Announcements, Assignments, and Reminders

- Visit the URL links that were covered in previous lectures.
- Homework1 is out, due Thursday, May 2nd, at 11:45pm via our dropbox (https://catalyst.uw.edu/collectit/dropbox/bilmes/26924).
Cumulative Outstanding Reading

- Read chapters 1 and 2 in our book (Huang, Acero, Hon, “Spoken Language Processing”).
- Read chapters 3 and 4 in our book (Huang, Acero, Hon, “Spoken Language Processing”).
- Read Chapter 6 in our book (Huang, Acero, Hon, “Spoken Language Processing”).
Speech Perception

- DTW
Speech Perception

- DTW
- DP
Speech Perception

- DTW
- DP
- DTW and speech
DTW Discussion

Advantages of DTW

- easy to implement
- gets reasonable results for limited vocabulary speech recognition
- adequate job aligning templates with different speaking rates

Problems with DTW

- Only one template used per utterance (possible to use multiple templates per utterance, and use the score from the lowest template to determine the unknown utterance)
- Ideally, we might like a template that is an “average” of utterances.
- path constraints are somewhat arbitrary
- path weights are somewhat arbitrary, ideally it should be automatic
- How do we do continuous speech (connected words)? Not so simple to generalize this system
- Not a formal system for dealing with uncertainty. Ideally, we’d like a formal probabilistic schema & use Bayes decision theory.

Solution: From DTW to Hidden Markov Models (HMMs)
Example alignment grid
Aligning source string $x = \text{"hhheeeeeeeelo"}$ with target string $y = \text{"chancellor"}$
Outline of today

- Morph from DTW to HMMs.
- Start HMMs and ASR.
Good books (for today)

- our book (Huang, Acero, Hon, “Spoken Language Processing”)
- Deller et al. “Discrete-time Processing of speech signals”
DTW was used for speech recognition for a long time before HMMs replaced them in the early 80s due to the research of Jelinek, Baker, Rabiner, Poritz, and others (IBM, AT&T, CMU, Dragon).
From DTW to HMM

- DTW was used for speech recognition for a long time before HMMs replaced them in the early 80s due to the research of Jelinek, Baker, Rabiner, Poritz, and others (IBM, AT&T, CMU, Dragon).
- See ICASSP proceedings, late 70s, early 80s.
DTW was used for speech recognition for a long time before HMMs replaced them in the early 80s due to the research of Jelinek, Baker, Rabiner, Poritz, and others (IBM, AT&T, CMU, Dragon).

See ICASSP proceedings, late 70s, early 80s.

Here, we “morph” from DTW to a Hidden Markov model (HMM) one step at a time.
From DTW to HMM

- DTW was used for speech recognition for a long time before HMMs replaced them in the early 80s due to the research of Jelinek, Baker, Rabiner, Poritz, and others (IBM, AT&T, CMU, Dragon).

- See ICASSP proceedings, late 70s, early 80s.

- Here, we “morph” from DTW to a Hidden Markov model (HMM) one step at a time.

- $y$ is template, and $x$ is unknown utterance.
An HMM has a certain number $n = |Q|$ of states.
From DTW to HMM

- An HMM has a certain number $n = |Q|$ of states
- rename the DTW examplar template length $T_y \rightarrow N$
From DTW to HMM

- An HMM has a certain number $n = |Q|$ of states
- rename the DTW examplar template length $T_y \rightarrow N$
- rename the examplar elements $y_1, y_2, \ldots, y_N$ to be the HMM states $q_1, q_2, \ldots, q_N$. 
From DTW to HMM

- An HMM has a certain number $n = |Q|$ of states.
- Rename the DTW examplar template length $T_y \rightarrow N$.
- Rename the examplar elements $y_1, y_2, \ldots, y_N$ to be the HMM states $q_1, q_2, \ldots, q_N$.
- Using the terminology of string alignment introduced last time, the target corresponds to the states, and we are aligning a source sequence of observations $x$ to the target states.

![Diagram of HMM states and observations]
Recall

- next slide from lecture 8.
Templates from path mergings

- Templates constructed from merging of paths:

(A) $P_1 = (+1, 0)$
$P_2 = (+1, +1)$
$P_3 = (+1, +1)$

(B) $P_1 = (+1, 0)$
$P_2 = (+1, +1)$
$P_3 = (0, +1)$

(C) $P_1 = (+1, +1)$
$P_2 = (+1, +1)$
$P_3 = (+1, +1)(0, +1)$

(D) $P_1 = (+2, +1)$
$P_2 = (+1, +1)$
$P_3 = (+1, +2)$

(E) $P_1 = (+1, 0)$
$P_2 = (+1, +1)$
$P_3 = (+1, +2)$
We will use Rabiner HMMs (to be described), so each state corresponds to an emission.
**From DTW to HMM**

- We will use Rabiner HMMs (to be described), so each state corresponds to an emission.
- To avoid this we will disallow purely vertical path transitions (so no insertions are allowed).

![Diagram showing valid and invalid paths]

Valid

Invalid

---

Valid paths such as $P_2$ in (B) and $P_3$ in (C) above do not apply. We must always consume one element of $x$ at each step, also, $x$ is never shorter than the shortest path through the set of states (which depends on the other constraints).

HMM's Markov chain first order, set of next allowable states must be determinable from current state, so paths $P_1$ and (once again) $P_3$ in the pattern (C) also do not apply. $P_1$ in (D) not allowed since that would correspond to moving to a state that emitted two observations.
From DTW to HMM

- We will use Rabiner HMMs (to be described), so each state corresponds to an emission.
- To avoid this we will disallow purely vertical path transitions (so no insertions are allowed).

Valid and Invalid paths such as $P_2$ in (B) and $P_3$ in (C) above do not apply.

Paths such as $P_1$ in (D) not allowed since that would correspond to moving to a state that emitted two observations.
From DTW to HMM

- We will use Rabiner HMMs (to be described), so each state corresponds to an emission.
- to avoid this we will disallow purely vertical path transitions (so no insertions are allowed)

![Diagram of valid and invalid paths]

- paths such as $P_2$ in (B) and $P_3$ in (C) above do not apply.
- we must always consume one element of $x$ at each step,
From DTW to HMM

- We will use Rabiner HMMs (to be described), so each state corresponds to an emission.
- to avoid this we will disallow purely vertical path transitions (so no insertions are allowed)

```
Valid          Invalid
```

- paths such as $P_2$ in (B) and $P_3$ in (C) above do not apply.
- we must always consume one element of $x$ at each step,
- also, $x$ is never shorter than the shortest path through the set of states (which depends on the other constraints).
From DTW to HMM

- We will use Rabiner HMMs (to be described), so each state corresponds to an emission.
- To avoid this we will disallow purely vertical path transitions (so no insertions are allowed).

```
<table>
<thead>
<tr>
<th></th>
<th>Valid</th>
<th>Invalid</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>[ ]</td>
<td>[ ]</td>
</tr>
<tr>
<td></td>
<td>[ ]</td>
<td>[ ]</td>
</tr>
</tbody>
</table>
```

- Paths such as \( P_2 \) in (B) and \( P_3 \) in (C) above do not apply.
- We must always consume one element of \( x \) at each step,
- Also, \( x \) is never shorter than the shortest path through the set of states (which depends on the other constraints).
- HMM's Markov chain first order, set of next allowable states must be determinable from current state, so paths \( P_1 \) and (once again) \( P_3 \) in the pattern (C) also do not apply.
From DTW to HMM

- We will use Rabiner HMMs (to be described), so each state corresponds to an emission.
- To avoid this we will disallow purely vertical path transitions (so no insertions are allowed).

