Lecture 13: How to train Observation Probability Densities

Mark Hasegawa-Johnson

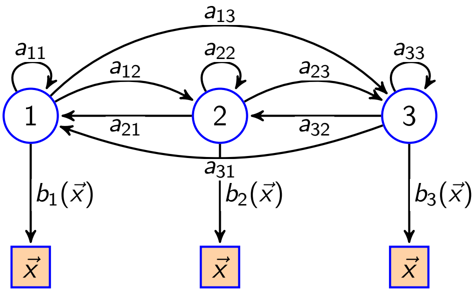
ECE 417: Multimedia Signal Processing, Fall 2020

1. Review: Hidden Markov Models
2. Softmax Observation Probabilities
3. Gaussian Observation Probabilities
4. Discrete Observation Probabilities
5. Summary

Outline

1. Review: Hidden Markov Models
2. Softmax Observation Probabilities
3. Gaussian Observation Probabilities
4. Discrete Observation Probabilities
5. Summary

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 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:**

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

3. **Terminate:**

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

The 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:**

$$\mathcal{L} = -\frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{N} \gamma_t(i) \ln b_i(\vec{x}_t)$$

Outline

1. Review: Hidden Markov Models
2. Softmax Observation Probabilities
3. Gaussian Observation Probabilities
4. Discrete Observation Probabilities
5. Summary

Review: Conditional Probability

The relationship among posterior, prior, evidence and likelihood is

$$p(q|\vec{x})p(\vec{x}) = p(\vec{x}|q)p(q)$$

Since softmax is normalized so that $1 = \sum_q \text{softmax}(e[q])$, it makes most sense to interpret $\text{softmax}(e[q]) = p(q|\vec{x})$. Therefore, the likelihood should be

$$b_q(\vec{x}) \equiv p(\vec{x}|q) = \frac{p(\vec{x})\,\text{softmax}(e[q])}{p(q)}$$

Relationship between the likelihood and the posterior

Therefore, the likelihood should be

$$b_q(\vec{x}) \equiv p(\vec{x}|q) = \frac{p(\vec{x})\,\text{softmax}(e[q])}{p(q)}$$

However,

- If we choose training data with equal numbers of each phone, then we can assume $p(q) = 1/N$.
- $p(\vec{x})$ is independent of q, so it doesn't affect recognition. So let's assume that $p(\vec{x}) = 1/N$ also.

Softmax Observation Probabilities

Given the assumptions that $p(q) = p(\vec{x}) = 1/N$,

$$b_q(\vec{x}) = p(\vec{x}|q) = p(q|\vec{x}) = \text{softmax}(e[q])$$

The assumptions are unrealistic. We sometimes need to adjust for low-frequency phones, in order to get good-quality recognition. But let's first derive the solution given these assumptions, and then we'll see if the assumptions can be relaxed.

Softmax Observation Probabilities

Given the assumptions that $p(q) = p(\vec{x}) = 1/N$,

$$b_q(\vec{x}) = \text{softmax}(e[q]) = \frac{\exp(e[q])}{\sum_{\ell=1}^{N} \exp(e[\ell])},$$

where $e[i]$ is the i^{th} element of the output excitation row vector, $\vec{e} = \vec{h}W$, computed as the product of a weight matrix W with the hidden layer activation row vector, $\vec{h}$.

Expected negative log likelihood

The neural net is trained to minimize the expected negative log likelihood, a.k.a. the cross-entropy between $\gamma_t(i)$ and $b_i(\vec{x}_t)$:

$$\mathcal{L}_{CE} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N}\gamma_t(i)\ln b_i(\vec{x}_t)$$

Remember that, since $\vec{e}=\vec{h}W$, the weight gradient is just:

$$\frac{d\mathcal{L}_{CE}}{dw_{jk}} = \sum_{t=1}^{T}\frac{d\mathcal{L}_{CE}}{de_t[k]}\frac{\partial e_t[k]}{\partial w_{jk}} = \sum_{t=1}^{T}\frac{d\mathcal{L}_{CE}}{de_t[k]}h_t[j],$$

where $h_t[j]$ is the j^{th} component of $\vec{h}$ at time t, and $e_t[k]$ is the k^{th} component of $\vec{e}$ at time t.

