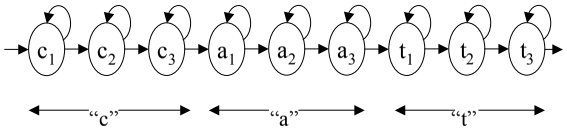
Lecture 16

HMM Training

- Training Hidden Markov Models
  - Initial model estimates
  - Viterbi training
  - Baum-Welch training

Hidden Markov Model Parameters

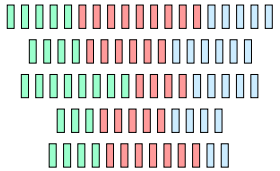


- A Hidden Markov Model for a word specifies the following parameters for each state s:
  - The mean and variance for each of the F elements of the parameter vector: $\boldsymbol{\mu}_s$ and $\boldsymbol{\sigma}_s^2$.
    - These allow us to calculate $d_s(\mathbf{x})$: the output probability density of input frame $\mathbf{x}$ in state s.
  - The transition probabilities $a_{s,j}$ to every possible successor state.
    - $a_{s,j}$ is often zero for all j except $j=s$ and $j=s+1$ it is then called a *left-to-right, no skips* model.
- For a Hidden Markov Model with S states we therefore have around $(2F+1)S$ parameters.
  - A typical word might have $S=15$ and $F=39$ giving 1200 parameters in all.
- Determining the correct values for all these parameters is called *training* the model.
  - We train the model by getting lots of people to say the word lots of times.

Training whole-word models

- Record >20 examples of each word in the vocabulary.
- Decide how many states to have for each word model.
- Choose an initial alignment for each training word.
- Repeat the following steps until alignments converge:
  - Estimate the model parameters from the aligned training examples
  - Align all the training words with the new models using Viterbi alignment.

5 aligned examples of "cat":



Means and Variances:

There are 31 frames aligned to state 2.

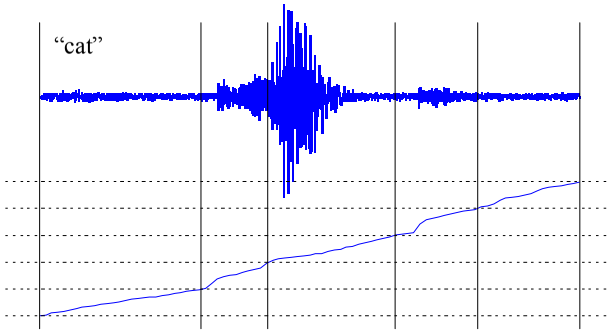
Take the mean and variance of the 31 feature vectors to get the output distribution for state 2.

Transition Probabilities:

Since 5 of the 31 frames are followed by a frame, the transition probability from state 2 to state 3 is 5/31.

Legal transitions that don't occur in the training data get a fixed low probability of, say, 0.001

Initial Alignment



- Calculate the Euclidean distance between the feature vectors from consecutive frames
- Plot the cumulative sum of these distances and divide the range up into N equal segments
  - Cumulative sum graph will be steepest where the spectrum is changing most rapidly
  - States will be short when spectrum is changing rapidly and long where it is constant.
- Initial alignment doesn't matter all that much. Some recognisers just divide the speech signal into equal length chunks.

Probabilistic Alignment

- Viterbi alignment finds the *best* alignment of a speech signal with a model.
  - There may be several other alignments that are almost as good as the best one.
- Our training procedure is called *Viterbi reestimation* of the model.
  - Viterbi reestimation only considers the best alignment of each training example and ignores all other possible alignments
- Baum-Welch reestimation considers all possible alignments and weights them with their production probabilities.
- We define two quantities:
  - $P(s,t)$ is the total probability density that the model generates $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t$ summed over all alignments having frame t in state s.
  - $Q(s,t)$ is the total probability density that the model generates $\mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \ldots, \mathbf{x}_T$ summed over all alignments having frame t in state s.
- It follows that $P(s,t)\times Q(s,t)$ is the probability density that the model generates the observed speech signal **and** that frame t is in state s .