![Valid and Invalid Paths]

- Paths such as $P_2$ in (B) and $P_3$ in (C) above do not apply.
- We must always consume one element of $x$ at each step.
- Also, $x$ is never shorter than the shortest path through the set of states (which depends on the other constraints).
- HMM’s Markov chain first order, set of next allowable states must be determinable from current state, so paths $P_1$ and (once again) $P_3$ in the pattern (C) also do not apply.
- $P_1$ in (D) not allowed since that would correspond to moving to a state that emitted two observations.
restrictions in the previous paragraph can be relaxed if we use an augmented output alphabet.
Restrictions

- restrictions in the previous paragraph can be relaxed if we use an augmented output alphabet.
- For example, if we allow epsilon observations (i.e., an alphabet $\emptyset \cup \{\varepsilon\}$), then strictly vertical arrows (such as $P_2$ in the pattern in-(B)) would be allowed.
Restrictions

- restrictions in the previous paragraph can be relaxed if we use an augmented output alphabet.
- For example, if we allow epsilon observations (i.e., an alphabet $\emptyset \cup \{\varepsilon\}$), then strictly vertical arrows (such as $P_2$ in the pattern in-(B)) would be allowed.
- we assume that these processes have already been done so that the assumptions in the previous paragraph have no loss of generality.
An HMM has a sequence of states corresponding to the Markov chain that is hidden. States named $q_1, q_2, \ldots, q_N$. 

HMMs
DTW to HMMs

- An HMM has a sequence of states corresponding to the Markov chain that is hidden. States named $q_1, q_2, \ldots, q_N$.
- So we align source with target (sequence of states).
DTW to HMMs

- An HMM has a sequence of states corresponding to the Markov chain that is hidden. States named $q_1, q_2, \ldots, q_N$.
- So we align source with target (sequence of states).
- Given an aligned state sequence, each state corresponds to consuming one input symbol.
DTW to HMMs

- An HMM has a sequence of states corresponding to the Markov chain that is hidden. States named $q_1, q_2, \ldots, q_N$.
- So we align source with target (sequence of states).
- Given an aligned state sequence, each state corresponds to consuming one input symbol.
- We can view this system as a finite state automata (or FSA).
An HMM has a sequence of states corresponding to the Markov chain that is hidden. States named $q_1, q_2, \ldots, q_N$.

So we align source with target (sequence of states).

Given an aligned state sequence, each state corresponds to consuming one input symbol.

We can view this system as a finite state automata (or FSA).

Original DTW path constraints determine allowable state transitions in Weighted Finite State Automata (WFSA).
states of an HMM have a natural ordering, and the transition matrix will obey this ordering by requiring that only strictly left-to-right transitions are possible.
states of an HMM have a natural ordering, and the transition matrix will obey this ordering by requiring that only strictly left-to-right transitions are possible.

E.g., sub-phones of a phone, phones of a word, words of a sentence, all of which is (mostly if not all) left to right.
DTW to HMMs

- states of an HMM have a natural ordering, and the transition matrix will obey this ordering by requiring that only strictly left-to-right transitions are possible.
- E.g., sub-phones of a phone, phones of a word, words of a sentence, all of which is (mostly if not all) left to right.
- banded upper diagonal transition matrix of a Markov chain.
states of an HMM have a natural ordering, and the transition matrix will obey this ordering by requiring that only strictly left-to-right transitions are possible.

E.g., sub-phones of a phone, phones of a word, words of a sentence, all of which is (mostly if not all) left to right.

banded upper diagonal transition matrix of a Markov chain.

Path constraints in DTW determine the structure of the HMM transition matrix.
DTW templates (left) and their corresponding Markov chains transition diagrams

- Path constraints in DTW determine the structure of the HMM transition matrix.
DTW templates (left) and their corresponding Markov chains transition diagrams

- Path constraints in DTW determine the structure of the HMM transition matrix.
- DTW horizontal path $\Rightarrow$ FSA self-transition
DTW templates (left) and their corresponding Markov chains transition diagrams

- Path constraints in DTW determine the structure of the HMM transition matrix.
- DTW horizontal path $\equiv$ FSA self-transition
- **Sequential strictly left-to-right**
DTW templates (left) and their corresponding Markov chains transition diagrams

- Path constraints in DTW determine the structure of the HMM transition matrix.
- DTW horizontal path == FSA self-transition
- Sequential strictly left-to-right
- Upper triangular strictly left-to-right
DTW templates (left) and their corresponding Markov chains transition diagrams

- **Successive bidirectional cycle**
DTW templates (left) and their corresponding Markov chains transition diagrams

- Successive bidirectional cycle

- Can encode preferences to stay in same state, move to next state, etc.
DTW templates (left) and their corresponding Markov chains transition diagrams

- Successive bidirectional cycle

- can encode preferences to stay in same state, move to next state, etc.
- vowels should want to stay longer in current state than consonants.
DTW templates (left) and their corresponding Markov chains transition diagrams

- Successive bidirectional cycle

- can encode preferences to stay in same state, move to next state, etc.
- vowels should want to stay longer in current state than consonants.
- Above example shows how cycles in HMM correspond to downward links in DTW.
path weights in DTW were encoded with $w^P(t) \in \mathbb{R}_+$ where $P$ is a path and $t$ is the relative position within this path.
- path weights in DTW were encoded with $w^P(t) \in \mathbb{R}_+$ where $P$ is a path and $t$ is the relative position within this path.

- weight might depend on where the path is applied in the DTW grid, even meaning that if $w^P(t_0)$ were applied in two different locations in the grid, we might have different scores.
path weights in DTW were encoded with $w^P(t) \in \mathbb{R}_+$ where $P$ is a path and $t$ is the relative position within this path.

- weight might depend on where the path is applied in the DTW grid, even meaning that if $w^P(t_0)$ were applied in two different locations in the grid, we might have different scores.

