Lecture 3

Gaussian Mixture Models and Introduction to HMM's

Michael Picheny, Bhuvana Ramabhadran, Stanley F. Chen

IBM T.J. Watson Research Center
Yorktown Heights, New York, USA
{picheny,bhuvana,stanchen}@us.ibm.com

24 September 2012

Administrivia

- Feedback (2+ votes):
    - Too fast (pace/content/talking): many.
    - More details/explanation of formulae: 5.
    - More examples + explanation: 4.
    - Talk more about labs: 2.
    - Earlier break or more breaks: 2
    - (Provide extra readings for people w/o DSP.)
- Will try to address most of these today.
- Muddiest topic: DTW (4), LPC (3), DSP (2), deltas (1).
- Lab 1 due Wednesday, October 3rd at 6pm.
    - Should have received username and password.
    - Courseworks discussion has been started.

Where Are We?

- Can extract features over time (LPC, MFCC, PLP) that ...
  - Characterize info in speech signal in compact form.
  - Every $\sim$10 ms, process window of samples ...
  - To get $\sim$40 features.
- DTW computes distance between feature vectors ...
  - While accounting for nonlinear time alignment.
- Learned basic concepts (*e.g.*, distances, shortest paths) ...
  - That will reappear throughout course.

DTW Revisited

$$w^* = \underset{w \in \text{vocab}}{\arg\min}\, \text{distance}(A'_{\text{test}}, A'_w)$$

- Training: collect audio A_w for each word w in vocab.
  - Generate features $\Rightarrow A'_w$ (*template* for w).
- Test time: given audio A_{test}, convert to A'_{test}.
  - For each w, compute distance(A'_{test}, A'_w) using DTW.
  - Return w with smallest distance.

What are Pros and Cons of DTW?

Pros

- Easy to implement.
- Lots of freedom — can model arbitrary time warpings.

Cons: It's *Ad Hoc*

- Distance measures completely heuristic.
    - Why Euclidean?
    - Weight all dimensions of feature vector equally?
- Warping paths heuristic.
    - Too much freedom not good for robustness?
    - Allowable local paths hand-derived.
- No guarantees of optimality or convergence.

Cons (cont'd)

- Doesn't scale well?
  - Run DTW for each template in training data.
  - What if large vocabulary? Lots of templates per word?
- Generalization.
  - Doesn't support mix and match between templates.

Can We Do Better?

- Key insight 1: Learn as much as possible from data.
  - *e.g.*, distance measure; graph weights; graph structure?
- Key insight 2: Use probabilistic modeling.
  - Use well-described theories and models from . . .
  - Probability, statistics, and computer science . . .
  - Rather than arbitrary heuristics with ill-defined properties.

Next Two Main Topics

- *Gaussian Mixture models* (today) — A probabilistic model of . . .
  - Feature vectors associated with a speech sound.
  - Principled distance between test frame . . .
  - And set of template frames.
- *Hidden Markov models* (next week) — A probabilistic model of . . .
  - Time evolution of feature vectors for a speech sound.
  - Principled generalization of DTW.

Part I

Gaussian Distributions

The Scenario

- Given alignment between training feats X, test feats Y.
  - Warping functions $\tau_x(t)$, $\tau_y(t)$, $t = 1, \ldots, T$.
  - *i.e.*, time $\tau_x(t)$ in X aligns with time $\tau_y(t)$ in Y.
- Total distance is sum of distance between aligned vectors.

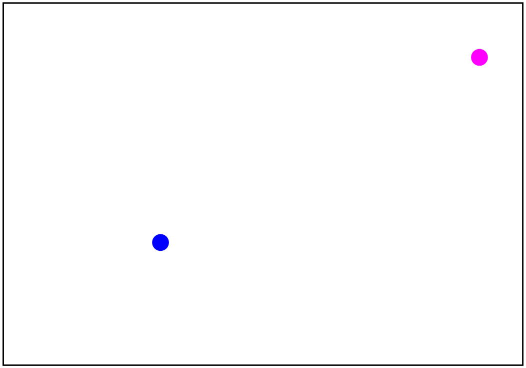
$$\text{distance}_{\tau_x,\tau_y}(X, Y) = \sum_{t=1}^{T} \text{framedist}(x_{\tau_x(t)}, y_{\tau_y(t)})$$

The Scenario

- Computing frame distance for pair of frames is easy.

$$\text{framedist}(x_{\tau_x(t)}, y_{\tau_y(t)})$$

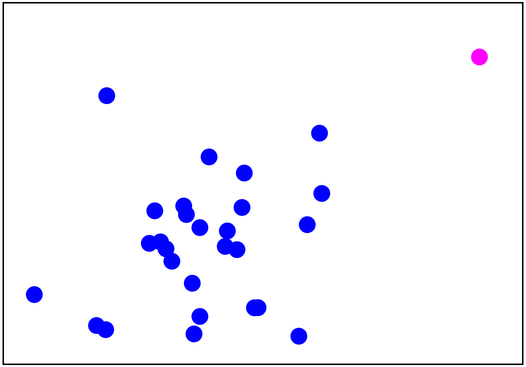
- Imagine 2d feature vectors instead of 40d for visualization.



Problem Formulation

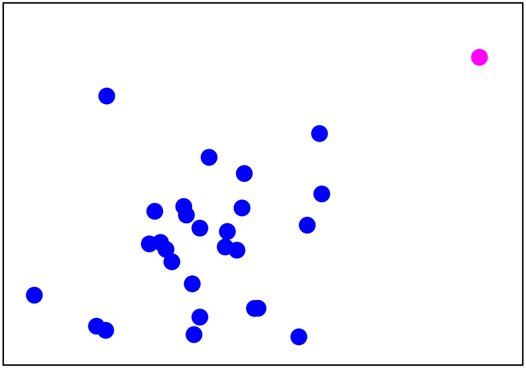
- What if instead of one training sample, have many?

$$\text{framedist}((x^1_{\tau_{x^1}(t)}, x^2_{\tau_{x^2}(t)}, x^3_{\tau_{x^3}(t)}, \ldots); y_{\tau_y(t)})$$



Ideas

- Average training samples; compute Euclidean distance.
- Find best match over all training samples.
- Make *probabilistic model* of training samples.



Probabilistic Modeling

- Old paradigm:

$$w^* = \operatorname*{arg\,min}_{w \in \text{vocab}} \text{distance}(A'_{\text{test}}, A'_w)$$

- New paradigm:

$$w^* = \operatorname*{arg\,min}_{w \in \text{vocab}} -\log P(A'_{\text{test}} | w)$$

  - $P(A'|w)$ is (relative) frequency with which w ...
  - Is realized as feature vector A'.

Why Probabilistic Modeling?

- If can estimate $P(A'|w)$ perfectly ...
  - Can perform classification optimally!
- *e.g.*, two-class classification (w/ classes equally frequent)
  - Choose *yes* iff $P(A'_{\text{test}}|yes) > P(A'_{\text{test}}|no)$.
  - This is best you can do!

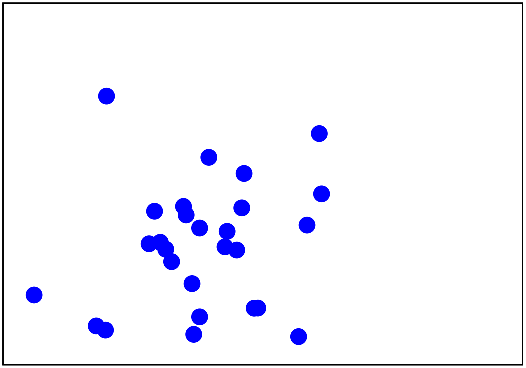
Why Probabilistic Modeling?

- In limit of infinite data, . . .
  - Can estimate probabilities perfectly (*consistency*).
- In real world situations (*e.g.*, sparse data) . . .
  - No guarantees.
  - Still, better to follow principles imperfectly . . .
  - Than to not have principles at all.

Where Are We?

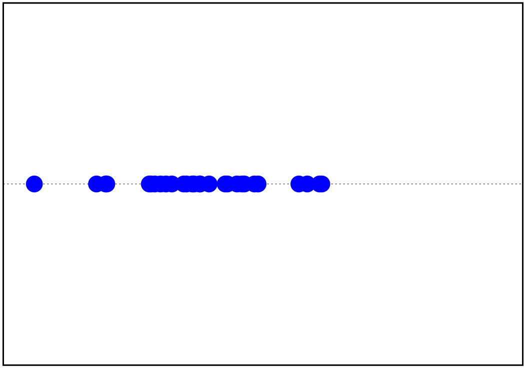
Problem Formulation, Two Dimensions

- Estimate $P(x_1, x_2)$, the "frequency" ...
  - That training sample occurs at location (x_1, x_2).



Let's Start With One Dimension

- Estimate $P(x)$, the "frequency" ...
  - That training sample occurs at location x.



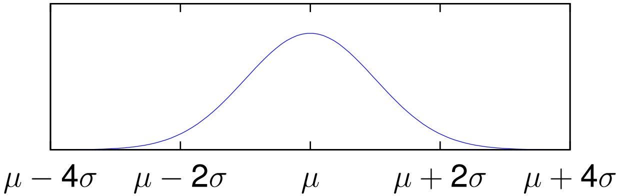
The Gaussian or Normal Distribution

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

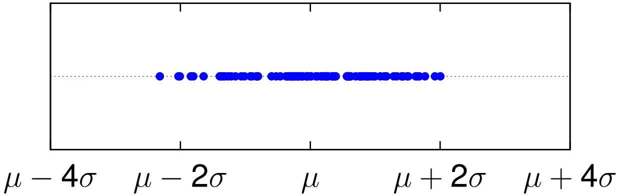
- Parametric distribution with two parameters:
  - μ = mean (the center of the data).
  - σ^2 = variance (how wide data is spread).

Visualization

- Density function:



- Sample from distribution:



Properties of Gaussian Distributions

- Is valid distribution.

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

- Central Limit Theorem: Sums of large numbers of identically distributed random variables tend to Gaussian.
  - Lots of different types of data look “bell-shaped”.
- Sums and differences of Gaussian random variables …
  - Are Gaussian.
- If X is distributed as $\mathcal{N}(\mu, \sigma^2)$ …
  - $aX + b$ is distributed as $\mathcal{N}(a\mu + b, (a\sigma)^2)$.
- Negative log looks like weighted Euclidean distance!

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

Where Are We?

Gaussians in Two Dimensions

$$\mathcal{N}(\mu_1, \mu_2, \sigma_1^2, \sigma_2^2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-r^2}}\, e^{-\frac{1}{2(1-r^2)}\left(\frac{(x_1-\mu_1)^2}{\sigma_1^2} - \frac{2rx_1x_2}{\sigma_1\sigma_2} + \frac{(x_2-\mu_2)^2}{\sigma_2^2}\right)}$$

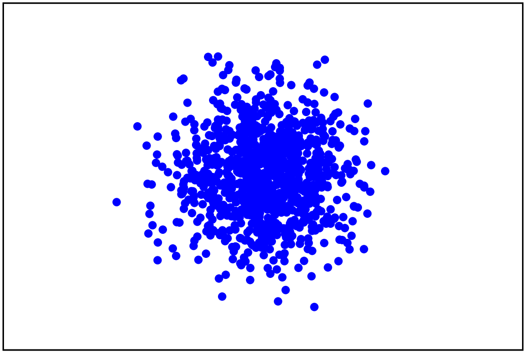
- If $r = 0$, simplifies to

$$\frac{1}{\sqrt{2\pi}\sigma_1}\, e^{-\frac{(x_1-\mu_1)^2}{2\sigma_1^2}} \frac{1}{\sqrt{2\pi}\sigma_2}\, e^{-\frac{(x_2-\mu_2)^2}{2\sigma_2^2}} = \mathcal{N}(\mu_1, \sigma_1^2)\mathcal{N}(\mu_2, \sigma_2^2)$$

  - *i.e.*, like generating each dimension independently.

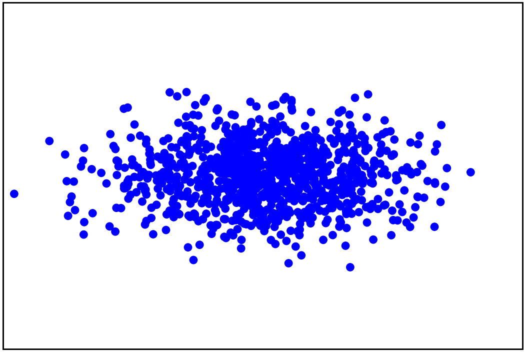
Example: $r = 0$, $\sigma_1 = \sigma_2$

- x_1, x_2 uncorrelated.
    - Knowing x_1 tells you nothing about x_2.



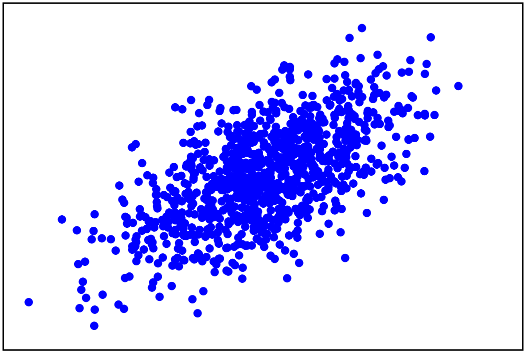
Example: $r = 0$, $\sigma_1 \neq \sigma_2$

- x_1, x_2 can be uncorrelated and have unequal variance.



Example: $r > 0$, $\sigma_1 \neq \sigma_2$

- x_1, x_2 correlated.
    - Knowing x_1 tells you something about x_2.



Generalizing to More Dimensions

- If we write following matrix:

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

  then another way to write two-dimensional Gaussian is:

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

  where $\mathbf{x} = (x_1, x_2)$, $\boldsymbol{\mu} = (\mu_1, \mu_2)$.
- More generally, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ can have arbitrary numbers of components.
  - *Multivariate* Gaussians.

Diagonal and Full Covariance Gaussians

- Let's say have 40d feature vector.
  - How many parameters in covariance matrix $\mathbf{\Sigma}$?
  - The more parameters, ...
  - The more data you need to estimate them.
- In ASR, usually assume $\mathbf{\Sigma}$ is diagonal $\Rightarrow$ d params.
  - This is why like having uncorrelated features!
- (Research direction: is there something in between?)

Computing Gaussian Log Likelihoods

- Why *log* likelihoods?
- Full covariance:

$$\log P(\mathbf{x}) = -\frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln|\Sigma| - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})$$

- Diagonal covariance:

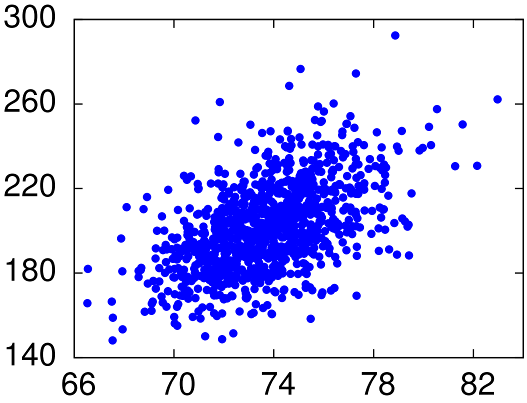
$$\log P(\mathbf{x}) = -\frac{d}{2}\ln(2\pi) - \sum_{i=1}^{d}\ln\sigma_i - \frac{1}{2}\sum_{i=1}^{d}(x_i-\mu_i)^2/\sigma_i^2$$

  - Again, note similarity to weighted Euclidean distance.
  - Terms on left independent of **x**; precompute.
  - A few multiplies/adds per dimension.

Where Are We?

Estimating Gaussians

- Give training data, how to choose parameters μ, $\mathbf{\Sigma}$?
- Find parameters so that resulting distribution ...
  - “Matches” data as well as possible.
- Sample data: height, weight of baseball players.



Maximum-Likelihood Estimation (Univariate)

- One criterion: data "matches" distribution well . . .
  - If distribution assigns high likelihood to data.
- Likelihood of string of observations $x_1, x_2, \ldots, x_N$ is . . .
  - Product of individual likelihoods.

$$\mathcal{L}(x_1^N|\mu, \sigma) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \, e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

- Maximum likelihood estimation: choose μ, σ . . .
  - That maximizes likelihood of training data.

$$(\mu, \sigma)_{\text{MLE}} = \arg\max_{\mu,\sigma} \mathcal{L}(x_1^N|\mu, \sigma)$$

Why Maximum-Likelihood Estimation?

- Assume we have “correct” model form.
- Then, in presence of infinite training samples . . .
    - ML estimates approach “true” parameter values.
    - For most models, MLE is asymptotically consistent, unbiased, and efficient.
- ML estimation is easy for many types of models.
    - Count and normalize!

What is ML Estimate for Gaussians?

- Much easier to work with log likelihood $L = \ln \mathcal{L}$:

$$L(x_1^N|\mu,\sigma) = -\frac{N}{2}\ln 2\pi\sigma^2 - \frac{1}{2}\sum_{i=1}^{N}\frac{(x_i-\mu)^2}{\sigma^2}$$

- Take partial derivatives w.r.t. μ, σ:

$$\begin{aligned}\frac{\partial L(x_1^N|\mu,\sigma)}{\partial\mu} &= \sum_{i=1}^{N}\frac{(x_i-\mu)}{\sigma^2}\\ \frac{\partial L(x_1^N|\mu,\sigma)}{\partial\sigma^2} &= -\frac{N}{2\sigma^2} + \sum_{i=1}^{N}\frac{(x_i-\mu)^2}{\sigma^4}\end{aligned}$$

- Set equal to zero; solve for μ, σ^2.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2$$

What is ML Estimate for Gaussians?

- Multivariate case.

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

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

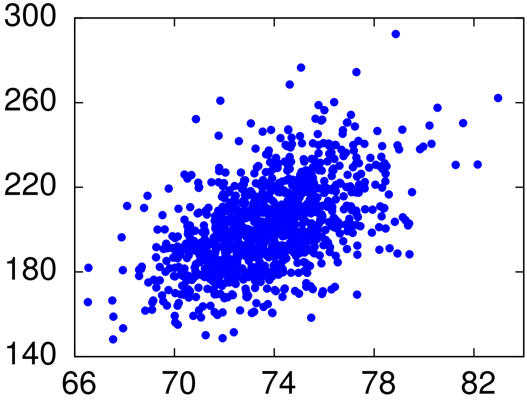
- What if diagonal covariance?
  - Estimate params for each dimension independently.

Example: ML Estimation

- Heights (in.) and weights (lb.) of 1033 pro baseball players.
    - Noise added to hide discretization effects.
- `~stanchen/e6870/data/mlb_data.dat`

height	weight
74.34	181.29
73.92	213.79
72.01	209.52
72.28	209.02
72.98	188.42
69.41	176.02
68.78	210.28
...	...
...	...

Example: ML Estimation



Example: Diagonal Covariance

$$\mu_1 = \frac{1}{1033}(74.34 + 73.92 + 72.01 + \cdots) = 73.71$$

$$\mu_2 = \frac{1}{1033}(181.29 + 213.79 + 209.52 + \cdots) = 201.69$$

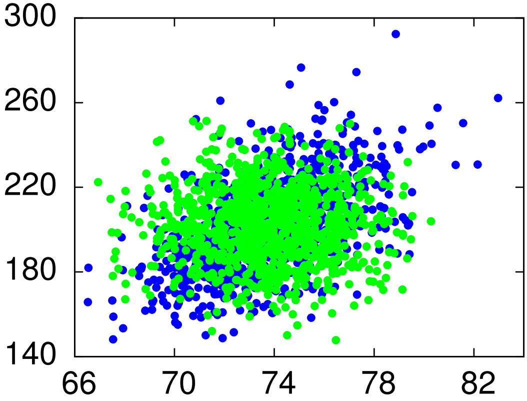
$$\sigma_1^2 = \frac{1}{1033}\left[(74.34 - 73.71)^2 + (73.92 - 73.71)^2 + \cdots)\right]$$

$$= 5.43$$

$$\sigma_2^2 = \frac{1}{1033}\left[(181.29 - 201.69)^2 + (213.79 - 201.69)^2 + \cdots)\right]$$

$$= 440.62$$

Example: Diagonal Covariance



Example: Full Covariance

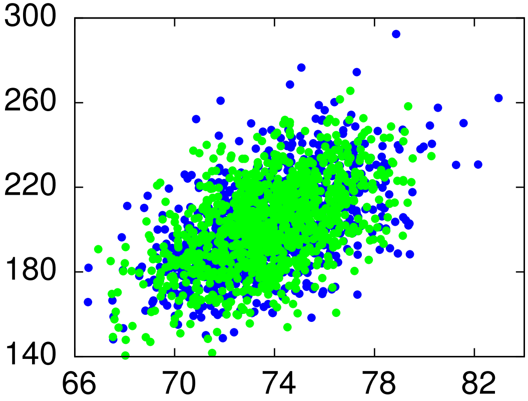
- Mean; diagonal elements of covariance matrix the same.

$$\begin{aligned}
\boldsymbol{\Sigma}_{12} = \boldsymbol{\Sigma}_{21} \\
&= \frac{1}{1033}[(74.34 - 73.71) \times (181.29 - 201.69) + \\
&\quad (73.92 - 73.71) \times (213.79 - 201.69) + \cdots)] \\
&= 25.43
\end{aligned}$$

$$\boldsymbol{\mu} = \begin{bmatrix} 73.71 & 201.69 \end{bmatrix}$$

$$\boldsymbol{\Sigma} = \begin{vmatrix} 5.43 & 25.43 \\ 25.43 & 440.62 \end{vmatrix}$$

Example: Full Covariance



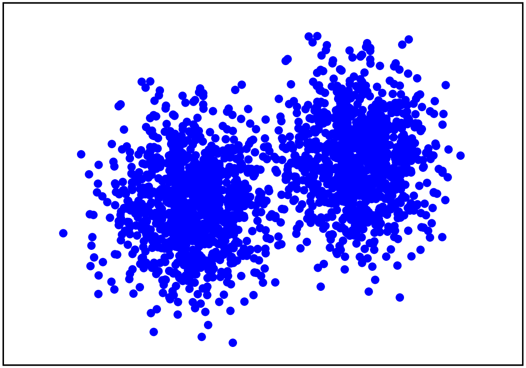
Recap: Gaussians

- Lots of data “looks” Gaussian.
    - Central limit theorem.
- ML estimation of Gaussians is easy.
    - Count and normalize.
- In ASR, mostly use diagonal covariance Gaussians.
    - Full covariance matrices have too many parameters.

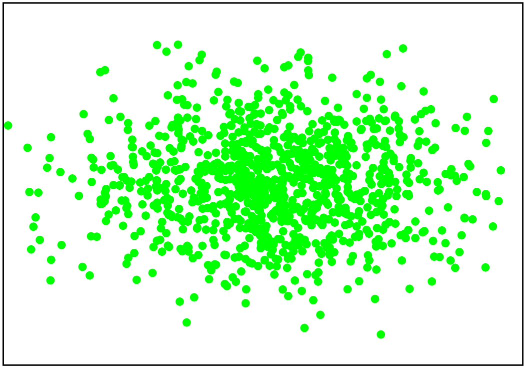
Part II

Gaussian Mixture Models

Problems with Gaussian Assumption

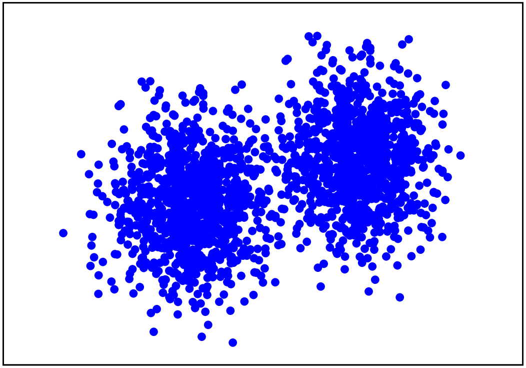


Problems with Gaussian Assumption



- Sample from MLE Gaussian trained on data on last slide.
- Not all data is Gaussian!

Problems with Gaussian Assumption



- What can we do? What about *two* Gaussians?

$$P(\mathbf{x}) = p_1 \times \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) + p_2 \times \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$$

where $p_1 + p_2 = 1$.

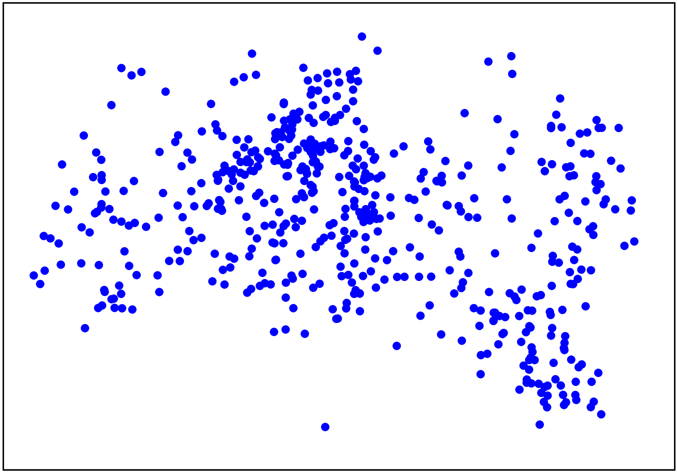
Gaussian Mixture Models (GMM's)

- More generally, can use arbitrary number of Gaussians:

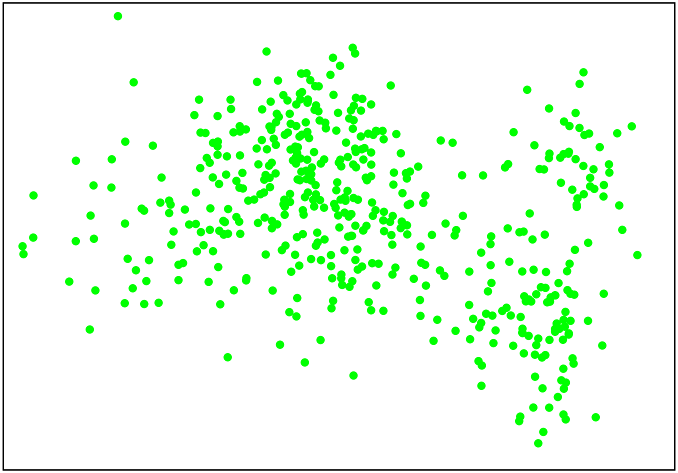
$$P(\mathbf{x}) = \sum_j p_j \frac{1}{(2\pi)^{d/2}|\Sigma_j|^{1/2}} \, e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)}$$

  where $\sum_j p_j = 1$ and all $p_j \geq 0$.
- Also called *mixture* of Gaussians.
- Can approximate any distribution of interest pretty well ...
  - If just use enough component Gaussians.

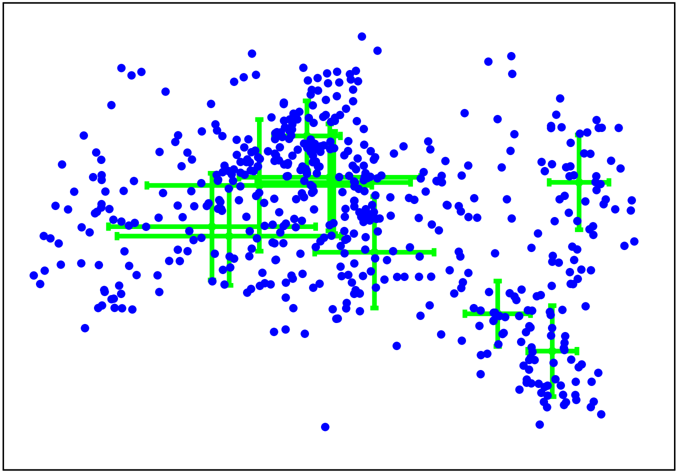
Example: Some Real Acoustic Data



Example: 10-component GMM (Sample)



Example: 10-component GMM (μ's, σ's)



ML Estimation For GMM's

- Given training data, how to estimate parameters ...
  - *i.e.*, the $\boldsymbol{\mu}_j$, $\boldsymbol{\Sigma}_j$, and mixture weights p_j ...
  - To maximize likelihood of data?
- No closed-form solution.
  - Can't just count and normalize.
- Instead, must use an optimization technique ...
  - To find good local optimum in likelihood.
  - Gradient search
  - Newton's method
- Tool of choice: The Expectation-Maximization Algorithm.

Where Are We?

Wake Up!

- This is another key thing to remember from course.
- Used to train GMM's, HMM's, and lots of other things.
- Key paper in 1977 by Dempster, Laird, and Rubin [2].

What Does The EM Algorithm Do?

- Finds ML parameter estimates for models . . .
    - With *hidden* variables.
- Iterative hill-climbing method.
    - Adjusts parameter estimates in each iteration . . .
    - Such that likelihood of data . . .
    - Increases (weakly) with each iteration.
- Actually, finds local optimum for parameters in likelihood.

What is a Hidden Variable?

- A random variable that isn't observed.
- Example: in GMMs, output prob depends on ...
  - The mixture component that generated the observation
  - But you can't observe it
- So, to compute prob of observed x, need to sum over ...
  - All possible values of hidden variable h:

$$P(x) = \sum_h P(h, x) = \sum_h P(h) P(x|h)$$

Mixtures and Hidden Variables

- Consider probability that is mixture of probs, *e.g.*, a GMM:

$$P(x) = \sum_j p_j \, \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

- Can be viewed as hidden model.
  - $h \Leftrightarrow$ Which component generated sample.
  - $P(h) = p_j$; $P(x|h) = \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$.

$$P(x) = \sum_h P(h)P(x|h)$$

The Basic Idea

- If nail down "hidden" value for each x_i, ...
  - Model is no longer hidden!
  - *e.g.*, data partitioned among GMM components.
- So for each data point x_i, assign single hidden value h_i.
  - Take $h_i = \arg\max_h P(h)P(x_i|h)$.
  - *e.g.*, identify GMM component generating each point.
- Easy to train parameters in non-hidden models.
  - Update parameters in $P(h)$, $P(x|h)$.
  - *e.g.*, count and normalize to get MLE for $\boldsymbol{\mu}_j$, $\boldsymbol{\Sigma}_j$, p_j.
- Repeat!

The Basic Idea

- Hard decision:
  - For each x_i, assign single $h_i = \arg\max_h P(h, x_i) \ldots$
  - With count 1.
- Soft decision:
  - For each x_i, compute for every $h \ldots$
  - the Posterior prob $\tilde{P}(h|x_i) = \frac{P(h,x_i)}{\sum_h P(h,x_i)}$.
  - Also called the “fractional count”
  - *e.g.*, partition event across *every* GMM component.
- Rest of algorithm unchanged.

The Basic Idea

- Initialize parameter values somehow.
- For each iteration . . .
- Expectation step: compute posterior (count) of h for each x_i.

$$\tilde{P}(h|x_i) = \frac{P(h, x_i)}{\sum_h P(h, x_i)}$$

- Maximization step: update parameters.
    - Instead of data x_i with hidden h, pretend . . .
    - *Non-hidden* data where . . .
    - (Fractional) count of each (h, x_i) is $\tilde{P}(h|x_i)$.

Example: Training a 2-component GMM

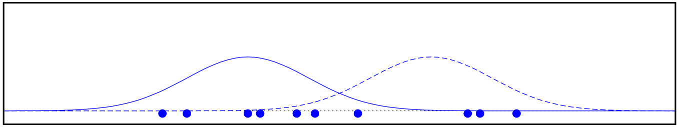
- Two-component univariate GMM; 10 data points.
- The data: $x_1, \ldots, x_{10}$

$$8.4, 7.6, 4.2, 2.6, 5.1, 4.0, 7.8, 3.0, 4.8, 5.8$$

- Initial parameter values:

p_1	μ_1	σ_1^2	p_2	μ_2	σ_2^2
0.5	4	1	0.5	7	1

- Training data; densities of initial Gaussians.



The E Step

x_i	$p_1 \cdot \mathcal{N}_1$	$p_2 \cdot \mathcal{N}_2$	$P(x_i)$	$\tilde{P}(1\vert x_i)$	$\tilde{P}(2\vert x_i)$
8.4	0.0000	0.0749	0.0749	0.000	1.000
7.6	0.0003	0.1666	0.1669	0.002	0.998
4.2	0.1955	0.0040	0.1995	0.980	0.020
2.6	0.0749	0.0000	0.0749	1.000	0.000
5.1	0.1089	0.0328	0.1417	0.769	0.231
4.0	0.1995	0.0022	0.2017	0.989	0.011
7.8	0.0001	0.1448	0.1450	0.001	0.999
3.0	0.1210	0.0001	0.1211	0.999	0.001
4.8	0.1448	0.0177	0.1626	0.891	0.109
5.8	0.0395	0.0971	0.1366	0.289	0.711

$$\tilde{P}(h|x_i) = \frac{P(h, x_i)}{\sum_h P(h, x_i)} = \frac{p_h \cdot \mathcal{N}_h}{P(x_i)} \qquad h \in \{1, 2\}$$

The M Step

- View: have *non-hidden* corpus for each component GMM.
  - For hth component, have $\tilde{P}(h|x_i)$ counts for event x_i.
- Estimating μ: fractional events.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \quad \Rightarrow \quad \mu_h = \frac{1}{\sum_i \tilde{P}(h|x_i)} \sum_{i=1}^{N} \tilde{P}(h|x_i) x_i$$

$$\mu_1 = \frac{1}{0.000 + 0.002 + 0.980 + \cdots} \times (0.000 \times 8.4 + 0.002 \times 7.6 + 0.980 \times 4.2 + \cdots)$$

$$= 3.98$$

- Similarly, can estimate σ_h^2 with fractional events.

The M Step (cont'd)

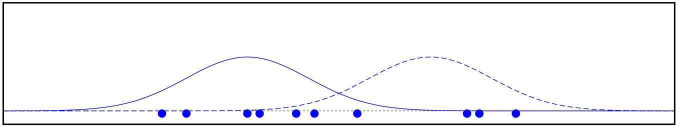
- What about the mixture weights p_h?
  - To find MLE, count and normalize!

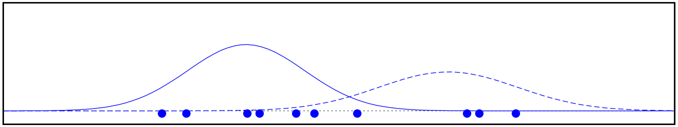
$$p_1 = \frac{0.000 + 0.002 + 0.980 + \cdots}{10} = 0.59$$

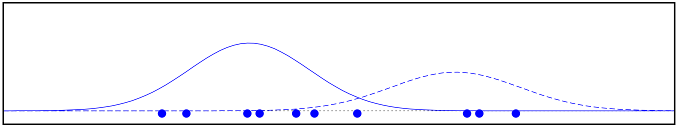
The End Result

iter	p_1	μ_1	σ_1^2	p_2	μ_2	σ_2^2
0	0.50	4.00	1.00	0.50	7.00	1.00
1	0.59	3.98	0.92	0.41	7.29	1.29
2	0.62	4.03	0.97	0.38	7.41	1.12
3	0.64	4.08	1.00	0.36	7.54	0.88
10	0.70	4.22	1.13	0.30	7.93	0.12

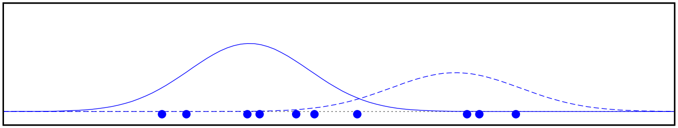
First Few Iterations of EM

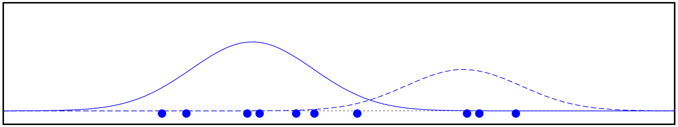

iter 0

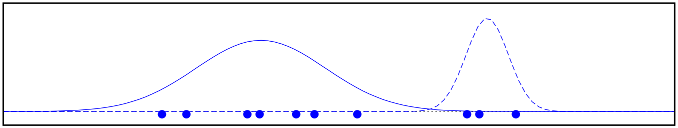

iter 1


iter 2

Later Iterations of EM


iter 2


iter 3


iter 10

Why the EM Algorithm Works [3]

- $\mathbf{x} = (x_1, x_2, \ldots)$ = whole training set; $\mathbf{h}$ = hidden.
- $\boldsymbol{\theta}$ = parameters of model.
- Objective function for MLE: (log) likelihood.

$$L(\boldsymbol{\theta}) = \log P(\mathbf{x}|\boldsymbol{\theta}) = \log \sum_{\mathbf{h}} P(\mathbf{h}, \mathbf{x}|\boldsymbol{\theta})$$

- Alternate objective function
  - Will show maximizing this equivalent to above

$$F(\tilde{P}, \boldsymbol{\theta}) = L(\boldsymbol{\theta}) - D(\tilde{P} \parallel P_{\boldsymbol{\theta}})$$

  - $P_{\boldsymbol{\theta}}(\mathbf{h}|\mathbf{x})$ = posterior over hidden.
  - $\tilde{P}(\mathbf{h})$ = distribution over hidden to be optimized ...
  - $D(\cdot \parallel \cdot)$ = Kullback-Leibler divergence.

Why the EM Algorithm Works

$$F(\tilde{P}, \boldsymbol{\theta}) = L(\boldsymbol{\theta}) - D(\tilde{P} \parallel P_{\boldsymbol{\theta}})$$

- Outline of proof:
  - Show that both E step and M step improve $F(\tilde{P}, \boldsymbol{\theta})$.
  - Will follow that likelihood $L(\boldsymbol{\theta})$ improves as well.

The E Step

$$F(\tilde{P}, \boldsymbol{\theta}) = L(\boldsymbol{\theta}) - D(\tilde{P} \parallel P_{\boldsymbol{\theta}})$$

- Properties of KL divergence.
  - Nonnegative; and zero iff $\tilde{P} = P_{\boldsymbol{\theta}}$.
- What is best choice for $\tilde{P}(\mathbf{h})$?
  - Compute the current posterior $P_{\boldsymbol{\theta}}(\mathbf{h}|\mathbf{x})$.
  - Set $\tilde{P}(\mathbf{h})$ equal to this posterior.
- Since $L(\boldsymbol{\theta})$ is not function of $\tilde{P}$ . . .
  - $F(\tilde{P}, \boldsymbol{\theta})$ can only improve in E step.

The M Step

- Lemma:

$$F(\tilde{P}, \boldsymbol{\theta}) = E_{\tilde{P}}[\log P(\mathbf{h}, \mathbf{x}|\boldsymbol{\theta})] + H(\tilde{P})$$

- Proof:

$$\begin{aligned}
F(\tilde{P}, \boldsymbol{\theta}) &= L(\boldsymbol{\theta}) - D(\tilde{P} \parallel P_{\boldsymbol{\theta}}) \\
&= \log P(\mathbf{x}|\boldsymbol{\theta}) - \sum_{\mathbf{h}} \tilde{P}(\mathbf{h}) \log \frac{\tilde{P}(\mathbf{h})}{P(\mathbf{h}|\mathbf{x}, \boldsymbol{\theta})} \\
&= \log P(\mathbf{x}|\boldsymbol{\theta}) - \sum_{\mathbf{h}} \tilde{P}(\mathbf{h}) \log \frac{\tilde{P}(\mathbf{h}) P(\mathbf{x}|\boldsymbol{\theta})}{P(\mathbf{h}, \mathbf{x}|\boldsymbol{\theta})} \\
&= \sum_{\mathbf{h}} \tilde{P}(\mathbf{h}) \log P(\mathbf{h}, \mathbf{x}|\boldsymbol{\theta}) - \sum_{\mathbf{h}} \tilde{P}(\mathbf{h}) \log \tilde{P}(\mathbf{h}) \\
&= E_{\tilde{P}}[\log P(\mathbf{h}, \mathbf{x}|\boldsymbol{\theta})] + H(\tilde{P})
\end{aligned}$$

The M Step (cont'd)

$$F(\tilde{P}, \boldsymbol{\theta}) = E_{\tilde{P}}[\log P(\mathbf{h}, \mathbf{x}|\boldsymbol{\theta})] + H(\tilde{P})$$

- $E_{\tilde{P}}[\cdots]$ = log likelihood of non-hidden corpus ...
  - Where each h gets $\tilde{P}(\mathbf{h})$ counts.
- $H(\tilde{P})$ = *entropy* of distribution $\tilde{P}(\mathbf{h})$.

- What do we do in M step?
  - Pick $\boldsymbol{\theta}$ to maximize term on left
  - Note this is just MLE of non-hidden corpus ...
  - Since we chose an estimate for h from the E step.
- Since $H(\tilde{P})$ is not function of $\boldsymbol{\theta}$ ...
  - $F(\tilde{P}, \boldsymbol{\theta})$ can only improve in M step.

Why the EM Algorithm Works

- Observation: $F(\tilde{P}, \boldsymbol{\theta}) = L(\boldsymbol{\theta})$ after E step (set $\tilde{P} = P_{\boldsymbol{\theta}}$).

$$F(\tilde{P}, \boldsymbol{\theta}) = L(\boldsymbol{\theta}) - D(\tilde{P} \parallel P_{\boldsymbol{\theta}})$$

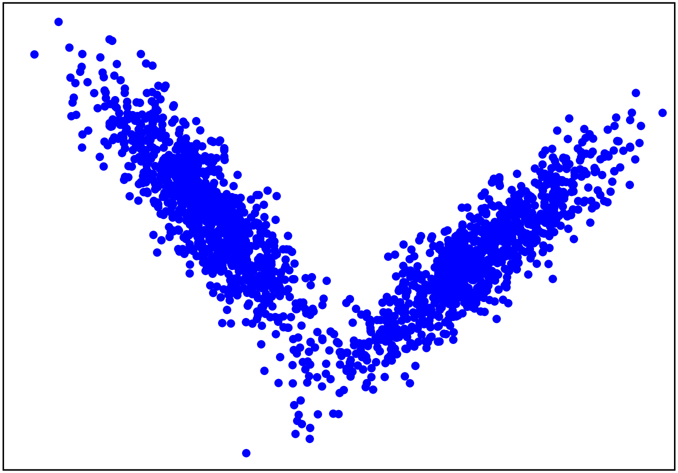
- If $F(\tilde{P}, \boldsymbol{\theta})$ improves with each iteration ...
  - And $F(\tilde{P}, \boldsymbol{\theta}) = L(\boldsymbol{\theta})$ after each E step ...
  - $L(\boldsymbol{\theta})$ improves after each iteration.
- There you go!

Discussion

- EM algorithm is elegant and general way to ...
  - Train parameters in hidden models ...
  - To optimize likelihood.
- Only finds local optimum.
  - *Seeding* is of paramount importance.
- Generalized EM algorithm.
  - $F(\tilde{P}, \theta)$ just needs to improve *some* in each step.
  - *i.e.*, $\tilde{P}(h)$ in E step need not be exact posterior.
  - *i.e.*, θ in M step need not be ML estimate.
  - *e.g.*, can optimize *Viterbi* likelihood.

Where Are We?

Another Example Data Set



Question: How Many Gaussians?

- Method 1 (most common): Guess!
- Method 2: Bayesian Information Criterion (BIC)[1].
    - Penalize likelihood by number of parameters.

$$\mathrm{BIC}(C_k) = \sum_{j=1}^{k} \{-\frac{1}{2} n_j \log |\Sigma_j|\} - Nk(d + \frac{1}{2} d(d+1))$$

- k = Gaussian components.
- d = dimension of feature vector.
- n_j = data points for Gaussian j; N = total data points.

The Bayesian Information Criterion

- View GMM as way of coding data for transmission.
  - Cost of transmitting model $\Leftrightarrow$ number of params.
  - Cost of transmitting data $\Leftrightarrow$ log likelihood of data.
- Choose number of Gaussians to minimize cost.

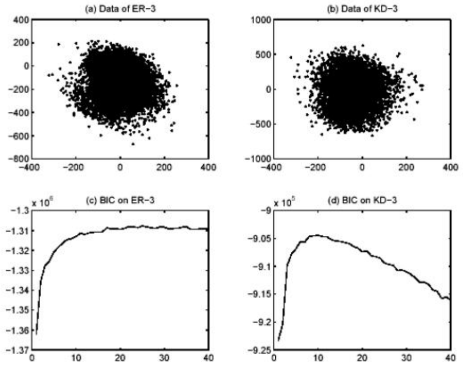


Figure 1. Different degrees of complexity in phone ER-3 and KD-3

Question: How To Initialize Parameters?

- Set mixture weights p_j to $1/k$ (for k Gaussians).
- Pick N data points at random and …
  - Use them to seed initial values of μ_j.
- Set all σ's to arbitrary value …
  - Or to global variance of data.
- Extension: generate multiple starting points.
  - Pick one with highest likelihood.

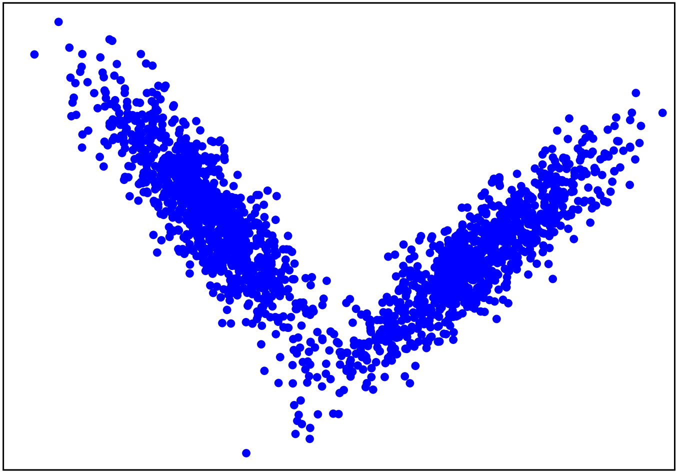
Another Way: Splitting

- Start with single Gaussian, MLE.
- Repeat until hit desired number of Gaussians:
  - Double number of Gaussians by perturbing means . . .
  - Of existing Gaussians by $\pm\epsilon$.
  - Run several iterations of EM.

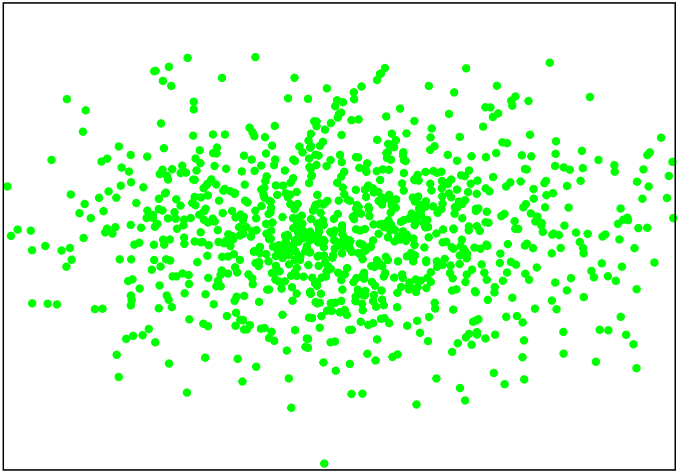
Question: How Long To Train?

- *i.e.*, how many iterations of EM?
- Guess.
- Look at performance on training data.
    - Stop when change in log likelihood per event . . .
    - Is below fixed threshold.
- Look at performance on held-out data.
    - Stop when performance no longer improves.

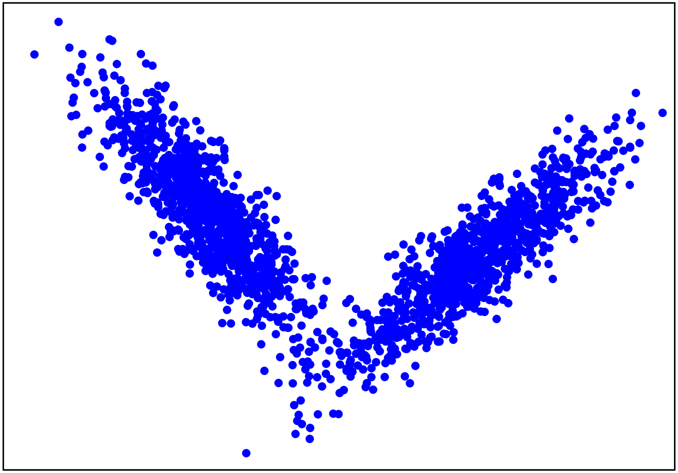
The Data Set



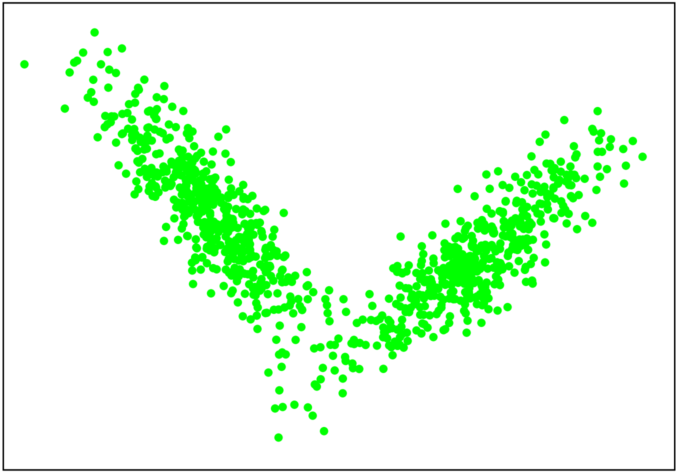
Sample From Best 1-Component GMM



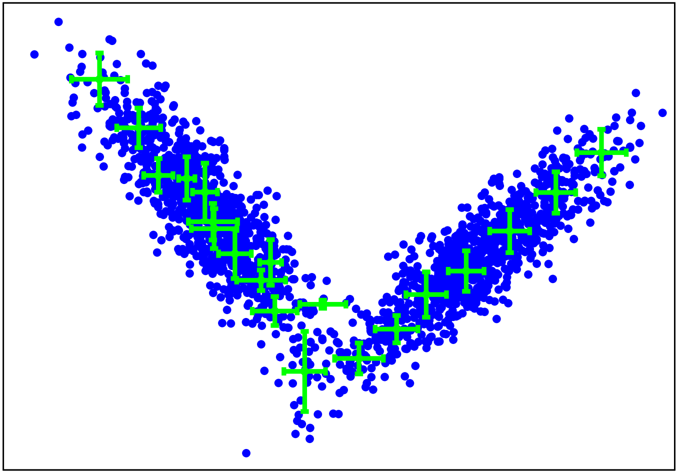
The Data Set, Again



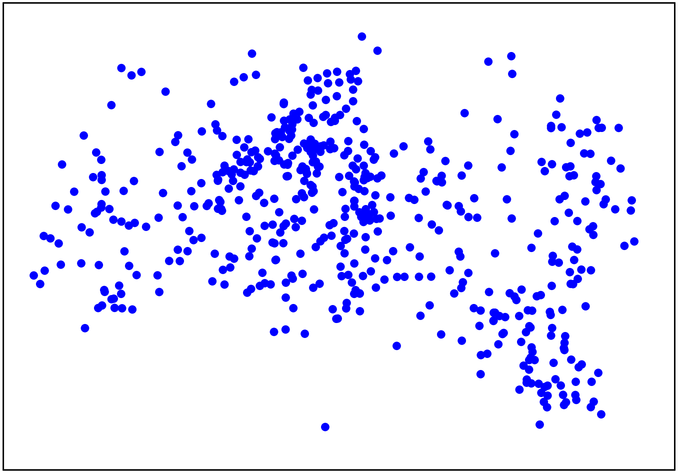
20-Component GMM Trained on Data



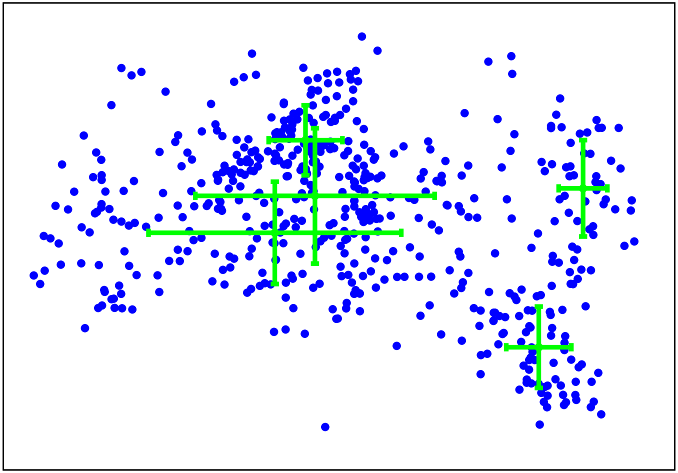
20-Component GMM μ's, σ's



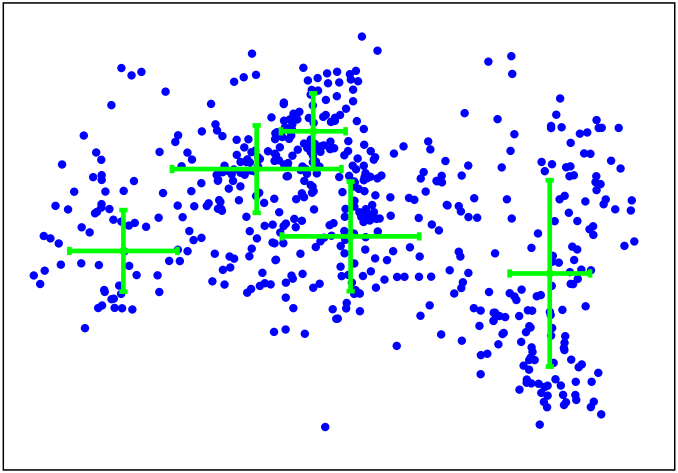
Acoustic Feature Data Set



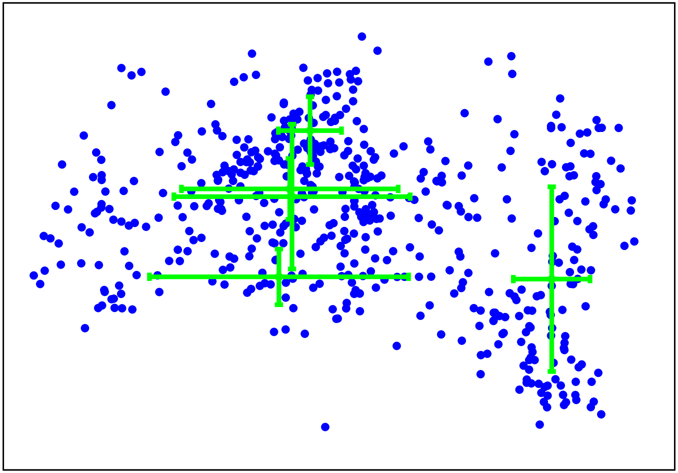
5-Component GMM; Starting Point A



5-Component GMM; Starting Point B



5-Component GMM; Starting Point C



Solutions With Infinite Likelihood

- Consider log likelihood; two-component 1d Gaussian.

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

- If $\mu_1 = x_1$, above reduces to

$$\ln \left(\frac{1}{2\sqrt{2\pi}\sigma_1} + \frac{1}{2\sqrt{2\pi}\sigma_2} e^{\frac{1}{2}\frac{(x_1-\mu_2)^2}{\sigma_2^2}} \right) + \sum_{i=2}^{N} \ldots$$

  which goes to ∞ as $\sigma_1 \to 0$.

- Only consider finite local maxima of likelihood function.
  - Variance flooring.
  - Throw away Gaussians with “count” below threshold.

Recap

- GMM’s are effective for modeling arbitrary distributions.
  - State-of-the-art in ASR for decades.
- The EM algorithm is primary tool for training GMM’s.
  - Very sensitive to starting point.
  - Initializing GMM’s is an art.

References

- S. Chen and P.S. Gopalakrishnan, “Clustering via the Bayesian Information Criterion with Applications in Speech Recognition”, ICASSP, vol. 2, pp. 645–648, 1998.
- A.P. Dempster, N.M. Laird, D.B. Rubin, “Maximum Likelihood from Incomplete Data via the EM Algorithm”, Journal of the Royal Stat. Society. Series B, vol. 39, no. 1, 1977.
- R. Neal, G. Hinton, “A view of the EM algorithm that justifies incremental, sparse, and other variants”, *Learning in Graphical Models*, MIT Press, pp. 355–368, 1999.

Where Are We: The Big Picture

- Given test sample, find nearest training sample.

$$w^* = \arg\min_{w \in \text{vocab}} \text{distance}(A'_{\text{test}}, A'_w)$$

- Total distance between training and test sample …
  - Is sum of distances between aligned frames.

$$\text{distance}_{\tau_x,\tau_y}(X, Y) = \sum_{t=1}^{T} \text{framedist}(x_{\tau_x(t)}, y_{\tau_y(t)})$$

- Goal: move from *ad hoc* distances to probabilities.

Gaussian Mixture Models

- Assume many training templates for each word.
  - Calc distance between *set* of training frames ...
  - And test frame.

$$\text{framedist}((x_1, x_2, \ldots, x_D); y)$$

- Idea: use $x_1, x_2, \ldots, x_D$ to train GMM: $P(x)$.

$$\text{framedist}((x_1, x_2, \ldots, x_D); y) \Rightarrow -\log P(y) \quad !$$

What's Next: Hidden Markov Models

- Replace DTW with probabilistic counterpart.
- Together, GMM's and HMM's comprise . . .
  - Unified probabilistic framework.
- Old paradigm:

$$w^* = \operatorname*{arg\,min}_{w \in \text{vocab}} \text{distance}(A'_{\text{test}}, A'_w)$$

- New paradigm:

$$w^* = \operatorname*{arg\,max}_{w \in \text{vocab}} P(A'_{\text{test}} | w)$$

Part III

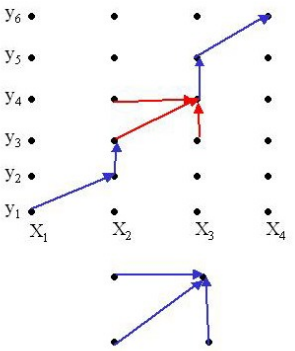
Introduction to Hidden Markov Models

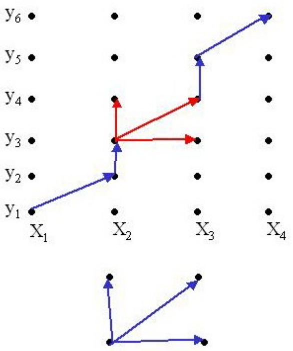
Introduction to Hidden Markov Models

- 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 weights from data?
- Take many utterances . . .



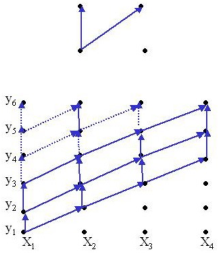


Learning Weights From Data

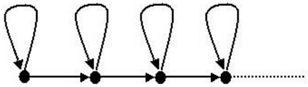
- For each node in DP path, count number of times move up $\uparrow$ right $\rightarrow$ and diagonally $\nearrow$.
- Normalize number of times each direction taken by total number of times node was actually visited.
- Take some constant times reciprocal as weight.
- Example: particular node visited 100 times.
    - Move $\nearrow$ 50 times; $\rightarrow$ 25 times; $\uparrow$ 50 times.
    - Set weights to 2, 4, and 4, respectively (or 1, 2, and 2).
- Point: weight distribution should reflect …
    - Which directions are taken more frequently at a node.
- Weight estimation not addressed in DTW …
    - But central part of *Hidden Markov models*.

DTW and Directed Graphs

- Take following Dynamic Time Warping setup:

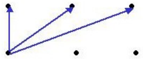


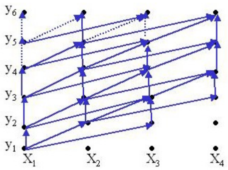
- Let's look at representation of this as directed graph:



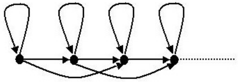
DTW and Directed Graphs

- Another common DTW structure:





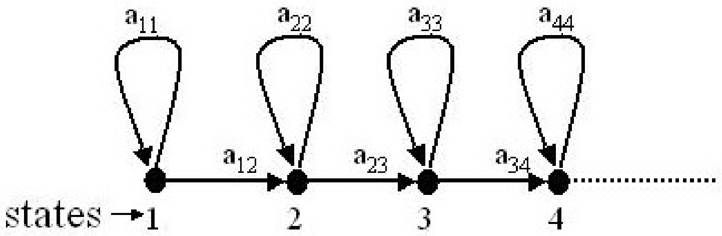
- As a directed graph:



- Can represent even more complex DTW structures . . .
  - Resultant directed graphs can get quite bizarre.

Path Probabilities

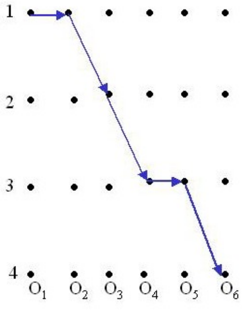
- Let's assign probabilities to transitions in directed graph:



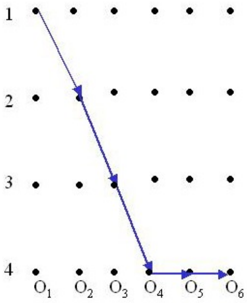
- a_{ij} is transition probability going from state i to state j, where $\sum_j a_{ij} = 1$.
- Can compute probability P of individual path just using transition probabilities a_{ij}.

Path Probabilities

- It is common to reorient typical DTW pictures:



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



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

- Above only describes path probabilities associated with transitions.
- Also need to include likelihoods associated with *observations*.

Path Probabilities

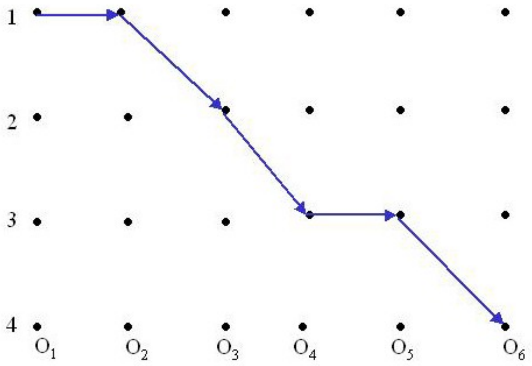
- As in GMM discussion, let us define likelihood of producing observation x_i from state j as

$$b_j(x_i) = \sum_m c_{jm} \frac{1}{(2\pi)^{d/2} |\Sigma_{jm}|^{1/2}} \; e^{-\frac{1}{2}(\mathbf{x_i} - \boldsymbol{\mu}_{jm})^T \boldsymbol{\Sigma}_{jm}^{-1} (\mathbf{x_i} - \boldsymbol{\mu}_{jm})}$$

  where c_{jm} are mixture weights associated with state j.
- This state likelihood is also called the *output probability* associated with state.

Path Probabilities

- In this case, likelihood of entire path can be written as:



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

Hidden Markov Models

- The output and transition probabilities define a *Hidden Markov Model* or *HMM*.
    - Since probabilities of moving from state to state only depend on current and previous state, model is Markov.
    - Since only see observations and have to infer states after the fact, model is *hidden*.
- One may consider HMM to be *generative* model of speech.
    - Starting at upper left corner of trellis, generate observations according to permissible transitions and output probabilities.
- Not only can compute likelihood of single path ...
    - Can compute overall likelihood of observation string ...
    - As sum over *all* paths in trellis.

HMM: The Three Main Tasks

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

Part IV

Epilogue

Sample Project List

Course Feedback

1. Was this lecture mostly clear or unclear? What was the muddiest topic?
2. Other feedback (pace, content, atmosphere)?
3. What is the chance you will do a non-reading project? If nonzero, what type of project appeals most to you right now? (Doesn’t have to be on list.)