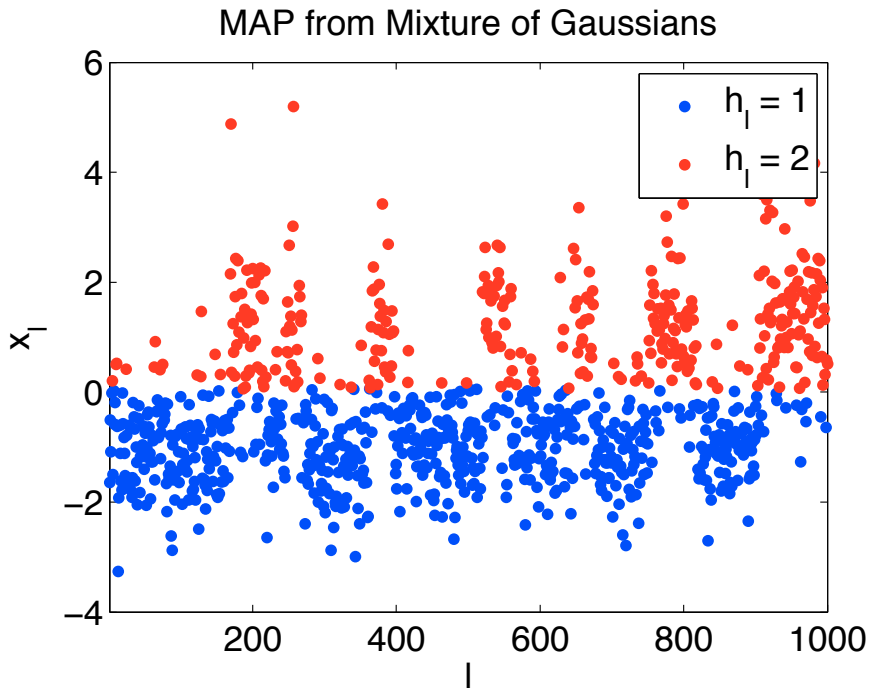
Once both forward and backward pass are done

$$
h_i^* = \operatorname*{argmax}_v m_{h_{i-1},h_i}(v)m_{h_{i+1},h_i}(v)
$$

Staring at some code ...

MAP from two different models on the same data instance





Color is based on MAP assignments of class labels (ground truth parameters were used).

MoG gets 8.2% missclassification error and HMM on MoG 0.3%.

Learning parameters of HMM

We will use exact EM:

$$E: q^{\text{new}} = \operatorname*{argmax}_{q} \sum_{t} \sum_{h^t} q(h^t) \log p(x^t, h^t|\theta) - \sum_{h} q(h^t) \log q(h^t)$$

$$M: \theta^{new} = \operatorname*{argmax}_{\theta} \sum_{t} \sum_{h^t} q^{\text{new}}(h^t) \log p(x^t, h^t|\theta)$$

Which parameters are we learning

$$
\begin{aligned}
p(h_1 = m|\pi) &= \pi_m \\
p(h_i|h_{i-1}, T) &= T(h_i, |h_{i-1}) \\
p(x_i|h_i, \nu) &= g(x_i; \nu_{h_i})
\end{aligned}
$$

and in the case of MoG $\nu_k = (\mu_k, \Sigma_k)$ mean and covariance matrix of the k^{th} class.

So the M-step

$$
M : \theta^{new} = \operatorname*{argmax}_{\theta} \sum_{t} \sum_{h^t} q^{\text{new}}(h^t) \log p(x^t, h^t|\theta)
$$

operates on $\theta = \{\pi, T, \nu_1, \ldots, \nu_K\}$.