- Assume weights are the same regardless of where the path is applied, so that if a path is applied in very different parts of the grid, the path weights would be the same.
path weights in DTW were encoded with $w^P(t) \in \mathbb{R}_+$ where $P$ is a path and $t$ is the relative position within this path.

weight might depend on where the path is applied in the DTW grid, even meaning that if $w^P(t_0)$ were applied in two different locations in the grid, we might have different scores.

Assume weights are the same regardless of where the path is applied, so that if a path is applied in very different parts of the grid, the path weights would be the same.

Hence, it is possible to encode the path constraints using a form of transition matrix $W = [w_{ij}]_{ij}$ where $w_{ij} \geq 0$. 


path weights in DTW were encoded with $w^P(t) \in \mathbb{R}_+$ where $P$ is a path and $t$ is the relative position within this path.

- Weight might depend on where the path is applied in the DTW grid, even meaning that if $w^P(t_0)$ were applied in two different locations in the grid, we might have different scores.

- Assume weights are the same regardless of where the path is applied, so that if a path is applied in very different parts of the grid, the path weights would be the same.

- Hence, it is possible to encode the path constraints using a form of transition matrix $W = [w_{ij}]_{ij}$ where $w_{ij} \geq 0$.

- Moving from state $i$ to $j$ costs $w_{ij}$ regardless of where in the grid such a transition occurs.
path weights in DTW were encoded with $w^P(t) \in \mathbb{R}_+$ where $P$ is a path and $t$ is the relative position within this path.

weight might depend on where the path is applied in the DTW grid, even meaning that if $w^P(t_0)$ were applied in two different locations in the grid, we might have different scores.

Assume weights are the same regardless of where the path is applied, so that if a path is applied in very different parts of the grid, the path weights would be the same.

Hence, it is possible to encode the path constraints using a form of transition matrix $W = [w_{ij}]_{ij}$ where $w_{ij} \geq 0$.

moving from state $i$ to $j$ costs $w_{ij}$ regardless of where in the grid such a transition occurs

A time-homogeneous assumption when horizontal axis seen as time.
In general, DTW weights can be a function of two template frames $i$ & $j$ (not just $i - j$).
DTW weights

- In general, DTW weights can be a function of two template frames $i$ & $j$ (not just $i - j$).
- Example (note, infinite costs are not drawn in state transition graph, correspond to no possible state transition case).
DTW weights

- In general, DTW weights can be a function of two template frames $i$ & $j$ (not just $i - j$).
- Example (note, infinite costs are not drawn in state transition graph, correspond to no possible state transition case).
DTW weights

- In general, DTW weights can be a function of two template frames $i$ & $j$ (not just $i - j$).
- Example (note, infinite costs are not drawn in state transition graph, correspond to no possible state transition case).

$$W = \begin{bmatrix} w_{ij} \end{bmatrix}_{i,j} = \begin{bmatrix} 1 & 4 & 3 & \infty & \infty & 3 & \infty & 8 & \infty & 7 & 2 & \infty & \infty & \infty & \infty \end{bmatrix}$$ (9.1)
In general, DTW weights can be a function of two template frames $i$ & $j$ (not just $i - j$).

Example (note, infinite costs are not drawn in state transition graph, correspond to no possible state transition case).

$$W = [w_{ij}]_{i,j} = \begin{pmatrix} 1 & 4 & 3 & \infty \\ \infty & 3 & \infty & 8 \\ \infty & 7 & 2 & \infty \\ \infty & \infty & \infty & \infty \end{pmatrix}$$ (9.1)
Costs in DTW

- need mapping between the costs in DTW and an observation distribution in an HMM.
Costs in DTW

- need mapping between the costs in DTW and an observation distribution in an HMM.
- The DTW cost is between any pair of element $x_i$ and $y_j$ and is computed using the function $d(x_i, y_j)$
Costs in DTW

- need mapping between the costs in DTW and an observation distribution in an HMM.
- The DTW cost is between any pair of element \( x_i \) and \( y_j \) and is computed using the function \( d(x_i, y_j) \)
- Note that if \( d(x_i, y_j) \) is the Mahalanobis distance, then the transformation \( \frac{1}{|2\pi A|^{1/2}} e^{\frac{1}{2} (x-y_j)^\top A^{-1} (x-y_j)} \) results in a Gaussian density on \( x \) with mean \( y \) and covariance matrix \( A \).
Costs in DTW

- need mapping between the costs in DTW and an observation distribution in an HMM.

- The DTW cost is between any pair of element $x_i$ and $y_j$ and is computed using the function $d(x_i, y_j)$

- Note that if $d(x_i, y_j)$ is the Mahalanobis distance, then the transformation

\[
\frac{1}{|2\pi A|^{1/2}} e^{-\frac{1}{2} (x-y_j)^\top A^{-1} (x-y_j)}
\]

results in a Gaussian density on $x$ with mean $y$ and covariance matrix $A$.

- Hence, if $p(x|y_q)$ is the observation distribution given state $q$, then

\[- \log p(x_i|y_q) + c = d(i, q)\]

is the mapping between HMM observation probability and DTW cost.
need mapping between the costs in DTW and an observation distribution in an HMM.

The DTW cost is between any pair of element \( x_i \) and \( y_j \) and is computed using the function \( d(x_i, y_j) \).

Note that if \( d(x_i, y_j) \) is the Mahalanobis distance, then the transformation

\[
\frac{1}{|2\pi A|^{1/2}} e^{\frac{1}{2}(x-y_j)^\top A^{-1}(x-y_j)}
\]

results in a Gaussian density on \( x \) with mean \( y \) and covariance matrix \( A \).

Hence, if \( p(x|y_q) \) is the observation distribution given state \( q \), then

\[-\log p(x_i|y_q) + c = d(i, q)\]

is the mapping between HMM observation probability and DTW cost.

the “states” in the HMM select a Gaussian mean and covariance matrix, and the so DTW distance is seen as a weighted distance to the mean of a Gaussian.
Costs in DTW

- need mapping between the costs in DTW and an observation distribution in an HMM.
- The DTW cost is between any pair of element $x_i$ and $y_j$ and is computed using the function $d(x_i, y_j)$
- Note that if $d(x_i, y_j)$ is the Mahalanobis distance, then the transformation
  \[ \frac{1}{|2\pi A|^{1/2}} e^{\frac{1}{2} (x-y_j)^\top A^{-1} (x-y_j)} \]
  results in a Gaussian density on $x$ with mean $y$ and covariance matrix $A$.
- Hence, if $p(x|y_q)$ is the observation distribution given state $q$, then
  \[ -\log p(x_i|y_q) + c = d(i, q) \]
  is the mapping between HMM observation probability and DTW cost.
- the “states” in the HMM select a Gaussian mean and covariance matrix, and the so DTW distance is seen as a weighted distance to the mean of a Gaussian.
- mapping can be used for all distances. E.g., if $p(x_i|y_q)$ is a Gaussian mixture, above mapping can be used as well to produce a distance.
Cost/Probability Mapping

At current frame, we have:

\[ d(x, y_{q_i}) = -\log p(x|q_i) + c \]  \hspace{1cm} (9.2)

or

\[ p(x|q_i) = e^{-d(x,y_{q_i})+c} \]  \hspace{1cm} (9.3)
Cost/Probability Mapping

- At current frame, we have:

\[ d(x, y_{q_i}) = -\log p(x|q_i) + c \]  \hspace{1cm} (9.2)

or

\[ p(x|q_i) = e^{-d(x, y_{q_i}) + c} \]  \hspace{1cm} (9.3)

- So as distance goes down, the probability goes up.
Costs and score

- With the above two transformations, can consider computing the "score" of a given path in a DTW vs. the score of a comparable HMM.
Costs and score

- With the above two transformations, can consider computing the “score” of a given path in a DTW vs. the score of a comparable HMM.
- Consider path;
Costs and score

- With the above two transformations, can consider computing the "score" of a given path in a DTW vs. the score of a comparable HMM.

- Consider path:

  ![Diagram of DTW grid](image)

- DTW grid is essentially the same as an HMM trellis (to be describe).
When viewed as an HMM, the probability of this path becomes:

\[
\Pr(x_1, x_2, \ldots, x_7, \text{path in figure}) = p(x_1|q_1)p(x_2|q_1)p(x_3|q_2)p(x_4|q_2)p(x_5|q_3)p(x_6|q_4)p(x_7|q_5) \\
\pi_1 a_{11} a_{12} a_{22} a_{23} a_{34} a_{45}
\]

where \( \pi_i = \Pr(Q_1 = i) \) is the initial state probability.
Costs and score

- When viewed as an HMM, the probability of this path becomes:

\[
\Pr(x_1, x_2, \ldots, x_7, \text{path in figure}) = p(x_1|q_1)p(x_2|q_1)p(x_3|q_2)p(x_4|q_2)p(x_5|q_3)p(x_6|q_4)p(x_7|q_5)
\pi_1a_{11}a_{12}a_{22}a_{23}a_{34}a_{45}
\]

(9.4)

where \(\pi_i = \Pr(Q_1 = i)\) is the initial state probability.

- The DTW score of the path in figure as follows:

\[
d(x_1, q_1)w_{01} + d(x_2, q_1)w_{11} + d(x_3, q_2)w_{12} \\
+ d(x_4, q_2)w_{22} + d(x_5, q_3)w_{23} + d(x_5, q_4)w_{34} + d(x_7, q_5)w_{45}
\]

(9.5)

where \(w_{01}\) is a preference for starting at state 1.
Costs and score

To compare the HMM and the DTW scores more directly, we use the following relationships between DTW score and HMM probability:

\[ a_{ij} = \frac{e^{-w_{ij}}}{\sum_{j'} e^{-w_{ij'}}} \quad \text{and} \quad p(x|q_i) = e^{-d(x,y_{q_i})+c} \quad (9.6) \]
Costs and score

- to compare the HMM and the DTW scores more directly, we use the following relationships between DTW score and HMM probability:

\[
a_{ij} = \frac{e^{-w_{ij}}}{\sum_{j'} e^{-w_{ij'}}}
\]

and

\[
p(x|q_i) = e^{-d(x, y_{q_i}) + c}
\]  

(9.6)

- In the case of DTW, the score associated with this path is:

\[
d_{\phi}(x, y) = \sum_{t=1}^{T} d(\phi_x(t), \phi_y(t))w_t^{\phi}
\]  

(9.7)
Costs and score

- To compare the HMM and the DTW scores more directly, we use the following relationships between DTW score and HMM probability:

\[ a_{ij} = \frac{e^{-w_{ij}}}{ \sum_{j'} e^{-w_{ij'}}} \quad \text{and} \quad p(x|q_i) = e^{-d(x,y_{q_i}) + c} \quad (9.6) \]

- In the case of DTW, the score associated with this path is:

\[ d_\phi(x, y) = \sum_{t=1}^{T} d(\phi_x(t), \phi_y(t))w_\phi^t \quad (9.7) \]

- In the case of HMMs, the score (i.e., log probability) associated with this path is:

\[ -\log \Pr(x_{1:T}, q_{1:T}) = -\sum_{t=1}^{T} d(x_t, y_{q_t}) + w_{q_t,q_{t+1}} + c \quad (9.8) \]

where \( c \) is a constant.
State duration distributions

- State duration distribution is geometric, namely:

\[
\Pr(Duration_i = \ell) = a_{ii}^{\ell-1}(1 - a_{ii}) \quad (9.9)
\]
State duration distributions

- State duration distribution is geometric, namely:

\[ \Pr(Duration_i = \ell) = a_{ii}^{\ell-1}(1 - a_{ii}) \]  

(9.9)

- Example duration distributions:
State duration distributions

- State duration distribution is geometric, namely:

  \[ \Pr(\text{Duration}_i = \ell) = a_{ii}^{\ell-1}(1 - a_{ii}) \]  

- Example duration distributions:

- Is this a good duration distribution for phones? vowels? consonants?
State duration distributions

- Sum of geometric random variables is negative binomial.
State duration distributions

- Sum of geometric random variables is negative binomial.

Can mimic this by chaining multiple states together — i.e., the duration distribution of a string of states each with self-transition is negative binomial.
DTW to HMM

- Stochastic transition matrix with observation annotations at states.
Min cost path vs. most probable (or Viterbi) path

- Bayes decision is how we make decisions about a set of words, i.e.:

\[ W^* = \arg\max_W \Pr(x_1:T|W)p(W) \]  \hspace{1cm} (9.10)

where \( W \) is a sequence of words, \( \Pr(x_1:T|W) \) is an HMM model, \( p(W) \) is a language model, \( W = (W_1, W_2, \ldots, W_N) \) is a string of words.
Min cost path vs. most probable (or Viterbi) path

1. Bayes decision is how we make decisions about a set of words, i.e.:

   \[ W^* = \arg\max_W \Pr(x_1:T|W)p(W) \]  \hspace{1cm} (9.10)

   where \( W \) is a sequence of words, \( \Pr(x_1:T|W) \) is an HMM model, \( p(W) \) is a language model, \( W = (W_1, W_2, \ldots, W_N) \) is string of words.

