Lecture 14: Hidden Markov Models

Mark Hasegawa-Johnson

ECE 417: Multimedia Signal Processing, Fall 2021

Outline

Bayesian Classifiers

A Bayesian classifier chooses a label, $y \in \{0 \ldots N_Y - 1\}$, that has the minimum probability of error given an observation, $\vec{x} \in \Re^D$:

$$\begin{aligned}
\hat{y} &= \underset{y}{\operatorname{argmin}} \Pr\left\{ Y \neq y | \vec{X} = \vec{x} \right\} \\
&= \underset{y}{\operatorname{argmax}} \Pr\left\{ Y = y | \vec{X} = \vec{x} \right\} \\
&= \underset{y}{\operatorname{argmax}}\, p_{Y|\vec{X}}(y|\vec{x}) \\
&= \underset{y}{\operatorname{argmax}}\, p_Y(\hat{y}) p_{\vec{X}|Y}(\vec{x}|y)
\end{aligned}$$

Review: Learning the Bayesian Probabilities

Here's how we can estimate the four Bayesian probabilities:

1. **Posterior, Evidence:** need lots of training data, and a neural net or kernel estimator.
2. **Prior:**

$$p_Y(y) = \frac{\#\text{ times } Y = y \text{ occurred in training data}}{\#\text{ frames in training data}}$$

3. **Likelihood:**

$$p_{\vec{X}|Y}(\vec{x}|y) = \sum_{k=0}^{K-1} c_{y,k} \mathcal{N}(\vec{x}|\vec{\mu}_{y,k}, \Sigma_{y,k})$$

where $c_{y,k}$, $\vec{\mu}_{y,k}$, $\Sigma_{y,k}$ might be estimated using EM.

Outline

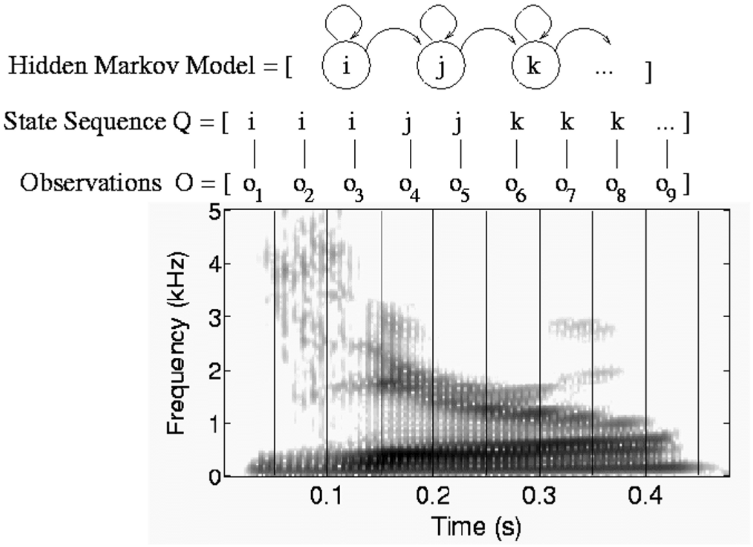
Notation: Inputs and Outputs

- Let's assume we have T consecutive observations, $X = [\vec{x}_1, \ldots, \vec{x}_T]$.
- A "hidden Markov model" represents those probabilities by assuming some sort of "hidden" state sequence, $Q = [q_1, \ldots, q_T]$, where q_t is the hidden (unknown) state variable at time t.

The idea is, can we model these probabilities well enough to solve problems like:

1. **Recognition:** What's $p(X)$ given the model?
2. **Segmentation:** What state is the model in at time t?
3. **Training:** Can we learn a model to fit some data?

