ENGR-E 511; ENGR-E 399

“Machine Learning for Signal Processing”

Module 03: Lecture 01:

Clustering

Minje Kim

Department of Intelligent Systems Engineering

Email: minje@indiana.edu

Website: http://minjekim.com

Research Group: http://saige.sice.indiana.edu

Meeting Request: http://doodle.com/minje

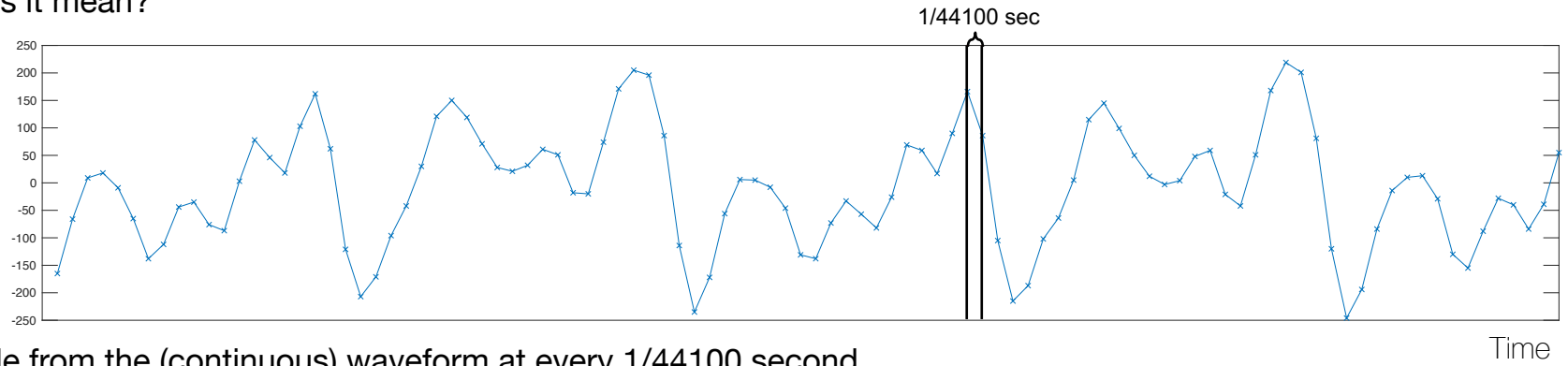
INDIANA UNIVERSITY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

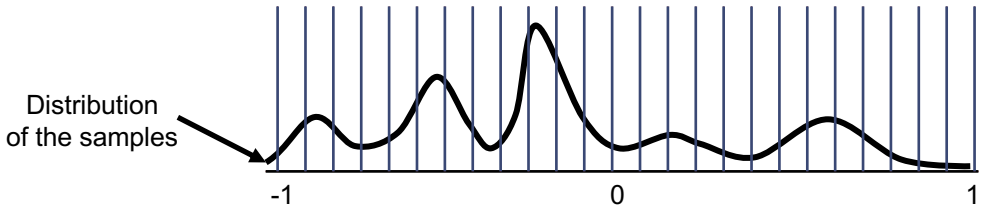
Motivating Problems

- CD

- How do we represent music in a CD?
  - What? What is CD?
  - 44.1 kHz, 16 bit, LPCM
  - What does it mean?



  - We sample from the (continuous) waveform at every 1/44100 second
  - Each sample is represented with one of 2^{16}=65536 values
    - `e.g. 0000 0000 0000 1000 : -0.9998`
    - `e.g. 1111 1111 1101 1100 : +0.9989`
    - `e.g. 1000 0000 0000 0000 : 0`
  - Can we do better?
    - (Real-valued) samples in the same block is represented by the same value (and I don't like it)



Motivating Problems

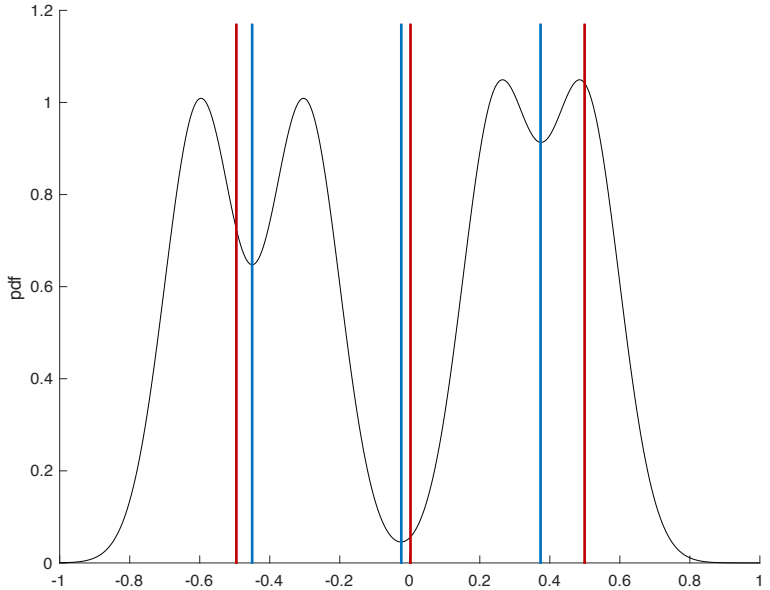
- The Lloyds-Max algorithm

- First of all, we need to save the bits
  - What if there's a lack of bits?
    - Some values are misrepresented

8bits

4bits

- Assume that the raw audio samples are following a distribution like:

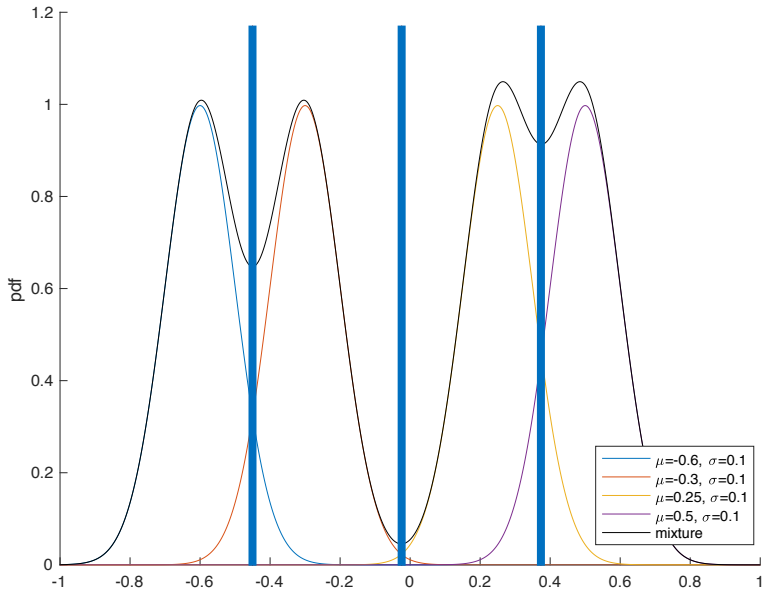


- Which do you prefer between the red boundaries and blue boundaries (they are all for 2 bit encoding)?

Motivating Problems

- The Lloyds-Max algorithm

- You prefer the blue boundaries because they go well with the underlying structure of the sample distribution
  - Underlying structure?



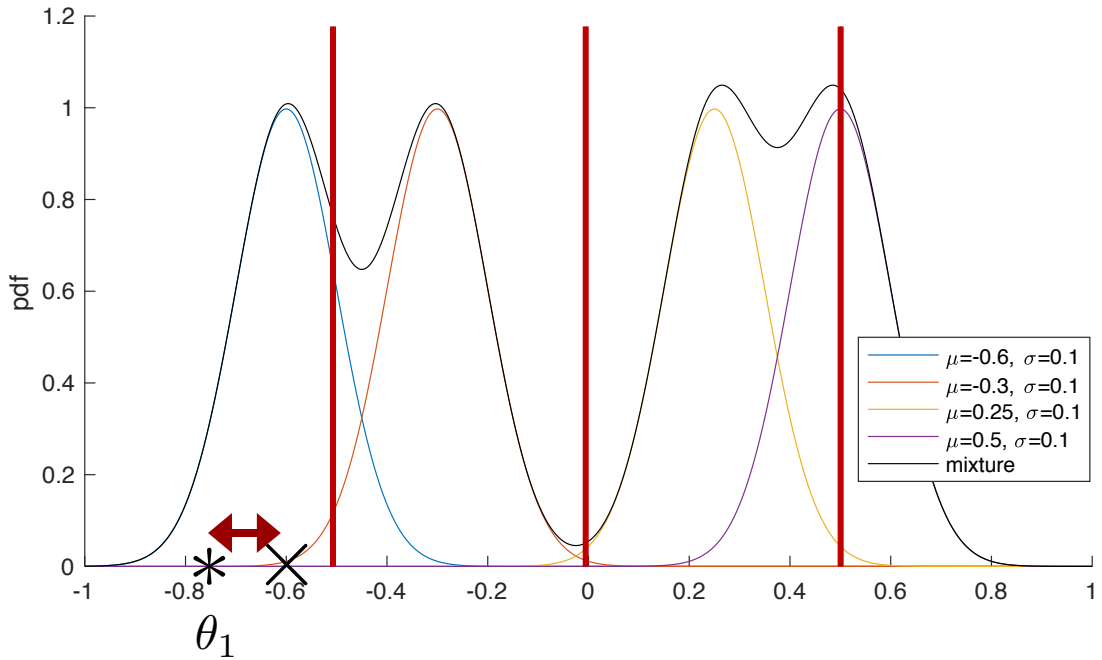
- And they sound better!

4bits (Lloyds) 4bits (Uniform) 8bits (Lloyds) 8bits (Uniform) Original

k-Means Clustering

- A scalar case

- Where's the distortion from?
- Let's start from the red boundaries we don't like
  - The discrepancy between the representative and the actual samples
  - We want to find the representative that creates the least discrepancy
    - For each quantization level
- How do we measure the amount of discrepancy?



