ELEN E6884 - Topics in Signal Processing
Topic: Speech Recognition
Lecture 3

Stanley F. Chen, Michael A. Picheny, and Bhuvana Ramabhadran

IBM T.J. Watson Research Center

Yorktown Heights, NY, USA

stanchen@us.ibm.com, picheny@us.ibm.com,

bhuvana@us.ibm.com

22 September 2009

Outline of Today's Lecture

- Recap
- Gaussian Mixture Models - A
- Gaussian Mixture Models - B
- Introduction to Hidden Markov Models

Administrivia

- main feedback from last lecture
  - EEs: Speed ok
  - CSs: Hard to follow
- Remedy: Only one more lecture will have serious signal processing content so don't worry!
- Lab 1 due Sept 30 (don't wait until the last minute!)

Where are We?

- Can extract feature vectors over time - LPC, MFCC, or PLPs - that characterize the information in a speech signal in a relatively compact form.
- Can perform simple speech recognition by
  - Building templates consisting of sequences of feature vectors extracted from a set of words
  - Comparing the feature vectors for a new utterance against all the templates using DTW and picking the best scoring template
- Learned about some basic concepts (e.g., graphs, distance measures, shortest paths) that will appear over and over again throughout the course

What are the Pros and Cons of DTW

Pros

- Easy to implement and compute
- Lots of freedom - can model arbitrary time warpings

Cons

- Distance measures completely heuristic.
  - Why Euclidean? Are all dimensions of the feature vector created equal?
- Warping paths heuristic
  - Too much freedom is not always a good thing for robustness
  - Allowable path moves all hand-derived
- No guarantees of optimality or convergence

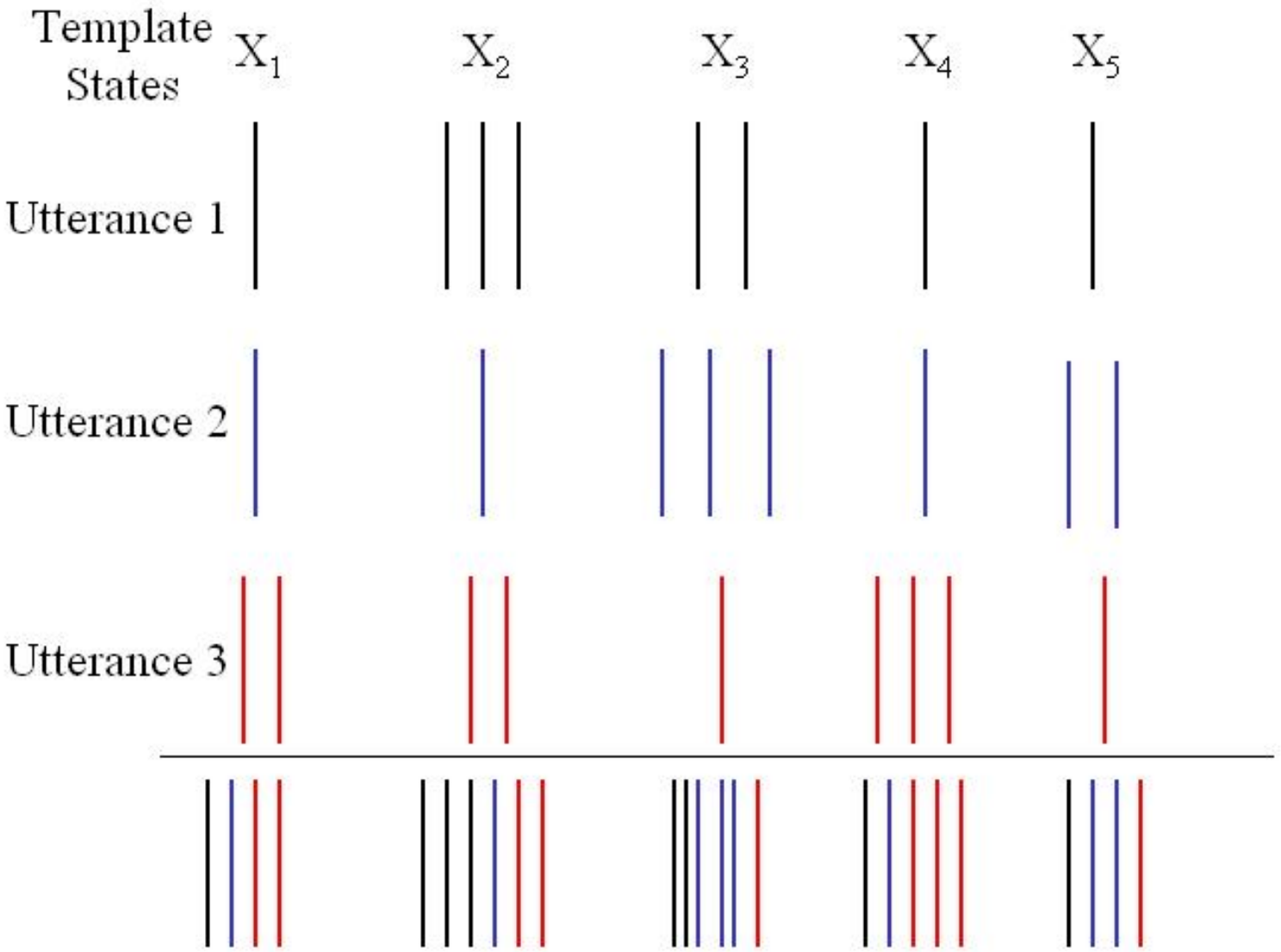
How can we Do Better?

- Key insight 1: Learn as much as possible from data - Distance measure, weights on graph, even graph structure itself (future research)
- Key insight 2: Use well-described theories and models from probability, statistics, and computer science to describe the data rather than developing new heuristics with ill-defined mathematical properties
- Start by modeling the behavior of the distribution of feature vectors associated with different speech sounds leading to a particular set of models called Gaussian Mixture Models - a formalism of the concept of the distance measure.
- Then derive models for describing the time evolution of feature vectors for speech sounds and words, called Hidden Markov Models, a generalization of the template idea in DTW.