Notation: Inputs and Outputs

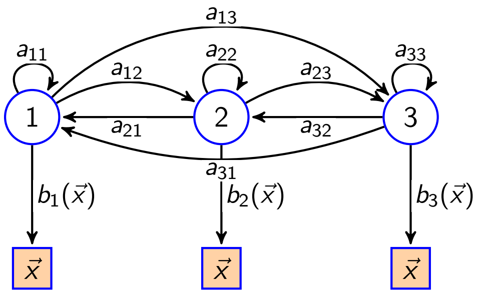

Hidden Markov Model = [i j k ...]
State Sequence Q = [i i i j j k k k ...]
Observations O = [o1 o2 o3 o4 o5 o6 o7 o8 o9]
Frequency (kHz)
0
1
2
3
4
5
0.1
0.2
0.3
0.4
Time (s)

HMM: Key Concepts

An HMM is a "generative model," meaning that it models the joint probability $p(Q, X)$ using a model of the way in which those data might have been generated. An HMM pretends the following generative process:

1. Start in state $q_t = i$ with pmf $\pi_i = p(q_1 = i)$.
2. Generate an observation, $\vec{x}$, with pdf $b_i(\vec{x}) = p(\vec{x}|q_t = i)$.
3. Transition to a new state, $q_{t+1} = j$, according to pmf $a_{ij} = p(q_{t+1} = j|q_t = i)$.
4. Repeat.

HMM: Finite State Diagram



1. Start in state $q_t = i$, for some $1 \leq i \leq N$.
2. Generate an observation, $\vec{x}$, with pdf $b_i(\vec{x})$.
3. Transition to a new state, $q_{t+1} = j$, according to pmf a_{ij}.
4. Repeat steps #2 and #3, T times each.

Notation: Model Parameters

Solving an HMM is possible if you **carefully keep track of notation**. Here's standard notation for the parameters:

- $\pi_i = p(q_1 = i)$ is called the **initial state probability**. Let N be the number of different states, so that $1 \le i \le N$.
- $a_{ij} = p(q_t = j | q_{t-1} = i)$ is called the **transition probability**, $1 \le i, j \le N$.
- $b_j(\vec{x}) = p(\vec{x}_t = \vec{x} | q_t = j)$ is called the **observation probability**. It is usually estimated by a neural network, though Gaussians, GMMs, and even lookup tables are possible.
- Λ is the complete set of **model parameters**, including all the π_i's and a_{ij}'s, and the Gaussian, GMM, or neural net parameters necessary to compute $b_j(\vec{x})$.

The Three Problems for an HMM

1. **Recognition:** Given two different HMMs, Λ_1 and Λ_2, and an observation sequence X. Which HMM was more likely to have produced X? In other words, is $p(X|\Lambda_1) > p(X|\Lambda_2)$?
2. **Segmentation:** What is $p(q_t = i|X, \Lambda)$?
3. **Training:** Given an initial HMM Λ, and an observation sequence X, can we find Λ' such that $p(X|\Lambda') > p(X|\Lambda)$?

Outline

The HMM Recognition Problem

- Given
  - $X = [\vec{x}_1, \ldots, \vec{x}_T]$ and
  - $\Lambda = \{\pi_i, a_{ij}, b_j(\vec{x}) \forall i, j\}$,

  what is $p(X|\Lambda)$?
- Let's solve a simpler problem first:
- Given
  - $X = [\vec{x}_1, \ldots, \vec{x}_T]$ and
  - $Q = [q_1, \ldots, q_T]$ and
  - $\Lambda = \{\pi_i, a_{ij}, b_j(\vec{x}) \forall i, j\}$,

  what is $p(X|\Lambda)$?

Joint Probability of State Sequence and Observation Sequence

The joint probability of the state sequence and the observation sequence is calculated iteratively, from beginning to end:

- The probability that $q_1 = q_1$ is π_{q_1}.
- Given q_1, the probability of $\vec{x}_1$ is $b_{q_1}(\vec{x}_1)$.
- Given q_1, the probability of q_2 is $a_{q_1 q_2}$.
- . . . and so on. . .

$$p(Q, X|\Lambda) = \pi_{q_1} b_{q_1}(\vec{x}_1) \prod_{t=2}^{T} a_{q_{t-1}q_t} b_{q_t}(\vec{x}_t)$$

Probability of the Observation Sequence

The probability of the observation sequence, alone, is somewhat harder, because we have to solve this sum:

$$p(X|\Lambda) = \sum_{Q} p(Q, X|\Lambda)$$
$$= \sum_{q_T=1}^{N} \cdots \sum_{q_1=1}^{N} p(Q, X|\Lambda)$$

On the face of it, this calculation seems to have complexity $\mathcal{O}\left\{N^T\right\}$. So for a very small 100-frame utterance, with only 10 states, we have a complexity of $\mathcal{O}\left\{10^{100}\right\}$ =one google.