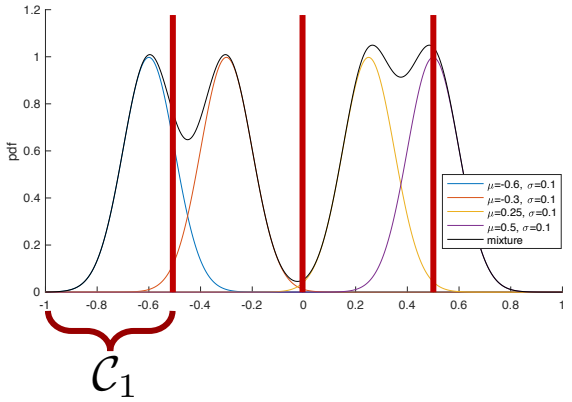
k-Means Clustering

- A scalar case

- So, the objective for j-th range is to find the representative value that minimizes the error

$$\arg\min_{\theta_j} \sum_{i \in \mathcal{C}_j} ||x_i - \theta_j||^2$$

  - For all examples that belong to the j-th range
    - $\mathcal{C}_j$ holds the indices for j-th range
  - So, what's the solution? (why?) $\theta_j = \frac{1}{|\mathcal{C}_j|} \sum_{i \in \mathcal{C}_j} x_i$



- Eventually, we need to do this for all J ranges

$$\arg\min_{\theta_1, \theta_2, \cdots, \theta_J} \sum_{j=1}^{J} \sum_{i \in \mathcal{C}_j} ||x_i - \theta_j||^2$$

- Let me put it in another way

$$\arg\min_{\theta_1, \theta_2, \cdots, \theta_J} \sum_{j=1}^{J} \sum_{i=1}^{N} u_{ij} ||x_i - \theta_j||^2$$

  - Where u_{ij} is a matrix that indicates the membership
    - e.g. $u_{ij} = 1$ then $i \in \mathcal{C}_j$
  - The membership matrix u_{ij} should also meet another constraint: $\sum_j u_{ij} = 1$

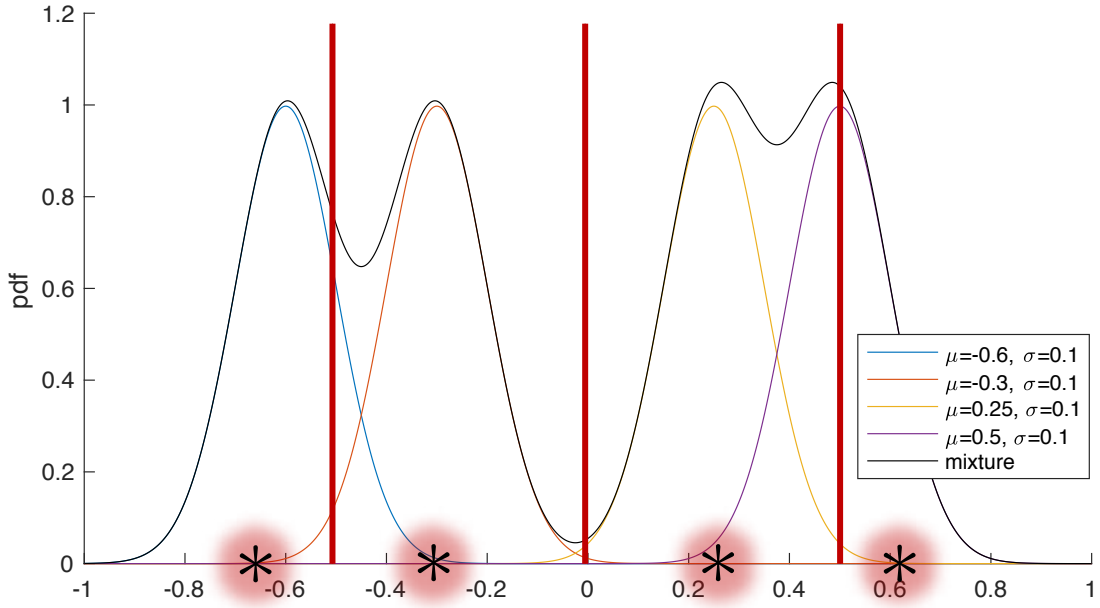
	j=1			j=J
i=1	1	0	0	0
	0	0	0	1
	1	0	0	0
	0	1	0	0
	0	0	1	0
	…	…	…	…
i=N	0	1	0	0

4th sample belongs to the 2nd cluster

k-Means Clustering

- A scalar case

- Are we done?
  - No, we assumed that the boundaries are correct, but they aren't



  - In other words, we don't know if the u_{ij} matrix is correct

k-Means Clustering

- A scalar case

- We need to optimize w.r.t. the membership matrix as well

$$\arg\min_{\boldsymbol{\theta}, \boldsymbol{U}} \sum_{j=1}^{J} \sum_{i=1}^{N} u_{ij} ||x_i - \theta_j||^2$$

- The k-means clustering algorithm on scalar samples
  - Initialize the means $\boldsymbol{\theta} = \{\theta_1, \theta_2, \cdots, \theta_J\}^\top$ with random numbers
  - Update the membership matrix
    - This is actually a complicated optimization problem (why?), but the solution is simple
    - For a fixed set of means

$$u_{ij} = \begin{cases} 1 & \text{if } j = \arg\min_{j'} ||x_i - \theta_{j'}||^2 \\ 0 & \text{otherwise} \end{cases}$$

  - Update the means

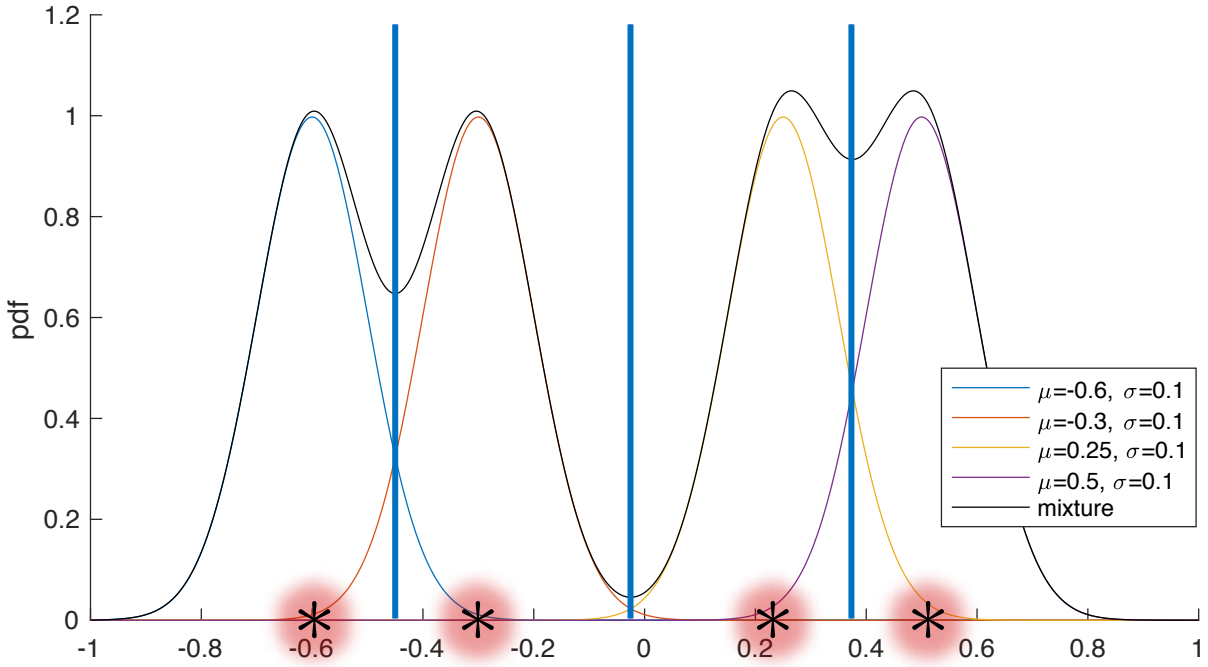
$$\theta_j = \frac{1}{|\mathcal{C}_j|} \sum_{i \in \mathcal{C}_j} x_i$$

$$= \frac{1}{|\mathcal{C}_j|} \sum_{i=1}^{N} u_{ij} x_i$$

k-Means Clustering

- A scalar case

- Let's get back to the CD encoding problem (Lloyd-Max algorithm)
- Now instead of all the possible real values between -1 and +1
- We replace the values within each range with their corresponding representatives, i.e. the means.



4bits (Lloyds)

4bits (Uniform)

8bits (Lloyds)

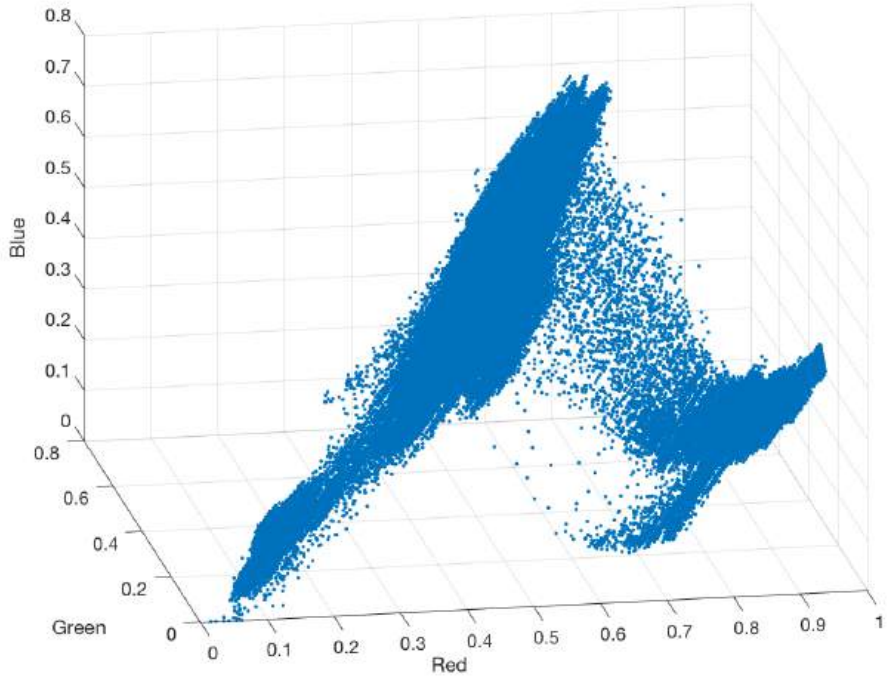
8bits (Uniform)