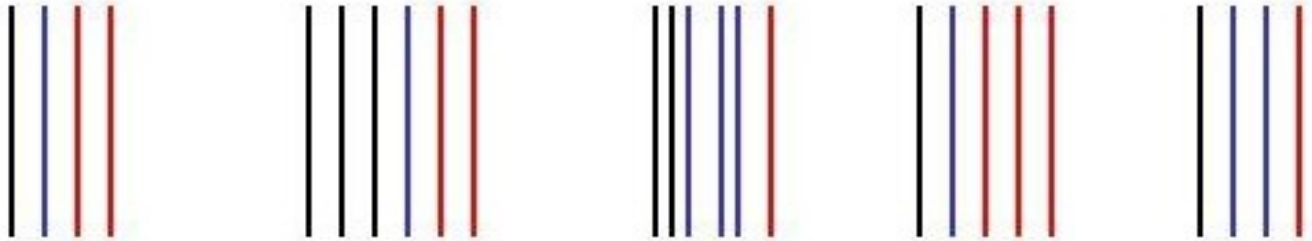
Gaussian Mixture Model Overview

- Motivation for using Gaussians
- Univariate Gaussians
- Multivariate Gaussians
- Estimating parameters for Gaussian Distributions
- Need for Mixtures of Gaussians
- Estimating parameters for Gaussian Mixtures
- Initialization Issues
- How many Gaussians?

How do we Capture Variability?



Data Models

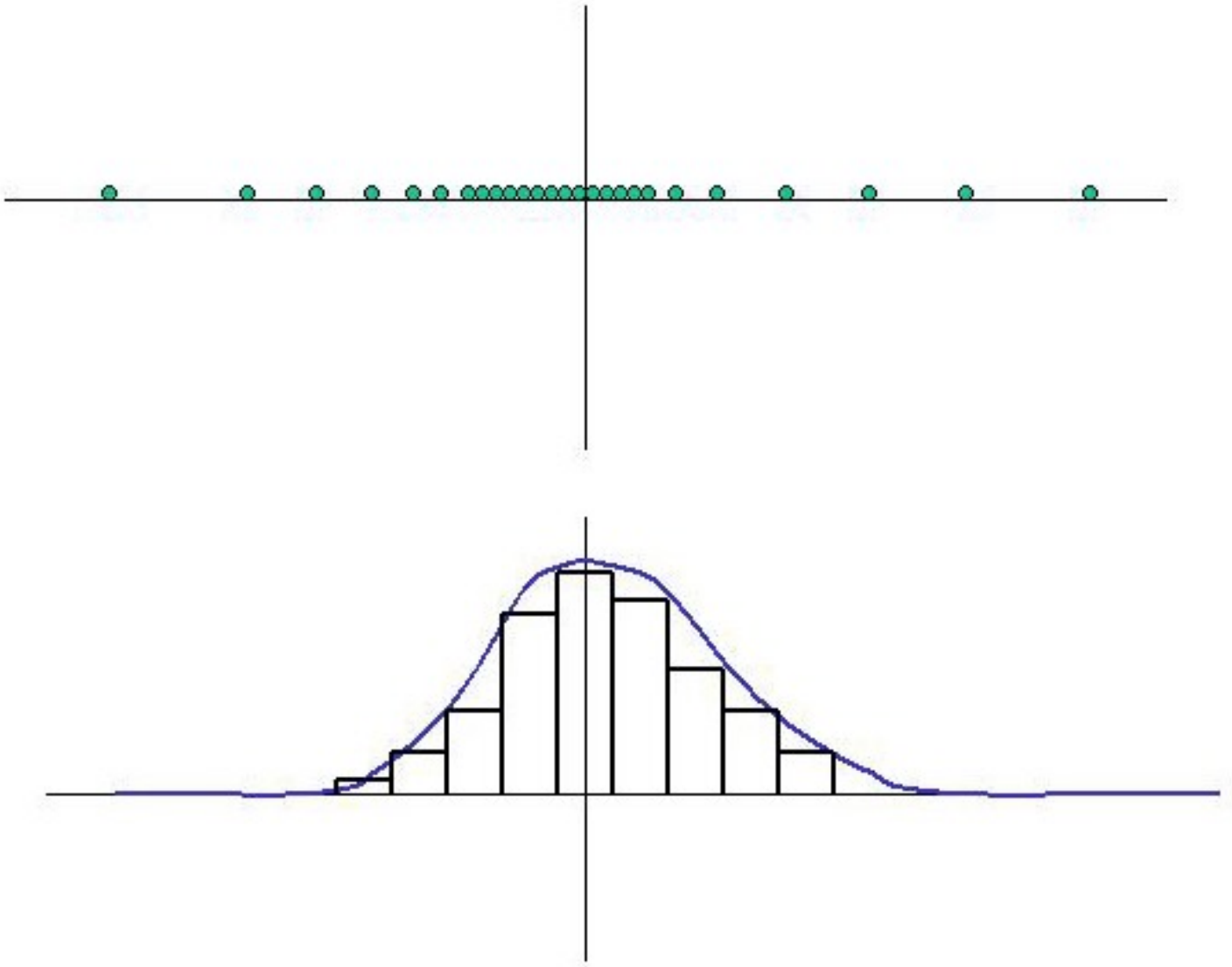


- We can average the data for each state and compute the Euclidean distance as usual.
- We can keep all the data for each state and find the best match

$$d(x_j, O_t) = \min_j(|O_t - x_{mj}|^2)$$

- We can try to make a MODEL for the data in a given state.

The Gaussian Distribution



A lot of different types of data are distributed like a “bell-shaped

curve". Mathematically we can represent this by what is called a *Gaussian* or *Normal* distribution:

$$\mathcal{N}(\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(O-\mu)^2}{2\sigma^2}}$$

μ is called the mean and σ^2 is called the variance. The value at a particular point O is called the *likelihood*. The integral of the above distribution is 1:

$$\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(O-\mu)^2}{2\sigma^2}} dO = 1$$

It is often easier to work with the logarithm of the above:

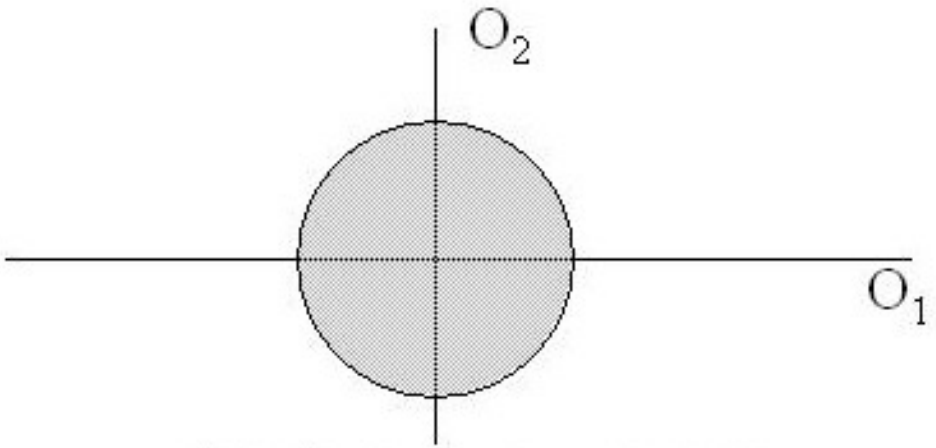
$$-\ln \sqrt{2\pi}\sigma - \frac{(O-\mu)^2}{2\sigma^2}$$

which looks suspiciously like a weighted Euclidean distance!

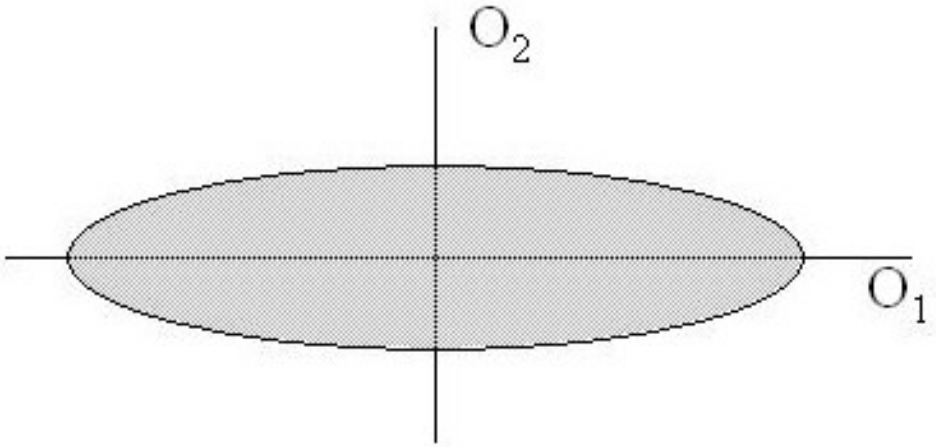
Advantages of Gaussian Distributions

- Central Limit Theorem: Sums of large numbers of identically distributed random variables tend to Gaussian
- The sums and differences of Gaussian random variables are also Gaussian
- If X is distributed as $\mathcal{N}(\mu, \sigma)$ then $aX + b$ is distributed as $\mathcal{N}(a\mu + b, (a\sigma)^2)$

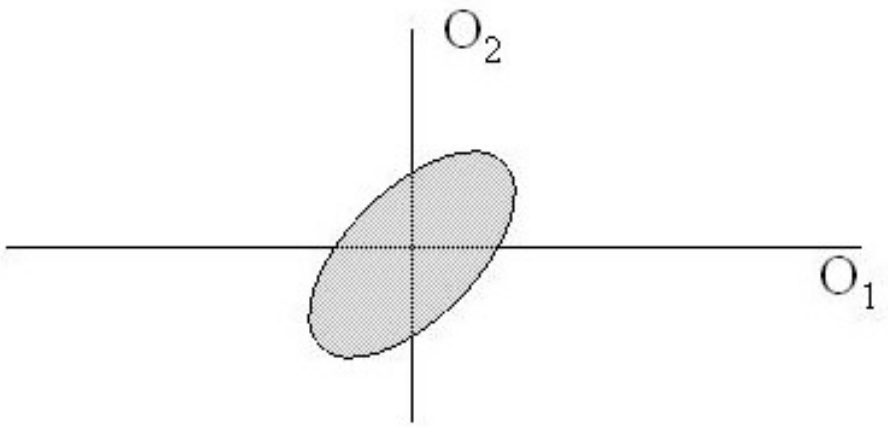
Gaussians in Two Dimensions



O_1 and O_2 are uncorrelated – knowing O_1 tells you nothing about O_2



O_1 and O_2 can be uncorrelated without having equal variances



O_1 and O_2 are correlated – knowing O_1 tells you something about O_2

$$\mathcal{N}(\mu_1, \mu_2, \sigma_1, \sigma_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-r^2}} e^{-\frac{1}{2(1-r^2)}\left(\frac{(O_1-\mu_1)^2}{\sigma_1^2} - \frac{2rO_1O_2}{\sigma_1\sigma_2} + \frac{(O_2-\mu_2)^2}{\sigma_2^2}\right)}$$

If $r = 0$ can write the above as

$$\frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(O_1-\mu_1)^2}{2\sigma_1^2}} \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(O_2-\mu_2)^2}{2\sigma_2^2}}$$

If we write the following matrix:

$$\Sigma = \begin{vmatrix} \sigma_1^2 & r\sigma_1\sigma_2 \\ r\sigma_1\sigma_2 & \sigma_2^2 \end{vmatrix}$$

using the notation of linear algebra, we can write

$$\mathcal{N}(\mu, \mathbf{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(\mathbf{O}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{O}-\mu)}$$

where $\mathbf{O} = (O_1, O_2)$ and $\mu = (\mu_1, \mu_2)$. More generally, μ and $\mathbf{\Sigma}$ can have arbitrary numbers of components, in which case the above is called a *multivariate* Gaussian.

We can write the logarithm of the multivariate likelihood of the Gaussian as:

$$-\frac{n}{2}\ln(2\pi) - \frac{1}{2}\ln|\Sigma| - \frac{1}{2}(\mathbf{O}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{O}-\mu)$$

For most problems we will encounter in speech recognition, we will assume that Σ is diagonal so we may write the above as:

$$-\frac{n}{2}\ln(2\pi) - \sum_{i=1}^{n}\ln\sigma_i - \frac{1}{2}\sum_{i=1}^{n}(O_i - \mu_i)^2/\sigma_i^2$$

Again, note the similarity to a weighted Euclidean distance.