State Assignment Probabilities

- For any alignment, frame $\mathbf{x}_t$ must be in one of the states $1, 2, \ldots, S$. Hence

$$\sum_{s=1}^{S} P(s,t)\times Q(s,t) = P$$

  where P is the total probability of generating the word summed over all possible alignments.
- With Viterbi reestimation, we used the best alignment to assign $\mathbf{x}_t$ to one particular state.
- With Baum-Welch reestimation we assign parts of $\mathbf{x}_t$ to many different states. The fraction of $\mathbf{x}_t$ that it is assigned to state s is given by

$$A(s,t) = \frac{P(s,t)\times Q(s,t)}{P} \quad \text{where} \quad \sum_{s=1}^{S} A(s,t) = 1$$

- The total "number" of frames assigned to state s is given by

$$\sum_{t=1}^{T} A(s,t)$$

Baum-Welch Reestimation

- We estimate the mean feature vector in each state by taking a weighted average of **all** frames from **all** N training examples of the word.

$$\mathbf{m}_s = \frac{\sum_{n=1}^{N}\sum_{t=1}^{T_n} A_n(s,t)\times \mathbf{x}_t}{\sum_{n=1}^{N}\sum_{t=1}^{T_n} A_n(s,t)}$$

- Similarly, we estimate the covariance matrix by taking a weighted average of the squared deviations from the mean.

$$\mathbf{C}_s = \frac{\sum_{n=1}^{N}\sum_{t=1}^{T_n} A_n(s,t)\times (\mathbf{x}_t - \mathbf{m}_s)(\mathbf{x}_t - \mathbf{m}_s)^T}{\sum_{n=1}^{N}\sum_{t=1}^{T_n} A_n(s,t)}$$

- If the elements of the feature vector are independent, then $\mathbf{C}$ will be a diagonal matrix.

Baum-Welch Transition Probabilities

- We calculate the transition probability a_{sk} by finding out what fraction of frames that were assigned to state s had the following frame assigned to state k.
- The total probability density of all alignments with $\mathbf{x}_t$ in state s and $\mathbf{x}_{t+1}$ in state k is

$$P(s,t)\times a_{sk}\times d_k(x_{t+1})\times Q(k,t+1)$$

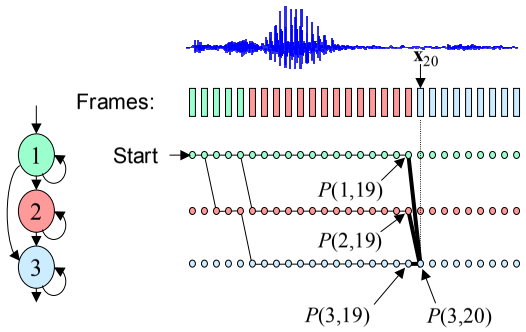
  the fraction of alignments with this property is therefore

$$P_n(s,t)\times a_{sk}\times d_k(x_{n,t+1})\times Q_n(k,t+1)/P_n$$

- We calculate a_{sk} by adding up the total number of times that consecutive frames are assigned to states s and k respectively and dividing by the total number of frames that are assigned to state s

$$a_{sk} = \frac{\sum_{n=1}^{N}\sum_{t=1}^{T_n} P_n(s,t)\times a_{sk}\times d_k(x_{n,t+1})\times Q_n(k,t+1)/P_n}{\sum_{n=1}^{N}\sum_{t=1}^{T_n} A_n(s,t)}$$

Forward Probability Calculation



- $P(s,t)$ is the total probability density that the model generates $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t$ summed over all alignments having frame t in state s.
- Any alignment going through (3,20) must go through either (1,19), (2,19) or (3,19). Hence:

$$P(3,20) = \left(\sum_{s=1}^{3} P(s,19) \times a_{s3} \right) \times d_3(\mathbf{x}_{20})$$

- In general, we can calculate $P(*,t)$ from $P(*,t-1)$:

$$P(k,t) = \left(\sum_{s=1}^{S} P(s,t-1) \times a_{sk} \right) \times d_k(\mathbf{x}_t)$$

Forward Probability Recursion

- The calculation of $P(s,t)$ is very similar to the calculation of $B(s,t)$ except that we add together the probabilities at each stage instead of just taking the biggest.

