TTIC 31110
Speech Technologies

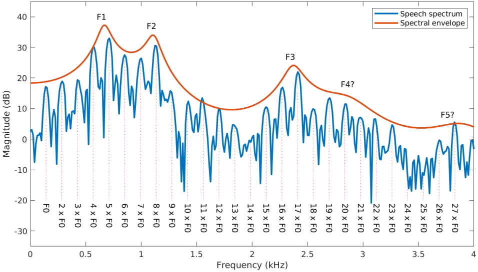
April 21, 2020

Announcements

- Karen office hours today after class
- Tutorial 2 slides and recording available on canvas

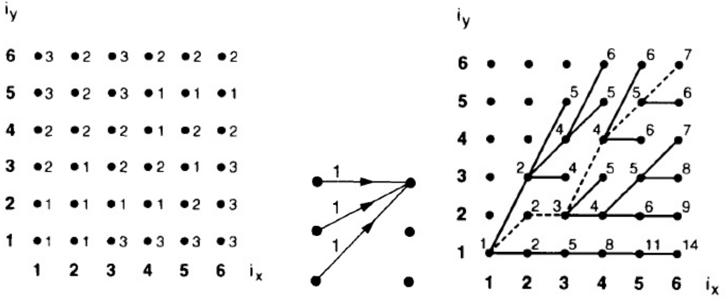
HW1 follow-up note

- In general, the class did quite well!
- Homework took roughly 4-15 hours



Outline

Recap: DTW example



Recap: DP algorithm for DTW

- Initialization: $D(1,1) = d(1,1)m(1)$
- Recursion: For $1 \leq i_x \leq T_x, \;\; 1 \leq i_y \leq T_y$

  $$D(i_x, i_y) = \min_{i'_x, i'_y} \left[D(i'_x, i'_y) + \theta\left((i'_x, i'_y), (i_x, i_y)\right) \right], \text{ where}$$

  $$\theta\left((i'_x, i'_y), (i_x, i_y)\right) = \sum_{l=0}^{L} d\left(\phi_x(T'-l), \phi_y(T'-l)\right) m(T'-l),$$

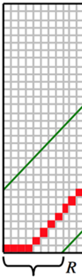
  where L is the number of steps from (i'_x, i'_y) to (i_x, i_y),
  $\phi_x(T') = i_x, \; \phi_y(T') = i_y, \; \phi_x(T'-L) = i'_x, \; \phi_y(T'-L) = i'_y$
- Termination: $d(X, Y) = \frac{D(T_x, T_y)}{M_\phi}$

Time complexity: $O(T_x T_y)$

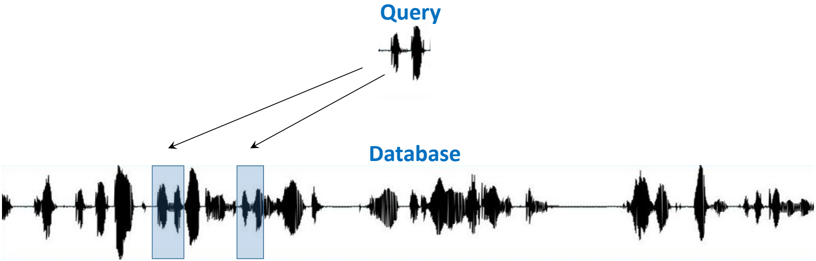
- Recently improved: Gold & Sharir, “Dynamic Time Warping and Geometric Edit Distance: Breaking the Quadratic Barrier,” *ACM Transactions on Algorithms*, 2018.
- Various *approximate* DTW algorithms exist as well

DTW: Extensions

- For ASR using DTW, use multiple templates per word: Average the distance over all templates per word, or pick the best match
- Impose global constraints on allowed paths
- Allow uncertain start/end times



