CSC401 Assignment 3

Tutorial 2 of 4

2021-03-17

Based on slides of previous years

Assignment 3

- Two parts:
  - Speaker identification: Determine which of 32 speakers an unknown test sample of speech comes from, given Gaussian mixture models you will train for each speaker.
  - Speech recognition: Compute word-error rates for speech recognition systems using Levenshtein distance.

Today's Agenda

- Speaker identification
- Fitting to data
- Gaussian mixture models
- Truth–lie detection

Speaker Data

- 32 speakers (e.g., S-3C, S-5A).
- Each speaker has up to 12 training utterances.
  - e.g., `/u/csc401/A3/data/S-3C/0.wav`
- Each utterance has 3 files:
  - `*.wav` : The original wave file.
  - `*.mfcc.npy` : The MFCC features in NumPy format
  - `*.txt` : Sentence-level transcription.

Speaker Data (cont.)

- All you need to know: A speech utterance is an Nxd matrix
  - Each row represents the features of a d-dimensional point in time.
  - There are N rows in a sequence of N frames.
  - The data is in numpy arrays `*.mfcc.npy`
  - To read the files: `np.load('1.mfcc.npy')`

data dimension

time

frames

	1	2		d
1	$X_1[1]$	$X_1[2]$	...	$X_1[d]$
2	$X_2[1]$	$X_2[2]$	...	$X_2[d]$
	...	...	...	...
N	$X_N[1]$	$X_N[2]$	...	$X_N[d]$

Speaker Data (cont.)

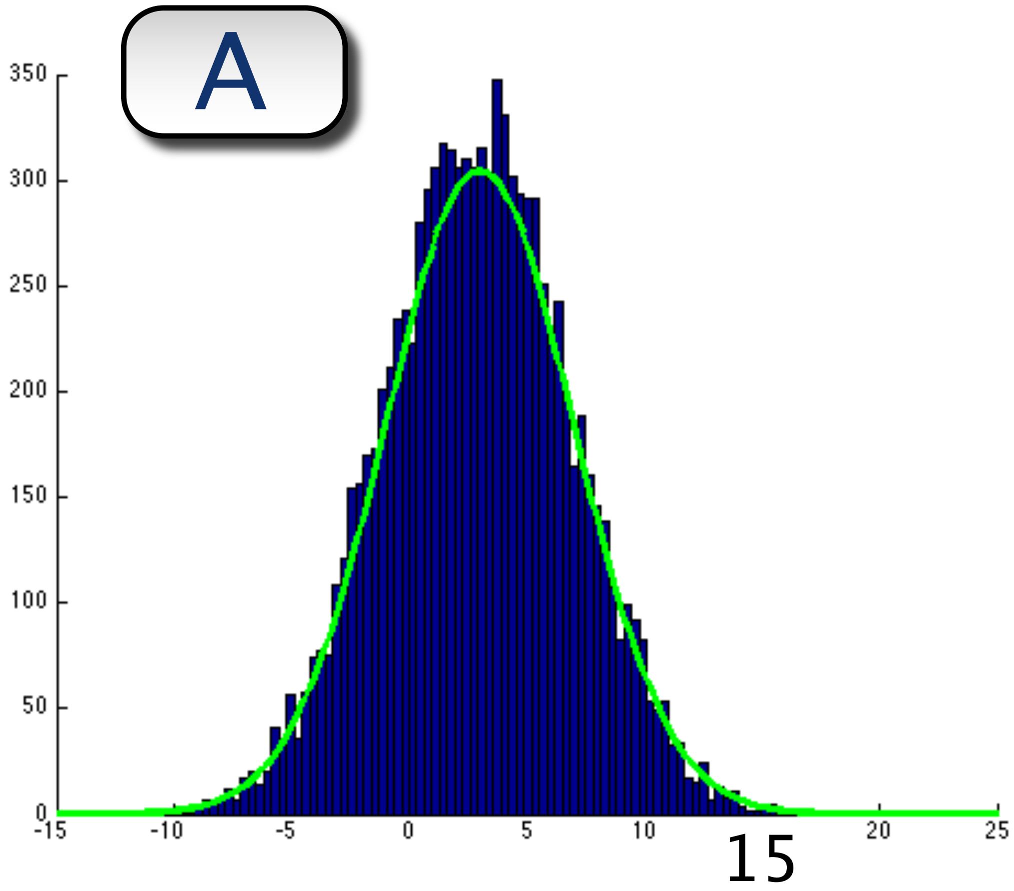
- You are given human transcriptions in `transcripts.txt`
- You are also given Kaldi and Google transcriptions in `transcripts.*.txt`.
- Ignore any symbols that are not words.

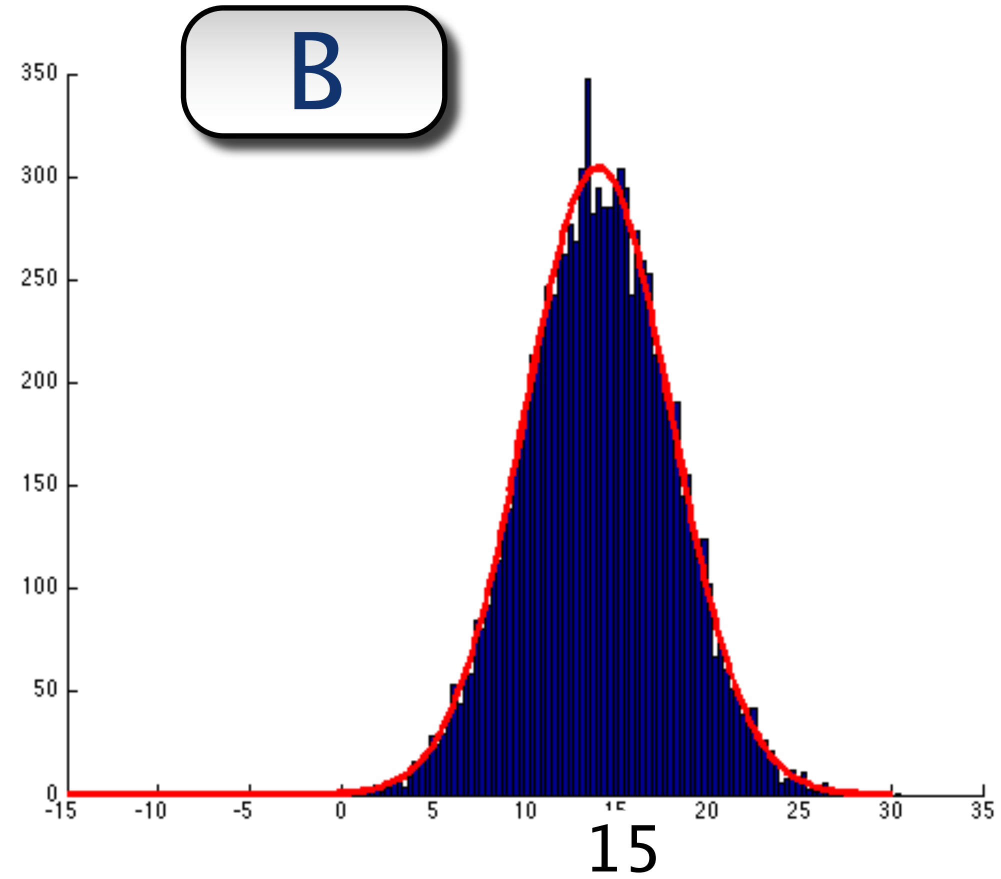
Speaker Identification

- The data is randomly split into training and testing utterances. The objective is to learn the characteristics of each speaker and classify each test utterance to its corresponding speaker.
- Every speaker occupies a characteristic part of the acoustic space.
- We want to learn a probability distribution for each speaker that describes their acoustic behavior.
  - Use those distributions to identify the speaker-dependent features of some unknown sample of speech data.

Some background: fitting to data

- Given a set of observations X of some random variable, we wish to know how X was generated.
- Here, we assume that the data was sampled from a Gaussian Distribution (validated by data).
- Given a new data point (x=15), It is more likely that x was generated by B.



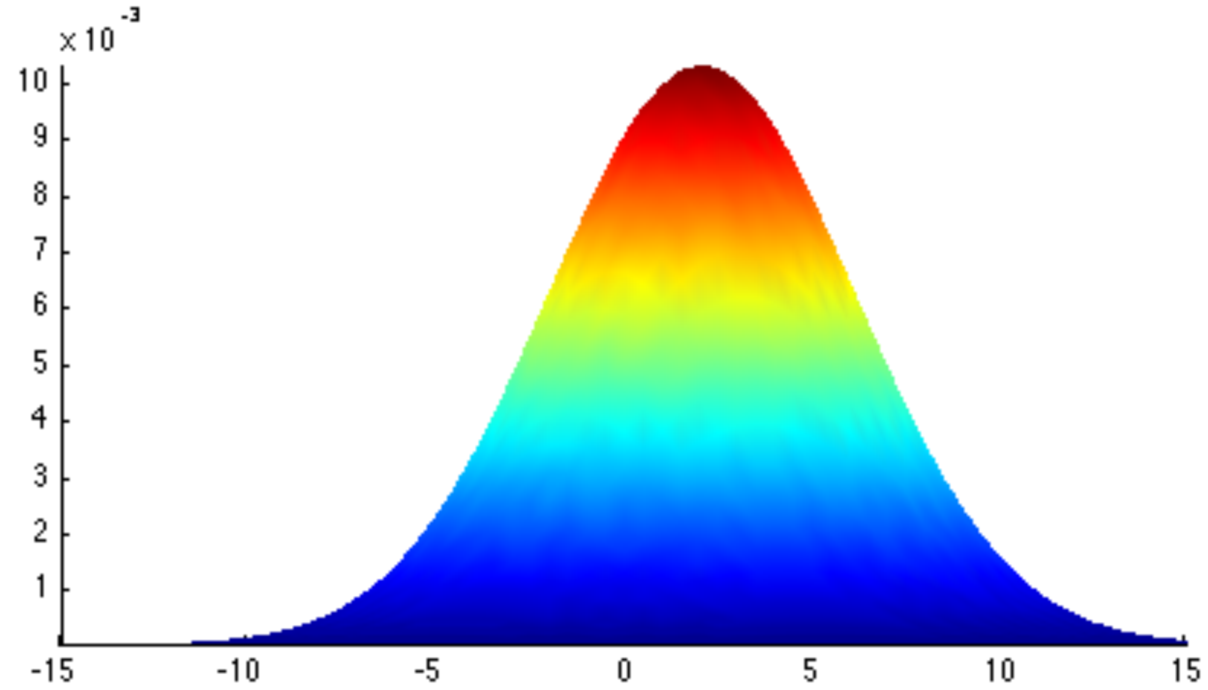


Finding parameters: 1D Gaussians

- Often called *Normal* distributions

$$p(x) = \frac{\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)}{\sqrt{2\pi}\sigma}$$

$$N(\mu, \sigma)$$



$$\mu = E(x) = \int xp(x)dx$$

$$\sigma^2 = E((x-\mu)^2) = \int (x-\mu)^2 p(x)dx$$

- The parameters we can adjust to fit the data are μ and σ^2 :

$$\theta = \langle \mu, \sigma \rangle$$

Maximum likelihood estimation

- Given data: $X = \{x_1, x_2, \ldots, x_n\}$
- and Parameter set: θ
- Maximum likelihood attempts to find the parameter set that maximizes the likelihood of the data.