M-step derivation

$$T^{new} = \operatorname*{argmax}_T \sum_t \sum_{\mathbf{h}^t} q(\mathbf{h}^t) \log \left\{ p(h_1^t) p(x_1^t | h_1^t) \prod_{l=2}^{L} T(h_l^t | h_{l-1}^t) p(x_l^t | h_l^t) \right\}$$

Let us simplify the expression under argmax

$$\operatorname*{argmax}_T \sum_t \sum_{\mathbf{h}^t} q(\mathbf{h}^t) \log \left\{ p(h_1^t) p(x_1^t | h1) \prod_{l=2}^{L} T(h_l^t | h_{l-1}^t) p(x_l^t | h_l^t) \right\} =$$

$$\operatorname*{argmax}_T \sum_t \sum_{\mathbf{h}^t} q(\mathbf{h}^t) \left(\log \left\{ \prod_{l=2}^{L} T(h_l^t | h_{l-1}^t) \right\} + \underbrace{\log \left\{ p(h_1^t) p(x_1^t | h_1^t) \prod_{l=2}^{L} p(x_l^t | h_l^t) \right\}}_{\text{no occurrence of T}} \right)$$

$$\operatorname*{argmax}_T \sum_t \sum_{l=2}^{L} \sum_{\mathbf{h}^t} q(\mathbf{h}^t) \log \left\{ T(h_l^t | h_{l-1}^t) \right\} =$$

$$\operatorname*{argmax}_T \sum_t \sum_{l=2}^{L} \sum_{h_l^t, h_{l-1}^t} q(h_l^t, h_{l-1}^t) \log \left\{ T(h_l^t | h_{l-1}^t) \right\}$$

M-step derivation

Again as with mixing proportions we are learning a matrix of multinomials

$$\sum_a T(a|b) = 1$$

So we need to solve a constrained optimization problem

$$\begin{aligned} &\underset{T}{\text{maximize}} \sum_t \sum_{l=2}^{L} \sum_{h_l^t, h_{l-1}^t} q(h_l^t, h_{l-1}^t) \log \left\{ T(h_l^t | h_{l-1}^t) \right\} \\ &\text{subject to} \sum_a T(a|b) = 1, \forall b \end{aligned}$$

and the Lagrangian for this problem is

$$\begin{aligned} L(T, \lambda) &= \sum_t \sum_{l=2}^{L} \sum_{h_l^t, h_{l-1}^t} q(h_l^t, h_{l-1}^t) \log \left\{ T(h_l^t | h_{l-1}^t) \right\} \\ &+ \sum_b \lambda_b \left(\sum_a T(a|b) - 1 \right) \end{aligned}$$

M-step derivation

The following first order conditions have to hold for an optimum

$$\begin{aligned} \frac{\partial L(T,\lambda)}{\partial T(a|b)} L(T^*,\lambda^*) &= 0 \\ \frac{\partial L(T,\lambda)}{\partial \lambda_b} L(T^*,\lambda^*) &= 0 \end{aligned}$$

and more explicitly

$$\begin{aligned} \sum_t \sum_{l=2}^{L} q(h_l^t = a, h_{l-1}^t = b) \frac{1}{T(a|b)} + \lambda_b &= 0 \\ \sum_a T(a|b) - 1 = 0 \end{aligned}$$

this last part you can push through yourselves to get

$$T^{\text{new}}(a|b) = \frac{\sum_t \sum_{l=2}^{L} q(h_l^t = a, h_{l-1}^t = b)}{\sum_t \sum_{l=2}^{L} q(h_{l-1}^t = b)}$$

M-step derivation

Similar gymnastics lead to

$$\pi_m^{\text{new}} = \frac{\sum_t q(h_1^t = m)}{\sum_t \sum_{h_1^t} q(h_1^t)} = \frac{\sum_t q(h_1^t = m)}{N}$$