Query-by-example search

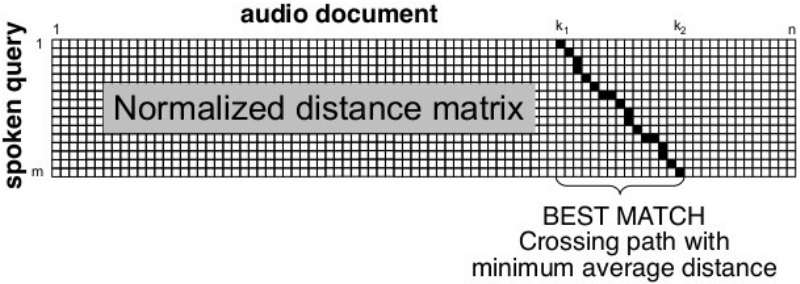


[Figure credit: Herman Kamper]

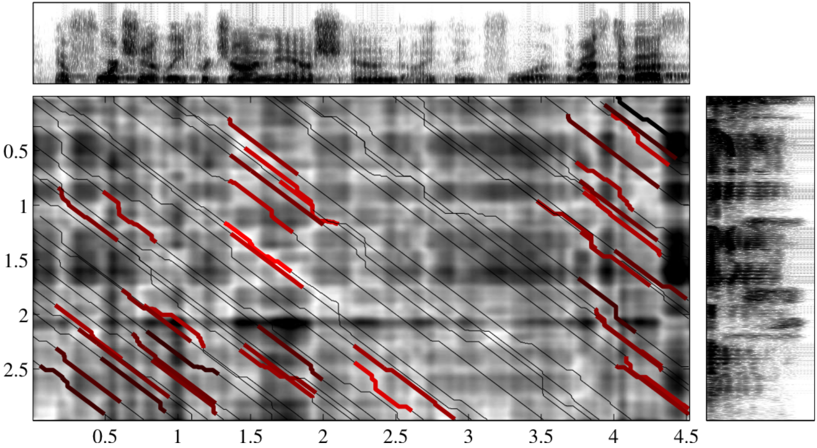
Applications:

- Open-vocabulary search
- Search in low-resource/unwritten/unknown language data
- Multilingual search

Example: Query-by-example

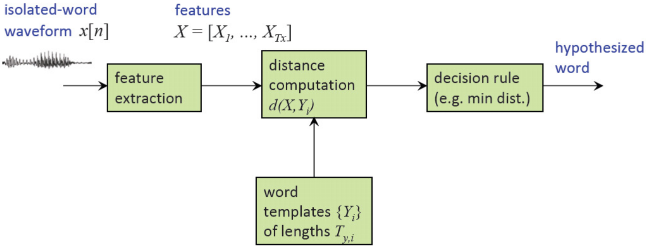


Example: Spoken term discovery



[Park & Glass 2007]

DTW: Summary



- "Efficient" algorithm for computing "distance" between two signals
- Enough for good performance in controlled conditions
- But, no principled approach for setting frame-level distances, move sets, weights $\longrightarrow$ DTW is easy to break!

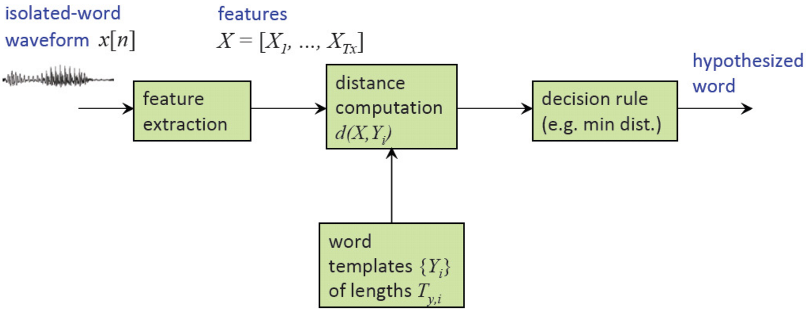
DTW: Summary

Alternatives:

- Learn some of the DTW parameters from data [Garreau+ 2014, Kamper+ 2015]
- Don't do DTW; instead, map each signal to a fixed-dimensional vector, and compute a distance between the two vectors (optionally learn the mapping parameters from data) [Levin+ 2013, Chen+ 2015, Settle+ 2017]
- Use a statistical model of the linguistic content, e.g. hidden Markov model – coming soon!