The Forward Algorithm

The solution is to use a kind of dynamic programming algorithm, called “the forward algorithm.” The forward probability is defined as follows:

$$\alpha_t(i) \equiv p(\vec{x}_1, \ldots, \vec{x}_t, q_t = i|\Lambda)$$

Obviously, if we can find $\alpha_t(i)$ for all i and all t, we will have solved the recognition problem, because

$$\begin{aligned} p(X|\Lambda) &= p(\vec{x}_1, \ldots, \vec{x}_T|\Lambda) \\ &= \sum_{i=1}^{N} p(\vec{x}_1, \ldots, \vec{x}_T, q_T = i|\Lambda) \\ &= \sum_{i=1}^{N} \alpha_T(i) \end{aligned}$$

The Forward Algorithm

So, working with the definition $\alpha_t(i) \equiv p(\vec{x}_1, \ldots, \vec{x}_t, q_t = i|\Lambda)$, let's see how we can actually calculate $\alpha_t(i)$.

1. **Initialize:**

$$\begin{aligned}\alpha_1(i) &= p(q_1 = i, \vec{x}_1|\Lambda) \\ &= p(q_1 = i|\Lambda)p(\vec{x}_1|q_1 = i, \Lambda) \\ &= \pi_i b_i(\vec{x}_1)\end{aligned}$$

The Forward Algorithm

Definition: $\alpha_t(i) \equiv p(\vec{x}_1, \ldots, \vec{x}_t, q_t = i|\Lambda)$.

1. **Initialize:**
$$\alpha_1(i) = \pi_i b_i(\vec{x}_1), \quad 1 \le i \le N$$
2. **Iterate:**
$$\begin{aligned}\alpha_t(j) &= p(\vec{x}_1, \ldots, \vec{x}_t, q_t = j|\Lambda)\\ &= \sum_{i=1}^{N} p(\vec{x}_1, \ldots, \vec{x}_{t-1}, q_{t-1} = i) p(q_t = j|q_{t-1} = i) p(\vec{x}_t|q_t = j)\\ &= \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(\vec{x}_t)\end{aligned}$$

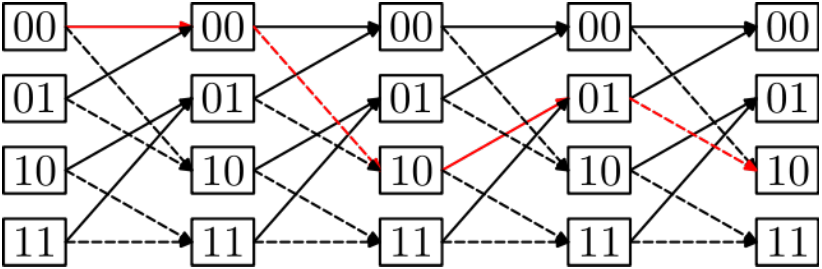
The Forward Algorithm

So, working with the definition $\alpha_t(i) \equiv p(\vec{x}_1, \ldots, \vec{x}_t, q_t = i|\Lambda)$, let's see how we can actually calculate $\alpha_t(i)$.

1. **Initialize:**
$$\alpha_1(i) = \pi_i b_i(\vec{x}_1), \quad 1 \leq i \leq N$$
2. **Iterate:**
$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(\vec{x}_t), \;\; 1 \leq j \leq N, \; 2 \leq t \leq T$$
3. **Terminate:**
$$p(X|\Lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

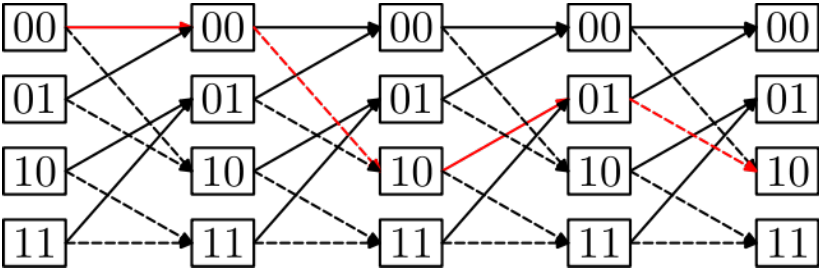
Visualizing the Forward Algorithm using a Trellis