M-step derivation

Which marginals of $q(\mathbf{h}^t)$ do we need for the update of ν

$$\nu^{\text{new}} = \underset{\nu}{\text{argmax}} \sum_t \sum_{\mathbf{h}^t} q(\mathbf{h}^t) \log \left\{ p(h_1^t) p(x_1^t | h_1^t) \prod_{l=2}^{L} T(h_l^t | h_{l-1}^t) p(x_l^t | h_l^t) \right\}$$

and we can (and you should!) push through simplification of the update to obtain

$$\begin{aligned} \nu^{\text{new}} &= \underset{\nu}{\text{argmax}} \sum_t \sum_l \sum_{h_l^t} q(h_l^t) \log \left\{ p(x_l^t | h_l^t) \right\} \\ &= \underset{\nu}{\text{argmax}} \sum_t \sum_l \sum_{h_l^t} q(h_l^t) \log \left\{ g(x_l^t | \nu_{h_l^t}) \right\} \end{aligned}$$

Equating the derivative of the expression under argmax with respect to ν and to zero yields the updates.

M-step derivation

In the case of the gaussian distribution specified by μ_k and Σ_k

$$\begin{aligned}
\mu_k^{\text{new}} &= \frac{\sum_t \sum_{l=1}^{L} q(h_l^t = k) x_l^t}{\sum_t \sum_{l=1}^{L} q(h_l^t = k)} \\
\Sigma_k^{\text{new}} &= \frac{\sum_t \sum_{l=1}^{L} q(h_l^t = k) x_l^t (x_l^t)'}{\sum_t \sum_{l=1}^{L} q(h_l^t = k)} - \mu_k^{\text{new}} (\mu_k^{\text{new}})'
\end{aligned}$$

Marginals

Recall that E-step is

$$E: q^{\text{new}} = \operatorname*{argmax}_q \sum_t \sum_{h^t} q(h^t) \log p(x^t, h^t|\theta) - \sum_h q(h^t) \log q(h^t)$$

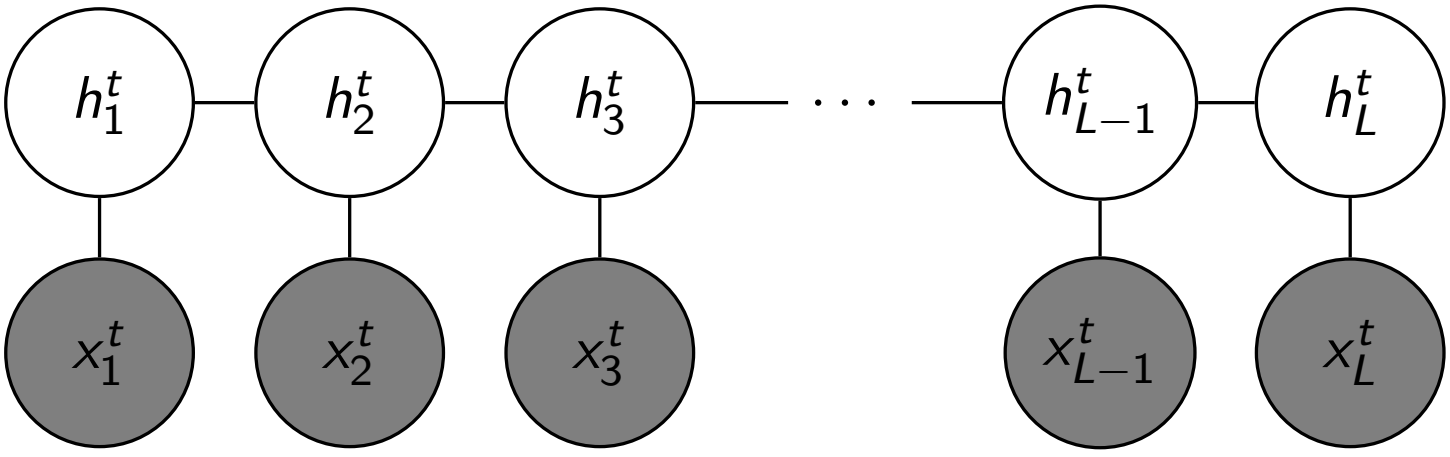
alternatively

$$\operatorname*{argmin}_q \mathrm{KL}(q(h^t)||p(x^t, h^t|\theta))$$

so

$$q(h^t) \propto p(x^t, h^t|\theta)$$

Clique tree



$$q(h^t) = \frac{1}{Z^t} p(h_1^t) p(x_1^t | h_1) \prod_{l=2}^{L-1} p(h_l^t | h_{l-1}^t) p(x_l^t | h_l^t)$$