Back-prop

Let's find the loss gradient w.r.t. $e_t[k]$. The loss is

$$\mathcal{L}_{CE} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N}\gamma_t(i)\ln b_i(\vec{x}_t)$$

so its gradient is

$$\frac{d\mathcal{L}_{CE}}{de_t[k]} = -\frac{1}{T}\sum_{i=1}^{N}\frac{\gamma_t(i)}{b_i(\vec{x}_t)}\frac{\partial b_i(\vec{x}_t)}{\partial e_t[k]}$$

Differentiating the softmax

The softmax is

$$b_i(\vec{x}) = \frac{\exp(e[i])}{\sum_\ell \exp(e[\ell])} = \frac{A}{B}$$

Its derivative is

$$\begin{aligned}
\frac{\partial b_i(\vec{x})}{\partial e[k]} &= \frac{1}{B}\frac{\partial A}{\partial e[k]} - \frac{A}{B^2}\frac{\partial B}{\partial e[k]} \\
&= \begin{cases} \frac{\exp(e[i])}{\sum_\ell \exp(e[\ell])} - \frac{\exp(e[i])^2}{\left(\sum_\ell \exp(e[\ell])\right)^2} & i = k \\ -\frac{\exp(e[i])\exp(e[k])}{\left(\sum_\ell \exp(e[\ell])\right)^2} & i \neq k \end{cases} \\
&= \begin{cases} b_i(\vec{x}) - b_i^2(\vec{x}) & i = k \\ -b_i(\vec{x})b_k(\vec{x}) & i \neq k \end{cases}
\end{aligned}$$

The loss gradient

The loss gradient it

$$
\begin{aligned}
\frac{d\mathcal{L}_{CE}}{de_t[k]} &= -\frac{1}{T}\sum_{i=1}^{N}\frac{\gamma_t(i)}{b_i(\vec{x}_t)}\frac{\partial b_i(\vec{x}_t)}{\partial e_t[k]} \\
&= -\frac{1}{T}\left(\gamma_t(k)(1-b_k(\vec{x}_t)) - \sum_{i\neq k}\gamma_t(i)b_k(t)\right) \\
&= -\frac{1}{T}\left(\gamma_t(k) - b_k(\vec{x}_t)\sum_{i=1}^{N}\gamma_t(i)\right) \\
&= -\frac{1}{T}\left(\gamma_t(k) - b_k(\vec{x}_t)\right)
\end{aligned}
$$

Summary: softmax observation probabilities

Training W to minimize the cross-entropy between $\gamma_t(i)$ and $b_i(t)$,

$$\mathcal{L}_{CE} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N}\gamma_t(i)\ln b_i(\vec{x}_t),$$

yields the following weight gradient:

$$\frac{d\mathcal{L}_{CE}}{dw_{jk}} = -\frac{1}{T}\sum_{t=1}^{T} h_t[j]\left(\gamma_t(k) - b_k(\vec{x}_t)\right)$$

which vanishes when the neural net estimates $b_k(\vec{x}_t) \rightarrow \gamma_t(k)$ as well as it can.

Summary: softmax observation probabilities

The Baum-Welch algorithm alternates between two types of estimation, often called the E-step (expectation) and the M-step (maximization or minimization):

1. **E-step:** Use forward-backward algorithm to re-estimate $\gamma_t(i) = p(q_t = i|X, \Lambda)$.
2. **M-step:** Train the neural net for a few iterations of gradient descent, so that $b_k(\vec{x}_t) \rightarrow \gamma_t(k)$.

Final note: Those ridiculous assumptions

As a final note, let's see if we can eliminate those ridiculous assumptions, $p(q) = p(\vec{x}) = 1/N$. **How?** Well, the weight gradient goes to zero when $\sum_{t=1}^{T} h_t[j]\left(\gamma_t(k) - b_k(\vec{x}_t)\right) = 0$. There are at least two ways in which this can happen:

1. $b_k(\vec{x}_t) = \gamma_t(k)$. The neural net is successfully estimating the posterior. This is the best possible solution if $p(q = i) = p(\vec{x}) = \frac{1}{N}$.
2. $b_k(\vec{x}_t) - \gamma_t(k)$ is uncorrelated with $h_t[j]$, e.g., because it is zero mean and independent of $\vec{x}_t$.