One way to think about the forward algorithm is by way of a **trellis**. A trellis is a matrix in which each time step is a column, and each row shows a different state. For example, here's a trellis with $N = 4$ states, and $T = 5$ frames:



Public domain image by Qef, 2009

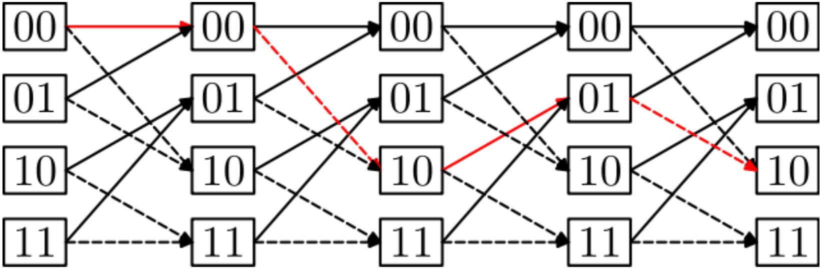
Visualizing the Forward Algorithm using a Trellis



Using a trellis, the **initialize** step computes probabilities for the first column of the trellis:

$$\alpha_1(i) = \pi_i b_i(\vec{x}_1), \quad 1 \leq i \leq N$$

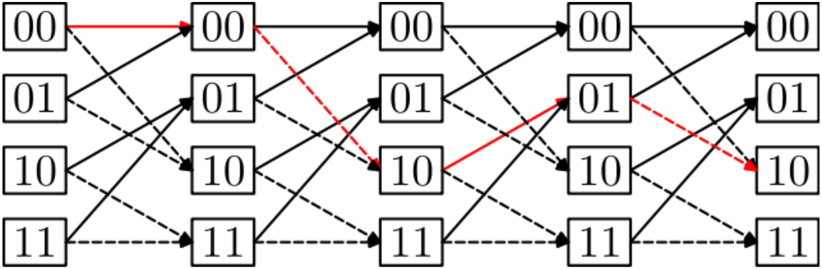
Visualizing the Forward Algorithm using a Trellis



The **iterate** step then computes the probabilities in the t^{th} column by adding up the probabilities in the $(t-1)^{\mathrm{st}}$ column, each multiplied by the corresponding transition probability:

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(\vec{x}_t), \quad 1 \le j \le N,\ 2 \le t \le T$$

Visualizing the Forward Algorithm using a Trellis



The **terminate** step then computes the likelihood of the model by adding the probabilities in the last column:

$$p(X|\Lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

The Forward Algorithm: Computational Complexity

Most of the computational complexity is in this step:

- **Iterate:**

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(\vec{x}_t), \quad 1 \le i,j \le N, \; 2 \le t \le T$$

Its complexity is:

- For each of $T-1$ time steps, $2 \le t \le T, \ldots$
- we need to calculate N different alpha-variables, $\alpha_t(j)$, for $1 \le j \le N, \ldots$
- each of which requires a summation with N terms.

So the total complexity is $\mathcal{O}\left\{TN^2\right\}$. For example, with $N = 10$ and $T = 100$, the complexity is only $TN^2 = 10,000$ multiplies (much, much less than N^T!!)

Outline

The Segmentation Problem

There are different ways to define the segmentation problem. Let's define it this way:

- We want to find the most likely state, $q_t = i$, at time $t, \ldots$
- given knowledge of the *entire* sequence $X = [\vec{x}_1, \ldots, \vec{x}_T]$, not just the current observation. So for example, we don't want to recognize state i at time t if the surrounding observations, $\vec{x}_{t-1}$ and $\vec{x}_{t+1}$, make it obvious that this choice is impossible. Also,...
- given knowledge of the HMM that produced this sequence, Λ.

In other words, we want to find the **state posterior probability**, $p(q_t = i|X, \Lambda)$. Let's define some more notation for the state posterior probability, let's call it

$$\gamma_t(i) = p(q_t = i|X, \Lambda)$$

Use Bayes' Rule

Suppose we already knew the **joint probability,** $p(X, q_t = i|\Lambda)$. Then we could find the state posterior using Bayes' rule:

$$\gamma_t(i) = p(q_t = i|X, \Lambda) = \frac{p(X, q_t = i|\Lambda)}{\sum_{j=1}^{N} p(X, q_t = j|\Lambda)}$$

Use the Forward Algorithm

Let's expand this:

$$p(X, q_t = i|\Lambda) = p(q_t = i, \vec{x}_1, \ldots, \vec{x}_T|\Lambda)$$

We already know about half of that:
$\alpha_t(i) = p(q_t = i, \vec{x}_1, \ldots, \vec{x}_t|\Lambda)$. We're only missing this part:

$$p(X, q_t = i|\Lambda) = \alpha_t(i) p(\vec{x}_{t+1}, \ldots, \vec{x}_T|q_t = i, \Lambda)$$

Again, let's try the trick of "solve the problem by inventing new notation." Let's define

$$\beta_t(i) \equiv p(\vec{x}_{t+1}, \ldots, \vec{x}_T|q_t = i, \Lambda)$$

The Backward Algorithm

Now let's use the definition $\beta_t(i) \equiv p(\vec{x}_{t+1}, \ldots, \vec{x}_T | q_t = i, \Lambda)$, and see how we can compute that.

1. **Initialize:**

$$\beta_T(i) = 1, \quad 1 \leq i \leq N$$

   This might not seem immediately obvious, but think about it. Given that there are no more $\vec{x}$ vectors after time T, what is the probability that there are no more $\vec{x}$ vectors after time T? Well, 1, obviously.

The Backward Algorithm

Now let's use the definition $\beta_t(i) \equiv p(\vec{x}_{t+1}, \ldots, \vec{x}_T | q_t = i, \Lambda)$, and see how we can compute that.

1. **Initialize:**
$$\beta_T(i) = 1, \quad 1 \le i \le N$$
2. **Iterate:**

$$\begin{aligned}
\beta_t(i) &= p(\vec{x}_{t+1}, \ldots, \vec{x}_T | q_t = i, \Lambda) \\
&= \sum_{j=1}^{N} p(q_{t+1} = j | q_t = i) p(\vec{x}_{t+1} | q_{t+1} = j) p(\vec{x}_{t+2}, \ldots, \vec{x}_T | q_{t+1} = j) \\
&= \sum_{j=1}^{N} a_{ij} b_j(\vec{x}_{t+1}) \beta_{t+1}(j)
\end{aligned}$$

The Backward Algorithm

Now let's use the definition $\beta_t(i) \equiv p(\vec{x}_{t+1}, \ldots, \vec{x}_T | q_t = i, \Lambda)$, and see how we can compute that.

1. **Initialize:**
$$\beta_T(i) = 1, \quad 1 \le i \le N$$
2. **Iterate:**
$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(\vec{x}_{t+1}) \beta_{t+1}(j), \quad 1 \le i \le N, \ 1 \le t \le T-1$$
3. **Terminate:**
$$p(X|\Lambda) = \sum_{i=1}^{N} \pi_i b_i(\vec{x}_1) \beta_1(i)$$

The Backward Algorithm: Computational Complexity

Most of the computational complexity is in this step:

- **Iterate:**

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(\vec{x}_{t+1}) \beta_{t+1}(j), \quad 1 \le i \le N, \ 2 \le t \le T$$

Its complexity is:

- For each of $T-1$ time steps, $1 \le t \le T-1$,...
- we need to calculate N different beta-variables, $\beta_t(i)$, for $1 \le i \le N$,...
- each of which requires a summation with N terms.

So the total complexity is $\mathcal{O}\left\{TN^2\right\}$.

Use Bayes' Rule

The segmentation probability is then

$$
\begin{aligned}
\gamma_t(i) &= \frac{p(X, q_t = i|\Lambda)}{\sum_{k=1}^N p(X, q_t = k|\Lambda)} \\
&= \frac{p(\vec{x}_1, \ldots, \vec{x}_t, q_t = i|\Lambda) p(\vec{x}_{t+1}, \ldots, \vec{x}_T|q_t = i, \Lambda)}{\sum_{k=1}^N p(\vec{x}_1, \ldots, \vec{x}_t, q_t = k|\Lambda) p(\vec{x}_{t+1}, \ldots, \vec{x}_T|q_t = k, \Lambda)} \\
&= \frac{\alpha_t(i)\beta_t(i)}{\sum_{k=1}^N \alpha_t(k)\beta_t(k)}
\end{aligned}
$$

What About State Sequences?

- Notice a problem: $\gamma_t(i)$ only tells us about one frame at a time! It doesn't tell us anything about the probability of a sequence of states, covering a sequence of frames!
- . . . but we can extend the same reasoning to cover two or more consecutive frames. For example, let's define:

$$\xi_t(i,j) = p(q_t = i, q_{t+1} = j | X, \Lambda)$$

- We can solve for $\xi_t(i,j)$ using the same reasoning that we used for $\gamma_t(i)$!

Segmentation: The Backward Algorithm

In summary, we now have three new probabilities, all of which can be computed in $\mathcal{O}\left\{TN^2\right\}$ time:

1. **The Backward Probability:**

$$\beta_t(i) = p(\vec{x}_{t+1},\ldots,\vec{x}_T|q_t = i,\Lambda)$$

2. **The State Posterior:**

$$\gamma_t(i) = p(q_t = i|X,\Lambda) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{k=1}^N \alpha_t(k)\beta_t(k)}$$

3. **The Segment Posterior:**

$$\begin{aligned}\xi_t(i,j) &= p(q_t = i, q_{t+1} = j|X,\Lambda)\\ &= \frac{\alpha_t(i)a_{ij}b_j(\vec{x}_{t+1})\beta_{t+1}(j)}{\sum_{k=1}^N\sum_{\ell=1}^N \alpha_t(k)a_{k\ell}b_\ell(\vec{x}_{t+1})\beta_{t+1}(\ell)}\end{aligned}$$

Outline

Example: Gumball Machines

"Gumball machines in a Diner at Dallas, Texas, in 2008," Andreas Praefcke, public domain image.

Example: Gumball Machines

- **Observation Probabilities:** Suppose we have two gumball machines, $q = 1$ and $q = 2$. Machine #1 contains 60% Grapefruit gumballs, 40% Apple gumballs. Machine #2 contains 90% Apple, 10% Grapefruit.

$$b_1(x) = \begin{cases} 0.4 & x = A \\ 0.6 & x = G \end{cases}, \quad b_2(x) = \begin{cases} 0.9 & x = A \\ 0.1 & x = G \end{cases}$$

- **Initial State Probabilities:** My friend George flips a coin to decide which machine to use first.

$$\pi_i = 0.5, \quad i \in \{1, 2\}$$

- **Transition Probabilities:** After he's used a machine, George flips two coins, and he only changes machines if both coins come up heads.

$$a_{ij} = \begin{cases} 0.75 & i = j \\ 0.25 & i \neq j \end{cases}$$

A Segmentation Problem

- George bought three gumballs, using three quarters. The three gumballs are $(x_1 = A, x_2 = G, x_3 = A)$.
- Unfortunately, George is a bit of a goofball. The second of the three "quarters" was actually my 1867 silver "Seated Liberty" dollar, worth $4467.
- Which of the two machines do I need to dismantle in order to get my coin back?

Image used with permission of the National Numismatic Collection, National Museum of American History.

The Forward Algorithm: $t = 1$

Remember, the observation sequence is $X = (A, G, A)$.

$$
\begin{aligned}
\alpha_1(i) &= \pi_i b_1(i) \\
&= \begin{cases} (0.5)(0.4) = 0.2 & i = 1 \\ (0.5)(0.9) = 0.45 & i = 2 \end{cases}
\end{aligned}
$$

The Forward Algorithm: $t = 2$

Remember, the observation sequence is $X = (A, G, A)$.

$$
\begin{aligned}
\alpha_2(j) &= \sum_{i=1}^{2} \alpha_1(i) a_{ij} b_j(x_2) \\
&= \begin{cases} \alpha_1(1) a_{11} b_1(x_2) + \alpha_1(2) a_{21} b_1(x_2) & j = 1 \\ \alpha_1(1) a_{12} b_2(x_2) + \alpha_1(2) a_{22} b_2(x_2) & j = 2 \end{cases} \\
&= \begin{cases} (0.2)(0.75)(0.6) + (0.45)(0.25)(0.6) = 0.04125 & j = 1 \\ (0.2)(0.25)(0.1) + (0.45)(0.75)(0.1) = 0.03875 & j = 2 \end{cases}
\end{aligned}
$$

The Backward Algorithm: $t = 3$

The backward algorithm always starts out with $\beta_T(i) = 1$!

$$\beta_3(i) = 1, \quad i \in \{1, 2\}$$

The Backward Algorithm: $t = 2$

Remember, the observation sequence is $X = (A, G, A)$.

$$\begin{aligned}
\beta_2(i) &= \sum_{j=1}^{2} a_{ij} b_j(x_3) \beta_3(j) \\
&= \begin{cases} a_{11} b_1(x_3) + a_{12} b_2(x_3) & i = 1 \\ a_{21} b_1(x_3) + a_{22} b_2(x_3) & i = 2 \end{cases} \\
&= \begin{cases} (0.75)(0.4) + (0.25)(0.9) = 0.525 & j = 1 \\ (0.25)(0.4) + (0.75)(0.9) = 0.775 & j = 2 \end{cases}
\end{aligned}$$

The Solution to the Puzzle

Given the observation sequence is $X = (A, G, A)$, the posterior state probability is

$$\gamma_2(i) = \frac{\alpha_2(i)\beta_2(i)}{\sum_{k=1}^{2} \alpha_2(k)\beta_2(k)}$$
$$= \begin{cases} \frac{(0.04125)(0.525)}{(0.04125)(0.525)+(0.03875)(0.775)} = 0.42 & i = 1 \\ \frac{(0.03875)(0.775)}{(0.04125)(0.525)+(0.03875)(0.775)} = 0.58 & i = 2 \end{cases}$$

So I should dismantle gumball machine #2, hoping to find my rare 1867 silver dollar. Good luck!

Outline

The Forward Algorithm

Definition: $\alpha_t(i) \equiv p(\vec{x}_1, \ldots, \vec{x}_t, q_t = i|\Lambda)$. Computation:

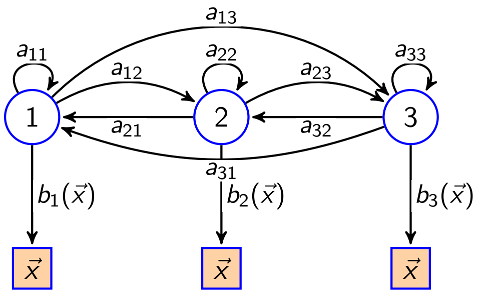
1. **Initialize:**
$$\alpha_1(i) = \pi_i b_i(\vec{x}_1), \quad 1 \leq i \leq N$$
2. **Iterate:**
$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(\vec{x}_t), \quad 1 \leq j \leq N, \; 2 \leq t \leq T$$
3. **Terminate:**
$$p(X|\Lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

The Backward Algorithm

Definition: $\beta_t(i) \equiv p(\vec{x}_{t+1}, \ldots, \vec{x}_T | q_t = i, \Lambda)$. Computation:

1. **Initialize:**
$$\beta_T(i) = 1, \quad 1 \le i \le N$$
2. **Iterate:**
$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(\vec{x}_{t+1}) \beta_{t+1}(j), \quad 1 \le i \le N, \ 1 \le t \le T-1$$
3. **Terminate:**
$$p(X|\Lambda) = \sum_{i=1}^{N} \pi_i b_i(\vec{x}_1) \beta_1(i)$$

Hidden Markov Model



1. Start in state $q_t = i$ with pmf π_i.
2. Generate an observation, $\vec{x}$, with pdf $b_i(\vec{x})$.
3. Transition to a new state, $q_{t+1} = j$, according to pmf a_{ij}.
4. Repeat.

Outline

Written Example

Joe's magic shop opens at random, and closes at random. To be more specific, if it's currently closed, the probability that it will open any time in the next hour is 10%; if it's currently open, the probability that it will close any time in the next hour is 10%. The shop is in a busy part of town; when the shop is open, the area gets even busier. If the shop is closed, the area is noisy with a probability of 40%. If it's open, the area is noisy with a probability of 70%.
At 1:00, you notice that the area is noisy, so you go to check; unfortunately, the shop is closed. At 2:00, the area is still noisy, but you decide that it's unlikely that the shop has opened in just one hour. At 3:00 the area is still noisy, and at 4:00, and at 5:00. How many hours in a row does the area need to be noisy before you decide that, with a probability of greater than 50%, the shop is open?