Sum product instantiation

$$
\begin{aligned}
m_{h_{i-1},h_i}(v) &= \sum_{h_{i-1}} p(h_i|h_{i-1})p(x_i|h_i)m_{h_{i-2},h_{i-1}}(h_{i-1}) \\
m_{h_{i+1},h_i}(v) &= \sum_{h_{i+1}} p(h_{i+1}|h_i)p(x_{i+1}|h_{i+1})m_{h_{i+2},h_{i+1}}(h_{i+1})
\end{aligned}
$$

Once both forward and backward pass are done

$$
\begin{aligned}
q(h_l = v) &= m_{h_{l-1},h_l}(v)m_{h_{l+1},h_l}(v) \\
q(h_l = v_1, h_{l+1} = v_2) &= m_{h_{l-1},h_l}(v_1)p(h_{l+1}|h_l)p(x_l|h_l)m_{h_{l+2},h_{l+1}}(v_2)
\end{aligned}
$$

Staring at more code ...

Conditional Random Fields

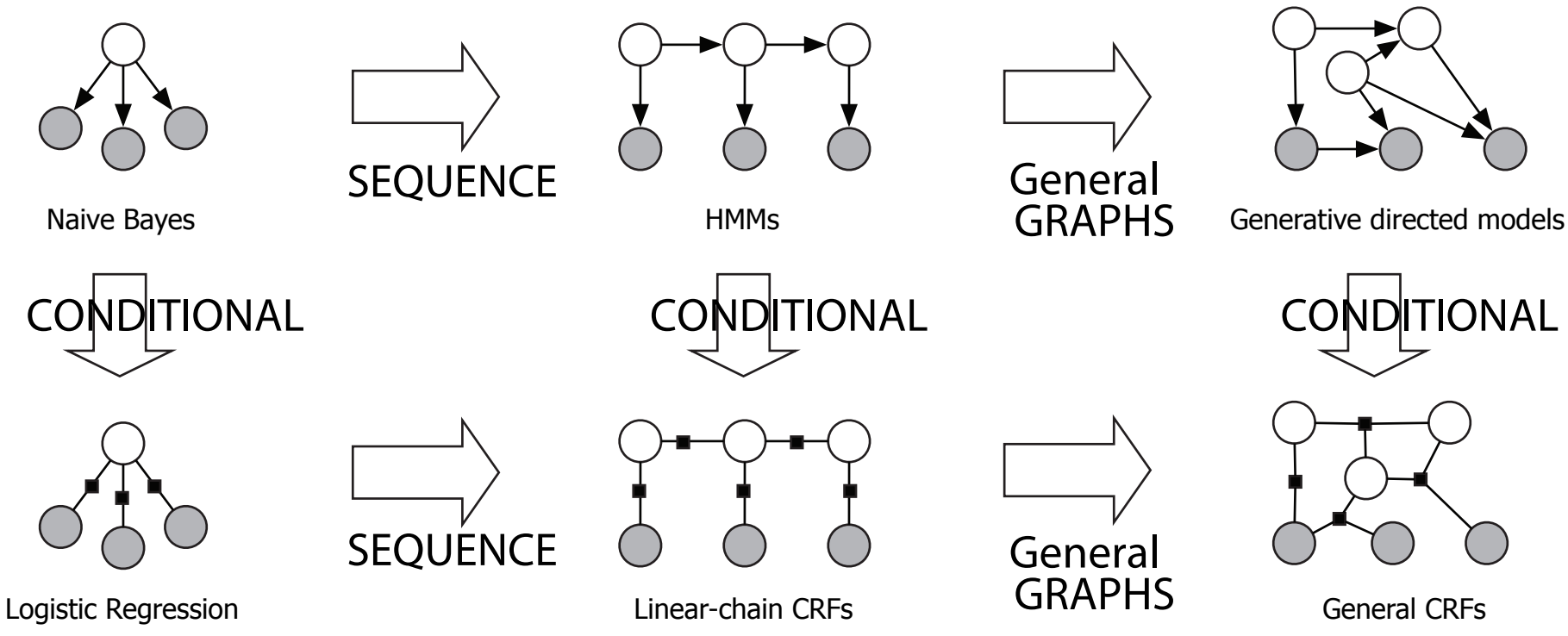
CRFs are the discriminative analog of MRFs.