Estimating Gaussians

Given a set of observations $\mathbf{O}_1, \mathbf{O}_2, \ldots, \mathbf{O}_N$ it can be shown that μ and $\mathbf{\Sigma}$ can be estimated as:

$$\mu = \frac{1}{N} \sum_{i=1}^{N} \mathbf{O}_i$$

and

$$\mathbf{\Sigma} = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{O}_i - \mu)^T (\mathbf{O}_i - \mu)$$

How do we actually derive these formulas?

Maximum-Likelihood Estimation

For simplicity, we will assume a univariate Gaussian. We can write the likelihood of a string of observations $O_1^N = O_1, O_2, \ldots, O_N$ as the product of the individual likelihoods:

$$\mathcal{L}(O_1^N|\mu,\sigma) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(O_i-\mu)^2}{2\sigma^2}}$$

It is much easier to work with $L = \ln \mathcal{L}$:

$$L(O_1^N|\mu,\sigma) = -\frac{N}{2}\ln 2\pi\sigma^2 - \frac{1}{2}\sum_{i=1}^{N}\frac{(O_i-\mu)^2}{\sigma^2}$$

To find μ and σ we can take the partial derivatives of the above

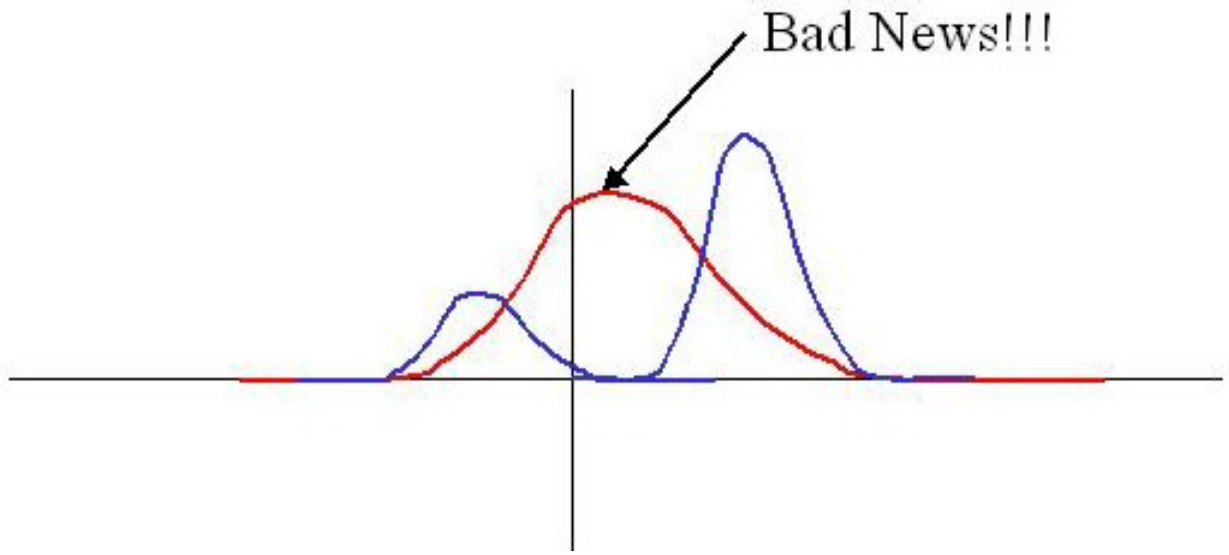
expressions:

$$\frac{\partial L(O_1^N|\mu,\sigma)}{\partial \mu} = \sum_{i=1}^{N} \frac{(O_i - \mu)}{\sigma^2} \quad (1)$$

$$\frac{\partial L(O_1^N|\mu,\sigma)}{\partial \sigma^2} = -\frac{N}{2\sigma^2} + \sum_{i=1}^{N} \frac{(O_i - \mu)^2}{\sigma^4} \quad (2)$$

By setting the above terms equal to zero and solving for μ and σ we obtain the classic formulas for estimating the means and variances. Since we are setting the parameters based on maximizing the likelihood of the observations, this process is called *Maximum-Likelihood* Estimation, or just *ML* estimation.

Problems with Gaussian Assumption



What can we do? Well, in this case, we can try modeling this with **two** Gaussians:

$$\mathcal{L}(O) = p_1 \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(O-\mu_1)^2}{2\sigma_1^2}} + p_2 \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(O-\mu_2)^2}{2\sigma_2^2}}$$

where $p_1 + p_2 = 1$.

More generally, we can use an arbitrary number of Gaussians:

$$\sum_i p_i \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(O-\mu_i)^2}{2\sigma_i^2}}$$

this is generally referred to as a *Mixture* of Gaussians or a *Gaussian Mixture Model* or *GMM*. Essentially any distribution of interest can be modeled with GMMs.

Issues with ML Estimation of GMMs

How many Gaussians? (to be discussed later....)

Infinite solutions: For the two-mixture case above, we can write the overall log likelihood of the data as:

$$\sum_{i=1}^{N} \ln \left(p_1 \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(O_i-\mu_1)^2}{2\sigma_1^2}} + p_2 \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(O_i-\mu_2)^2}{2\sigma_2^2}} \right)$$

Say we set $\mu_1 = O_1$. We can then write the above as

$$\ln \left(\frac{1}{2\sqrt{2\pi}\sigma_1} + \frac{1}{2\sqrt{2\pi}\sigma_2} e^{\frac{1}{2}\frac{(O_1-\mu_2)^2}{\sigma_2^2}} \right) + \sum_{i=2}^{N} \ldots$$

which clearly goes to ∞ as $\sigma_1 \rightarrow 0$. Empirically we can restrict our attention to the finite local maxima of the likelihood function.

This can be done by such techniques as flooring the variance and eliminating solutions in which μ is estimated from essentially a single data point.

Solving the equations: Very ugly. Unlike single Gaussian case, a closed form solution does not exist.

What are some methods you could imagine?

Estimating Mixtures of Gaussians - Intuition

Can we break down the problem? (Let's focus on the two Gaussian case for now).

- For each data point O_i, if we knew which Gaussian it belonged to, we could just compute μ_1, the mean of the first Gaussian, as

$$\mu_1 = \frac{1}{N_1} \sum_{O_i \in G_1} O_i$$

  where G_1 is the first Gaussian. Similar formulas follow for the other parameters.
- Well, we don't know which one it belongs to. So let's devise a scheme to divvy each data point across the Gaussians.
- First, make some initial reasonable guesses about the parameter values (more on this later)

- Second, divvy up the data point using the following formula

$$C(i,j) = \left(p_j \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(O_i - \mu_j)^2}{2\sigma_j^2}} \right) / \mathcal{L}(O_i)$$

  Observe that this will be a number between 0 and 1. This is called the *a posteriori probability* of Gaussian j producing O_i. In speech recognition literature, this is also called the *Count*.
- This probability is a measure of how much a data point can be assumed to "belong" to a particular Gaussian given a set of parameter values for the means and covariances.
- We then estimate μ and σ using a modified version of the Gaussian estimation equations presented earlier:

$$\mu_j = \frac{1}{C(j)} \sum_{i=1}^{N} O_i C(i,j)$$

and

$$\sigma_j = \frac{1}{C(j)} \sum_{i=1}^{N} (O_i - \mu_j)^2 C(i,j)$$

where $C(j) = \sum_i C(i,j)$.

- Use these estimates and repeat the process several times.
- A typical stopping criteria is to compute the log likelihood of the data after each iteration and compare it to the value on the previous iteration.
- The beauty of this estimation process is that it can be shown that this not only increases the likelihood but eventually converges to a local minimum (E-M Algoritm, more details in a later lecture).

Two-mixture GMM Solution

The log-likelihood of the two mixture case is

$$\sum_{i=1}^{N} \ln \left(p_1 \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(O_i-\mu_1)^2}{2\sigma_1^2}} + p_2 \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(O_i-\mu_2)^2}{2\sigma_2^2}} \right)$$

We can use Lagrange multiplers to satisfy the constraint that $p_1 + p_2 = 1$. We take the derivative with respect to each parameter and set the result equal to zero:

$\frac{\partial}{\partial p_1}$:

$$\sum_{i=1}^{N} \frac{\frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(O_i-\mu_1)^2}{2\sigma_1^2}}}{\frac{p_1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(O_i-\mu_1)^2}{2\sigma_1^2}} + \frac{p_2}{\sqrt{2\pi}\sigma_2} e^{-\frac{(O_i-\mu_2)^2}{2\sigma_2^2}}} + \lambda = 0$$

Define

$$C(i,j) = \frac{\frac{p_j}{\sqrt{2\pi}\sigma_j} e^{-\frac{(O_i-\mu_j)^2}{2\sigma_j^2}}}{\frac{p_1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(O_i-\mu_1)^2}{2\sigma_1^2}} + \frac{p_2}{\sqrt{2\pi}\sigma_2} e^{-\frac{(O_i-\mu_2)^2}{2\sigma_2^2}}}$$

The above equation is then:

$$\sum_{i=1}^{N} C(i,1)/p_1 + \lambda = 0$$

So we can write

$$\begin{aligned} p_1 &= -\tfrac{1}{\lambda}\textstyle\sum_i C(i,1) &&= -\tfrac{1}{\lambda}C(1) \\ p_2 &= -\tfrac{1}{\lambda}\textstyle\sum_i C(i,2) &&= -\tfrac{1}{\lambda}C(2) \end{aligned}$$

Since $p_1 + p_2 = 1$ we can write

$$-\frac{1}{\lambda}(C(1) + C(2)) = 1 \Rightarrow \lambda = -\frac{1}{C(1) + C(2)}$$

and

$$p_1 = C(1)/(C(1) + C(2)); p_2 = C(2)/(C(1) + C(2))$$

Similarly, $\frac{\partial}{\partial \mu_1}$:

$$\sum_{i=1}^{N} C(i, 1)(O_i - \mu_1) = 0$$

implies that

$$\mu_1 = \sum_{i=1}^{N} C(i, 1)O_i/C(1)$$

and with similar manipulation,

$$\sigma_1 = \sum_{i=1}^{N} C(i,1)(O_i - \mu_1)^2 / C(1)$$

The n-dimensional case is derived in the handout from Duda and Hart.

Initialization

How do we come up with initial values of the parameters? One solution:

- set all the p_is to $1/N$
- pick N data points at random and use them to seed the initial values of μ_i
- set all the initial σs to an arbitrary value, or the global variance of the data.

Similar solution: Try multiple starting points, pick the one with the highest overall likelihood (why would one do this?)

Splitting:

- Initial: Compute global mean and variance
- Repeat: Perturb each mean by $\pm\epsilon$ (doubling the number of Gaussians)

- Run several iterations of the GMM parameter mixture estimation algorithm.

Have never seen a comprehensive comparison of the above two schemes!

Number of Gaussians

Method 1 (most common): Guess!

Method 2: Penalize the likelihood by the number of parameters (Bayesian Information Criterion (BIC)[1]):

$$BIC(C_k) = \sum_{i=1}^{k} \{-\frac{1}{2} n_i \log |\Sigma_i|\} - Nk(d + \frac{1}{2} d(d+1))$$

where k is the number of clusters, n_i the number of data points in cluster i, N the total number of data points, and d the dimensionality of the parameter vector.

Such penalty terms can be derived by viewing a GMM as a way

of coding data by transmitting the id of the closest Gaussian. In such a case, the number of bits required for transmission of the data also includes the cost of transmitting the model itself; the bigger the model, the larger the cost.