Original

Motivating Problems

- Black cat, red wall, gray ground

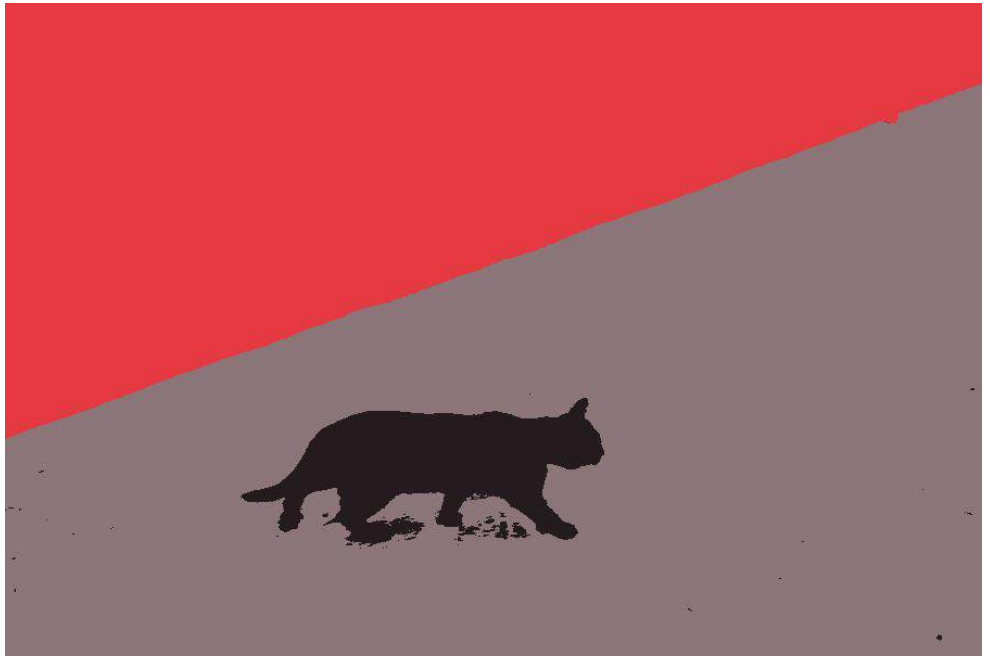
- Now let's move on to the multi-dimensional case
- In general, how do we quantize a vector?
- First off, can you (verbally) describe this picture?

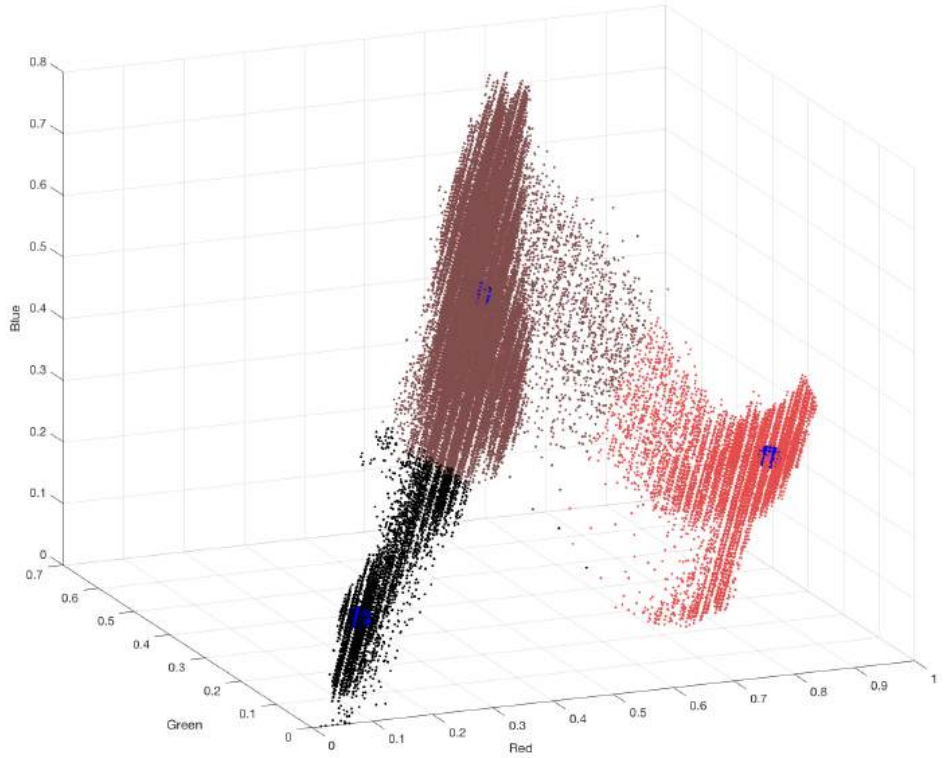


Motivating Problems

- Black cat, red wall, gray ground

- k-means with three clusters





Motivating Problems

- Black cat, red wall, gray ground

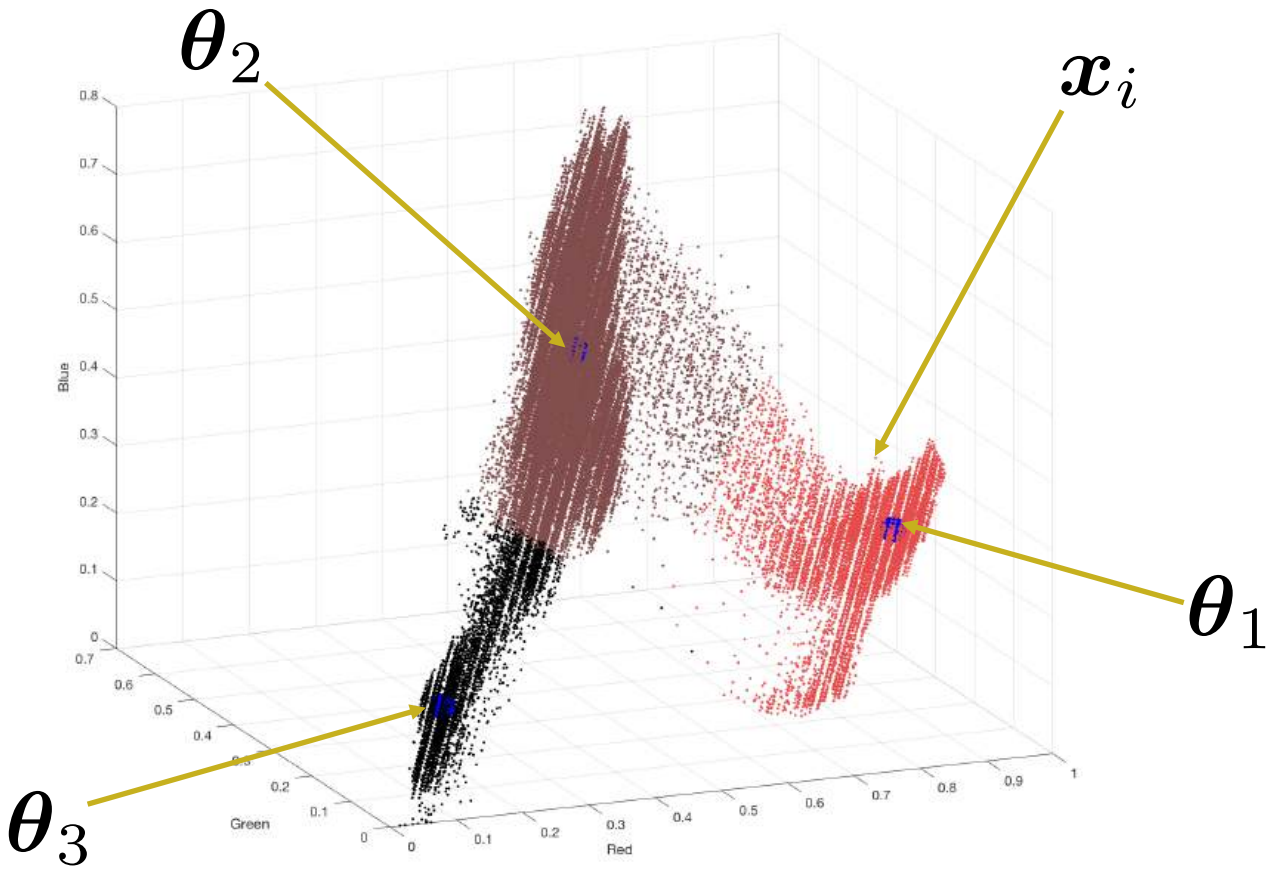
- k-means with 8 clusters and 16 clusters

- Algorithm-wise everything is the same except for the fact that the input samples are 3D (RGB) vectors

$$\arg\min_{\boldsymbol{\theta},\boldsymbol{U}} \sum_{j=1}^{J}\sum_{i=1}^{N} u_{ij}||x_i - \theta_j||^2 \longrightarrow \arg\min_{\boldsymbol{\Theta},\boldsymbol{U}} \sum_{j=1}^{J}\sum_{i=1}^{N} u_{ij}||\boldsymbol{x}_i - \boldsymbol{\theta}_j||^2$$

Motivating Problems

- Black cat, red wall, gray ground



Vector Quantization

- Clustering on multi-dimensional samples

- What we did is something called **Vector Quantization (VQ)**
  - Do clustering
  - Replace vector samples with the mean of the cluster they belong to

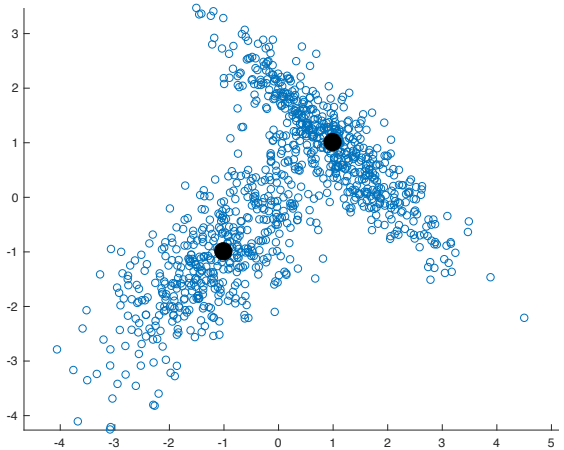
- In image, we had 3D vectors (RGB)
  - If we assume J clusters, we need $\log_2 J$ bits to encode a pixel
    - Rather than 8 bits per pixel