In particular, a linear CRF is a discriminative analog of HMM.

This is the same relationship that held between Naive Bayes and Logistic Regression.

CRFs can be seen as generalization of Logistic Regression to a structured set of labels.

CRFs and Generative models



Source: Charles Sutton

From HMM to CRF

A joint probability of an HMM (dropping the instance index t for clarity)

$$p(\mathbf{x}, \mathbf{y}) = \prod_l p(x_l|y_l)p(y_l|y_{l-1})$$

can be rewritten as

$$p(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} \exp\left\{\sum_l \log\{p(y_l|y_{l-1})\} + \log\{p(x_l|y_l)\}\right\}$$

and using indicator functions

$$p(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} \exp\left\{\sum_l \sum_{a,b} \lambda_{ab}[y_l = a][y_{l-1} = b] + \sum_l \sum_a \sum_o \xi_{ao}[y_l = a][x_l = o]\right\}$$

in the case of the parameterization we used earlier $\lambda_{ab} = \log T(a|b)$ and $\xi_{ao} = \log g(o; \nu_a)$

From HMM to CRF

$$p(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} \exp \left\{ \sum_l \sum_{a,b} \lambda_{ab}[y_l = a][y_{l-1} = b] + \sum_l \sum_a \sum_o \xi_{ao}[y_l = a][x_l = o] \right\}$$

and we can construct features

$$f_{ab}(y^1, y^2, x) = [y^1 = a][y^2 = b]$$
$$f_{ao}(y^1, y^2, x) = [y^1 = a][x = o]$$

Using this set of features we can rewrite the probability as

$$p(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} \exp \left\{ \sum_l \sum_r \beta_r f_r(y_l, y_{l-1}, x_l) \right\}$$

Finally we obtain the conditional

$$p(\mathbf{y}|\mathbf{x}) = \frac{p(\mathbf{y}, \mathbf{x})}{p(\mathbf{x})} = \frac{\exp \left\{ \sum_l \sum_r \beta_r f_r(y_l, y_{l-1}, x_l) \right\}}{\sum_{\mathbf{h}} \exp \left\{ \sum_l \sum_r \beta_r f_r(h_l, h_{l-1}, x_l) \right\}}$$

Linear chain CRF

The distribution of sequential labels $\mathbf{y}$ given sequential data $\mathbf{x}$

$$p(\mathbf{y}|\mathbf{x}) = \frac{p(\mathbf{y},\mathbf{x})}{p(\mathbf{x})} = \frac{\exp\left\{\sum_l \sum_r \beta_r f_r(y_l, y_{l-1}, x_l)\right\}}{\sum_{\mathbf{h}} \exp\left\{\sum_l \sum_r \beta_r f_r(h_l, h_{l-1}, x_l)\right\}}$$

is called linear-chain Conditional Random Field.

In the context of CRFs both labels $\mathbf{y}$ and $\mathbf{x}$ are observed on the training set and the objective is to maximize the (conditional) log likelihood

$$\begin{aligned}
\mathrm{LL}(\beta) &= \sum_t \log p(\mathbf{y}^t|\mathbf{x}^t) \\
&= \sum_t \sum_l \sum_r \beta_r f_r(y_l^t, y_{l-1}^t, x_l) \\
&- \sum_t \underbrace{\log\left\{\sum_{\mathbf{h}^t} \exp\left\{\sum_l \sum_r \beta_r f_r(h_l^t, h_{l-1}^t, x_l^t)\right\}\right\}}_{\text{log partition function } \log Z(\beta, \mathbf{x}^t)}
\end{aligned}$$

Optimizing the conditional log likelihood for CRF

We maximize $\mathrm{LL}(\beta)$ with respect to β and we can use the same regularization terms as before (ridge/lasso).

The good news is that $\log Z$ is convex in β ($\log \sum \exp$ of a linear combination of βs).

Gradient computation is non-trivial though due to the $\log Z$ term.

$$
\begin{aligned}
\frac{\partial}{\partial \beta_r}\mathrm{LL}(\beta) &= \sum_t \sum_l f_r(y_l^t, y_{l-1}^t, x_l) \\
&- \sum_t \frac{\sum_{\mathbf{h}^t} \exp\left\{\sum_l \sum_r \beta_r f_r(h_l, h_{l-1}, x_l)\right\} \sum_l f_r(h_l, h_{l-1}, x_l)}{\sum_{\mathbf{h}^t} \exp\left\{\sum_l \sum_r \beta_r f_r(h_l^t, h_{l-1}^t, x_l^t)\right\}} \\
&= \underbrace{\sum_t \sum_l f_r(y_l^t, y_{l-1}^t, x_l)}_{\text{feature count in the data}} - \underbrace{\sum_t E_p\left[\sum_l f_r(h_l, h_{l-1}, x_l)\right]}_{\text{expected feature count}}
\end{aligned}
$$

Computing the expectations

We have two types of features

$$\sum_t E_p\left[\sum_l f_{ab}(h_l^t, h_{l-1}^t, x_l^t)\right] = \sum_t \sum_l p(h_l^t, h_{l-1}^t | \mathbf{x}^t, \beta)[h_l = a][h_{l-1}^t = b]$$

$$\sum_t E_p\left[\sum_l f_{ao}(h_l^t, h_{l-1}^t, x_l^t)\right] = \sum_t \sum_l p(h_l^t | \mathbf{x}^t, \beta)[h_l^t = a][x_l^t = o]$$

we can see which marginals we need to compute.

Sum product again

We've done this several times over so I will just write out the potentials

$$
\begin{aligned}
p(\mathbf{h}|\mathbf{x}) &\propto \exp\left\{\sum_l \sum_r \beta_r f_r(h_l, h_{l-1}, x_l)\right\} \\
&= \prod_l \exp\left\{\sum_{ab} \beta_{ab}[h_l = a, h_{l-1} = b]\right\} \exp\left\{\sum_{ai} \beta_{ao}[h_l, x_l = o]\right\} \\
&= \prod_l \phi_l(h_l, h_{l-1})\psi_l(h_l, x_l)
\end{aligned}
$$

So computation of the marginals $p(h_l^t, h_{l-1}^t|\mathbf{x}^t, \beta)$ and $p(h_l^t|\mathbf{x}^t, \beta)$ should be a walk in the arboretum[2]

[2] full of new sights and smells but ultimately just as easy as a walk in the park

Optimization options

$$\frac{\partial}{\partial \beta_r} \mathrm{LL}(\beta) = \underbrace{\sum_t \sum_l f_r(y_l^t, y_{l-1}^t, x_l)}_{\text{feature count in the data}} - \underbrace{\sum_t E_p \left[\sum_l f_r(h_l, h_{l-1}, x_l) \right]}_{\text{expected feature count}} - \sum_r \frac{\gamma}{2} \beta_r$$

You can compute the gradients so options are

- gradient descent
- conjugate gradients
- L-BFGS

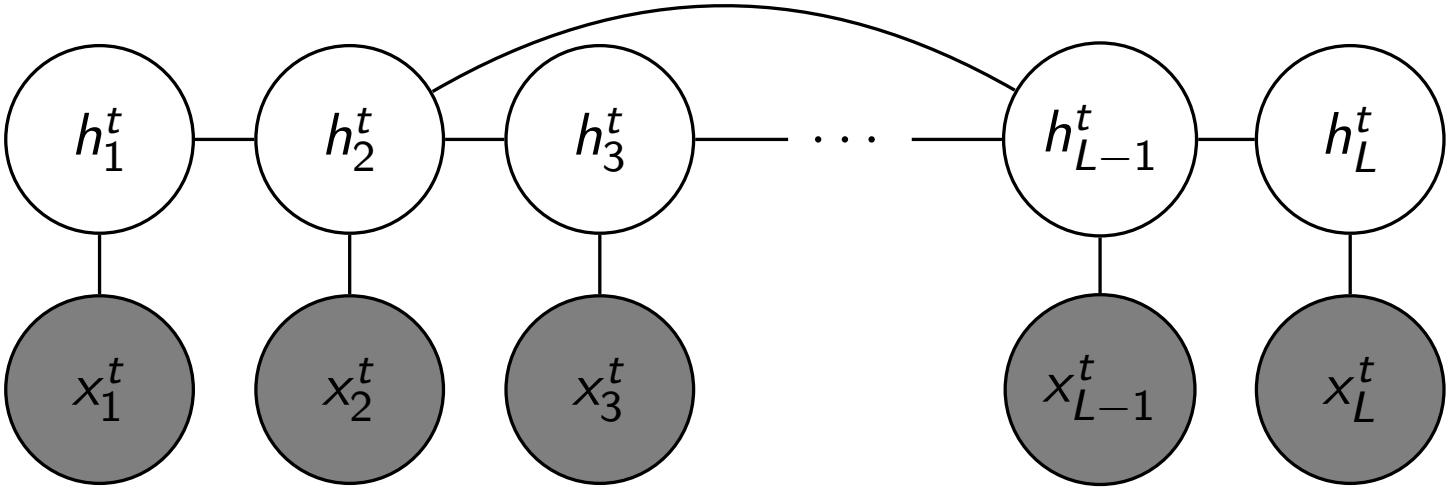
Prediction

Unsurprisingly, you can run max-product on the same distribution

$$
\begin{aligned}
p(\mathbf{h}|\mathbf{x}) &\propto \exp\left\{\sum_l \sum_r \beta_r f_r(h_l, h_{l-1}, x_l)\right\} \\
&= \prod_l \exp\left\{\sum_{ab} \beta_{ab}[h_l = a, h_{l-1} = b]\right\} \exp\left\{\sum_{ai} \beta_{ao}[h_l, x_l = o]\right\} \\
&= \prod_l \phi_l(h_l, h_{l-1})\psi_l(h_l, x_l)
\end{aligned}
$$

to obtain $\mathbf{h}^*$ for a given $\mathbf{x}$, the most likely annotation of the sequence $\mathbf{x}$.

Skip-chain CRF



Features that operate on non-local parts of sequence, e.g.

$$f_r(x_1, x_{100}, h_1, h_{100})$$

Modifying HMM or CRF to include these features can make exact inference intractable.

CRFs are not inherently easier to train than HMMs.

The standard approximation is to run forward-backward without forming the full cliques (loopy belief prop).

Advantages of CRFs

An NLP and CV workhorse, CRFs empirically perform better than HMMs in sequence annotation.

If you are doing sequence annotation or segmentation CRF should be your first choice.

If you are doing unsupervised learning you will have to resort to HMMs.

Either way parameterization and state space structure is the key.

We did ...

- Max-product
- Hidden Markov Models (inference, learning)
- Conditional Random Fields (inference, learning)