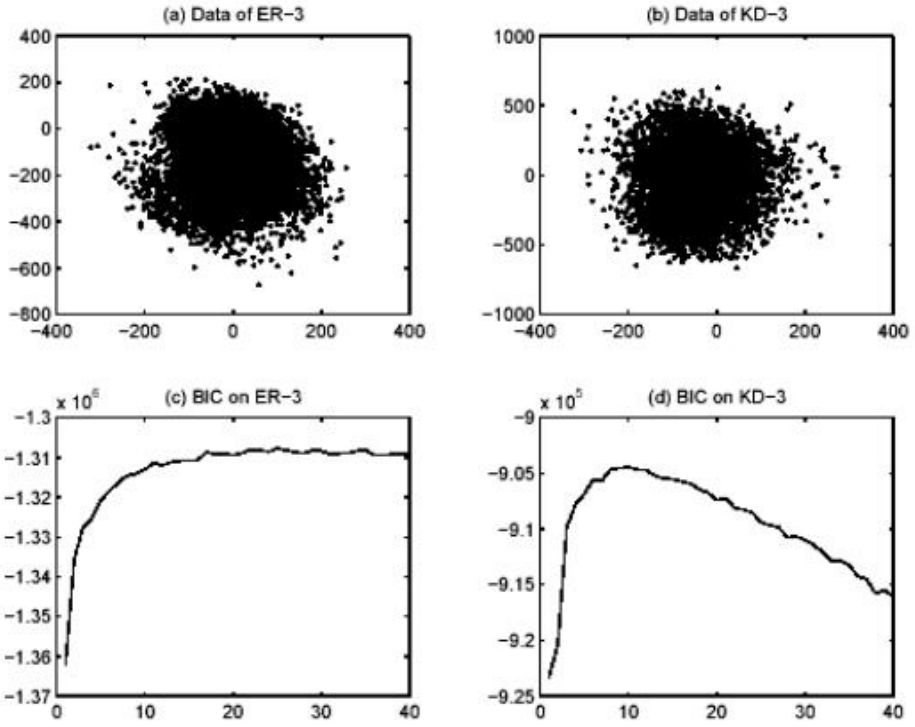


Figure 1. Different degrees of complexity in phone ER-3 and KD-3

References

[1] S. Chen and P. S. Gopalakrishnan (1998)“Clustering via the Bayesian Information Criterion with Applications in Speech Recognition”, ICASSP-98, Vol. 2 ppp 645-648.

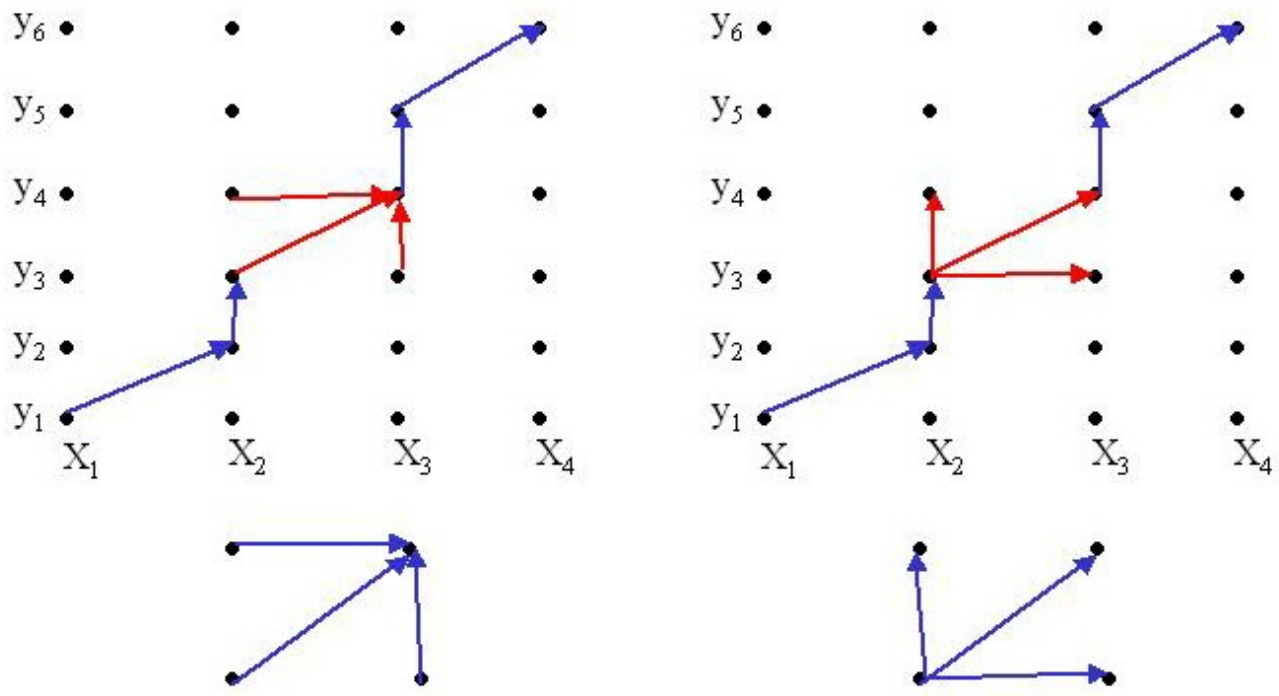
Introduction to Hidden Markov Models

- The issue of weights in DTW
- Interpretation of DTW grid as Directed Graph
- Adding Transition and Output Probabilities to the Graph gives us an HMM!
- The three main HMM operations

Another Issue with Dynamic Time Warping

Weights are completely heuristic!

Maybe we can learn the weights from data?



- Take many utterances
- For each node in the DP path, count number of times move up

$\uparrow$ right $\rightarrow$ and diagonally $\nearrow$.

- Normalize number of times each direction taken by the total number of times the node was actually visited.
- Take some constant times the reciprocal as the weight.

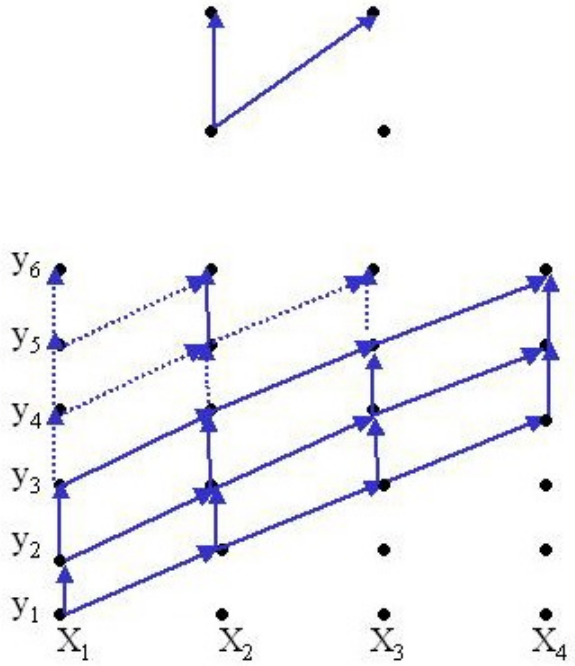
For example, if a particular node was visited 100 times, and after alignment, the diagonal path was taken 50 times, and the "up" and "right" paths 25 times each, the weights could be set to 2, 4, and 4, respectively, or (more commonly) 1, 2 and 2.

Point is that it seems to make sense that if you observe out of a given node, a particular direction is favored, the weight distribution should reflect it.