From DTW to hidden Markov models, via Gaussians

Speech recognition with DTW:



- This week: Improve this idea with Gaussians and hidden Markov models
- These will serve as “fancier” distances and move sets

From DTW to hidden Markov models, via Gaussians

Suppose:

- We use a single template per word
- We use Euclidean distance as the frame distance in DTW

This is equivalent to:

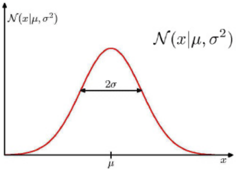
- Assuming that test frame is drawn from a particular Gaussian distribution defined by the template frame, and
- Using the density of the test frame under that Gaussian as the distance function.
- What if we use other densities? Gaussian mixture densities are a popular choice.

Notation (a la Bishop textbook)

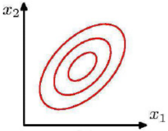
- x : scalar
- $\mathbf{x} = (x_1, \ldots, x_D)^T$: D-dimensional column vector
- $\mathbf{x}^T = (x_1, \ldots, x_D)$: row vector
- $\mathbf{X}$: matrix
- $p(x)$: density of x if it is continuous, probability mass function (PMF) if discrete
- $p(x|y)$: conditional density/PMF of x given y
- $p(x|\theta)$: density/PMF of x with parameters θ
- (Note: sometimes "distribution" interchangeable with "density")
- $p(\mathbf{X}|\theta) = f(\theta)$: likelihood function of θ

Gaussian (normal) distributions

1-D and 2-D Gaussian densities:



$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

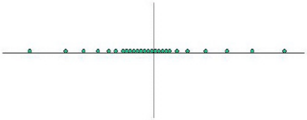


$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

(What does the covariance matrix look like in the 2-D case?)

Gaussian (normal) distributions

Random draws from a 1-D Gaussian:



Maximum-likelihood (ML) estimation of Gaussian parameters

Given data $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$, want to maximize data (log) prob
$\ln p(\mathbf{X}|\mu, \boldsymbol{\Sigma}) =$

$$-\frac{ND}{2}\ln(2\pi) - \frac{N}{2}\ln|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{n=1}^{N}(\mathbf{x}_n - \mu)^T \boldsymbol{\Sigma}^{-\mathbf{1}}(\mathbf{x}_n - \mu)$$

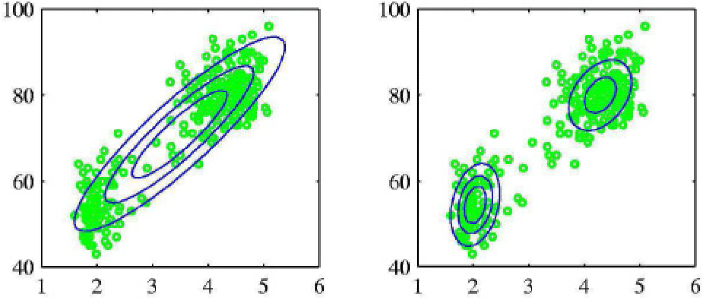
Setting the derivative to zero:

$$\frac{\partial}{\partial\mu}\ln p(\mathbf{X}|\mu, \boldsymbol{\Sigma}) = \sum_{n=1}^{N}\boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \mu) = 0$$

which gives $\mu_{ML} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$

Similarly, $\boldsymbol{\Sigma}_{ML} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \mu_{ML})(\mathbf{x}_n - \mu_{ML})^T$

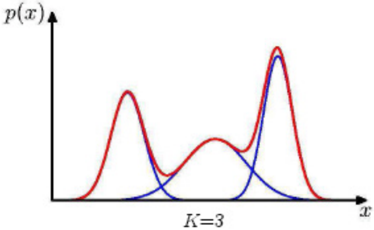
Gaussian mixtures (Gaussian mixture models, GMM)