- What we need:
  - A good clustering
    - A small number of means that are representative enough
  - Dictionary (codebook)
    - A codeword corresponds to one of the means
  - Index to the codebook
    - Index of the pixel-wise membership

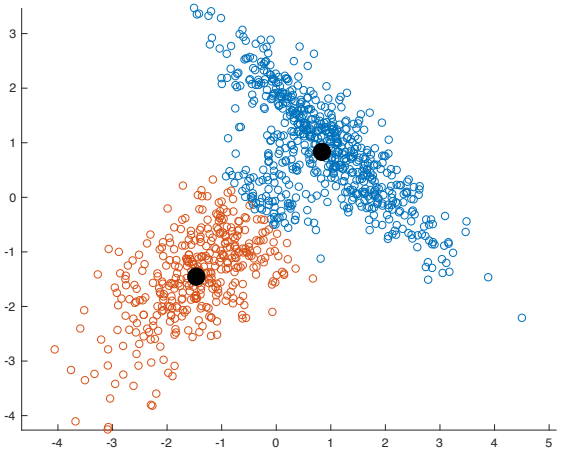
Gaussian Mixture Model

- What's wrong with k-means?

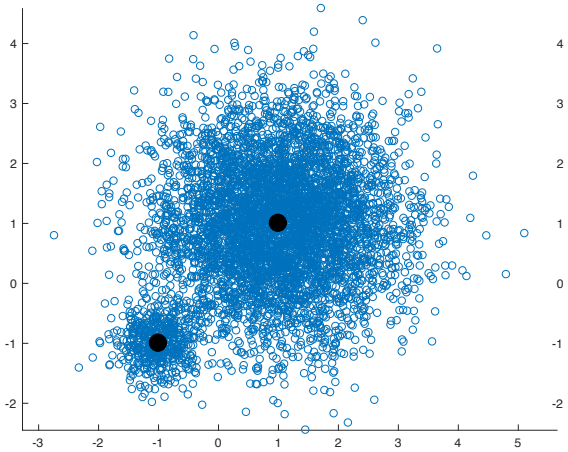
- What I don't like about k-means
  - Euclidean distance, hard decision, equiprobable clusters, diagonal cov…



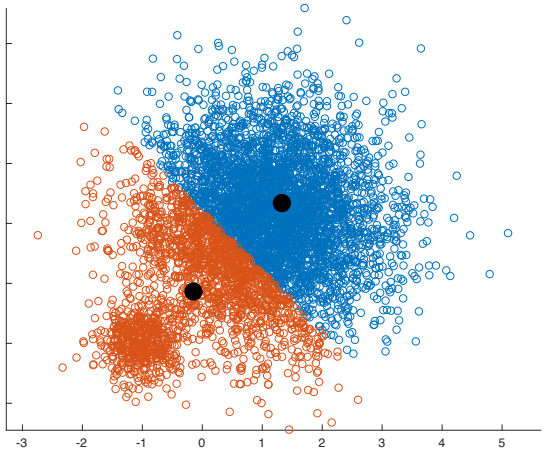
Input data with GT means



K-means clustering results



Input data with GT means



K-means clustering results

Gaussian Mixture Model

- An alternative: Mahalanobis distance

- Let's tweak k-means
- First, let's take variance into account for the distance metric
  - Mahalanobis distance:

$$\mathcal{D}_M(x_i||\mu_j) = \sqrt{\frac{(x_i - \mu_j)^2}{\sigma_j^2}}$$

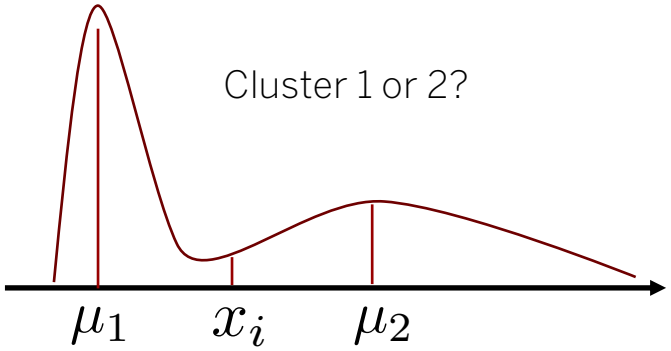
μ_j Mean of j-th cluster

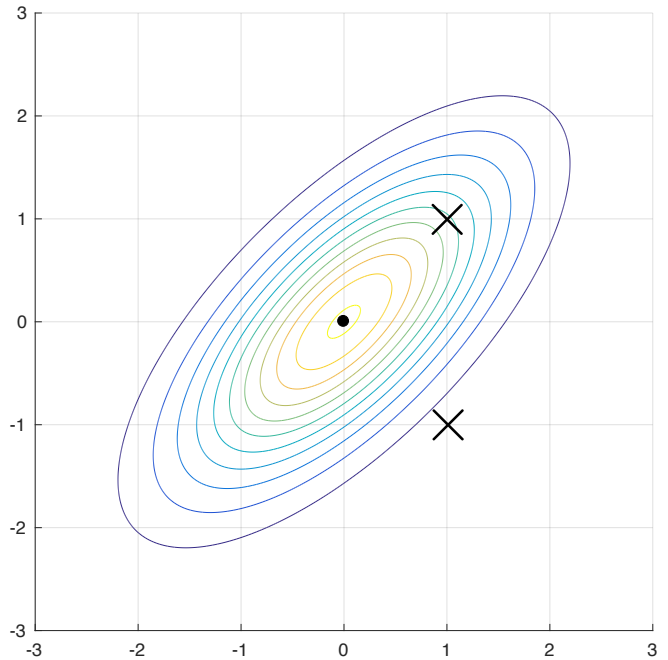
σ_j Standard dev. of j-th cluster

  - Multi-dimensional cases with covariance:

$$\mathcal{D}_M(\boldsymbol{x}_i||\boldsymbol{\mu}_j) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_j)}$$

  - For a 2D Gaussian with $\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$
  - (1,1)
    - Euclidean: $\sqrt{2}$
    - Mahalanobis: 1.0847
  - (1,-1)
    - Euclidean: $\sqrt{2}$
    - Mahalanobis: 2.5820

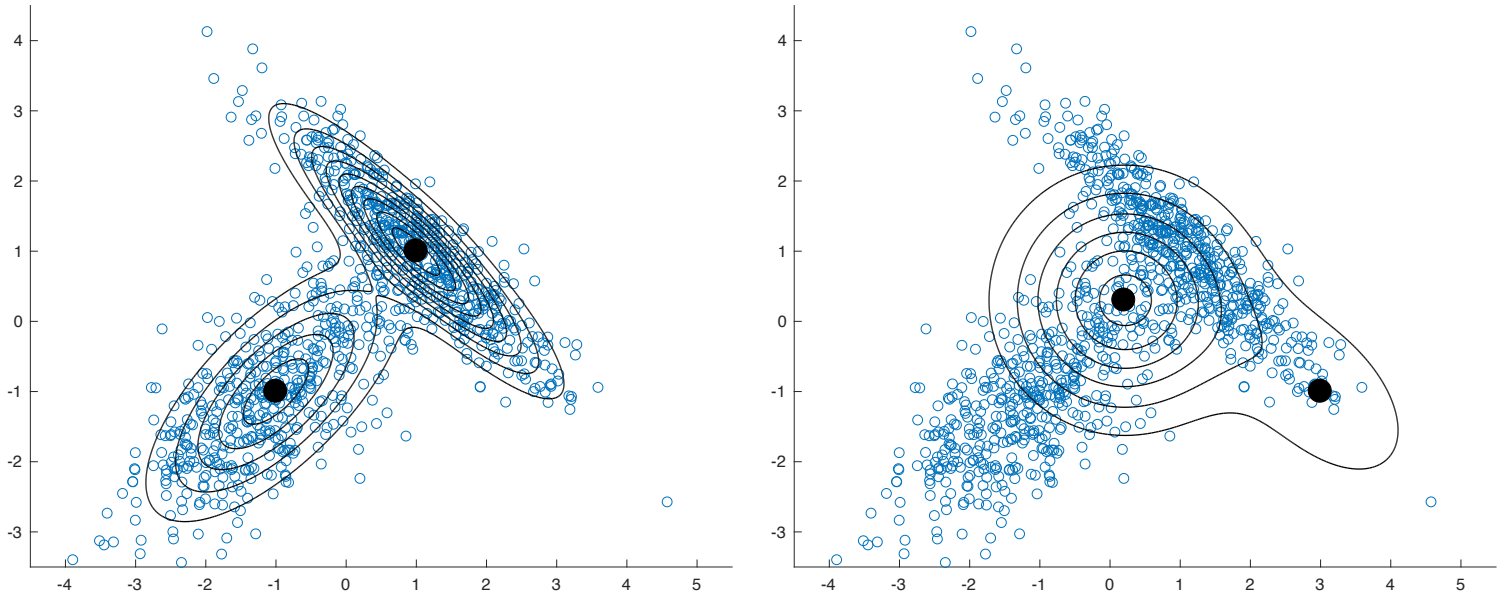




Gaussian Mixture Model

- Maximum Likelihood

- Mixture of Gaussians (MoG) or Gaussian Mixture Model (GMM)
  - A maximum likelihood problem
    - Given the data, find the best fit among the family of prob. distributions with a certain parametric form
- Which one is a better fit?