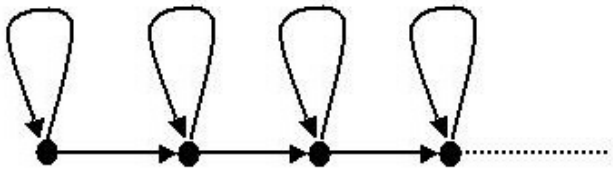
There is no real solution to weight estimation in DTW but something called a *Hidden Markov Model* tries to put the weight estimation ideas and the GMM concepts in a single consistent probabilistic framework.

DTW and Directed Graphs

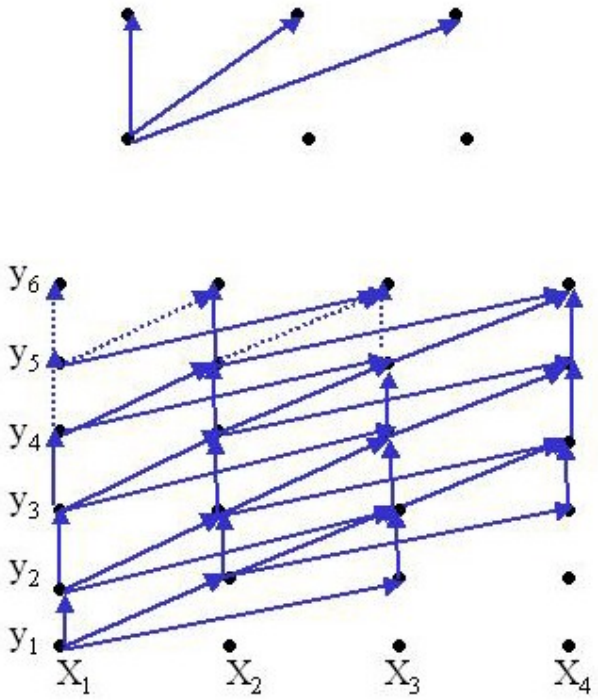
Take the following Dynamic Time Warping setup:



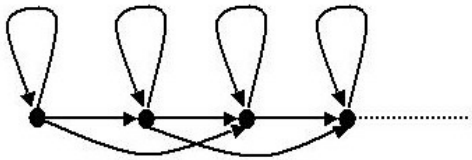
Let's look at a compact representation of this as a directed graph:



Another common DTW structure:



and a directed graph:

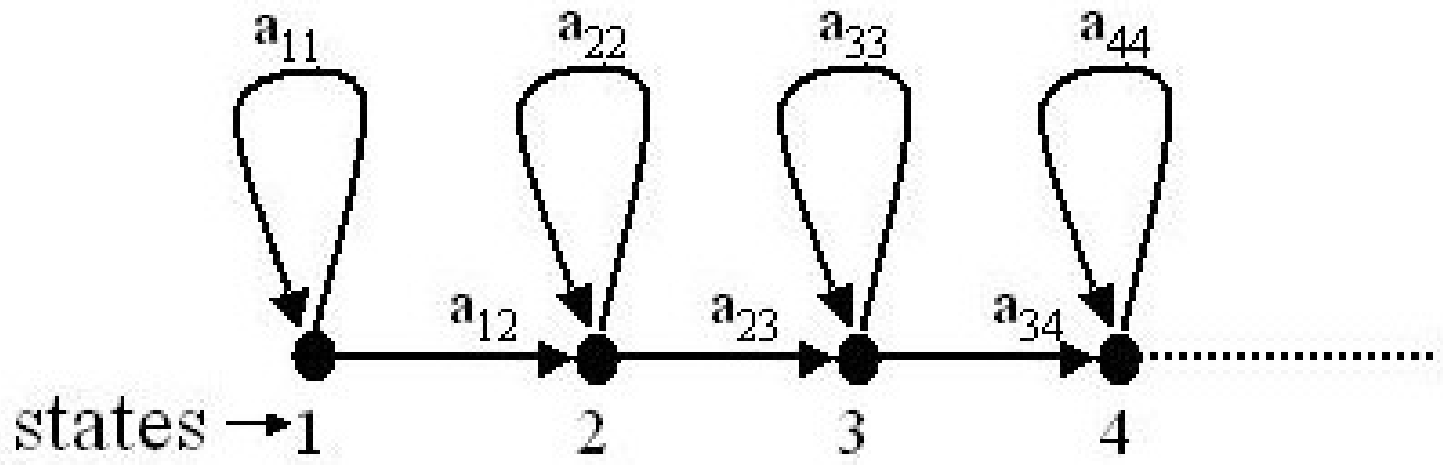


One can represent even more complex DTW structures though

the resultant directed graphs can get quite bizarre looking....

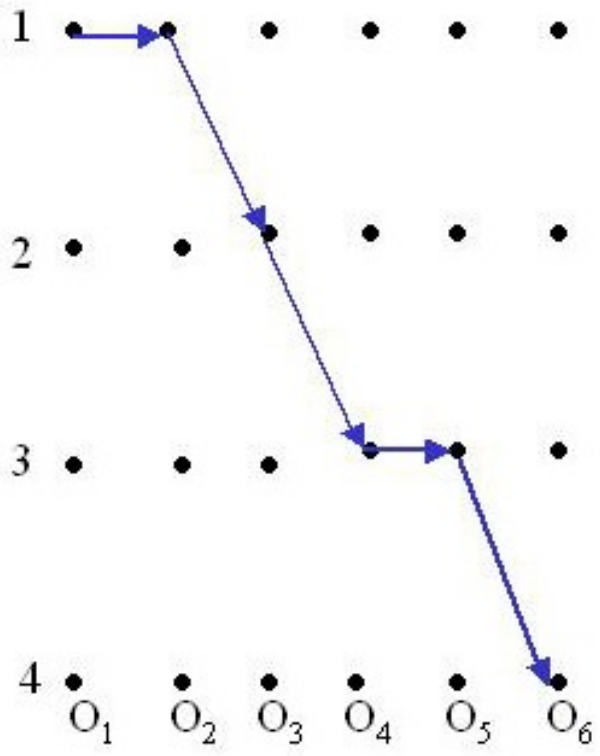
Path Probabilities

Let us now assign probabilities to the transitions in the directed graph:

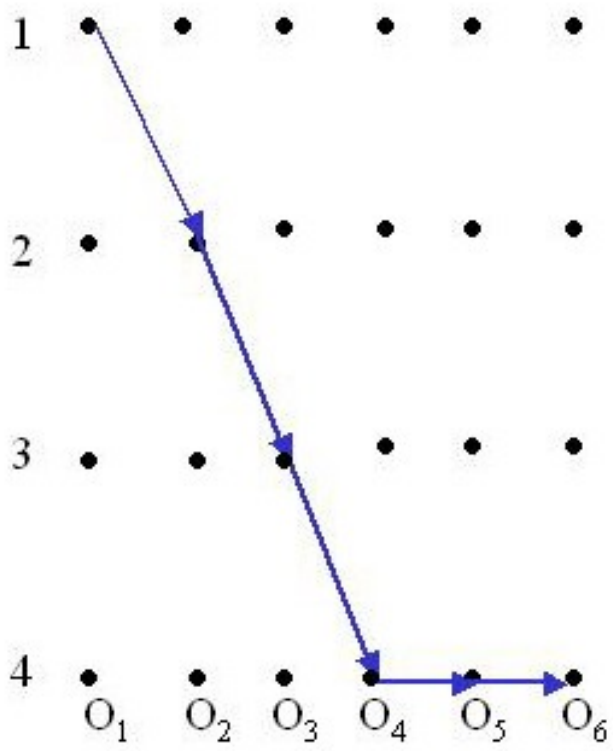


Where a_{ij} is the transition probability going from state i to state j. Note that $\sum_j a_{ij} = 1$. We can compute the probability P of an individual path just using the transition probabilities a_{ij}.

It is also common (just to be confusing!) to reorient the typical DTW picture:



$$P = a_{11}\, a_{12}\, a_{23}\, a_{33}\, a_{34}$$



$$P = a_{12}\, a_{23}\, a_{34}\, a_{44}\, a_{44}$$

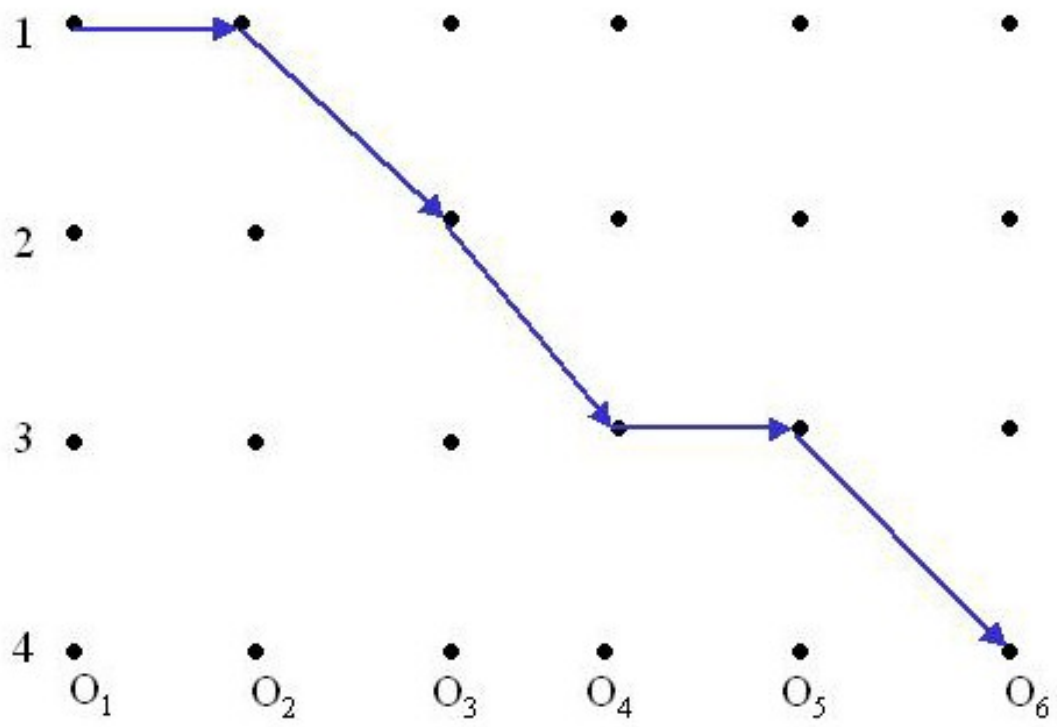
The above only describes the path probability associated with the transition. We also need to include the likelihoods associated with

the observations.

As in the GMM discussion, previously, let us define the likelihood, of producing observation O_i from state j as

$$b_j(O_i) = \sum_m c_{jm} \frac{1}{(2\pi)^{n/2}|\Sigma_{jm}|^{1/2}} e^{-\frac{1}{2}(\mathbf{O_i}-\mu_{jm})^T \mathbf{\Sigma}_{jm}^{-1}(\mathbf{O_i}-\mu_{jm})}$$

where c_{jm} are the mixture weights associated with state j. This state likelihood is also called the *output probability* associated with the state. In this case the likelihood of the entire path can be written as:



$$P = b_1(O_1)\, a_{11}\, b_1(O_2)\, a_{12}\, b_2(O_3)\, a_{23}\, b_3(O_4)\, a_{33}\, b_3(O_5)\, a_{34}\, b_4(O_6)$$

The output and transition probilities define what is called a *Hidden Markov Model* or *HMM*. Since the probabilities of moving from state to state only depend on the current and previous state, the model is Markov. Since we only see the observations and have to

infer the states after the fact, we add the term *Hidden*

One may consider an HMM to be a *generative* model of speech. One starts at the upper left corner of the trellis, and generates observations according to the permissible transitions and output probabilities.

Note also that one can not only compute the likelihood of a single path through the HMM but one can also compute the overall likelihood of producing a string of observations from the HMM as the sum of the likelihoods of the individual paths through the HMM.

HMM-The Three Main Tasks

Given the above formulation, the three main computations associated with an HMM are:

- Compute the likelihood of generating a string of observations from the HMM (the *Forward* algorithm)
- Compute the best path from the HMM (the *Viterbi* algorithm)
- Learn the parameters (output and transition probabilities) of the HMM from data (the *Baum-Welch* a.k.a. *Forward-Backward* algorithm)

COURSE FEEDBACK

- Was this lecture mostly clear or unclear? What was the muddiest topic?
- Other feedback (pace, content, atmosphere)?