2. In DTW, the score is the minimum cost path, i.e.,

   \[ d(x, y) = \min_{\phi} d_\phi(x, y) \]  \hspace{1cm} (9.11)

   In an HMM, via the way cost is related to probability, we need the most probable path (called Viterbi path), namely:

   \[ \text{Viterbi score} \left(x_1:T\right) = \max_{\text{all paths}} \Pr(x_1:T, \text{path}) \]  \hspace{1cm} (9.12)

   \[ \text{Viterbi path} = \arg\max_{\text{all paths}} \Pr(x_1:T, \text{path}) \]  \hspace{1cm} (9.13)

This last equation is called Viterbi decoding.
Min cost path vs. most probable (or Viterbi) path

- Bayes decision is how we make decisions about a set of words, i.e.:
  \[
  W^* = \arg\max_W \Pr(x_{1:T}|W)p(W) \tag{9.10}
  \]
  where \( W \) is a sequence of words, \( \Pr(x_{1:T}|W) \) is an HMM model, \( p(W) \) is a language model, \( W = (W_1, W_2, \ldots, W_N) \) is string of words.

- In DTW, the score is the minimum cost path, i.e.,
  \[
  d(x, y) = \min_{\phi} d_\phi(x, y) \tag{9.11}
  \]

- In an HMM, via the way cost is related to probability, we need the most probable path (called Viterbi path), namely:
  \[
  \text{Viterbi score}(x_{1:T}) = \max_{\text{all paths}} \Pr(x_{1:T}, \text{path}) \tag{9.12}
  \]
  \[
  \text{Viterbi path} = \arg\max_{\text{all paths}} \Pr(x_{1:T}, \text{path}) \tag{9.13}
  \]
Min cost path vs. most probable (or Viterbi) path

- Bayes decision is how we make decisions about a set of words, i.e.:
  \[ W^* = \operatorname*{argmax}_W \Pr(x_1:T|W)p(W) \]  
  \[ (9.10) \]
  where \( W \) is a sequence of words, \( \Pr(x_1:T|W) \) is an HMM model, \( p(W) \) is a language model, \( W = (W_1, W_2, \ldots, W_N) \) is a string of words

- In DTW, the score is the minimum cost path, i.e.,
  \[ d(x, y) = \min_{\phi} d_\phi(x, y) \]  
  \[ (9.11) \]

- In an HMM, via the way cost is related to probability, we need the most probable path (called Viterbi path), namely:
  \[ \text{Viterbiscore}(x_1:T) = \max_{\text{all paths}} \Pr(x_1:T, \text{path}) \]  
  \[ (9.12) \]
  
  \[ \text{Viterbi path} = \operatorname*{argmax}_{\text{all paths}} \Pr(x_1:T, \text{path}) \]  
  \[ (9.13) \]

- This last equation is called Viterbi decoding
Costs and score: distance similarities and differences

- DTW has multiplicative but HMM has additive weight (although HMM can do this two with Jelinek HMMs to be defined).
Costs and score: distance similarities and differences

- DTW has multiplicative but HMM has additive weight (although HMM can do this two with Jelinek HMMs to be defined).
  - in DTW, weight has interpretation of slope penalty.
Costs and score: distance similarities and differences

- DTW has multiplicative but HMM has additive weight (although HMM can do this two with Jelinek HMMs to be defined).
  - in DTW, weight has interpretation of slope penalty.
  - in HMM, “weight” has interpretation of state transition.
Costs and score: distance similarities and differences

- DTW has multiplicative but HMM has additive weight (although HMM can do this two with Jelinek HMMs to be defined).
  - in DTW, weight has interpretation of slope penalty.
  - in HMM, “weight” has interpretation of state transition.

- HMMs “absorbs” an utterance frame (but can be extended)
Costs and score: distance similarities and differences

- DTW has multiplicative but HMM has additive weight (although HMM can do this two with Jelinek HMMs to be defined).
  - in DTW, weight has interpretation of slope penalty.
  - in HMM, “weight” has interpretation of state transition.
- HMMs “absorbs” an utterance frame (but can be extended)
- HMMs use stochastic matrix for state transitions
Costs and score: distance similarities and differences

- DTW has multiplicative but HMM has additive weight (although HMM can do this two with Jelinek HMMs to be defined).
  - in DTW, weight has interpretation of slope penalty.
  - in HMM, “weight” has interpretation of state transition.
- HMMs “absorbs” an utterance frame (but can be extended)
- HMMs use stochastic matrix for state transitions
- DTW has two stage moves (can skip utterance frames), while HMM can simulate this with alphabet addition and more states
Costs and score: distance similarities and differences

- DTW has multiplicative but HMM has additive weight (although HMM can do this two with Jelinek HMMs to be defined).
  - in DTW, weight has interpretation of slope penalty.
  - in HMM, “weight” has interpretation of state transition.
- HMMs “absorbs” an utterance frame (but can be extended)
- HMMs use stochastic matrix for state transitions
- DTW has two stage moves (can skip utterance frames), while HMM can simulate this with alphabet addition and more states
- HMMs might not allow utterances to be shorter than number states.
Costs and score: distance similarities and differences

- DTW has multiplicative but HMM has additive weight (although HMM can do this two with Jelinek HMMs to be defined).
  - in DTW, weight has interpretation of slope penalty.
  - in HMM, “weight” has interpretation of state transition.
- HMMs “absorbs” an utterance frame (but can be extended)
- HMMs use stochastic matrix for state transitions
- DTW has two stage moves (can skip utterance frames), while HMM can simulate this with alphabet addition and more states
- HMMs might not allow utterances to be shorter than number states.
- HMM has nice probabilistic interpretation, in fact:

\[
\int_x \sum_{\text{all paths}} p(x, \text{path}) = 1 \quad (9.14)
\]
Costs and score: distance similarities and differences

- DTW has multiplicative but HMM has additive weight (although HMM can do this two with Jelinek HMMs to be defined).
  - in DTW, weight has interpretation of slope penalty.
  - in HMM, “weight” has interpretation of state transition.

- HMMs “absorbs” an utterance frame (but can be extended)

- HMMs use stochastic matrix for state transitions

- DTW has two stage moves (can skip utterance frames), while HMM can simulate this with alphabet addition and more states

- HMMs might not allow utterances to be shorter than number states.

- HMM has nice probabilistic interpretation, in fact:
  \[
  \int_x \sum_{\text{all paths}} p(x, \text{path}) = 1 \quad (9.14)
  \]

- Next, we turn our attention to HMMs.
Hidden Markov Models (HMMs)

- Hidden Markov Models (HMMs) are a ubiquitously used model in many fields, including speech recognition, natural language processing, bioinformatics, financial markets, and many time-series problems.
Hidden Markov Models (HMMs)

- Hidden Markov Models (HMMs) are a ubiquitously used model in many fields, including speech recognition, natural language processing, bioinformatics, financial markets, and many time-series problems.

- HMMs are powerful and diverse enough to be interesting, but still simple enough so that they are often computationally feasible (it is also a good example to get a strong feel for exact probabilistic inference).
Hidden Markov Models (HMMs) are a ubiquitously used model in many fields, including speech recognition, natural language processing, bioinformatics, financial markets, and many time-series problems.

HMMs are powerful and diverse enough to be interesting, but still simple enough so that they are often computationally feasible (it is also a good example to get a strong feel for exact probabilistic inference).

HMMs can be described in many ways, including using one of a variety of different graphical models (e.g., Bayesian networks or Markov random fields, which means that they are decomposable).
Hidden Markov Models (HMMs)

- Hidden Markov Models (HMMs) are a ubiquitously used model in many fields, including speech recognition, natural language processing, bioinformatics, financial markets, and many time-series problems.
- HMMs are powerful and diverse enough to be interesting, but still simple enough so that they are often computationally feasible (it is also a good example to get a strong feel for exact probabilistic inference).
- HMMs can be described in many ways, including using one of a variety of different graphical models (e.g., Bayesian networks or Markov random fields, which means that they are decomposable).
- As we will see, HMMs are relatively easy to deal with ($O(TN^2)$ complexity and $O(TN)$ memory).
Recall

- The following slide is from Lecture 1.
Inspiration: Communication and Information Theory

Semantic Intent \( S \) \( \rightarrow \) Linguistic Representation \( L \) \( \rightarrow \) Word Sequence \( W \) \( \rightarrow \) Sub-Word Sequence \( U \)

glotal & articulatory control sequence \( A \) \( \rightarrow \) Markov state sequence \( Q \) \( \rightarrow \) acoustic waveform \( X \)

acoustic waveform at ear \( Y \)
Speech Encoding via an HMM and (Viterbi) Decoding

- Why is it called (Viterbi) decoding?
Speech Encoding via an HMM and (Viterbi) Decoding

- Why is it called (Viterbi) decoding?
- **Source-channel model of communications (from Information Theory)**

![Diagram of source-channel model](image)

Consider the source being generated by a Markov chain, and the "channel" being each symbol corrupted by some channel noise (observation distribution).
Speech Encoding via an HMM and (Viterbi) Decoding

- Why is it called (Viterbi) decoding?
- Source-channel model of communications (from Information Theory)

Consider the source being generated by Markov chain, and the “channel” being each symbol corrupted by some channel noise (observation distribution).
Speech Encoding via an HMM and (Viterbi) Decoding

- Why is it called (Viterbi) decoding?
- Source-channel model of communications (from Information Theory)

Consider the source being generated by Markov chain, and the “channel” being each symbol corrupted by some channel noise (observation distribution).
Homogeneous Markov chains

- The statistical evolution of a 1st-order Markov chain is determined by the state transition probabilities $a_{ij}(t) \overset{\Delta}{=} P(Q_t = j|Q_{t-1} = i)$.
- function both of the states at successive time steps and of the current time $t$.
- sometimes, there is no dependence on $t$, and this is called *time-homogeneous* (or just *homogeneous*) because $a_{ij}(t) = a_{ij}$ for all $t$.
- Hence, a transition matrix $A$ with $(i,j)^{th}$ entry $a_{ij}$ is sufficient to represent all transition probabilities in a time-homogeneous 1st-order Markov chain.
Homogeneous Markov Chain - Transition Matrix/Graph

\[ A = \begin{pmatrix}
    a_{00} & a_{01} & a_{02} & 0 & 0 & 0 & 0 & 0 \\
    0 & a_{11} & 0 & a_{13} & a_{14} & 0 & 0 & 0 \\
    0 & 0 & a_{22} & a_{23} & 0 & 0 & a_{26} & 0 \\
    0 & 0 & 0 & a_{33} & a_{34} & a_{35} & 0 & 0 \\
    0 & 0 & 0 & 0 & 0 & 0 & a_{46} & 0 \\
    0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{57} \\
    0 & a_{61} & 0 & 0 & 0 & 0 & 0 & a_{67} \\
    a_{70} & 0 & 0 & 0 & 0 & 0 & 0 & a_{77}
\end{pmatrix} \]
Graphical model gives the factorization assumptions about a joint probability distribution $p(x_1, x_2, \ldots, x_5)$. 
Graphical model gives the factorization assumptions about a joint probability distribution $p(x_1, x_2, \ldots, x_5)$.

On the left is a 1st order Markov chain, so

$$p(x_1, \ldots, x_5) = p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_3)p(x_5|x_4).$$
Markov Chains and Graphical Models

- Graphical model gives the factorization assumptions about a joint probability distribution $p(x_1, x_2, \ldots, x_5)$.

- On the left is a 1st order Markov chain, so
  $$p(x_1, \ldots, x_5) = p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_3)p(x_5|x_4).$$

- The structure (and “meaning”) of the Markov chain is entirely encoded in the transition matrix $[a_{ij}]_{i,j}$ where
  $$p(X_t = j|X_{t-1} = i) = a_{ij}$$  \hspace{1cm} (9.15)
Graphical model gives the factorization assumptions about a joint probability distribution $p(x_1, x_2, \ldots, x_5)$.

On the left is a 1st order Markov chain, so

$$p(x_1, \ldots, x_5) = p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_3)p(x_5|x_4).$$

The structure (and “meaning”) of the Markov chain is entirely encoded in the transition matrix $[a_{ij}]_{i,j}$ where

$$p(X_t = j|X_{t-1} = i) = a_{ij}$$

(9.15)

Middle is 2nd order Markov chain.

$$p(x_1, \ldots, x_5) = p(x_1)p(x_2|x_1)p(x_3|x_2, x_1)p(x_4|x_3, x_2)p(x_5|x_4, x_3).$$
Graphical model gives the factorization assumptions about a joint probability distribution \( p(x_1, x_2, \ldots, x_5) \).