Single Gaussian

Mixture of two Gaussians

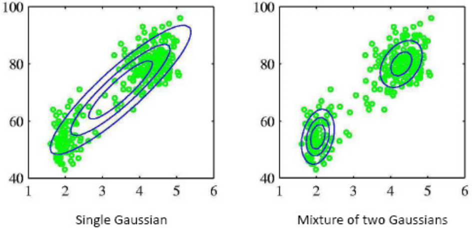
Gaussian mixtures



$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \mathbf{\Sigma}_k)$$

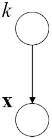
$$\pi_k \geq 0, \ \ \sum_{k=1}^{K} \pi_k = 1$$

Gaussian mixtures



Motivations:

- Approximate arbitrary distributions
- Express the existence of *latent (hidden) variables*
- Graphical model notation *a la* Bishop textbook:



$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \mathbf{\Sigma}_k)$$

Gaussian mixtures: Issues with ML estimation

$$\ln p(X|\pi,\theta) = \sum_{i=1}^{N} \ln \sum_{k=1}^{K} \pi_k N(x_i|\mu_k,\Sigma_k)$$

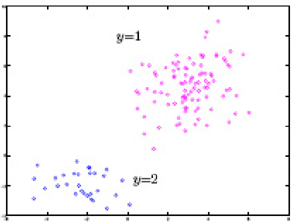
- No closed-form solution
- Infinite solution: Just set $\mu_1 = x_1$, let $\sigma_1 \to 0$
- How many Gaussians? How about one for each data point?
- Need *regularization* – but we'll get back to that later

Gaussian mixtures: ML estimation

- Introduce a set of binary indicator variables $z_{i1}, \ldots, z_{iK}$, where $z_{ik} = 1$ if x_i came from Gaussian component k and $z_{ik} = 0$ otherwise
- Count of examples from k^{th} component: $N_k = \sum_{i=1}^{N} z_{ik}$

ML estimation with known component labels

- If we know z_i, then the ML estimates of the Gaussian components are just like in the single-Gaussian case:



$$\hat{\pi}_k = \frac{N_k}{N}$$

$$\hat{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N} z_{ik} x_i$$

$$\hat{\Sigma}_k = \frac{1}{N\,k} \sum_{i=1}^{N} z_{ik} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T$$

ML estimation with unknown component labels: The credit assignment problem

- When we don't know $\mathbf{z}_i$, we face a credit assignment problem: Which component was reponsible for $\mathbf{x}_i$?
- Suppose we *do* know the component parameters $\theta = \{\mu_k, \mathbf{\Sigma}_k\}$
- Then the *posterior probability* of the indicator variables is (using Bayes' rule)

$$\gamma_{ik} = P(z_{ik} = 1 | \mathbf{x}_i, \theta) = \frac{\pi_k p(\mathbf{x}_i | \mu_k, \mathbf{\Sigma}_k)}{\sum_{l=1}^{K} \pi_l p(\mathbf{x}_i | \mu_l, \mathbf{\Sigma}_l)}$$

- γ_{ik} is called the *responsibility* of the k^{th} component for $\mathbf{x}_i$. Note that $\sum_{k=1}^{K} \gamma_{ik} = 1$

ML estimation with unknown component labels (cont'd)

Now, "pretend" that the γ_{ik} are the indicator variables (instead of z_{ik}), i.e., that $N_k = \sum_{i=1}^{N} \gamma_{ik}$:

$$
\begin{aligned}
\hat{\pi}_k &= \frac{\sum_{i=1}^{N} \gamma_{ik}}{N} \\
\hat{\mu}_k &= \frac{1}{\sum_{i=1}^{N} \gamma_{ik}} \sum_{i=1}^{N} \gamma_{ik} \mathbf{x}_i \\
\hat{\mathbf{\Sigma}}_k &= \frac{1}{\sum_{i=1}^{N} \gamma_{ik}} \sum_{i=1}^{N} \gamma_{ik} (\mathbf{x}_i - \hat{\mu}_k)(\mathbf{x}_i - \hat{\mu}_k)^T
\end{aligned}
$$