$$L(X,\theta) = p(X \mid \theta) = p(x_1, x_2, \ldots, x_n \mid \theta) = \prod_{i=1}^{n} p(x_i \mid \theta)$$

- The likelihood function $L(X,\theta)$ provides a surface over all possible parameterizations. In order to find the Maximum Likelihood, we set the derivative to zero:

$$\frac{\partial}{\partial\theta} L(X,\theta) = 0$$

MLE – 1D Gaussian

- Estimate $\hat{\mu}$:

$$L(X,\mu) = p(X \mid \mu) = \prod_{i=1}^{n} p(x_i \mid \mu) = \prod_{i=1}^{n} \frac{\exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right)}{\sqrt{2\pi}\sigma}$$

$$\log L(X,\mu) = -\frac{\sum_i (x_i-\mu)^2}{2\sigma^2} - n\log\sqrt{2\pi}\sigma$$

$$\frac{\partial}{\partial\mu}\log L(X,\mu) = \frac{\sum_i (x_i-\mu)}{\sigma^2} = 0$$

$$\hat{\mu} = \frac{\sum_i x_i}{n}$$

- A similar approach gives the MLE estimate of $\hat{\sigma}^2$:

$$\hat{\sigma}^2 = \frac{\sum_i (x_i-\hat{\mu})^2}{n}$$

Multidimensional Gaussians

- When your data is d-dimensional, the input variable is

  $$\vec{x} = \langle x[1], x[2], \ldots, x[d] \rangle$$

  the mean vector is

  $$\vec{\mu} = E(\vec{x}) = \langle \mu[1], \mu[2], \ldots, \mu[d] \rangle$$

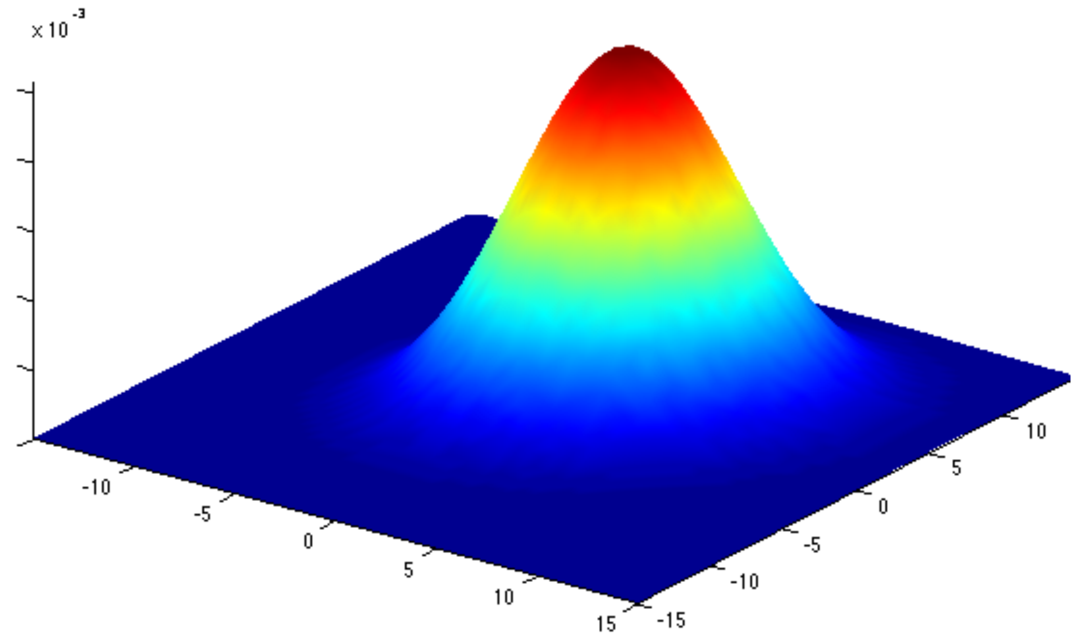
  the covariance matrix is

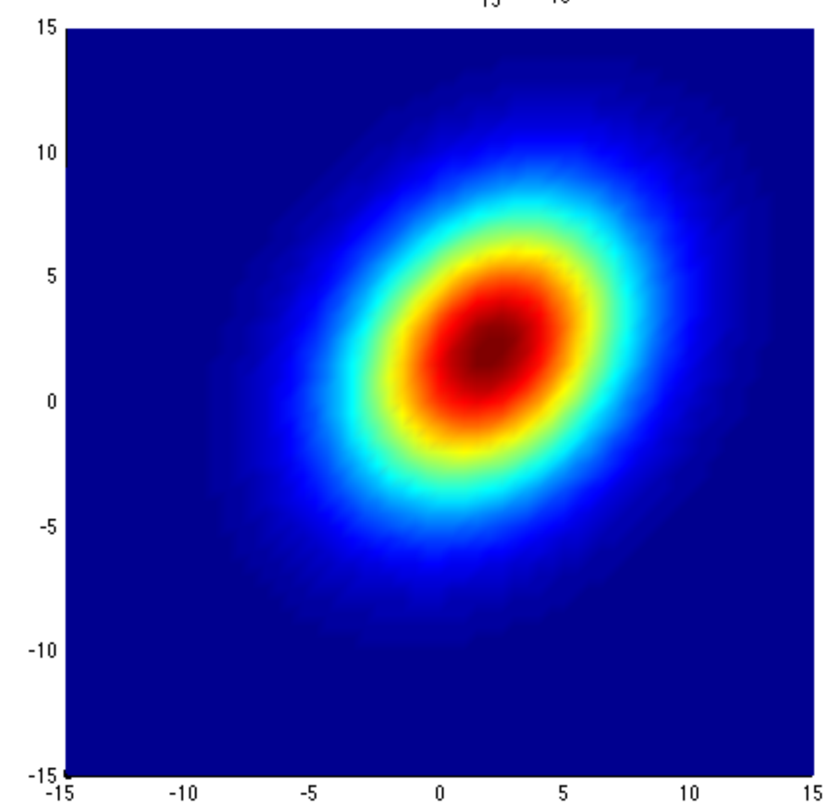
  $$\Sigma = E((\vec{x} - \vec{\mu})(\vec{x} - \vec{\mu})^T)$$

  with $\Sigma[i,j] = E(x[i]x[j]) - \mu[i]\mu[j]$

  and

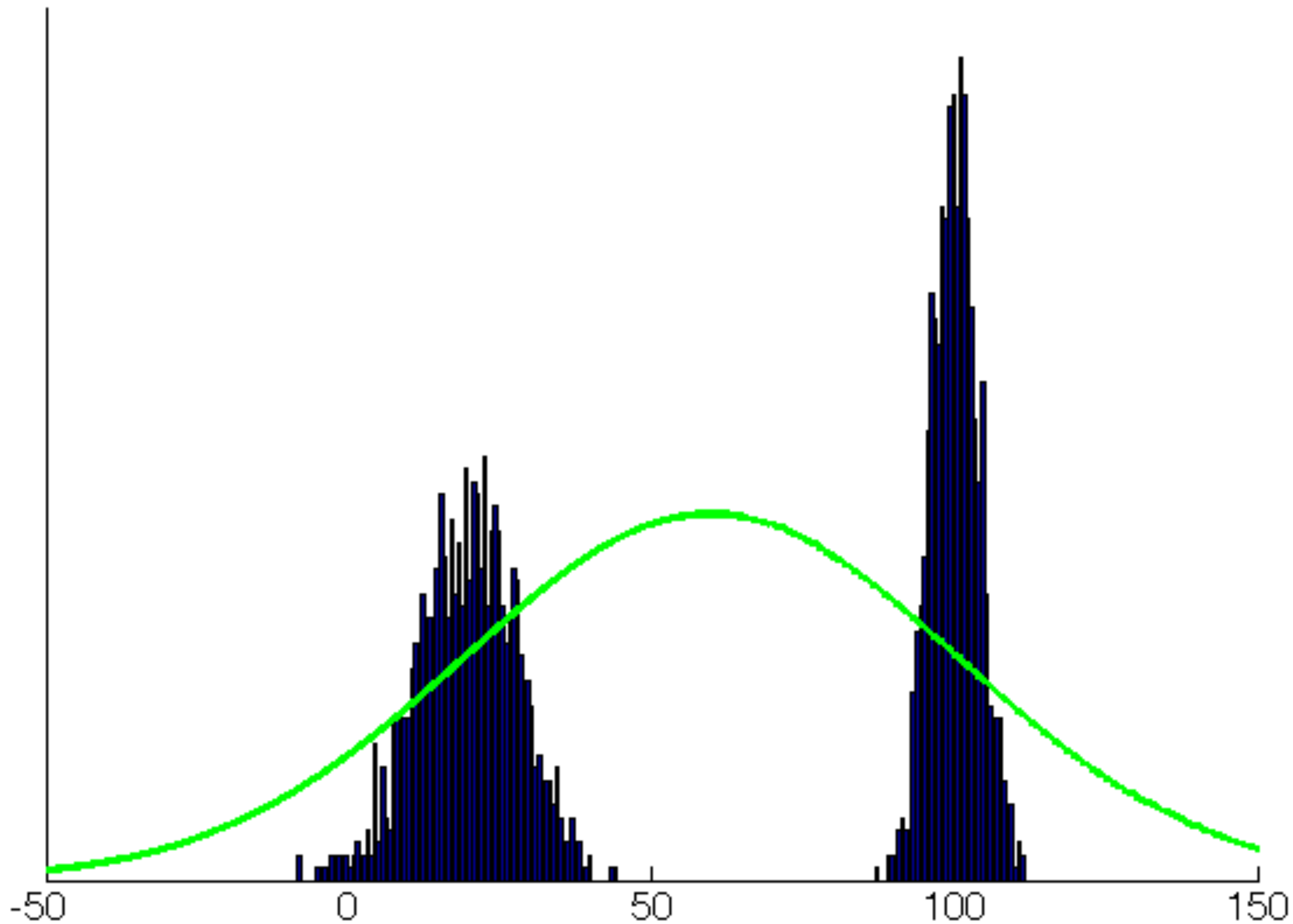
  $$p(\vec{x}) = \frac{\exp\left(-\frac{(\vec{x}-\vec{\mu})^T \Sigma^{-1} (\vec{x}-\vec{\mu})}{2}\right)}{(2\pi)^{d/2} |\Sigma|^{1/2}}$$





Non-Gaussian data

- Our speaker data does not behave unimodally.
  - i.e., we can't use just 1 Gaussian per speaker.
- E.g., observations below occur mostly bimodally, so fitting 1 Gaussian would not be representative.



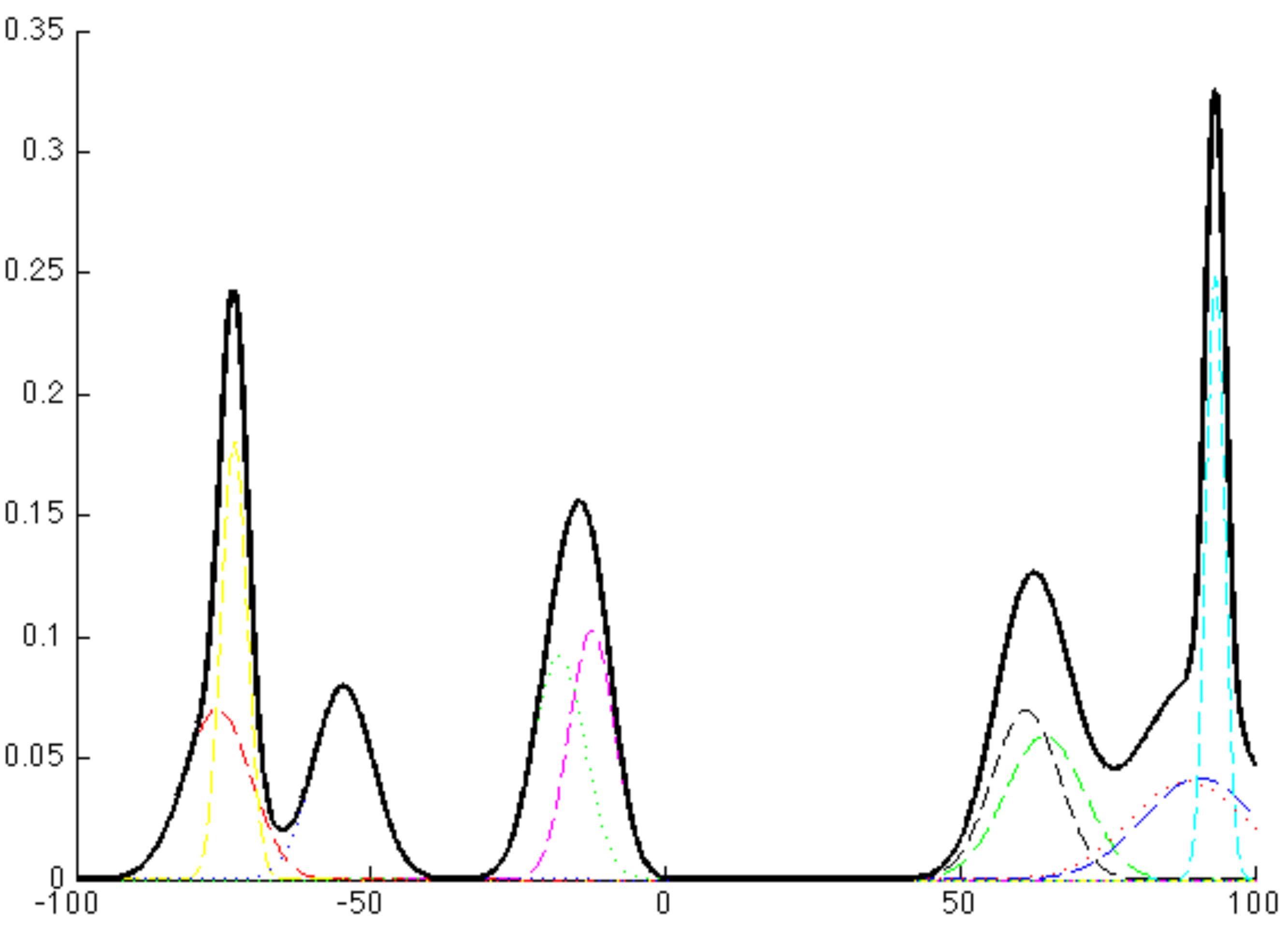
UNIVERSITY OF TORONTO

Gaussian mixtures

- Gaussian mixtures are a weighted linear combination of M component gaussians.

$$\langle \Gamma_1, \ldots, \Gamma_M \rangle$$

$$p(\vec{x}) = \sum_{j=1}^{M} p(\Gamma_j) p(\vec{x} \mid \Gamma_j)$$



MLE for Gaussian mixtures

- For notational convenience, $\omega_m = p(\Gamma_m), \; b_m(\vec{x_t}) = p(\vec{x_t} \mid \Gamma_m)$
- So

$$p_\Theta(\vec{x_t}) = \sum_{m=1}^{M} \omega_m b_m(\vec{x_t}), \; \Theta = \langle \omega_m, \vec{\mu_m}, \Sigma_m \rangle, \; m = 1, \ldots, M$$

$$\boxed{b_m(\vec{x_t}) = \frac{\exp\left(-\frac{1}{2}\sum_{i=1}^{d} \frac{(x_t[i] - \mu_m[i])^2}{\sigma_m^2[i]}\right)}{(2\pi)^{d/2}\left(\prod_{i=1}^{d} \sigma_m^2[i]\right)^{1/2}}}$$

- To find $\hat{\Theta}$, we solve $\nabla_\Theta \log L(X, \Theta) = 0$ where

$$\log L(X, \Theta) = \sum_{t=1}^{N} \log p_\Theta(\vec{x_t}) = \sum_{t=1}^{N} \log \left(\sum_{m=1}^{M} \omega_m b_m(\vec{x_t}) \right)$$

...see Appendix for more

MLE for Gaussian mixtures (pt. 2)

- Given $$\frac{\partial \log L(X,\Theta)}{\partial \mu_m[n]} = \sum_{t=1}^{N} \frac{1}{p_\Theta(\vec{x_t})} \left[\frac{\partial}{\partial \mu_m[n]} \omega_m b_m(\vec{x_t}) \right]$$
- Since $$\frac{\partial}{\partial \mu_m[n]} b_m(\vec{x_t}) = b_m(\vec{x_t}) \frac{x_t[n] - \mu_m[n]}{\sigma_m^2[n]}$$
- We obtain $\hat{\mu_m}[n]$ by solving for $\mu_m[n]$ in :

$$\frac{\partial \log L(X,\Theta)}{\partial \mu_m[n]} = \sum_{t=1}^{N} \frac{\omega_m}{p_\Theta(\vec{x_t})} b_m(\vec{x_t}) \frac{x_t[n] - \mu_m[n]}{\sigma_m^2[n]} = 0$$

$$b_m(\vec{x_t}) = p(\vec{x_t} \mid \Gamma_m)$$
$$p(\Gamma_m \mid \vec{x_t}, \Theta) = \frac{\omega_m}{p_\Theta(\vec{x_t})} b_m(\vec{x_t})$$

and:

$$\hat{\mu_m}[n] = \frac{\sum_t p(\Gamma_m \mid \vec{x_t}, \Theta) x_t[n]}{\sum_t p(\Gamma_m \mid \vec{x_t}, \Theta)}$$

Recipe for GMM ML estimation

- Do the following for each speaker individually. Use all the frames available in their respective **Training** directories

1. **Initialize**: Guess $\Theta = \langle \omega_m, \vec{\mu_m}, \Sigma_m \rangle,\ m = 1, \ldots, M$ with M random vectors in the data, or by performing M-means clustering.
2. **Compute likelihood**: Compute $b_m(\vec{x_t})$ and $P(\Gamma_m \mid \vec{x_t}, \Theta)$
3. **Update parameters**:

$$\hat{\omega_m} = \frac{1}{T}\sum_{t=1}^{T} p(\Gamma_m \mid \vec{x_t}, \Theta)$$

$$\hat{\vec{\sigma}}_m^{\,2} = \frac{\sum_t p(\Gamma_m \mid \vec{x_t}, \Theta)\vec{x_t}^{\,2}}{\sum_t p(\Gamma_m \mid \vec{x_t}, \Theta)} - \hat{\vec{\mu}}_m^{\,2}$$

$$\hat{\vec{\mu_m}} = \frac{\sum_t p(\Gamma_m \mid \vec{x_t}, \Theta)\vec{x_t}}{\sum_t p(\Gamma_m \mid \vec{x_t}, \Theta)}$$

Repeat 2&3 until converges

$$\log p(X \mid \hat{\Theta}_{i+1}) - \log p(X \mid \hat{\Theta}_i) < \epsilon$$

Cheat sheet

$$b_m(\vec{x_t}) = p(\vec{x_t} \mid \Gamma_m)$$

$$b_m(\vec{x_t}) = \frac{\exp\left(-\frac{1}{2}\sum_{i=1}^{d}\frac{(x_t[i]-\mu_m[i])^2}{\sigma_m^2[i]}\right)}{(2\pi)^{d/2}\left(\prod_{i=1}^{d}\sigma_m^2[i]\right)^{1/2}}$$

Probability of observing x_t in the m^{th} Gaussian

$$\omega_m = p(\Gamma_m)$$

Prior probability of the m^{th} Gaussian

$$p(\Gamma_m \mid \vec{x_t}, \Theta) = \frac{\omega_m}{p_\Theta(\vec{x_t})} b_m(\vec{x_t})$$

Probability of the m^{th} Gaussian, given x_t

$$p_\Theta(\vec{x_t}) = \sum_{m=1}^{M} \omega_m b_m(\vec{x_t})$$

Probability of x_t in the GMM

Initializing theta

$$\Theta = \langle \omega_1, \mu_1, \Sigma_1, \omega_2, \mu_2, \Sigma_2, \ldots, \omega_M, \mu_M, \Sigma_M \rangle$$

- Initialize each mu_m to a random vector from the data.
- Initialize Sigma_m to a `random' diagonal matrix (or identity matrix).
- Initialize omega_m randomly, with these constraints:

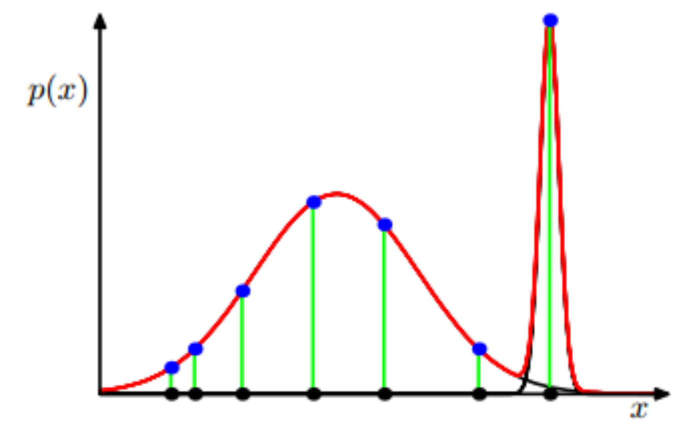
$$0 \leq \omega_m \leq 1$$

$$\sum_m \omega_m = 1$$

- A good choice would be to set omega_m to 1/M

A note: Overfitting GMMs

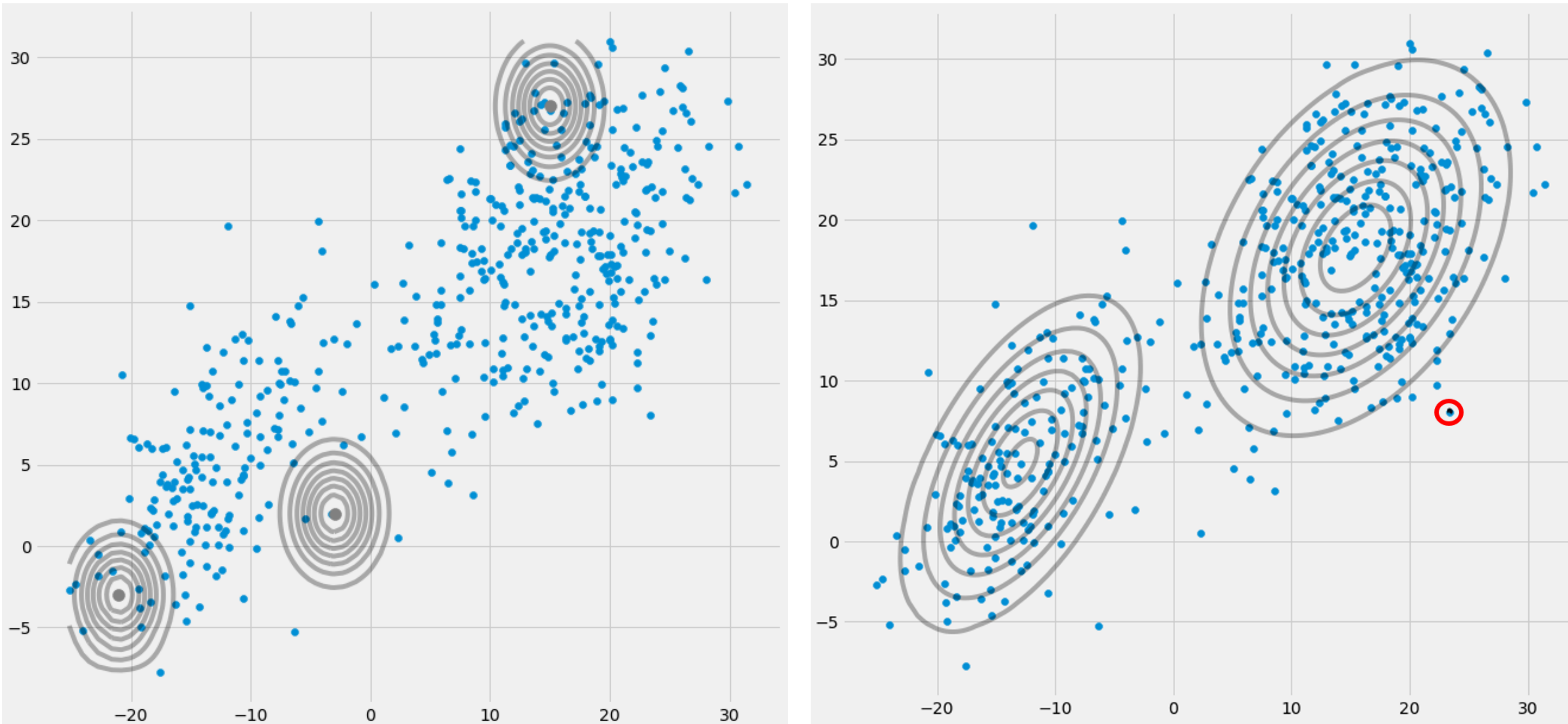
- Consider when a Gaussian mixture has M components, each with covariance matrix $\Sigma_m = \sigma_m^2 \mathbf{I}$
- If the *j*–th component of the GMM has its mean equal exactly to the *n*–th data point, then: $\mu_j = x_n$



- Then $\mathcal{N}(x_n | x_n,\ \sigma_j^2 \mathbf{I}) = \frac{1}{(2\pi)^{d/2}} \frac{1}{\sigma_j^d}$
- When $\sigma_j \longrightarrow 0$:
  - The covariance matrix is singular (not invertible)
  - The likelihood of the data spikes to infinity
  - The MLE of the GMM in this case becomes ill–defined
- But how can this happen? **The problem of choosing M!**

Overfitting GMMS: Choosing M

- Your data has 2 clusters, but you choose M = 3.



- This isn't possible when we fit a single Gaussian. Why?

UNIVERSITY OF TORONTO

Overfitting GMMs: Possible solutions

- Choose a smaller M
- Although the approach above can solve your issue, some other solutions that are used include (**<u>you will not need to worry about these for the assignment</u>**):
  - Resetting the mean and variance
  - Use MAP/Bayesian estimation instead of MLE

Your Task

- For each speaker, train a GMM, using the EM algorithm, assuming diagonal covariance.
- Identify the speaker of each test utterance.
- Experiment with the number of mixture elements in the models, the improvement threshold, number of possible speakers, etc.
- Comment on the results

Practical tips for MLE of GMMs

- We assume diagonal covariance matrices. This reduces the number of parameters and can be sufficient in practice given enough components.

- Numerical Stability: Compute likelihoods in the log domain (especially when calculating the likelihood of a sequence of frames).

$$\log b_m(\vec{x_t}) = -\sum_{n=1}^{d} \frac{(\vec{x_t}[n] - \vec{\mu_m}[n])^2}{2\vec{\sigma_m}^2[n]} - \frac{d}{2}\log 2\pi - \frac{1}{2}\log \prod_{n=1}^{d} \vec{\sigma_m}^2[n]$$

- Here, $\vec{x_t},\ \vec{\mu_m}$ and $\vec{\sigma_m}^2$ are d-dimensional vectors.

UNIVERSITY OF TORONTO

Practical tips (pt. 2)

- Efficiency: Pre-compute terms not dependent on $\vec{x_t}$

$$\log b_m(\vec{x_t}) = -\sum_{n=1}^{d}\left(\frac{1}{2}\vec{x_t}[n]^2\vec{\sigma_m}^{-2}[n] - \vec{\mu_m}[n]\vec{x_t}[n]\vec{\sigma_m}^{-2}[n]\right)$$
$$-\left(\sum_{n=1}^{d}\frac{\vec{\mu_m}[n]^2}{2\vec{\sigma_m}^{2}[n]} + \frac{d}{2}\log 2\pi + \frac{1}{2}\log\prod_{n=1}^{d}\vec{\sigma_m}^{2}[n]\right)$$

Practical tips (pt. 3): LogSumExp

- *log_p_m_x*, *logLik* use *log_b_m_x*. But how should we implement *log_b_m_x*? Do we actually have the *b_m_x* (non-logged) values? <u>Answer</u>: *No*

- Naïve way of implementing: exponentiate the log_b_m_x. Problem?

- *scipy.special.logsumexp*

Incorrect Approach

$$\log \sum_{i=1}^{n} x_i = \log \sum_{i=1}^{n} e^{\log x_i}$$

This is not the correct approach. If one of $\log x_i$ is extremely large, you would be just calculating $\log \infty$ in the computer program. On the other hand, if all the $\log x_i$ are extremely small, you would be just calculating $\log 0$ in the computer program. In either case, there would be a problem.

Correct Approach

$$\log \sum_{i=1}^{n} x_i = \log \sum_{n=1}^{N} e^{\log x_i} = \log(e^a \times \sum_{n=1}^{N} e^{\log x_i - a}) = a + \log \sum_{n=1}^{N} e^{\log x_i - a}$$

where $a = \max(\log x_1, \log x_2, \dots, \log x_n)$.

Credit: https://leimao.github.io/blog/LogSumExp/

UNIVERSITY OF TORONTO

Truth/Lie Detection

- Using the transcripts labelled with the truthfulness of the utterances, we build a GRU model for lie detection.
- The starter code has provided you with everything to run the training.
- **What you need to do (file: *model.py*):**
  - Finish the __init__ method of the class LieDetector by filling in the code to create a **unidirectional** GRU with **one hidden layer**
  - Also, initialize a linear layer to project the GRU's output to prediction space. What should the number of output features be?
  - Following the instructions from the handout, run experiments by varying the size of the hidden layer and record the performance of the model. Comment on these results as asked in the handout.

Appendices

Multidimensional Gaussians, pt. 2

- If the i^th and j^th dimensions are statistically independent,

$$E(x[i]x[j]) = E(x[i])E(x[j])$$

  and

$$\Sigma[i,j] = 0$$

- If all dimensions are statistically independent, $\Sigma[i,j] = 0, \ \forall i \neq j$ and the covariance matrix becomes diagonal, which means

$$p(\vec{x}) = \prod_{i=1}^{d} p(x[i])$$

  where

$$p(x[i]) \sim N(\mu[i], \Sigma[i,i])$$
$$\Sigma[i,i] = \sigma^2[i]$$

MLE example – dD Gaussians

- The MLE estimates for parameters $\Theta = \langle \theta_1, \theta_2, \ldots, \theta_d \rangle$ given i.i.d. training data $X = \langle \vec{x}_1, \ldots, \vec{x}_n \rangle$ are obtained by maximizing the joint likelihood

$$L(X, \Theta) = p(X \mid \Theta) = p(\vec{x}_1, \ldots, \vec{x}_n \mid \Theta) = \prod_{i=1}^{n} p(\vec{x}_i \mid \Theta)$$

- To do so, we solve $\nabla_\Theta L(X, \Theta) = 0$, where

$$\nabla_\Theta = \left\langle \frac{\partial}{\partial \theta_1}, \ldots, \frac{\partial}{\partial \theta_d} \right\rangle$$

- Giving

$$\hat{\vec{\mu}} = \frac{\sum_{t=1}^{n} \vec{x}_t}{n} \qquad \hat{\Sigma} = \frac{\sum_{t=1}^{n} \left(\vec{x}_t - \hat{\vec{\mu}}\right)\left(\vec{x}_t - \hat{\vec{\mu}}\right)^T}{n}$$

MLE for Gaussian mixtures (pt1.5)

- Given $\log L(X, \Theta) = \sum_{t=1}^{N} \log p_\Theta(\vec{x}_t)$ and $p_\Theta(\vec{x}_t) = \sum_{m=1}^{M} \omega_m b_m(\vec{x}_t)$

  - Obtain an ML estimate, $\hat{\vec{\mu}}_m$, of the mean vector by maximizing $\log L(X, \vec{\mu}_m)$ w.r.t. $\mu_m[n]$

$$\frac{\partial \log L(X, \Theta)}{\partial \mu_m[n]} = \sum_{t=1}^{N} \frac{\partial}{\partial \mu_m[n]} \log p_\Theta(\vec{x}_t) = \sum_{t=1}^{N} \frac{1}{p_\Theta(\vec{x}_t)} \left[\frac{\partial}{\partial \mu_m[n]} \omega_m b_m(\vec{x}_t) \right]$$

- Why?

d of sum = sum of d

d rule for $\log_e$

d wrt μ_m is 0 for all other mixtures in the sum in $p_\Theta(\vec{x}_t)$