On the left is a 1st order Markov chain, so
\[
p(x_1, \ldots, x_5) = p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_3)p(x_5|x_4).
\]

The structure (and “meaning”) of the Markov chain is entirely encoded in the transition matrix \([a_{ij}]_{i,j}\) where
\[
p(X_t = j|X_{t-1} = i) = a_{ij} \tag{9.15}
\]

Middle is 2nd order Markov chain.
\[
p(x_1, \ldots, x_5) = p(x_1)p(x_2|x_1)p(x_3|x_2, x_1)p(x_4|x_3, x_2)p(x_5|x_4, x_3).
\]

Right is 3rd order Markov chain.
HMM: As generative process

- Classic urn/ball example. We have $N$ urns, each urn has a set of different colored balls in them.
- We choose urns according to a 1st-order Markov chain. Each time we choose an urn, we sample (with replacement) a ball from that urn.
- The output is the sequence of ball colors, but not the sequence of urn choices.
- A goal might be, based on the sequence of ball colors, and knowing the corresponding distributions, deduce the most probable sequence of urns that could possibly have generated that ball sequence (this is the recognition problem).
HMM: As generative process urns/balls
Figure: A 4-state HMM viewed as balls in urns. Each HMM state corresponds to an urn and the directed edges between urns give the state transition matrix, where missing edges correspond to zeros in the matrix. In each state, there is a distribution on colored balls. For example, in state one, we have
\[
\Pr(\text{color} = \text{yellow}|\text{state} = 1) = \frac{4}{7} \quad \text{while} \quad \Pr(\text{color} = \text{red}|\text{state} = 1) = \frac{2}{7},
\]
\[
\Pr(\text{color} = \text{green}|\text{state} = 1) = \frac{1}{7}, \quad \text{and} \quad \Pr(\text{color} = \text{blue}|\text{state} = 1) = 0.
\]
That is, in each state (urn), the probability of drawing a color is equal to the number of balls of that color in that urn divided by the total number of balls.
Ball and urns example

- An HMM as a stochastic finite state automata, informally seen as a Markov chain with hanging observation distributions.
Ball and urns example

- An HMM as a stochastic finite state automata, informally seen as a Markov chain with hanging observation distributions.

- Urn example:

  - Urn 1: $P(R|U = 1) = 3/7$, $P(B|U = 1) = 4/7$
  - Urn 2: $P(R|U = 2) = 1/2$
  - Urn 3: $P(R|U = 3) = 5/6$
  - Urn 4: $P(R|U = 4) = 1/5$
HMMs and balls/urns

- In HMM, some sequence of urns is chosen, and from each urn, a ball chosen w. replacement.
HMMs and balls/urns

- In HMM, some sequence of urns is chosen, and from each urn, a ball chosen w. replacement.
- We know only the balls, but not which urns were used for each ball.
HMMs and balls/urns

- In HMM, some sequence of urns is chosen, and from each urn, a ball chosen w. replacement.
- We know only the balls, but not which urns were used for each ball.

Underlying generative process:

<table>
<thead>
<tr>
<th>Time t:</th>
<th>1 2 3 4 5 6 7 8 9 10 11 12 13 14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Urn Sequence qₜ:</td>
<td>1 1 1 3 3 3 2 2 1 1 2 4 4 4</td>
</tr>
<tr>
<td>Ball Sequence xₜ:</td>
<td>🍒 🍒 🍒 🍒 🍒 🍒 🍒 🍒 🍒 🍒 🍒 🔴 🔴 🔴</td>
</tr>
<tr>
<td>x₁ x₂ x₃ x₄ x₅ x₆ x₇ x₈ x₉ x₁₀ x₁₁ x₁₂ x₁₃ x₁₄</td>
<td></td>
</tr>
<tr>
<td>Transition Probs:</td>
<td>1 1/3 1/3 1/3 2/3 2/3 1/3 ½ 1/4 1/3 1/3 ¼ 1 1</td>
</tr>
<tr>
<td>Obs. Probs:</td>
<td>3/7 4/7 4/7 5/6 1/6 5/6 ½ ½ 4/7 4/7 ½ 4/5 1/5 4/5</td>
</tr>
</tbody>
</table>
Definition 9.4.1

**Hidden Markov Model**  A hidden Markov model (HMM) is a collection of random variables consisting of a set of $T$ discrete scalar variables $Q_{1:T}$ and a set of $T$ other variables $X_{1:T}$ which may be either discrete or continuous (and either scalar- or vector-valued). These variables, collectively, possess the following conditional independence properties:

\[
\{Q_{t:T}, X_{t:T}\} \perp \perp \{Q_{1:t-2}, X_{1:t-1}\} | Q_{t-1} \tag{9.16}
\]

and

\[
X_t \perp \perp \{Q_{-t}, X_{-t}\} | Q_t \tag{9.17}
\]

for each $t \in 1 : T$. No other conditional independence properties are true in general, unless they follow from Equations 9.16 and 9.17. The length $T$ of these sequences is itself an integer-valued random variable having a complex distribution.
HMM: As a set of independence properties

- $Q_t$ takes values from finite set, so $Q_t \in D_Q$ where $D_Q$ is called the state space, cardinality $|D_Q|$.

- Equation 9.16 states that the future is conditionally independent of the past given the present. Therefore, $Q_t \perp \perp Q_{1:t-2} | Q_{t-1}$ which means the variables $Q_{1:T}$ form a discrete-time, discrete-valued, first-order Markov chain.

- Also, $Q_t \perp \perp \{Q_{1:t-2}, X_{1:t-1}\} | Q_{t-1}$ so that $X_\tau$ is unable, given $Q_{t-1}$, to affect $Q_t$ for $\tau < t$. Does not imply, given $Q_{t-1}$, that $Q_t$ is unaffected by future variables.
HMM: As a set of independence properties

What the definition doesn’t require:

- does not limit the number of states $|D_Q|$ in the Markov chain, only that it is finite.
HMM: As a set of independence properties

What the definition doesn’t require:

- does not limit the number of states \(|D_Q|\) in the Markov chain, only that it is finite.

- Does not require the observations \(X_{1:T}\) to be either discrete, continuous, scalar-, or vector-valued, does not designate the implementation of the dependencies (e.g., general regression, probability table, neural network, etc.),
HMM: As a set of independence properties

What the definition doesn’t require:

- does not limit the number of states $|D_Q|$ in the Markov chain, only that it is finite.
- Does not require the observations $X_{1:T}$ to be either discrete, continuous, scalar-, or vector-valued, does not designate the implementation of the dependencies (e.g., general regression, probability table, neural network, etc.),
- does not determine the model families for each of the variables (e.g., Gaussian, Laplace, etc.),
HMM: As a set of independence properties

What the definition doesn’t require:

- does not limit the number of states $|D_Q|$ in the Markov chain, only that it is finite.
- Does not require the observations $X_{1:T}$ to be either discrete, continuous, scalar-, or vector-valued, does not designate the implementation of the dependencies (e.g., general regression, probability table, neural network, etc.),
- does not determine the model families for each of the variables (e.g., Gaussian, Laplace, etc.),
- does not force the underlying Markov chain to be time-homogeneous,
HMM: As a set of independence properties

What the definition doesn't require:

- does not limit the number of states $|D_Q|$ in the Markov chain, only that it is finite.
- Does not require the observations $X_{1:T}$ to be either discrete, continuous, scalar-, or vector-valued, does not designate the implementation of the dependencies (e.g., general regression, probability table, neural network, etc.),
- does not determine the model families for each of the variables (e.g., Gaussian, Laplace, etc.),
- does not force the underlying Markov chain to be time-homogeneous,
- does not fix the parameters or any tying mechanism
HMM: As a set of independence properties