Summary so far

- If we know the *parameters* and *indicators* (component labels) then we are done
- If we know the *indicators* but not the parameters, we can do ML estimation of the parameters – and we are done
- If we have a *guess for the parameters*, we can compute the posteriors of indicators; then estimate parameters that maximize the expected likelihood – and then we are done
- In reality, we know neither the parameters nor the indicators

The expectation-maximization (EM) algorithm

- Initialization: Guess θ, π
- Iterate:

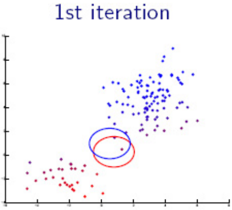
  E-step: Compute γ_{ik} using current estimates of θ, π

  M-step: Estimate new parameters, maximizing the expected likelihood, given the current γ_{ik}

- Until log likelihood converges

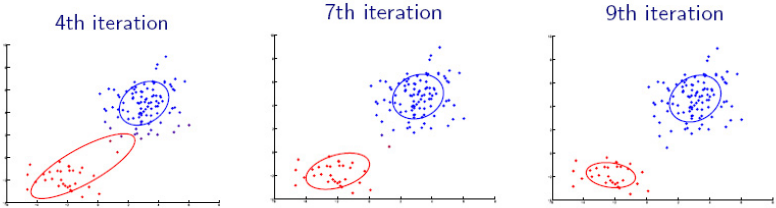
EM for Gaussian mixtures: Example

Colors represent γ_{ik} after the E-step



EM for Gaussian mixtures: Example

Colors represent γ_{ik} after the E-step



Convergence of EM

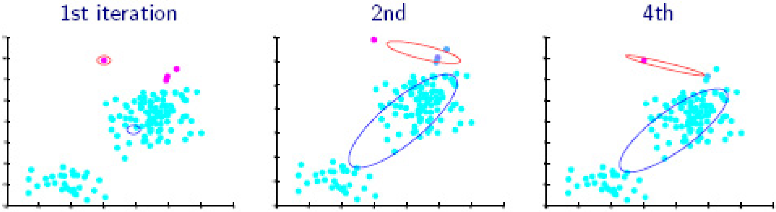
- Let $\ell^{(t)}$ be $\ln p(\mathbf{X}|\hat{\mu}, \hat{\mathbf{\Sigma}}, \hat{\pi})$ after t iterations
- Can show:

  $\ell^{(0)} \ \leq \ \ell^{(1)} \ \leq \ldots \leq \ell^{(t)} \ldots$

- EM converges, but possibly to a local maximum of the likelihood
- One solution (as in k-means): Use multiple initializations and pick the result with highest likelihood

Issues with EM for GMs

We can be very unlucky with the initialization:



Why? $\lim_{\sigma^2 \to 0} \mathcal{N}(\mathbf{x}|\mu = \mathbf{x}, \mathbf{\Sigma} = \sigma^2 \mathbf{I}) = \infty$

Regularizing EM

- Impose a prior distribution on θ
- Instead of maximizing the expected likelihood in the M-step, maximize the *posterior* probability of θ

$$\theta = \operatorname*{argmax}_{\theta} E_{z_{ik}|\mathbf{X},\pi,\theta}[\ln p(\mathbf{X},\mathbf{Z}|\pi,\theta)] + \ln p(\theta)$$

- This is the *maximum a posteriori* (MAP) estimate (as opposed to ML)
- In practice: Often just impose a minimum variance on each dimension of each component

Issues with EM for GMs: What is K?