Gaussian Mixture Model

- Maximum Likelihood

- We know how to solve a maximum likelihood problem:

$$\arg\max_{\boldsymbol{\Theta}} \prod_{i=1}^{N} p(\boldsymbol{x}_i; \boldsymbol{\Theta})$$

- For the GMM case,
  we can break down the likelihood as follows:

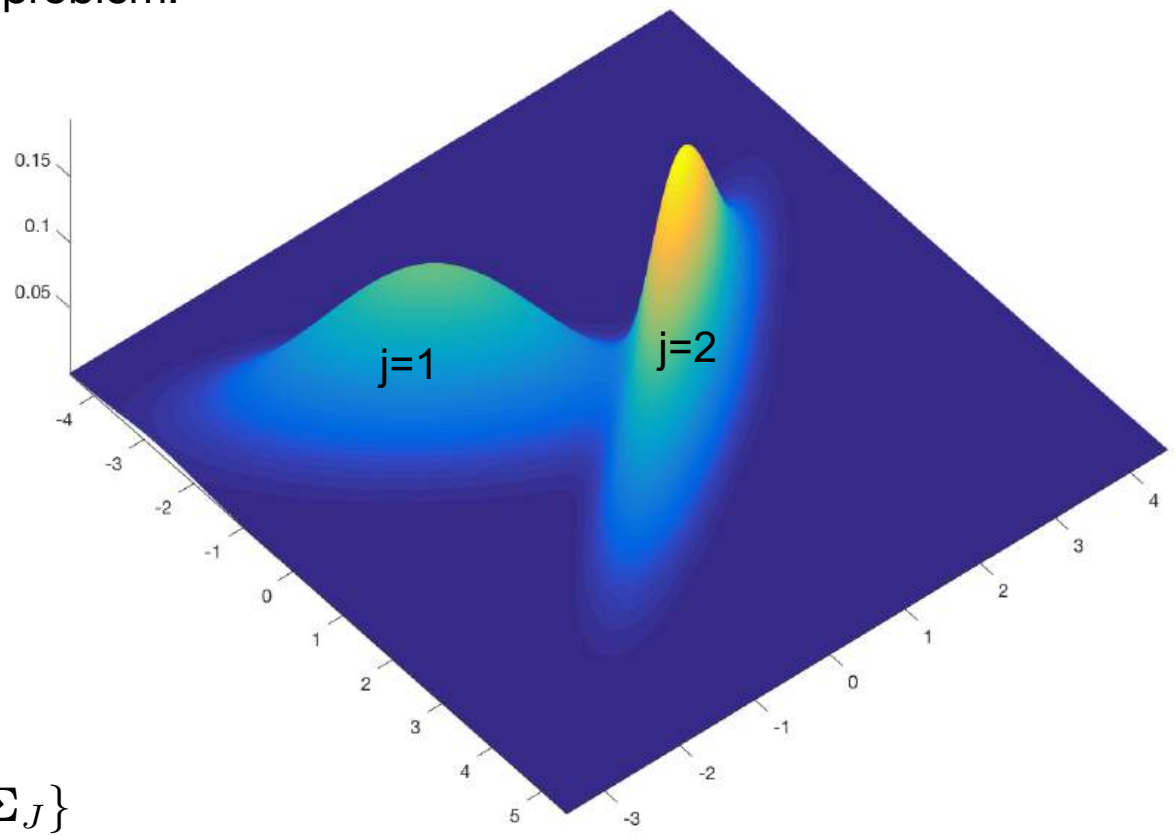
$$\mathcal{L} = \prod_{i=1}^{N} \sum_{j=1}^{J} P_j \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

  - Because,

$$p(\boldsymbol{x}_i; \boldsymbol{\Theta}) = \sum_{j=1}^{J} P_j \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

- Note that:

$$\boldsymbol{\Theta} = \{P_1, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, P_2, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2, \cdots, P_J, \boldsymbol{\mu}_J, \boldsymbol{\Sigma}_J\}$$



Gaussian Mixture Model

- Maximum Likelihood

- We also know the p.d.f. of a Gaussian:

$$\mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_j|^{1/2}} \exp\Big(-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^\top \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_j)\Big)$$

- Therefore, the likelihood is

$$\begin{aligned}\mathcal{L} &= \prod_{i=1}^{N}\sum_{j=1}^{J} P_j \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \\ &= \prod_{i=1}^{N}\sum_{j=1}^{J} P_j \left(\frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_j|^{1/2}} \exp\Big(-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^\top \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_j)\Big)\right)\end{aligned}$$

- Then the log-likelihood is:

$$\begin{aligned}\mathcal{LL} &= \sum_{i=1}^{N} \log\left(\sum_{j=1}^{J} P_j \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\right) \\ &= \sum_{i=1}^{N} \log\left(\sum_{j=1}^{J} P_j \left(\frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_j|^{1/2}} \exp\Big(-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^\top \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_j)\Big)\right)\right)\end{aligned}$$

Gaussian Mixture Model

- Maximum Likelihood

- The objective function:

$$\arg\max_{\boldsymbol{\Theta}} \mathcal{LL} + \lambda\left(\sum_{j=1}^{J} P_j - 1\right)$$

$$\arg\max_{\boldsymbol{\Theta}} \sum_{i=1}^{N} \log\left(\sum_{j=1}^{J} P_j \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\right) + \lambda\left(\sum_{j=1}^{J} P_j - 1\right)$$

- Differentiation is difficult
  - Why?
    - Because of the summation inside the logarithm
- What should we do?
  - Jensen's inequality

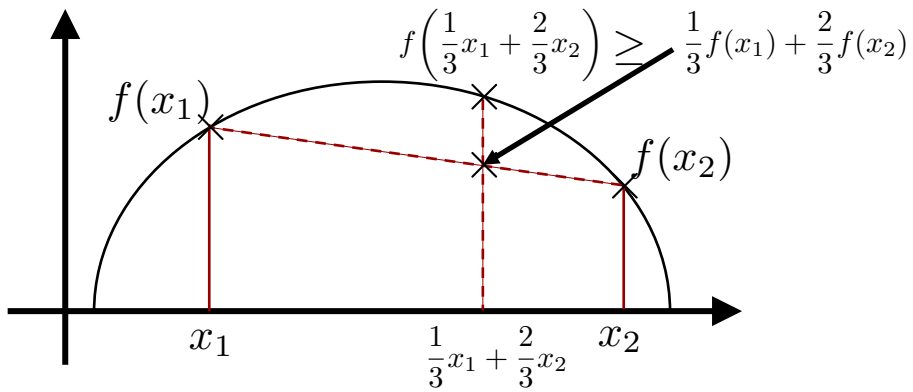
$$\log\left(\sum_{j=1}^{J} \frac{P_j \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \boldsymbol{U}_{ij}}{\boldsymbol{U}_{ij}}\right) \geq \sum_{j=1}^{J} \boldsymbol{U}_{ij} \log\left(\frac{P_j \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\boldsymbol{U}_{ij}}\right)$$

  - What? Why?

Gaussian Mixture Model

- Jensen's Inequality

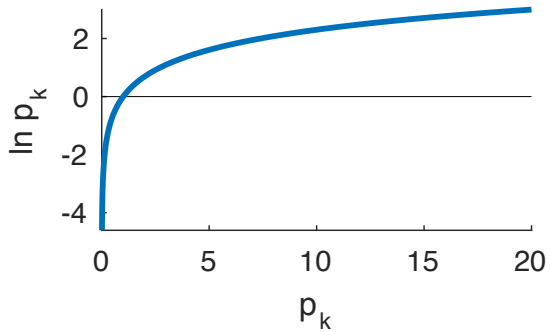
- For example



- For a concave function f: $f\left(\frac{\sum a_i x_i}{\sum a_i}\right) \geq \frac{\sum a_i f(x_i)}{\sum a_i}$
  - Or: $f\left(\sum a_i x_i\right) \geq \sum a_i f(x_i) \quad \text{if } \sum a_i = 1 \text{ and } a_i \geq 0$
- Logarithmic functions are concave
  - Why?

$$\log'(x) = \frac{1}{x}$$

$$\log''(x) = -\frac{1}{x^2}$$



Gaussian Mixture Model

- Expectation Maximization (EM)

- Let's get back to the ML problem for GMM

$$\mathcal{LL} = \sum_{i=1}^{N} \log \left(\sum_{j=1}^{J} \frac{P_j \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \boldsymbol{U}_{ij}}{\boldsymbol{U}_{ij}} \right) \qquad \sum_j \boldsymbol{U}_{ij} = 1 \text{ and } \boldsymbol{U}_{ij} \geq 0$$

$$\geq \sum_{i=1}^{N} \sum_{j=1}^{J} \boldsymbol{U}_{ij} \log \left(\frac{P_j \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\boldsymbol{U}_{ij}} \right) \qquad \text{Jensen's inequality}$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{J} \boldsymbol{U}_{ij} \log \left(\frac{p(j|\boldsymbol{x}_i) p(\boldsymbol{x}_i)}{\boldsymbol{U}_{ij}} \right) \qquad \because p(j|\boldsymbol{x}_i) = \frac{p(\boldsymbol{x}_i|j)p(j)}{\sum_j p(\boldsymbol{x}_i|j)p(j)} = \frac{\mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) P_j}{p(\boldsymbol{x}_i)}$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{J} \boldsymbol{U}_{ij} \log \left(\frac{p(j|\boldsymbol{x}_i)}{\boldsymbol{U}_{ij}} \right) + \sum_{j=1}^{J} \boldsymbol{U}_{ij} \log p(\boldsymbol{x}_i)$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{J} \boldsymbol{U}_{ij} \log \left(\frac{p(j|\boldsymbol{x}_i)}{\boldsymbol{U}_{ij}} \right) + \log p(\boldsymbol{x}_i) \; = \sum_{i=1}^{N} -\mathcal{D}_{KL}\left(\boldsymbol{U}_{ij} \middle|\middle| p(j|\boldsymbol{x}_i)\right) + \log p(\boldsymbol{x}_i)$$

- What if we fix all the other parameters and maximize $\mathcal{LL}$ w.r.t. $\boldsymbol{U}_{ij}$?
  - When the KL divergence is minimal: $\boldsymbol{U}_{ij} = p(j|\boldsymbol{x}_i)$
- This procedure is the **E-step** of the EM algorithm for GMM

Gaussian Mixture Model

- Expectation Maximization (EM)

$$\mathcal{N}(\boldsymbol{x}_i;\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)=\frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_j|^{1/2}}\exp\Big(-\frac{1}{2}(\boldsymbol{x}_i-\boldsymbol{\mu}_j)^\top\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu}_j)\Big)$$

- M-step
  - We find $\boldsymbol{\Theta}$ that maximizes $\mathcal{LL}+\lambda\big(\sum_{j=1}^{J}P_j-1\big)$

$$\mathcal{LL}\geq\sum_{i=1}^{N}\sum_{j=1}^{J}\boldsymbol{U}_{ij}\log\left(\frac{P_j\mathcal{N}(\boldsymbol{x}_i;\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)}{\boldsymbol{U}_{ij}}\right)$$

$$=\sum_{i=1}^{N}\sum_{j=1}^{J}\boldsymbol{U}_{ij}\log\big(P_j\mathcal{N}(\boldsymbol{x}_i;\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)\big)-\sum_{i=1}^{N}\sum_{j=1}^{J}\boldsymbol{U}_{ij}\log\boldsymbol{U}_{ij}$$

constant

  - Therefore the final objective function for the M-step is

$$\arg\max_{\boldsymbol{\Theta}}\sum_{i=1}^{N}\sum_{j=1}^{J}\boldsymbol{U}_{ij}\log\big(P_j\mathcal{N}(\boldsymbol{x}_i;\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)\big)+\lambda\Big(\sum_j P_j-1\Big)$$

  - More specifically (since we're going to do the partial differentiation)

$$\arg\max_{\boldsymbol{\mu}_j}\mathcal{J}_{\boldsymbol{\mu}_j},\quad\mathcal{J}_{\boldsymbol{\mu}_j}=\sum_{i=1}^{N}-\frac{1}{2}\boldsymbol{U}_{ij}(\boldsymbol{x}_i-\boldsymbol{\mu}_j)^\top\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu}_j)+const.$$

$$\arg\max_{P_j}\mathcal{J}_{P_j},\quad\mathcal{J}_{P_j}=\sum_{i=1}^{N}\boldsymbol{U}_{ij}\log P_j+\lambda(\sum_j P_j-1)+const.$$

$$\arg\max_{\boldsymbol{\Sigma}_j}\mathcal{J}_{\boldsymbol{\Sigma}_j},\quad\mathcal{J}_{\boldsymbol{\Sigma}_j}=\sum_{i=1}^{N}\boldsymbol{U}_{ij}\Big(-\frac{1}{2}\log|\boldsymbol{\Sigma}_j|-\frac{1}{2}(\boldsymbol{x}_i-\boldsymbol{\mu}_j)^\top\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu}_j)\Big)+const.$$

Gaussian Mixture Model

- Expectation Maximization (EM)

- M-step
  - Partial differentiation w.r.t. the parameters and find the local maxima
    - For the means:

$$\frac{\partial \mathcal{J}_{\boldsymbol{\mu}_j}}{\partial \boldsymbol{\mu}_j} = \sum_{i=1}^{N} -\boldsymbol{U}_{ij}(\boldsymbol{x}_i - \boldsymbol{\mu}_j) = 0, \quad \boldsymbol{\mu}_j = \frac{\sum_{i=1}^{N} \boldsymbol{U}_{ij}\boldsymbol{x}_i}{\sum_{i=1}^{N} \boldsymbol{U}_{ij}}$$

$$\arg\max_{\boldsymbol{\mu}_j} \mathcal{J}_{\boldsymbol{\mu}_j}, \quad \mathcal{J}_{\boldsymbol{\mu}_j} = \sum_{i=1}^{N} -\frac{1}{2}\boldsymbol{U}_{ij}(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^\top \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_j) + const.$$

    - For the priors:

$$\frac{\partial \mathcal{J}_{P_j}}{\partial \lambda} = (\sum_j P_j - 1) = 0 \qquad \frac{\partial \mathcal{J}_{P_j}}{\partial P_j} = \frac{\sum_{i=1}^{N} \boldsymbol{U}_{ij}}{P_j} + \lambda = 0$$

$$\Leftrightarrow \sum_{j=1}^{J}\sum_{i=1}^{N} \boldsymbol{U}_{ij} = -\lambda \sum_{j=1}^{J} P_j$$

$$\Leftrightarrow \lambda = -\sum_{j=1}^{J}\sum_{i=1}^{N} \boldsymbol{U}_{ij} = -N$$

$$\Leftrightarrow P_j = \frac{\sum_{i=1}^{N} \boldsymbol{U}_{ij}}{N}$$

$$\arg\max_{P_j} \mathcal{J}_{P_j}, \quad \mathcal{J}_{P_j} = \sum_{i=1}^{N} \boldsymbol{U}_{ij} \log P_j + \lambda(\sum_j P_j - 1) + const.$$

    - For the covariance (see matrixcookbook 2.1.2 and 2.2):

$$\boldsymbol{\Sigma}_j = \frac{\sum_i \boldsymbol{U}_{ij}(\boldsymbol{x}_i - \boldsymbol{\mu}_j)(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^\top}{\sum_i \boldsymbol{U}_{ij}}$$

Gaussian Mixture Model

- Expectation Maximization (EM)

- E-step: calculate posterior probabilities

$$\boldsymbol{U}_{ij} = p(j|\boldsymbol{x}_i) = \frac{P_j \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_j P_j \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

- M-step: update parameters

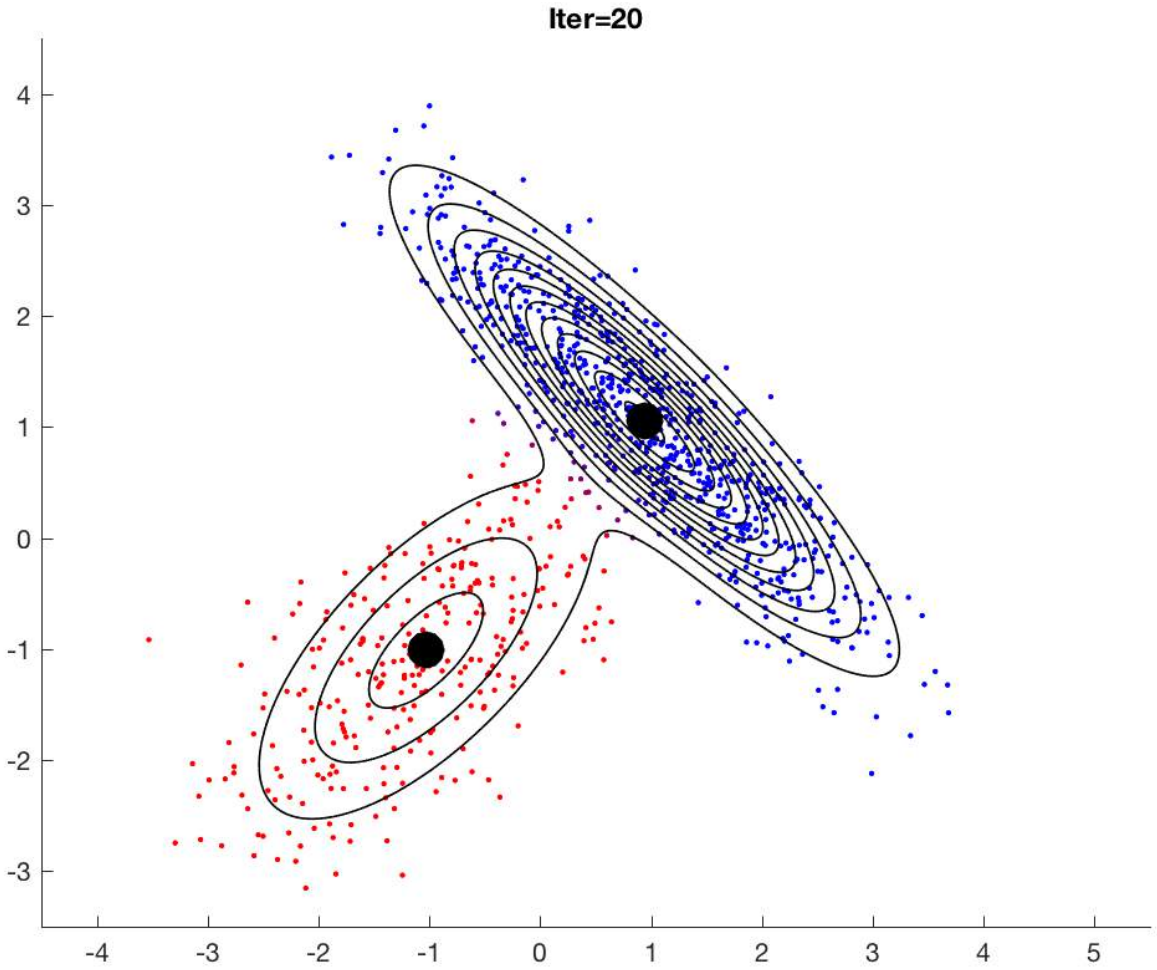
$$\boldsymbol{\mu}_j = \frac{\sum_{i=1}^{N} \boldsymbol{U}_{ij} \boldsymbol{x}_i}{\sum_{i=1}^{N} \boldsymbol{U}_{ij}}$$

$$P_j = \frac{\sum_{i=1}^{N} \boldsymbol{U}_{ij}}{N}$$

$$\boldsymbol{\Sigma}_j = \frac{\sum_i \boldsymbol{U}_{ij} (\boldsymbol{x}_i - \boldsymbol{\mu}_j)(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^\top}{\sum_i \boldsymbol{U}_{ij}}$$

Gaussian Mixture Model

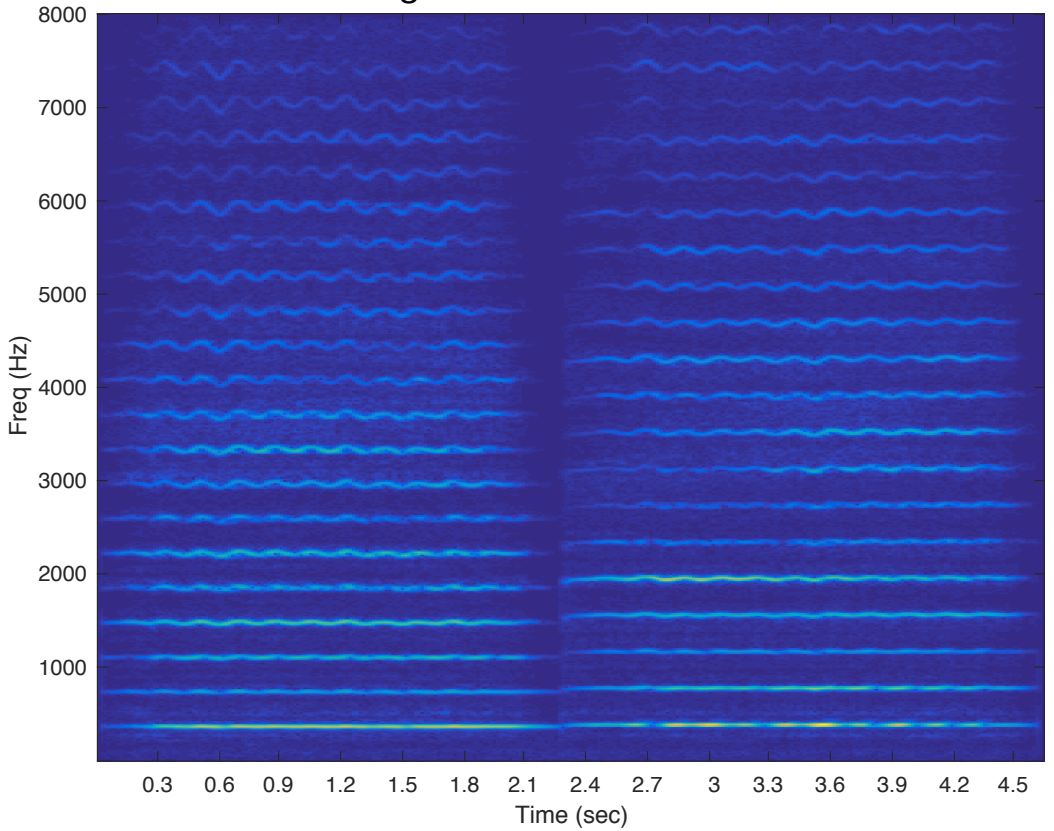
- Too much math?



Mixture of Multinomial Distributions

- Clustering Musical Notes

- How many clusters? (samples are magnitude spectra)
  - I'm curious what kind of notes are there in the signal



Mixture of Multinomial Distributions

- Clustering Musical Notes

- For i-th spectrum $\boldsymbol{x}_i$:
  - Although the magnitudes are real numbers we can scale them to convert them into integers:

$$[1.2, 1.35, 0.1, 5.525] = 0.025 \times [48, 54, 4, 221]$$

  - Then, we can think of this as an observation from a multinomial dist.

$$\mathcal{M}(\boldsymbol{x}_i; \boldsymbol{\theta}) = \frac{N!}{\prod_d x_{id}!} \prod_d \theta_d^{x_{id}}$$

- EM for mixture of multinomial distributions
  - Initialize two mean spectra (random numbers) $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}_+^D$
  - Initialize two prior prob (random numbers that sum to one) $P_1 + P_2 = 1$
  - Calculate posterior prob (E-step)

$$p(j = 1 | \boldsymbol{x}_i) = \frac{P_1 \mathcal{M}(\boldsymbol{x}_i; \boldsymbol{\theta}_1)}{P_1 \mathcal{M}(\boldsymbol{x}_i; \boldsymbol{\theta}_1) + P_2 \mathcal{M}(\boldsymbol{x}_i; \boldsymbol{\theta}_2)}$$

It's usually fine not to convert the spectra into integers, although it's not strictly correct. In this example I just normalized each spectrum.

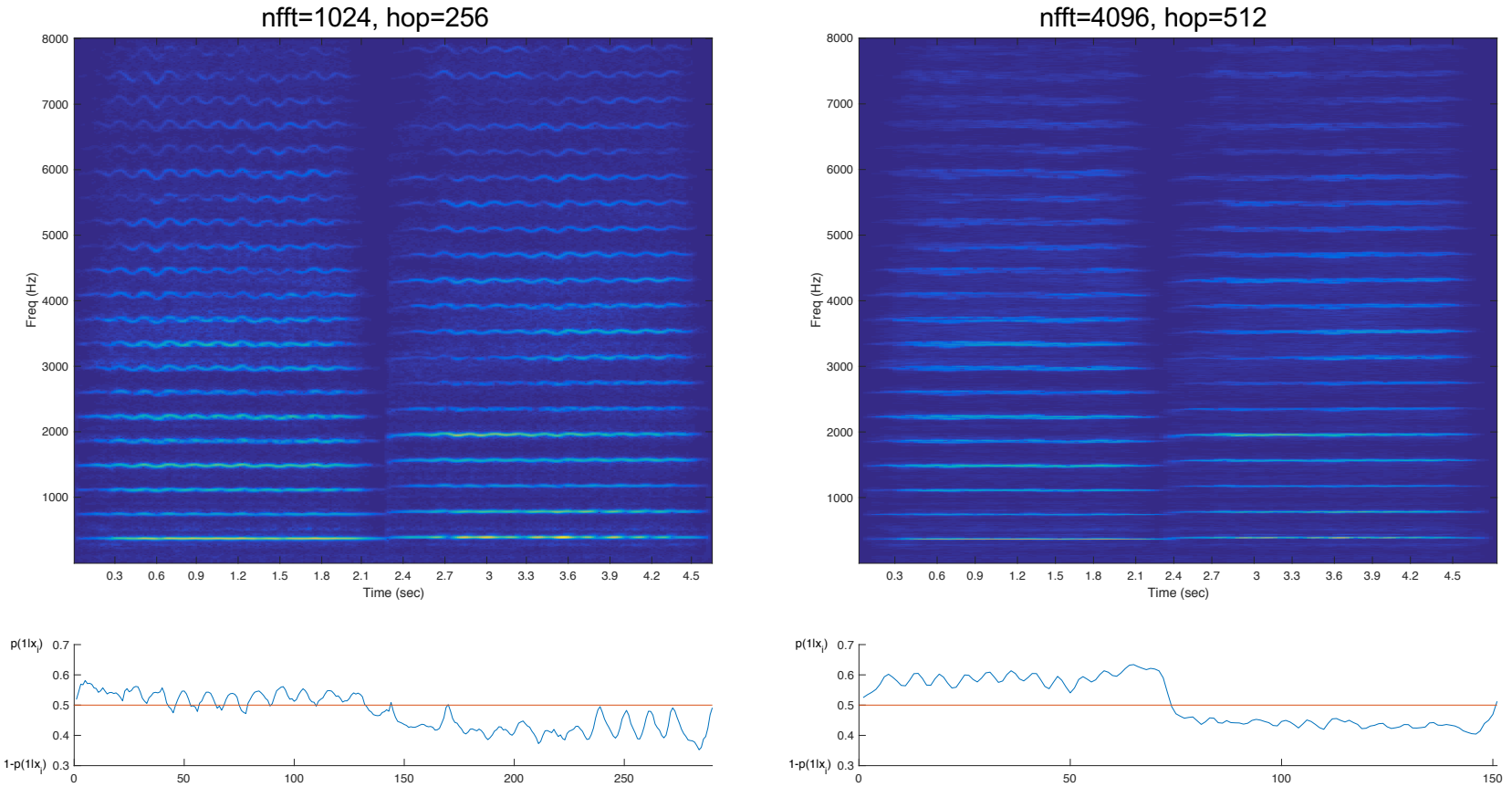
$$= \frac{P_1 \frac{N!}{\prod_d x_{id}!} \prod_d \theta_{1d}^{x_{id}}}{P_1 \frac{N!}{\prod_d x_{id}!} \prod_d \theta_{1d}^{x_{id}} + P_2 \frac{N!}{\prod_d x_{id}!} \prod_d \theta_{2d}^{x_{id}}} = \frac{P_1 \prod_d \theta_{1d}^{x_{id}}}{P_1 \prod_d \theta_{1d}^{x_{id}} + P_2 \prod_d \theta_{2d}^{x_{id}}}$$

  - Update means (M-step)

Mixture of Multinomial Distributions

- Clustering Musical Notes

- It's actually a difficult clustering task with a lot of spurious local minima



Locality Sensitive Hashing

- From clustering to hashing

- Hashing is a popular concept in database
  - There is a query to the database
  - Instead of comparing the original representations, a hash function maps the query down to an integer (binary) address
  - The address is associated with a **bucket**. It can contain a few different database records
    - We say that those records collide
  - Then, we refine the search inside the bucket
  - This is cheaper than seeing the entire database
- Traditional challenges
  - Records are better off evenly distributed (for the speed)
  - Overflow

Locality Sensitive Hashing

- From clustering to hashing

- There's another hashing concept in machine learning
  - Locality sensitive hashing or semantic hashing
- For the data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in a D-dimensional space
  - If they are close enough $\mathcal{D}(\boldsymbol{x}_i||\boldsymbol{x}_j) < \tau$
  - Then, the Hamming distance between them after hashing is zero $\mathcal{H}\big(\phi(\boldsymbol{x}_i)||\phi(\boldsymbol{x}_j)\big) = 0$ with probability p
  - If they are far enough $\mathcal{D}(\boldsymbol{x}_i||\boldsymbol{x}_j) \geq c\tau$
  - Then, the Hamming distance between them after hashing is zero $\mathcal{H}\big(\phi(\boldsymbol{x}_i)||\phi(\boldsymbol{x}_j)\big) = 0$ with probability q
  - $p > q$!
- The hash function $\phi(\boldsymbol{x}_i)$ that meets above conditions are said to be in the locality sensitive hash function family
- In other words…
  - Originally similar items collide in the same bucket
  - Share the same address
  - Quantized using the same binary string
  - Are in the same cluster