$P(1,1) = d_1(\mathbf{x}_1)$; $P(s,1) = 0$ for $s \neq 1$
for t=2:T
  for k=1:S

$$P(k,t) = \left(\sum_{s=1}^{S} P(s,t-1) \times a_{sk} \right) \times d_k(\mathbf{x}_t)$$

  end
end

- Since all alignments have to have the last frame in state S, the total probability density over all alignments is given by:

$$P = P(S,T) \times a_{S=}$$

  - $a_{S=}$ is the exit probability from the final state

Backward Probability Recursion

- $Q(k,t)$ is the total probability density that the model generates $\mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \ldots, \mathbf{x}_T$ summed over all alignments having frame t in state k.
- All alignments having frame t in state k must have frame t+1 in some other state s. We can therefore calculate $Q(k,t)$ in terms of $Q(*,t+1)$.
- We use a recursion going backwards in time:

$Q(S,T) = a_{S=}$; $Q(s,T) = 0$ for $s \neq S$
for t=T–1:–1:1
  for k=1:S

$$Q(k,t) = \sum_{s=1}^{S} a_{ks} \times d_s(\mathbf{x}_{t+1}) \times Q(s,t+1)$$

  end
end

- Since all alignments must have frame 1 in state 1, we have

$$P = d_1(\mathbf{x}_1) \times Q(1,1)$$

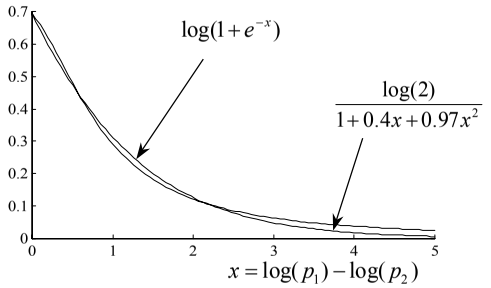
Log Probabilities

- Probability products get very small so we use log probabilities to avoid underflow problems. The Viterbi calculation only requires probability products but the forward/backward calculations also requires probability sums.
- Logs of products are easy but logs of sums are difficult:

$$\log(p_1 \times p_2) = \log(p_1) + \log(p_2)$$

$$\log(p_1 + p_2) = \log(p_1) + \log\left(1 + \frac{p_2}{p_1}\right)$$

- We can always choose $p_1 \geq p_2$ so $0 \leq p_2/p_1 \leq 1$. We can then approximate $\log(1+p_2/p_1)$ over this range as a function of $\log(p_1/p_2)$:



Summary of HMM Training Procedure

- The following steps must be performed for each different word that you need to recognise:
  - Make a crude model: we have seen one of several possible methods.
  - Do Viterbi training until the model converges:
    - Align each of your training examples with your current model.
    - Calculate new values for $\mathbf{m}_s$ and $\mathbf{C}_s$ by averaging over all frames that align with state s.
    - Calculate new values for a_{sk} by taking the fraction of the above frames for which the following frame aligned with state k.
  - Do a few (up to 10) iterations of Baum-Welch training:
    - Calculate $P(s,t)$ and $Q(s,t)$ for all t and s. These are called the *forward* and *backward* probabilities respectively.
    - Calculate new values for $\mathbf{m}_s$ and $\mathbf{C}_s$ by forming a weighted average over all the training frames.
    - Calculate new values for a_{sk} by taking the ratio of the pdf of paths through both s and k to that of all paths through s.