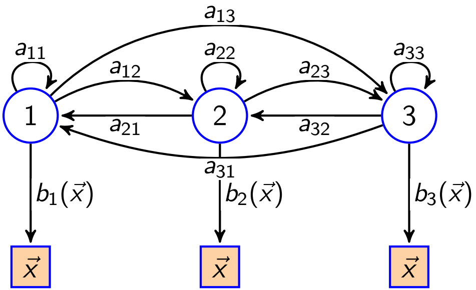
Lecture 15: Baum-Welch

Mark Hasegawa-Johnson

ECE 417: Multimedia Signal Processing, Fall 2021

Outline

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.

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, $p(X|\Lambda_1) > p(X|\Lambda_2)$?
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 $p(X|\Lambda') > p(X|\Lambda)$?

The Forward Algorithm

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

1. **Initialize:**

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

2. **Iterate:**

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

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:**
$$p(X|\Lambda) = \sum_{i=1}^{N} \pi_i b_i(\vec{x}_1)\beta_1(i)$$

Segmentation

1. **The State Posterior:**

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

2. **The Segment Posterior:**

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

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, $p(X|\Lambda_1) > p(X|\Lambda_2)$?
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

Maximum Likelihood Training

Suppose we're given several observation sequences of the form $X = [\vec{x}_1, \ldots, \vec{x}_T]$. Suppose, also, that we have some initial guess about the values of the model parameters (our initial guess doesn't have to be very good). Maximum likelihood training means we want to compute a new set of parameters, $\Lambda' = \left\{\pi'_i, a'_{ij}, b'_j(\vec{x})\right\}$ that maximize $p(X|\Lambda')$.

1. **Initial State Probabilities:** Find values of π'_i, $1 \le i \le N$, that maximize $p(X|\Lambda')$.
2. **Transition Probabilities:** Find values of a'_{ij}, $1 \le i, j \le N$, that maximize $p(X|\Lambda')$.
3. **Observation Probabilities:** Learn $b'_j(\vec{x})$. What does that mean, actually?

Learning the Observation Probabilities

There are four typical ways of learning the observation probabilities, $b_j(\vec{x})$.

1. Vector quantize $\vec{x}$, using some VQ method. Suppose $\vec{x}$ is the k^{th} codevector; then we just need to learn $b_j(k)$ such that

$$b_j(j) \geq 0, \quad \sum_{k=0}^{K-1} b_j(k) = 1$$

2. Model $b_j(k)$ as a Gaussian or mixture Gaussian, and learn its parameters.
3. Model $b_j(k)$ as a neural net, and learn its parameters.

Maximum Likelihood Training

For now, suppose that we have the following parameters that we need to learn:

1. **Initial State Probabilities:** π_i' such that

$$\pi_i' \geq 0, \qquad \sum_{i=1}^{N} \pi_i' = 1$$

2. **Transition Probabilities:** a_{ij}' such that

$$a_{ij}' \geq 0, \qquad \sum_{j=1}^{N} a_{ij}' = 1$$

3. **Observation Probabilities:** $b_j'(k)$ such that

$$b_j'(k) \geq 0, \qquad \sum_{k=1}^{K} b_j'(k) = 1$$

Maximum Likelihood Training with Known State Sequence

Impossible assumption: Suppose that we actually know the state sequences, $Q = [q_1, \ldots, q_T]$, matching with each observation sequence $X = [\vec{x}_1, \ldots, \vec{x}_T]$. Then what would be the maximum-likelihood parameters?

Maximum Likelihood Training with Known State Sequence

Our goal is to find $\Lambda = \{\pi_i, a_{ij}, b_j(k)\}$ in order to maximize

$$\begin{aligned}
\mathcal{L}(\Lambda) &= \ln p(Q, X|\Lambda) \\
&= \ln \pi_{q_1} + \ln b_{q_1}(x_1) + \ln a_{q_1,q_2} + b_{q_2}(x_2) + \dots \\
&= \ln \pi_{q_1} + \sum_{i=1}^{N} \left(\sum_{j=1}^{N} n_{ij} \ln a_{ij} + \sum_{k=1}^{K} m_{ik} \ln b_i(k) \right)
\end{aligned}$$

where

- n_{ij} is the number of times we saw $(q_t = i, q_{t+1} = j)$,
- m_{ik} is the number of times we saw $(q_t = i, k_t = k)$

Maximum Likelihood Training with Known State Sequence

$$\mathcal{L}(\Lambda) = \ln \pi_{q_1} + \sum_{i=1}^{N} \left(\sum_{j=1}^{N} n_{ij} \ln a_{ij} + \sum_{k=1}^{K} m_{ik} \ln b_i(k) \right)$$

When we differentiate that, we find the following derivatives:

$$\frac{\partial \mathcal{L}}{\partial \pi_i} = \begin{cases} \frac{1}{\pi_i} & i = q_1 \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{\partial \mathcal{L}}{\partial a_{ij}} = \frac{n_{ij}}{a_{ij}}$$

$$\frac{\partial \mathcal{L}}{\partial b_j(k)} = \frac{m_{jk}}{b_j(k)}$$

These derivatives are **never** equal to zero! What went wrong?

Maximum Likelihood Training with Known State Sequence

Here's the problem: we forgot to include the constraints $\sum_i \pi_i = 1$, $\sum_j a_{ij} = 1$, and $\sum_k b_j(k) = 1$!
We can include the constraints using the method of Lagrange multipliers. If we do that, we wind up with the solutions

$$\pi_i' = \begin{cases} \frac{1}{\lambda} & i = q_1 \\ 0 & \text{otherwise} \end{cases}$$

$$a_{ij}' = \frac{n_{ij}}{\mu_i}$$

$$b_j(k)' = \frac{m_{jk}}{\nu_j}$$

where λ, μ_i, and ν_j are **arbitrary constants** (called Lagrange multipliers) that we can set to any value we want, provided that the constraints are satisfied.

Maximum Likelihood Training with Known State Sequence

Using the Lagrange multiplier method, we can show that the maximum likelihood parameters for the HMM are:

1. **Initial State Probabilities:**

$$\pi'_i = \frac{\#\text{ state sequences that start with } q_1 = i}{\#\text{ state sequences in training data}}$$

2. **Transition Probabilities:**

$$a'_{ij} = \frac{\#\text{ frames in which } q_{t-1} = i, q_t = j}{\#\text{ frames in which } q_{t-1} = i}$$

3. **Observation Probabilities:**

$$b'_j(k) = \frac{\#\text{ frames in which } q_t = j, k_t = k}{\#\text{ frames in which } q_t = j}$$

Outline

Expectation Maximization

When the true state sequence is unknown, then we can't maximize the likelihood $p(X, Q|\Lambda')$ directly. Instead, we maximize the *expected* log likelihood. This is an instance of the EM algorithm, where the visible training dataset is

$$\mathcal{D}_v = \{\vec{x}_1, \ldots, \vec{x}_T\}$$

and the hidden dataset is

$$\mathcal{D}_h = \{q_1, \ldots, q_T\}$$

Expectation Maximization: the M-Step

In the M-step of EM, we use the E-step probabilities to calculate the **expected** maximum likelihood estimators:

1. **Initial State Probabilities:**

$$\pi_i' = \frac{E\left[\#\text{ state sequences that start with } q_1 = i\right]}{\#\text{ state sequences in training data}}$$

2. **Transition Probabilities:**

$$\pi_i' = \frac{E\left[\#\text{ frames in which } q_{t-1} = i, q_t = j\right]}{E\left[\#\text{ frames in which } q_{t-1} = i\right]}$$

3. **Observation Probabilities:**

$$b_j'(k) = \frac{E\left[\#\text{ frames in which } q_t = j, k_t = k\right]}{E\left[\#\text{ frames in which } q_t = j\right]}$$

Expectation Maximization: the E-Step

In order to find quantities like "the expected number of times $q_1 = i$," we need to do the E-Step of EM. The E-step calculates probabilities like:

$$p(\mathcal{D}_h | \mathcal{D}_v, \Lambda)$$

For example, in order to re-estimate $b_j(k)$, we need to know the # frames in which $q_t = i$. For that, we need

$$p(q_t = i | \vec{x}_1, \vec{x}_2, \ldots, \vec{x}_T, \Lambda)$$

... but this is something we already know! It is

$$p(q_t = i | \vec{x}_1, \vec{x}_2, \ldots, \vec{x}_T, \Lambda) = \gamma_t(i)$$

Expectation Maximization: the E-Step

Similarly, in order to re-estimate a_{ij}, we need to know the # frames in which $q_{t-1} = i$ and $q_t = j$. For that, we need $p(q_{t-1} = i, q_t = j | \vec{x}_1, \vec{x}_2, \ldots, \vec{x}_T, \Lambda)$:

- In the t^{th} frame, the event $q_t = i, q_{t+1} = j$ either happens, or it doesn't happen.
- So the following expectation is actually just a probability:

$$E\left[\#\text{ times during the } t^{\text{th}} \text{ frame, in which } q_t = i, q_{t+1} = j\right]$$
$$= p(q_t = i, q_{t+1} = j)$$
$$= \xi_t(i, j)$$

The Baum-Welch Algorithm

1. **Initial State Probabilities:**

$$\pi_i' = \frac{E\left[\#\text{ state sequences that start with } q_1 = i\right]}{\#\text{ state sequences in training data}}$$
$$= \frac{\sum_{sequences} \gamma_1(i)}{\#\text{ sequences}}$$

The Baum-Welch Algorithm

1. **Initial State Probabilities:**

$$\pi'_i = \frac{\sum_{sequences} \gamma_1(i)}{\#\ \text{sequences}}$$

2. **Transition Probabilities:**

$$\begin{aligned} a'_{ij} &= \frac{E\left[\#\ \text{frames in which } q_{t-1} = i, q_t = j\right]}{E\left[\#\ \text{frames in which } q_{t-1} = i\right]} \\ &= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{j=1}^{N} \sum_{t=1}^{T-1} \xi_t(i,j)} \end{aligned}$$

The Baum-Welch Algorithm

1. **Initial State Probabilities:**

$$\pi_i' = \frac{\sum_{sequences} \gamma_1(i)}{\#\ \text{sequences}}$$

2. **Transition Probabilities:**

$$a_{ij}' = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{j=1}^{N} \sum_{t=1}^{T-1} \xi_t(i,j)}$$

3. **Observation Probabilities:**

$$b_j'(k) = \frac{E\left[\#\ \text{frames in which}\ q_t = j, k_t = k\right]}{E\left[\#\ \text{frames in which}\ q_t = j\right]}$$

The Baum-Welch Algorithm

1. **Initial State Probabilities:**

$$\pi_i' = \frac{\sum_{sequences} \gamma_1(i)}{\#\ \text{sequences}}$$

2. **Transition Probabilities:**

$$a_{ij}' = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{j=1}^{N} \sum_{t=1}^{T-1} \xi_t(i,j)}$$

3. **Observation Probabilities:**

$$b_j'(k) = \frac{\sum_{t:\vec{x}_t=k} \gamma_t(j)}{\sum_t \gamma_t(j)}$$

Outline

Baum-Welch with Gaussian Probabilities

The requirement that we vector-quantize the observations is a problem. It means that we can't model the observations very precisely.

It would be better if we could model the observation likelihood, $b_j(\vec{x})$, as a probability density in the space $\vec{x} \in \Re^D$. One way is to use a parameterized function that is guaranteed to be a properly normalized pdf. For example, a Gaussian:

$$b_i(\vec{x}) = \mathcal{N}\left(\vec{x}; \vec{\mu}_i, \Sigma_i\right)$$

Diagonal-Covariance Gaussian pdf

Let's assume the feature vector has D dimensions, $\vec{x}_t = [x_{t,1}, \ldots, x_{t,D}]$. The Gaussian pdf is

$$b_i(\vec{x}_t) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} e^{-\frac{1}{2}(\vec{x}_t - \vec{\mu}_i)\Sigma_i^{-1}(\vec{x}_t - \vec{\mu}_i)^T}$$

The logarithm of a Gaussian is

$$\ln b_i(\vec{x}_t) = -\frac{1}{2}\left((\vec{x}_t - \vec{\mu}_i)^T\Sigma_i^{-1}(\vec{x}_t - \vec{\mu}_i) + \ln|\Sigma_i| + C\right)$$

where the constant is $C = D\ln(2\pi)$.

Expectation maximization

Expectation maximization maximizes the expected log probability, i.e.,

$$E\left[\ln b_i(\vec{x}_t)\right] = -\frac{1}{2}\sum_{i=1}^{N}\gamma_t(i)\left((\vec{x}_t-\vec{\mu}_i)^T\Sigma_i^{-1}(\vec{x}_t-\vec{\mu}_i)+\ln|\Sigma_i|+C\right)$$

If we include all of the frames, then we get

$$\begin{aligned}E\left[\ln p(X,Q|\Lambda)\right] &= \text{other terms}\\ &\quad -\frac{1}{2}\sum_{t=1}^{T}\sum_{i=1}^{N}\gamma_t(i)\left((\vec{x}_t-\vec{\mu}_i)^T\Sigma_i^{-1}(\vec{x}_t-\vec{\mu}_i)+\ln|\Sigma_i|+C\right)\end{aligned}$$

where the "other terms" are about a_{ij} and π_i, and have nothing to do with $\vec{\mu}_i$ or Σ_i.

M-Step: optimum $\vec{\mu}$

First, let's optimize $\vec{\mu}$. We want

$$0 = \nabla_{\vec{\mu}_q} \sum_{t=1}^{T} \sum_{i=1}^{N} \gamma_t(i)(\vec{x}_t - \vec{\mu}_i)^T \Sigma_i^{-1}(\vec{x}_t - \vec{\mu}_i)$$

Re-arranging terms, we get

$$\vec{\mu}_q' = \frac{\sum_{t=1}^{T} \gamma_t(q)\vec{x}_t}{\sum_{t=1}^{T} \gamma_t(q)}$$

M-Step: optimum Σ

Second, let's optimize Σ_i. In order to do this, we need to talk about the gradient of a scalar w.r.t. a matrix. Let's suppose that

$$\Sigma = \left[\begin{array}{ccc} \sigma_1^2 & \cdots & \rho_{1,D} \\ \vdots & \ddots & \vdots \\ \rho_{D,1} & \cdots & \sigma_D^2 \end{array}\right]$$

When we talk about $\nabla_\Sigma f(\Sigma)$, for some scalar function $f(\cdot)$, what we mean is the matrix whose elements are

$$\nabla_\Sigma f(\Sigma) = \left[\begin{array}{ccc} \frac{\partial f}{\partial \sigma_1^2} & \cdots & \frac{\partial f}{\partial \rho_{1,D}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial \rho_{D,1}} & \cdots & \frac{\partial f}{\partial \sigma_D^2} \end{array}\right]$$

M-Step: optimum Σ

In particular, for a positive-definite, symmetric Σ, it's possible to show that

$$\nabla_{\Sigma} \ln |\Sigma| = \Sigma^{-1}$$

and

$$\nabla_{\Sigma}(\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu}) = -\Sigma^{-1}(\vec{x}-\vec{\mu})(\vec{x}-\vec{\mu})^T\Sigma^{-1}$$

Minimizing the cross-entropy: optimum σ

Taking advantage of those facts, let's find

$$0 = \nabla_{\Sigma_q} \sum_{t=1}^{T} \sum_{i=1}^{N} \gamma_t(i) \left(\ln |\Sigma_i| + (\vec{x}_t - \vec{\mu}_i)^T \Sigma_i^{-1} (\vec{x}_t - \vec{\mu}_i) \right)$$

Re-arranging terms, we get

$$\Sigma_q' = \frac{\sum_{t=1}^{T} \gamma_t(q)(\vec{x}_t - \vec{\mu}_q)(\vec{x}_t - \vec{\mu}_q)^T}{\sum_{t=1}^{T} \gamma_t(q)}$$

Summary: Gaussian Observation PDFs

So we can use Gaussians for $b_j(\vec{x})$:

- **E-Step:**

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i'} \alpha_t(i')\beta_t(i')}$$

- **M-Step:**

$$\vec{\mu}_i' = \frac{\sum_{t=1}^{T} \gamma_t(i)\vec{x}_t}{\sum_{t=1}^{T} \gamma_t(i)}$$

$$\Sigma_i' = \frac{\sum_{t=1}^{T} \gamma_t(i)(\vec{x}_t - \vec{\mu}_i)(\vec{x}_t - \vec{\mu}_i)^T}{\sum_{t=1}^{T} \gamma_t(i)}$$

Outline

The Baum-Welch Algorithm: Initial and Transition Probabilities

1. **Initial State Probabilities:**

$$\pi_i' = \frac{\sum_{sequences} \gamma_1(i)}{\#\ \text{sequences}}$$

2. **Transition Probabilities:**

$$a_{ij}' = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{j=1}^{N} \sum_{t=1}^{T-1} \xi_t(i,j)}$$

The Baum-Welch Algorithm: Observation Probabilities

1. **Discrete Observation Probabilities:**

$$b_j'(k) = \frac{\sum_{t:\vec{x}_t=k} \gamma_t(j)}{\sum_t \gamma_t(j)}$$

2. **Gaussian Observation PDFs:**

$$\vec{\mu}_i' = \frac{\sum_{t=1}^{T} \gamma_t(i)\vec{x}_t}{\sum_{t=1}^{T} \gamma_t(i)}$$

$$\Sigma_i' = \frac{\sum_{t=1}^{T} \gamma_t(i)(\vec{x}_t - \vec{\mu}_i)(\vec{x}_t - \vec{\mu}_i)^T}{\sum_{t=1}^{T} \gamma_t(i)}$$

Outline

Written Example

In a second-order Markov process, q_t depends on both q_{t-2} and q_{t-1}, thus the model parameters are:

$$\pi_{ij} = p(q_1 = i, q_2 = j) \tag{1}$$
$$a_{ijk} = p(q_t = k | q_{t-2} = i, q_{t-1} = i) \tag{2}$$
$$b_k(\vec{x}) = p(\vec{x} | q_t = k) \tag{3}$$

Suppose you have a sequence of observations for which you have already $\alpha_t(i,j)$ and $\beta_t(i,j)$, defined as

$$\alpha_t(i,j) = p(\vec{x}_1, \ldots, \vec{x}_t, q_{t-1} = i, q_t = j | \Lambda) \tag{4}$$
$$\beta_t(i,j) = p(\vec{x}_{t+1}, \ldots, \vec{x}_T | q_{t-1} = i, q_t = j, \Lambda) \tag{5}$$

In terms of the quantities defined in Eqs. (1) through (5), find a formula that re-estimates a'_{ijk} so that, unless a_{ijk} is already optimal,

$$p(X | \pi_i, a'_{ijk}, b_j(\vec{x})) > p(X | \pi_i, a_{ijk}, b_j(\vec{x}))$$