Final note: Those ridiculous assumptions

The weight gradient goes to zero if $\gamma_t(k) - b_k(\vec{x}_t)$ is zero mean and independent of $\vec{x}_t$. For example,

- $b_k(\vec{x})$ might differ from $\gamma_t(k)$ by a global scale factor. Instead of softmax, we might use some other normalization, either because (a) it's scaled more like a likelihood, or (b) it has nice numerical properties. An example of (b) is:

$$b_i(\vec{x}) = \frac{\exp(e[i])}{\max_j \exp(e[j])}$$

- $b_k(\vec{x})$ might differ from $\gamma_t(k)$ by a phone-dependent scale factor, e.g., we might choose

$$b_i(\vec{x}) = \frac{p(q=i|\vec{x})}{p(q=i)} = \frac{\exp(e[i])}{p(q=i)\sum_{j=1}^{N}\exp(e[j])}$$

Outline

1. Review: Hidden Markov Models
2. Softmax Observation Probabilities
3. Gaussian Observation Probabilities
4. Discrete Observation Probabilities
5. Summary

Baum-Welch with Gaussian Probabilities

Baum-Welch asks us to minimize the cross-entropy between $\gamma_t(i)$ and $b_i(\vec{x}_t)$:

$$\mathcal{L}_{CE} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N}\gamma_t(i)\ln b_i(\vec{x}_t)$$

In order to force $b_i(\vec{x}_t)$ to be a likelihood, rather than a posterior, one way is to use a 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, $\vec{x} = [x_1, \ldots, x_D]$. The Gaussian pdf is

$$\mathcal{N}(\vec{x}; \vec{\mu}, \Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(\vec{x}-\vec{\mu})\Sigma^{-1}(\vec{x}-\vec{\mu})^T}$$

Let's assume a diagonal covariance matrix, $\Sigma = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_D^2)$, so that

$$\mathcal{N}(\vec{x}; \vec{\mu}, \Sigma) = \frac{1}{\sqrt{\prod_{d=1}^{D} 2\pi\sigma_d^2}} e^{-\frac{1}{2}\sum_{d=1}^{D} \frac{(x_d-\mu_d)^2}{\sigma_d^2}}$$

Logarithm of a diagonal covariance Gaussian

The logarithm of a diagonal-covariance Gaussian is

$$\ln b_i(\vec{x}) = -\frac{1}{2}\sum_{d=1}^{D}\frac{(x_d-\mu_d)^2}{\sigma_d^2} - \frac{1}{2}\sum_{d=1}^{D}\ln\sigma_d^2 - \frac{D}{2}\ln(2\pi)$$

Minimizing the cross-entropy

Surprise! The cross-entropy between $\gamma_t(i)$ and $b_i(\vec{x}_t)$ can be minimized in closed form, if $b_i(\vec{x})$ is Gaussian.

$$\mathcal{L}_{CE} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N}\gamma_t(i)\ln b_i(\vec{x}_t)$$
$$= \frac{1}{2T}\sum_{t=1}^{T}\sum_{i=1}^{N}\gamma_t(i)\left(\sum_{d=1}^{D}\frac{(x_{td}-\mu_{id})^2}{\sigma_{id}^2} + \sum_{d=1}^{D}\ln\sigma_{id}^2 + D\ln(2\pi)\right)$$

It's possible to choose μ_{id} and σ_{id}^2 so that

$$\frac{d\mathcal{L}_{CE}}{d\mu_{qd}} = \frac{d\mathcal{L}_{CE}}{d\sigma_{qd}^2} = 0$$

Minimizing the cross-entropy: optimum μ

First, let's optimize μ_{id}. We want

$$0 = \frac{d}{d\mu_{qd}} \sum_{t=1}^{T} \sum_{i=1}^{N} \gamma_t(i) \left(\sum_{d=1}^{D} \frac{(x_{td} - \mu_{id})^2}{\sigma_{id}^2} \right)$$

Re-arranging terms, we get

$$\mu_{qd} = \frac{\sum_{t=1}^{T} \gamma_t(q) x_{td}}{\sum_{t=1}^{T} \gamma_t(q)}$$

Minimizing the cross-entropy: optimum σ

Second, let's optimize σ_{id}^2. We want

$$0 = \frac{d}{d\sigma_{qd}^2} \sum_{t=1}^{T} \sum_{i=1}^{N} \gamma_t(i) \left(\sum_{d=1}^{D} \frac{(x_{td} - \mu_{id})^2}{\sigma_{id}^2} + \sum_{d=1}^{D} \ln \sigma_{id}^2 \right)$$

Re-arranging terms, we get

$$\sigma_{qd}^2 = \frac{\sum_{t=1}^{T} \gamma_t(q)(x_{td} - \mu_{qd})^2}{\sum_{t=1}^{T} \gamma_t(q)}$$

Summary: Gaussian observation probabilities

A Gaussian pdf can be optimized in closed form.

1. The mean is the weighted average of feature vectors:

$$\mu_{id} = \frac{\sum_{t=1}^{T} \gamma_t(i) x_{td}}{\sum_{t=1}^{T} \gamma_t(i)}$$

2. The variance is the weighted average of squared feature vectors:

$$\sigma_{id}^2 = \frac{\sum_{t=1}^{T} \gamma_t(i)(x_{td} - \mu_{id})^2}{\sum_{t=1}^{T} \gamma_t(i)}$$

. . . and then we would re-compute $\gamma_t(i)$ using forward-backward, and so on.

Outline

1. Review: Hidden Markov Models
2. Softmax Observation Probabilities
3. Gaussian Observation Probabilities
4. Discrete Observation Probabilities
5. Summary

Baum-Welch with Discrete Probabilities

Finally, suppose that x_t is discrete, for example, $x_t \in \{1, \ldots, K\}$. In this case, a pretty reasonable way to model the observations is using a lookup table:

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

Optimizing a discrete observation pmf

Again, Baum-Welch asks us to minimize the cross-entropy between $\gamma_t(i)$ and $b_i(x_t)$:

$$\mathcal{L}_{CE} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N}\gamma_t(i)\ln b_i(x_t),$$

but now we also have this constraint to satisfy:

$$1 = \sum_{k=1}^{K} b_i(k)$$

The Lagrangian

We can find the values $b_i(k)$ that minimize $\mathcal{L}_{CE}$ subject to the constraint using a method called Lagrangian optimization. Basically, we create a *Lagrangian*, which is defined to be the original criterion plus λ times the constraint:

$$\mathcal{L} = -\sum_{t=1}^{T}\sum_{i=1}^{N}\gamma_t(i)\ln b_i(x_t) + \lambda\left(1 - \sum_{k=1}^{K} b_i(k)\right)$$

The idea is that there are an infinite number of solutions that will set $\frac{d\mathcal{L}}{db_q(k)} = 0$; we will choose the one that also sets $\sum_k b_i(k) = 1$.

Differentiating The Lagrangian

Differentiating the Lagrangian gives

$$\frac{d\mathcal{L}}{db_q(k)} = -\sum_{t:x_t=k} \frac{\gamma_t(q)}{b_q(k)} - \lambda$$

Setting $\frac{d\mathcal{L}}{db_q(k)} = 0$ gives

$$b_q(k) = \frac{1}{\lambda} \sum_{t:x_t=k} \gamma_t(q)$$

Then we choose λ so that $\sum b_q(k) = 1$.

Outline

1. Review: Hidden Markov Models
2. Softmax Observation Probabilities
3. Gaussian Observation Probabilities
4. Discrete Observation Probabilities
5. Summary

Summary: Estimating the Observation Probability Densities

The Baum-Welch algorithm alternates between two steps, sometimes called the E-step (expectation) and the M-step (maximization or minimization):

1. **E-step:** Use forward-backward algorithm to re-estimate the posterior probability of the hidden state variable, $\gamma_t(i) = p(q_t = i|X, \Lambda)$, given the current model parameters.
2. **M-step:** re-estimate the model parameters, in order to minimize the cross-entropy between $\gamma_t(i)$ and $b_i(x_t)$:

$$\mathcal{L}_{CE} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N}\gamma_t(i)\ln b_i(x_t).$$

Three Types of Observation Probabilities

- Minimizing $\mathcal{L}_{CE}$ for a softmax gives

$$\frac{d\mathcal{L}_{CE}}{dw_{jk}} = -\frac{1}{T}\sum_{t=1}^{T} h_t[j]\left(\gamma_t(k) - b_k(\vec{x}_t)\right)$$

- Minimizing $\mathcal{L}_{CE}$ for a Gaussian gives

$$\mu_{id} = \frac{\sum_{t=1}^{T}\gamma_t(i)x_{td}}{\sum_{t=1}^{T}\gamma_t(i)}$$

$$\sigma_{id}^2 = \frac{\sum_{t=1}^{T}\gamma_t(i)(x_{td}-\mu_{id})^2}{\sum_{t=1}^{T}\gamma_t(i)}$$

- Minimizing $\mathcal{L}_{CE}$ for a discrete pmf gives

$$b_i(k) = \frac{\sum_{t:x_t=k}\gamma_t(i)}{\sum_{t=1}^{T}\gamma_t(i)}$$