- This is the *model selection* problem
- Idea 1: Choose K to maximize the likelihood
  - Result: A separate, tiny Gaussian for each training example
  - In the limit $\mathbf{\Sigma} \to 0$, this yields infinite likelihood
- Some solutions involve optimizing the likelihood plus a "complexity" penalty

Regularization & model selection for GMs, in practice

In practice, most often we use some held-out data to tune *hyperparameters* like K or the minimum variance

- Divide training set into *train* and *development* (or *held-out*, or *validation*) sets
- For each choice of hyperparameters, estimate parameters via EM on the training set and test on the development set
- Choose those hyperparameters that yield the best dev set performance
- Another way of choosing K:
  - Start with a single Gaussian, do EM until convergence
  - Repeat: *Split* each Gaussian into two by adding small random values to parameters, run EM, test on dev set
  - Until no performance improvement on dev set

Hidden Markov models (HMMs)

- Can serve as alternative to “templates” and “path weights” in DTW
- Ubiquitous model for speech recognition
- Today and next time:
  - Introduction to HMMs
  - Solving the 3 Problems: Scoring, decoding, training
  - Implementation issues, extensions

Hidden Markov models (HMMs): An example

- Your friend is in a distant city. You talk with him once per day and assess his mood. His mood depends on the weather at his location.
- 3 weather conditions (*states*): Fair (sunny), Cloudy, Rainy.
- 2 moods (i.e. observations): Happy, Unhappy
- Example observation sequence for 4 days: $\mathbf{O} = HHUH$
- The 3 problems:
  1. What is the probability of this sequence? (the *scoring* problem)
  2. What is your best guess of the sequence of weather states, $q_1, \ldots, q_4$? (the *decoding* problem)
  3. Given a large number of observations, how could you learn a model of the weather-mood system? (the *training* problem)

HMMs: An example (2)

Modeling the weather-mood system as a hidden Markov model:

- On the first day, the *a priori* probabilities of the three weather states F, C, R are $\{0.4, 0.3, 0.3\}$, respectively
- On any other day t, the weather state q_t depends on the previous day's weather (and nothing else), according to $P(q_{t+1}|q_t)$
  - I.e., the state sequence is a *Markov chain*
  - It is *hidden* since you don't observe the weather.
- Each day, your friend's mood o_t depends probabilistically on that day's weather (and on nothing else), according to $P(o_t|q_t)$

$P(o_t \mid q_t)$		
	$o_t = H$	$o_t = U$
$q_t = F$	0.9	0.1
$q_t = C$	0.5	0.5
$q_t = R$	0.2	0.8

$P(q_{t+1} \mid q_t)$			
	$q_{t+1} = F$	$q_{t+1} = C$	$q_{t+1} = R$
$q_t = F$	0.8	0.2	0
$q_t = C$	0.3	0.4	0.3
$q_t = R$	0	0.3	0.7

Elements of a (discrete) HMM

- N: Number of states; state at time t: $q_t \in \{1, \ldots, N\}$
- $V = \{v_1, \ldots, v_M\}$: Set of M possible observation labels (or vectors, in general); observation at time t: $o_t \in V$
- $\pi = \{\pi_i\}$: Initial state distribution,
  $\pi_i = P(q_1 = i), \;\; 1 \leq i \leq N$
- $\mathbf{A} = \{a_{ij}\}$: $N \times N$ state transition probability matrix,
  $a_{ij} = P(q_{t+1} = j | q_t = i), \;\; 1 \leq i, j, \leq N$
- $\mathbf{B} = \{b_i(k)\}$: Observation (or *emission*) distribution in state i,
  $b_i(k) = P(o_t = v_k | q_t = i), \;\; 1 \leq i \leq N, 1 \leq k \leq M$

The entire model can be denoted $\lambda = \{\mathbf{A}, \mathbf{B}, \pi\}$

Elements of the weather-mood HMM

- States: $N = 3$; let state 1 be F, state 2 be C, state 3 be R
- Obervation labels: $M = 2$; let $v_1 = H, v_2 = U$
- Model probabilities $\lambda = \{\mathbf{A}, \mathbf{B}, \pi\}$:

$$\pi = [0.4\ 0.3\ 0.3], \quad A = \begin{bmatrix} 0.8 & 0.2 & 0 \\ 0.3 & 0.4 & 0.3 \\ 0 & 0.3 & 0.7 \end{bmatrix}, \quad B = \begin{bmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \\ 0.2 & 0.8 \end{bmatrix}$$

- $\mathbf{A}$ can also be represented via a *state transition diagram*: