Logistics

Reminder: class web links and infrastructure

- Check in with our web page (http://j.ee.washington.edu/~bilmes/classes/ee596a_winter_2013/) for up to date announcements, homeworks, etc.
- All homeworks will be due via our dropbox (https://catalyst.uw.edu/collectit/dropbox/bilmes/25379)
- All questions should be posted to our discussion board (https://catalyst.uw.edu/gopost/board/bilmes/31332)
- You can contact me anonymously if you wish via anonymous email (https://catalyst.uw.edu/umail/form/bilmes/4144)
Relevant Readings: Readings/handouts

- Readings are in a sub-directory “reading_drafts” directly below our web page (http://j.ee.washington.edu/~bilmes/classes/ee596a_winter_2013/).
- uid is this class name (lower case) and pwd are the quarter/year of the class.
- Note, the PDF file is password protected. Send me email if you have trouble (adobe reader should have no problems reading it).

Cumulative Outstanding Reading

- Read the section in ‘doc.pdf’ on the island algorithm.
- NEW: read the section leading to DGMs (including factorial HMMs, and hierarchical HMMs) in 'doc.pdf'
On Final Project

- Will be held Monday, March 18th, 2013.
- Project should ideally be on some aspect of the material we have learnt, some aspect of dynamic graphical models. Possible good projects include:
  - an implementation (i.e., a fast implementation of some DGMs algorithm) and reporting and experience that you gain in doing this. Application to real data.
  - A paper summary, of papers that we are not going to cover in this class.
  - A new idea of your own, new algorithms and/or theoretical results. (e.g., approximation error for a sequential model).
  - Application of a DGM to a data domain (e.g., application of dynamic Bayesian networks to speech/language/biology/surgery or some other sequential data domain).
  - Applied use of GMTK on new applications domain.

The ideal project should be research-oriented, it is not acceptable to propose whatever machine learning task you are currently working on (e.g., “An application of SVMs to protein folding” would not be acceptable).
- should be dynamic
- Ideal project would lead to a conference and/or journal paper.
- Fine to combine it with your own research.
- Deadline today, 2/11, 5:00pm
- Deadline every Monday, 5:00pm up until day of final project 3/18.
Final Project - GMTK

- GMTK (the graphical models toolkit) is a C++ system for expressing and using dynamic Bayesian networks and generalizations on data.
- It is highly optimized in C++ and has many features, but the documentation is currently scattered about (combination latex and wiki).
- Possible project would be to use GMTK for some novel project.

Island algorithm time/space tradeoffs

- We can pay more (in terms of time) to recover some (in terms of memory).
- We can reduce memory complexity of HMM inference to $O(N \log T)$ while the compute complexity increase to $O(N^2T \log T)$.
- That is, additional factor of computation $\log T$ modest, while the reduction factor in memory is large $T/ \log T$.
- Other variants possible as well, for example we can have $O(TN^2)$ compute while using only $O(N\sqrt{T})$ memory, asymptotically always better than the default!!
Island algorithm: idea

- divide a length $T$ segment into $b$ subsegments each of length $T/b$.
- During forward pass, delete memory for every frame of the forward pass except for “islands” that occur only every $T/b$ frames.
- AT (right most) length-$T/b$ subsegment, perform a forward-backwards pass as normal within that subsegment.
- then delete memory associated with this right-most subsegment.
- re-do the forward pass of penultimate segment, can do that since we have stored its left-most frame.
- Then, continue the backwards pass of the penultimate subsegment.
- When done, we delete penultimate subsegment’s memory
- Next, re-do the forward pass of antepenultimate segment, can do that since we have stored its left-most frame.
- Then, continue the backwards pass of the antepenultimate subsegment.
- When done, we delete antepenultimate subsegment’s memory.
- repeat going back.