What the definition doesn’t require:

- does not limit the number of states $|D_Q|$ in the Markov chain, only that it is finite.
- Does not require the observations $X_{1:T}$ to be either discrete, continuous, scalar-, or vector-valued, does not designate the implementation of the dependencies (e.g., general regression, probability table, neural network, etc.),
- does not determine the model families for each of the variables (e.g., Gaussian, Laplace, etc.),
- does not force the underlying Markov chain to be time-homogeneous,
- does not fix the parameters or any tying mechanism
- HMM makes no marginal independence assumptions — nothing in an HMM is independent of anything else (no statements of form $A \perp \perp B$). Only conditional independence statements exist.
Using the conditional independence statements mentioned above, we can derive the following factorization:

\[
p(x_{1:T}, q_{1:T}) = p(x_T, q_T | x_{1:T-1}, q_{1:T-1})p(x_{1:T-1}, q_{1:T-1}) \tag{9.18}
\]

\[
= p(x_T | q_T, x_{1:T-1}, q_{1:T-1})p(q_T | x_{1:T-1}, q_{1:T-1})p(x_{1:T-1}, q_{1:T-1}) \tag{9.19}
\]

\[
p(x_{1:T-1}, q_{1:T-1}) \tag{9.20}
\]

\[
= p(x_T | q_T)p(q_T | q_{T-1})p(x_{1:T-1}, q_{1:T-1}) \tag{9.21}
\]

\[
= \ldots \tag{9.22}
\]

\[
= p(q_1) \prod_{t=2}^{T} p(q_t | q_{t-1}) \prod_{t=1}^{T} p(x_t | q_t) \tag{9.23}
\]

This last equation is the classical factorization expression for an HMM joint distribution over \(x_{1:T}, q_{1:T}\).
HMMs are a sequential model simultaneously over both sequences $X_{1:T}$ and $Q_{1:T}$, i.e., $p(x_{1:T}, q_{1:T})$.

HMMs represents a joint distribution, i.e., $p(x_{1:T}, q_{1:T})$, not a conditional distribution.
HMM parameters

- Parameters of HMM, depend on nature of underlying Markov chain.
- If time-homogeneous, we have an initial state distribution (typically $\pi$) with $p(Q_1 = i) = \pi_i$, and a state transition matrix $A$.
- We also have the set of observation distributions $b_j(x) = p(X_t = x|Q_t = j)$ in the time-homogeneous case. In time homogeneous case, we might have $b_{t,j}(x)$. Also, $B = \{b_i(\cdot)\}_i$.
- HMM, conditional on parameters $\lambda$, is given as $p(x_{1:T}|\lambda)$.
- All three parameters (initial distribution, transition matrix, and observation distribution) together refer to using $\lambda = (\pi, A, B)$.
- Sampling from an HMM means: 1) first randomly choose an assignment to $Q_{1:T}$ and then 2) randomly choose an assignment to $X_{1:T}$.
- Each new $X$ sample requires a new $Q$ sample.
An HMM is a tree (in the GM sense) and so we know that computing inference (over cliques) has cost $O(r^2)$. Since there are $T$ such cliques, overall cost should be $O(Tr^2)$.

Also, we view $X_{1:T}$ as the stochastic process with an underlying generative hidden chain $Q_{1:T}$. Thus, we might want to compute

$$p(\bar{x}_{1:T}) = \sum_{q_{1:T}} p(\bar{x}_{1:T}, q_{1:T})$$

again, naïvely, an exponential computation.

This is the probability of evidence computation, and again due to the query and the fact that it is a tree, it is relatively easy, at least from the perspective of clique size (there are many difficulties when $r$ gets extremely large, as we’ll see).
To compute $p(x_{1:T})$, we start out with $p(x_{1:t})$

\[
p(x_{1:t}, q_t, q_{t-1}) = p(x_{1:t-1}, q_{t-1}, x_t, q_t)
\]

\[\overset{(A)}{=} p(x_t, q_t | x_{1:t-1}, q_{t-1})p(x_{1:t-1}, q_{t-1})
\]

\[= p(x_t | q_t, x_{1:t-1}, q_{t-1})p(q_t | x_{1:t-1}, q_{t-1})p(x_{1:t-1}, q_{t-1})
\]

\[\overset{(B)}{=} p(x_t | q_t)p(q_t | q_{t-1})p(x_{1:t-1}, q_{t-1})
\]

where (A) follows from the chain rule of probability, and (B) follows since $X_t \perp \{X_{1:t-1}, Q_{1:t-1}\} | Q_t$ and $Q_t \perp \{X_{1:t-1}, Q_{1:t-2}\} | Q_{t-1}$
HMM 3 tasks

1) efficiently compute \( p(x_{1:T}) \) for an HMM. More generally, for each utterance \( M \), compute \( p(x_{1:T}|M, \lambda) \) where \( M \) is the utterance and \( \lambda \) is the set of parameters.
HMM 3 tasks

1) efficiently compute \( p(x_{1:T}) \) for an HMM. More generally, for each utterance \( M \), compute \( p(x_{1:T}|M, \lambda) \) where \( M \) is the utterance and \( \lambda \) is the set of parameters.

2) Finding most likely state sequence (compute Viterbi path via Viterbi decoding)

\[
q_{1:T}^* \in \arg\max_{q_{1:T}} p(x_{1:T}, q_{1:T}|M, \lambda) \tag{9.25}
\]

the Viterbi path in some sense “best” explains the utterance \( x_{1:T} \) (i.e., best set of urns for a given set of balls). Note: we are given \( x_{1:T} \) and we need to find best \( q_{1:T} \)
HMM 3 tasks

1) efficiently compute \( p(x_{1:T}) \) for an HMM. More generally, for each utterance \( M \), compute \( p(x_{1:T}|M, \lambda) \) where \( M \) is the utterance and \( \lambda \) is the set of parameters.

2) Finding most likely state sequence (compute Viterbi path via Viterbi decoding)

\[
q^{*}_{1:T} \in \arg\max_{q_{1:T}} p(x_{1:T}, q_{1:T}|M, \lambda) \tag{9.25}
\]

the Viterbi path in some sense “best” explains the utterance \( x_{1:T} \) (i.e., best set of urns for a given set of balls). Note: we are given \( x_{1:T} \) and we need to find best \( q_{1:T} \)

3) Given data set \( \mathcal{D} = \{x^{(1)}_{1:T}, x^{(2)}_{1:T}, \ldots, x^{(N)}_{1:T}\} \), find most likely parameters:

\[
\lambda^{*} \in \arg\max_{\lambda} \Pr(\mathcal{D}|\lambda) \tag{9.26}
\]

This is maximum likelihood training, solved by EM algorithm.
DTW (and DP) is one early method people used to recognize speech, and is based on templates.