Island algorithm: idea, recursion

- Each length $T/b$ segment at some point looks like a normal forward/backwards pass.
- Hence, each such forward/backwards pass can have the same process repeated.
- Hence, further divide each of the length-$T/b$ subsegments into subsubsegments of length $T/b^2$.
- Each of the subsubsegments look again like a standard forward-backwards procedure and so can be still further subdivided into length-$T/b^3$ sequence.
- The recursion proceeds to a depth of $k$, leading to segment lengths $T/b^k$ (base case) where we perform normal forward-backward.
- Depth can be decided based on some threshold $\ell$ so that we stop when $T/b^k < \ell$, or we choose $k$ so that $T/b^k \approx 1$. 
Island Steps Figure Discussion

- Previous Figure shows case for $b = 2$, only the top-level recursion is shown.
- At step 4 and step 5, standard forward/backward algorithm on a sequence of length $T/2$ hence, can run the island algorithm recursively on the right-half subsegment.
- Steps 7 and 8 also constitute a length $T/2$ standard forward/backward algorithm, so again island can be applied recursively.
- Within each recursive application lies further standard forward/backwards procedures of successively shorter lengths $T/4$, $T/8$.
- Recurse, until a base-case threshold is reached
- Bottom out with standard standard forward-backward.
Island Analysis

- Standard forward/backward (linear) inference is $O(T)$ memory and $O(T)$ compute (looking only at time for the moment, all costs are really multiplied by the average state cost, $O(|D_Q|^2)$ in the case of time, and $O(|D_Q|)$ in the case of space)
- island case, pay $O(b \times \log_b(T))$ memory and $O(T \log_b(T))$ compute, where $b$, the base of the logarithm, is the number of subsegments a segment is divided into at each recursion level (we used $b = 2$ in figure).
- This can be quite a significant reduction in memory costs, but added computation is nominal (although $\log T$ can be large, say $T = 1024$, and $b = 2$, then we pay an extra time cost of a factor of 10, e.g., 1 day to 10 days)!

Analysis Detail

- We are given $\ell$, the “linear section threshold”
- we keep a linear segment of length $\ell$ in memory, so we divide $T$ into $b^k$ chunks, and once the chunks are smaller than or equal to $\ell$, we can stop dividing.
- We make $k$ as small as possible in order to divide the length $T$ segment into as few as possible chunks, thereby making the final linear chunks as large as possible, while still meeting the constraint that the chunk is $\leq \ell$.
- We can solve for $k$ exactly, as $k = \lceil \log_b(T/\ell) \rceil$. 
at the top level, we need $b - 1$ islands (since we don’t store the right most one).

At the next level, we store only islands at any given time for one of the length $T/b$ segments, and within that segment we store, again at most $b - 1$ islands.

This repeats, and since there are $k$ levels (the depth is $k$), there is an additional memory cost of $k(b - 1)$.

Therefore, for a segment of length $T$, this algorithm never keeps simultaneously in memory more than about $M$ expanded frames, where

$$M = T/b^k + k \cdot (b - 1) \tag{10.1}$$

where $k$ is the smallest integer value such that

$$T/b^k \leq \ell \tag{10.2}$$

and where $\ell$ is the linear segment threshold, i.e., the segment length threshold below which linear forward-backward inference is performed.
Since $k$ is such that $T/b^k \leq \ell$, we keep in memory no more than about

$$M \leq \ell + k \ast (b - 1)$$  \hspace{1cm} (10.3)

segments, so our memory storage requirements is $O(M)$.

Also that $k = \log_b(T/\ell)$.

Ex: if $\ell = 3$, and $b = 3$, then we keep simultaneously in memory no more than about $3 + 2 \log_3(T/3)$ frames, which has only logarithmic growth in $T$. This is where we get the $O(\log_b(T))$ memory.

In general, the memory that we need is

$$\ell + k(b - 1) = \ell + \log_b(T/\ell)(b - 1)$$  \hspace{1cm} (10.4)

$$= \ell + b \log_b(T/\ell) - \log_b(T/\ell)$$  \hspace{1cm} (10.5)

$$= O(\ell + b \log_b(T))$$  \hspace{1cm} (10.6)

frames.

Multiple redundant forward messages but only one sweep of backwards messages (unfortunately, forward messages are often more expensive for DGMs).

Given a length $T$ segment, we need to do $R(T)$ forward messages in total.

We must at least do $T$ forward messages

Must also do $(b - 1)$ forward sweeps through each of the top-level subsegments of length $T/b$,

and the number of forward messages needed for each subsegment is $R(T/b)$.

recurrence relationship:

$$R(T) = T + (b - 1) \ast R(T/b).$$  \hspace{1cm} (10.7)
Time Cost - Solving the Recurrence

- To solve it, we can simplify by bounding $R(T)$ as follows

$$R(T) < T + b \cdot R(T/b)$$

(10.8)

- Then, expand $R(T)$, to get

$$R(T) < T + T + \ldots + T = O(T \log_b T)$$

(10.9)

We are also defining $d = \log_b(T)$ which is an important parameter.

- each forward message (except for the islands themselves) must be computed $d = \log_b(T)$ times, which is what leads to the additional computational penalty factor of $d$.

- In actuality, there are $k$ terms in the sum, since we stop when we’ve reached a threshold of $\ell$.

Tradeoffs

- can decrease running time and increase space requirements either by increasing $b$ (from 2 on up) or alternatively by increasing $\ell$ (from 2 on up).

- we increase memory usage by increasing $b$ since

$$O(b \cdot \log_b(T)) = O((b/\ln(b)) \ln(T))$$

(10.10)

so memory usage is growing as $b/\ln(b)$.

- should set $b$ and $\ell$ such that everything just fits in available main memory.
Setting the parameters

- How to set the parameters $\ell$ and $b$? Several interesting points:
  - First, note that
    
    $$ \argmin_{b \in \{2, 3, 4, \ldots\}} \frac{b}{\ln(b)} = 3 $$  \hspace{1cm} (10.11) 
    
    so $b = 3$ is a good default value for $b$.
  - Then one can set $\ell$ based on how much memory your machine has.
  - In general, one would want to set $\ell$ to be as large as possible without causing the algorithm either to run out of memory or start swapping.

- Alternatively, can take $b = \sqrt{T}$ (so log base changes for different segment lengths).
  - we get memory complexity of
    $$ O(b \times \log_b(T)) = O(\sqrt{T} \times 2) = O(\sqrt{T}) $$ memory
  - compute becomes $O(T \log_b(T)) = O(2 \times T) = O(T)$ compute.
  - so algorithm takes at most twice as long but uses $2/\sqrt{T}$ as much memory.
  - Hence, asymptotically, this means that there is an algorithm that is still linear in $T$ while memory usage is only square-root in $T$!
  - for any $b = T^{1/d}$ (so that $d = \log_b T$), the algorithm has $O(T^{1/d})$ memory and $O(dT)$ compute.
  - asymptotically, therefore, $O(T)$ memory point is suboptimal in the tradeoff curve.
Island and $k$-best

- Island can work with any semi-ring, and can also immediately work with the R-semi-module $K$-best algorithm.

<table>
<thead>
<tr>
<th></th>
<th>compute</th>
<th>memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R$-module</td>
<td>$O(kTN^2)$</td>
<td>$O(kTN^2)$</td>
</tr>
<tr>
<td>max-margin</td>
<td>$O(TN^2 + kT \log(kT) + kTN)$</td>
<td>$O(TN^2 + kT)$</td>
</tr>
<tr>
<td>$R$-module island I</td>
<td>$O(kT \log(T)N^2)$</td>
<td>$O(k \log(T)N^2)$</td>
</tr>
<tr>
<td>$R$-module island II</td>
<td>$O(kTN^2)$</td>
<td>$O(k\sqrt{TN^2})$</td>
</tr>
</tbody>
</table>

Table: Comparison of computational and memory resources needed by various $K$-best algorithms.

Towards Dynamic Bayesian Networks

We’ve been talking about HMMs so far, many of the algorithms for them ($k$-best, etc.). All of the algorithms generalize to DBNs.

The key-idea is expandable models:

- static graphical model, the graph is static and we can use properties of the graph to inference how costly it is to do inference
- simple plate models: repeat a model within a plate (graph node expansion).
- dynamic graphical model: a template plus rules for template expansion. Template & rules are sufficient to infer computational properties of the model at any expansion length.
- general template models: $\exists$ other template + rules models (e.g., PRMs, Markov logic) but there the expanded models can have such flexibility and complexity that it is not possible to infer complexity for any possible expansion based only on the template.
By “structure”, we mean that at each time frame, we have more than just a monolithic integer-valued state variable $Q_t$.

Instead, $Q_t$ might be a vector of values, and each element of this vector might have its own direct relationship with other individual elements within $Q_t$ and other individual elements within $Q_{\tau}$ for various $\tau \neq t$.

We start with some simple generalizations of HMM.

We will start using $Y_t$ rather than $Q_t$ for the state variable (this should be clear before we begin). $X_t$ will still represent the observed vector at each time.

$p(y_t)$ is no longer a distribution over a monolithic integer random variable $Y_t$.

Instead, $Y_t$ is broken down into components that can be represented by a vector of length $M$.

random variable $Y_t$ has the equivalence relationship $Y_t \equiv (Y^1_t, Y^2_t, \ldots, Y^M_t)$.

Domain relationship:

$$D_{Y_t} = D_{Y^1_t} \times D_{Y^2_t} \times D_{Y^3_t} \cdots \times D_{Y^M_t}, \quad (10.12)$$

We also say that $Y_t$ is a flattened representation of $(Y^1_t, Y^2_t, \ldots, Y^M_t)$. 
Vector HMM

- can form an HMM not based on the original flattened random variable $Y_t$ but instead based on the vector of random variables $(Y_t^1, Y_t^2, \ldots, Y_t^M)$.
- resulting joint distribution for this HMM has the form:

$$p(x_1:T, y_1^1:T) = p(y_1^1:T) \prod_{t=2}^T p(x_t | y_1^1:t)p(y_1^1:t | y_{t-1}^1:T) \quad (10.13)$$

- Still temporal Markov property (past independent of future given present), where present is a vector.
- No assumptions about how successive vectors relate to each other.

HMM and Vectorial HMM

HMM with fully coupled factored state space, where $Y_t \equiv (Y_t^1, Y_t^2, \ldots, Y_t^M)$

- Standard HMM

- Equivalent HMM with fully coupled vectorized state space
Vector HMM complexity

- With $M$ elements, Vector HMM has complexity $O(T \left( \prod_{i=1}^{M} N_i \right)^2)$
- But this is same as $O(TN^2)$, so nothing gained nothing lost
- graphical model allows factors of the form
  \[ p(y_t^i, y_{t+1}^i, \ldots, y_t^M, y_t^{1:M}) \quad \text{for} \quad i \in \{1, \ldots, M\} \]
- i.e., any of the variables $Y_t^i$ may directly interact, simultaneously with any of the other variables variables $Y_t^j$ with $j \neq i$, with any of the preceding variables $Y_t^{i'}$ (for any $i'$) or following variables $Y_t^{i''}$ (for any $i''$)
- Only utility of this is representational, where it is useful to think of $Y_t^i$ not as a monolithic integer but as a vector
- e.g., each $Y_t^i$ could represent a dimension in some $M$ dimensional space, so if $M = 2$ or $M = 3$ the hidden variables might represent position in 2D or 3D, robot navigation.
- or , hierarchically related (e.g., in speech recognition, they might represent sentences, words, phones, sub-phones).

Real Structure

- Becomes more important when we start making assumptions about the “structure” or pattern of the direct interactions between variable.
- Terminology Reminder: again, a lack of an edge indicates a conditional independence, an edge in a graphical model means an allowable (but not required) direct interaction between variables (meaning the variables might be involved together in a factor in the probability distribution).
- The next several structures have the above structure, but have edges removed.
The Factorial HMM removes all edges except for those between two successive corresponding variables $Y_{t-1}^i \rightarrow Y_t^i$ (persistent edges), and for those that lead from each state to the observation $Y_t^i \rightarrow X_t$.

The equation for the factorial HMM is:

$$p_{FHMM}(x_{1:T}, y_{1:M}^{1:T}) = p(x_1|y_1^M) \prod_{m=1}^m p(y_t^m) \prod_{t=2}^T p(x_t|y_t^{1:M}) \prod_{m=1}^M p(y_t^m|y_{t-1}^m)$$

(10.14)
Factorial HMM hidden marginal

- we see that

$$\sum_{x_1:T} p_{\text{FHMM}}(x_1:T, y_1^M_1) = \prod_{m=1}^{M} \prod_{t} p(y_t^m | y_{t-1}^m) \quad (10.15)$$

- hence, we are left with a set of $M$ independent Markov chains, meaning that $Y^i_{1:T} \perp \perp Y^j_{1:T}$ for $i \neq j$.
- Model is: multiple independent chains of events that go into explaining the observables.
- Ex: speech signal in a noisy environment with two Markov chains one each for speech and noise.
- Ex: music, learn temporal independent components in a set of, say, Bach chorals.

Factorial HMM inference cost

- $N_f = |D_{Y_i^j}|$ be the common state space of each of the hidden Markov chains in the factorial HMM.
- So $N = (N_f)^M$
- A flattened vectorial HMM complexity, again, would be $O(TN^2) = O(TN_f^{2M})$.
- In factorial HMM, eliminate the variables slice by slice, using the elimination order $Y^M_1, Y^{M-1}_1, \ldots, Y^1_1$
- In fact, we have

**Theorem 10.4.1**

The factorial HMM has an optimal computational complexity of $O(TMN_f^{M+1})$. 
Factorial HMM inference example, $T = 7$, $M = 4$

Moralized Factorial HMM with $T = 7$ and $M = 4$

After eliminating $Y^4_1$
Factorial HMM inference example, $T = 7, M = 4$

After eliminating $Y_2^4$

After eliminating $Y_3^4$
Factorial HMM inference example, $T = 7$, $M = 4$

After eliminating $Y^4_4$. The graph here looks the same as we started, but missing the first frame.

Other elimination orders

- Is there benefit to eliminating variables starting from the middle of the model (i.e., not frame 1 or $T$)?
- Consider:

  - What is clique size of eliminating any middle node?
  - What is clique size of eliminating any $t = 1$ or $t = T$ node?
for long enough $T \geq 3$, the above elimination order cannot be improved upon when one wishes to obtain optimal exact inference. 

- treewidth is $M$
- The elimination procedure has largest clique size of $M + 1$. 
- hence

**Theorem 10.4.2**

*The factorial HMM has an optimal computational complexity of $O(TM^2N^{M+1})$.**
DBNs are just Bayesian networks, where some edges are specific to time (or position).

**Definition 10.4.3 (Classic Definition of a DBN)**

A DBN of length $T$ is a directed acyclic graph $G = (V, E) = (\bigcup_{t=1}^{T} V_t, E_T \cup \bigcup_{t=1}^{T-1} E_t \cup E_t^{-})$ with node set $V$, and edge set $E$ comprising pairs of nodes. If $uv \in E$ for $u, v \in V$, then $uv$ is an edge of $G$. The sets $V_t$ are the nodes at time slice $t$, $E_t$ are the *intra-slice* edges between nodes in $V_t$, and $E_t^{-}$ are the *inter-slice* edges between nodes in $V_t$ and $V_{t+1}$.
A DBN, however, does not typically have this much flexibility.

DBN is specified using a “rolled up” template giving nodes that are repeated in each slice, the intra-slice edges among those nodes, and the inter-slice edges between nodes of adjacent slices.

This template is then unrolled to any desired length $T$ to yield the DBN $G$.

In a DBN, the CPF of a node is shared (or tied) with the CPF of all other nodes that have come from the same underlying node in the DBN template. Therefore, it is possible to represent a DBN of unbounded length, but with only a finite description length and a finite number of parameters.
DBNs and temporal independence

- Like any DGM, a DBN makes temporal Markov assumption: future & past independent given present.
- But this general property is a little more powerful than the classic definition above.
- We may wish a portion of the model to be unique at the beginning and at the end of the sequence (assuming that there is an end).
- Moreover, the repeated section might last longer than one time slice.

More general DBN definition

**Definition 10.4.4 (Definition of a DBN)**

There are three sections of a graph, $\mathcal{G}^p = (V_p, E_p)$ called the prologue, $\mathcal{G}^c = (V_c, E_c)$ called the chunk, and $\mathcal{G}^e = (V_e, E_e)$ called the epilogue. Either $\mathcal{G}^p$ or $\mathcal{G}^c$ may be empty. Each of these sections can be any number of frames long. There are an additional set of edges $E^{++}$ between each of $\mathcal{G}^p$ and $\mathcal{G}^c$, and $\mathcal{G}^c$ and $\mathcal{G}^e$, and these edges might skip chunks. A DBN is a directed acyclic graph $\mathcal{G}$ that is formed by starting with $\mathcal{G}^p$, repeating $\mathcal{G}^c$ any number of times, and ending with $\mathcal{G}^e$. Successive sections are connected with $E^{++}$.

We will make this definition more precise as we go along and show examples.
More general DBN definition: example

DBN vs HMMs

- First of all, an HMM is one instance of a DBN.
- On the other hand, an HMM can often represent a DBN exactly (we saw before all of those amazing properties of HMMs). That is, a DBN can sometimes be flattened into HMM.
Consider the following DBN, extended in time in both directions.

\[ Q_{t-1}^1 \rightarrow Q_t^1 \rightarrow Q_{t+1}^1 \]
\[ \cdots \]
\[ Q_{t-1}^2 \rightarrow Q_t^2 \rightarrow Q_{t+1}^2 \]

- Notice that there are two 1st order Markov chains, and that they are coupled by a link between the two at successive time intervals.
- We have \( p(Q_t^2|Q_{t-1}^2, Q_t^1, Q_{t-1}^1) p(Q_t^1|Q_{t-1}^1) \) at each time.

**Definition 10.4.5 (Persistent edge in a DBN)**

Any edge between two variables identical except for successive time slides is called “persistent”

So the above model has two persistent edges and one non-persistent edge per slice.

---

Considering previous DBN with chains \((Q_t^1, Q_t^2)\).
A flattened HMM would have one chain \( R_t \equiv (Q_t^1, Q_t^2) \) with transition probabilities set as follows:

\[
P(R_t = r_t|R_{t-1} = r_{t-1}) = P(Q_t^1 = q_t^1, Q_t^2 = q_t^2|Q_{t-1}^1 = q_{t-1}^1, Q_{t-1}^2 = q_{t-1}^2)
\]

(10.16)

where \( r_t \equiv (q_t^1, q_t^2) \) is the joint HMM state space. Such flattening, however, ignores the factorization constraint expressed by the graph, which is:

\[
P(Q_t^1 = q_t^1, Q_t^2 = q_t^2|Q_{t-1}^1 = q_{t-1}^1, Q_{t-1}^2 = q_{t-1}^2) = P(Q_t^1 = q_t^1|Q_{t-1}^1 = q_{t-1}^1) P(Q_t^2 = q_t^2|Q_{t-1}^2 = q_{t-1}^2)\]

(10.17)

(10.18)

In other words, not all possible CPFs are allowed given the graph.
Thus, we see that the HMM contains the DBN, but does not express this constraint.

If $G_{hmm}$ is the HMM graph, and $G_{dbn}$ is the just previously mentioned DBN graph, then we have that $\mathcal{F}(G_{hmm}, R) \subseteq \mathcal{F}(G_{dbn}, R)$ and also $\mathcal{F}(G_{dbn}, R) \subseteq \mathcal{F}(G_{hmm}, R)$ so the families are identical.

On the other hand, the DBN expresses constraints. I.e.,

Constrained HMM: could be defined as: $p \in \mathcal{F}(G_{hmm}, R)$ where there exits unique integers $q^1, q^2$ with $r = q^1 + \alpha q^2$, and where there exists CPTs $p(q^1|q^1')$ and $p(q^2|q^1, q^2')$ such that for all values $r$ and $r'$

$$p(r|r') = p(q^1|q^1')p(q^2|q^1, q^2')$$ (10.19)

Factorization constraints are naturally represented by graphical models, thus DBNs in this case.

Simple Hierarchical HMM

- compare this with the speech recognition case study.
This recognizes a single word, consisting of a number of phonetic constituents.

- wavy edges are random, others are deterministic functions.
- This basic structure is a widely used (modular) component in a variety of DBNs, so it is important to understand it.
- There are other modular components of DBNs that, once understood individually, make understanding larger components easier.

Corresponds precisely to the bi-gram \( p(w_t|w_{t-1}) \) ASR system that we saw last lecture.
Corresponds precisely to the tri-gram $p(w_t | w_{t-1}, w_{t-2})$ ASR system that we saw last lecture.

Note how we need to have extra variable to keep track of previous word, regardless of how far back into the past it is.
Linear Lexicons in ASR

Conceptually easy, but large state space (lots of shared prefixes not taken advantage of)

Prefix Tree Lexicons in ASR

State sharing of common word prefixes
Significant state space reduction relative to previous case.
Again, can get the idea of what such a system is doing only by looking at the DBN graph.

With a DBN, observations might be a function of any DBN variable.

What would an HMM need to do to express this?
Explains the variability in the observations with multiple streams that may go in and out of sync.

Generative articulatory modeling in speech recognition.
Can this be flattened into an equivalent HMM?

This only represents a static scene
Associated temporal edges exist as well.

Sources for Today’s Lecture

- “doc.pdf” sections 8.1 - 8.3