Locality Sensitive Hashing

- From clustering to hashing

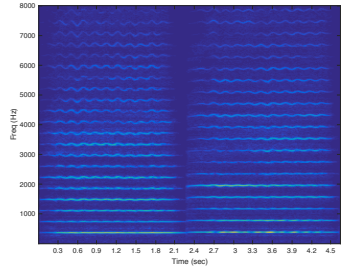
- How do we find the hash function?
  - Well, it's not easy
- One way is to rely on a bunch of random projections

$$\mathrm{K}\left\{\begin{bmatrix} +1 \\ -1 \\ \vdots \\ -1 \end{bmatrix}\right. = \mathrm{sign}\left(\begin{bmatrix} -1 & +1 & \cdots & -1 \\ +1 & -1 & \cdots & +1 \\ \vdots & \vdots & \ddots & \vdots \\ +1 & +1 & \cdots & -1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix}\right)$$

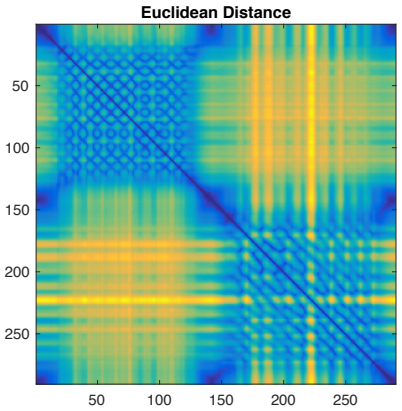
  - With *K* projections, we can represent *D* dimensional data with *K* bits
    - e.g. 513 magnitude Fourier coefficients (real) with *K*=128 bits
- Why does it work?
  - Johnson-Lindenstrauss Theorem
- It's also related to sparse coding and compressive sensing

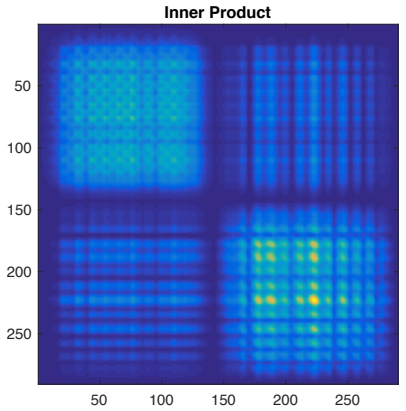
Locality Sensitive Hashing

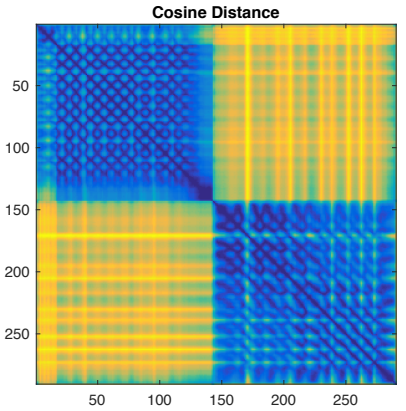
- From clustering to hashing



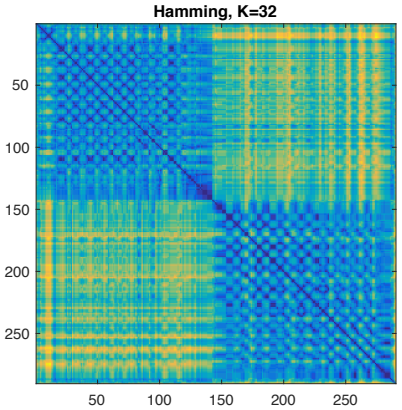
- Let's see how the original spectra are similar to each other

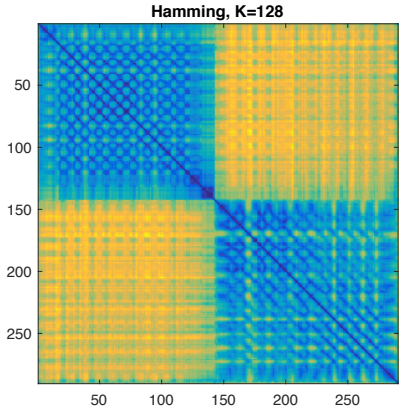


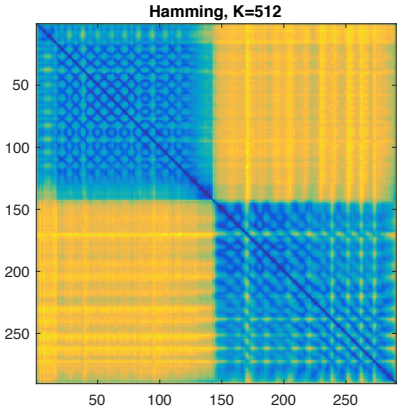




- And in terms of Hamming distance after random projection







Spectral Hashing

- More machine learning involved

- Just another hashing technique, but it tries to minimize the difference between

Original pairwise similarity

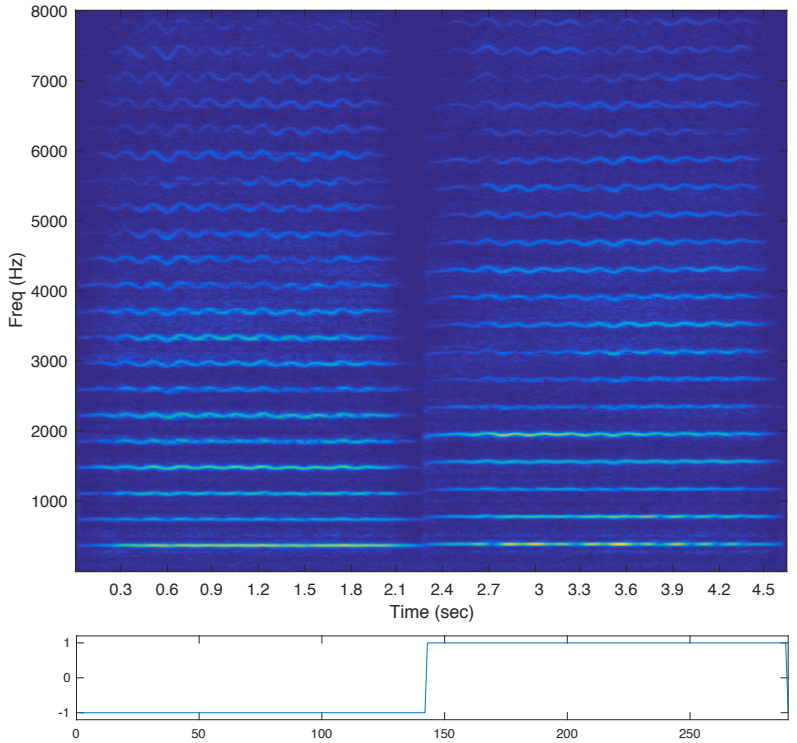
Pairwise Hamming distance

$$\min \sum_{ij} \boldsymbol{W}_{ij} \mathcal{H}(\boldsymbol{y}_i || \boldsymbol{y}_j)$$

$$\text{subject to: } \boldsymbol{y}_i \in \{-1, +1\}^K$$

$$\sum_i \boldsymbol{y}_i = \mathbf{0}$$

$$\boldsymbol{y}_i^\top \boldsymbol{y}_j = 0 \quad \text{if } i \neq j$$



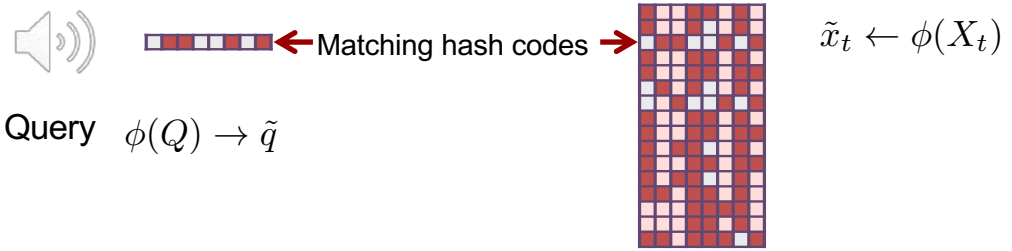
- See the paper for the more optimization detail
  - But, the basic idea is to see the problem as an eigendecomposition problem
  - That's why it's called **spectral** hashing

Weiss, Yair, Antonio Torralba, and Rob Fergus. "Spectral hashing." *Advances in neural information processing systems*. 2009.

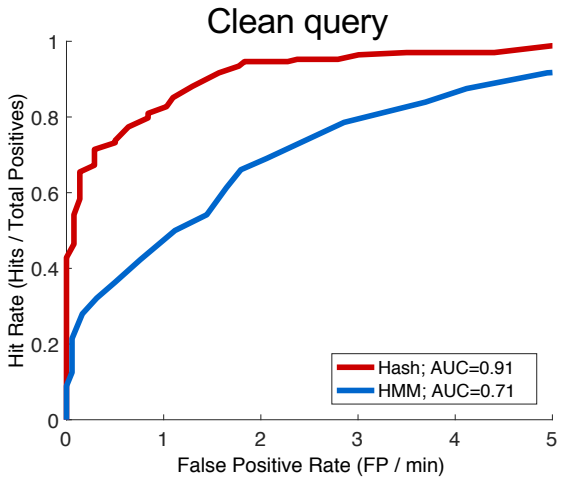
Locality Sensitive Hashing

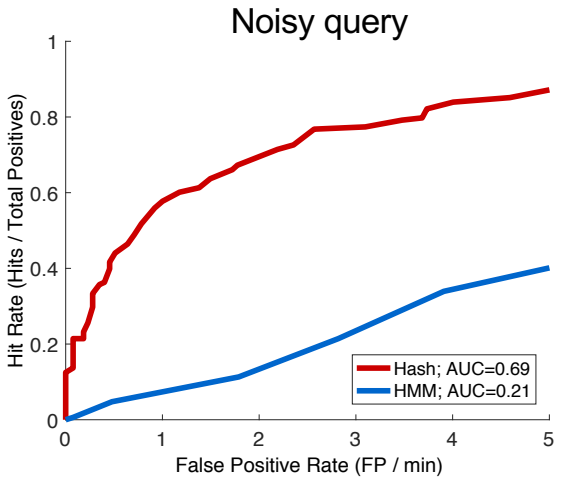
- Why is it useful?

- For faster detection



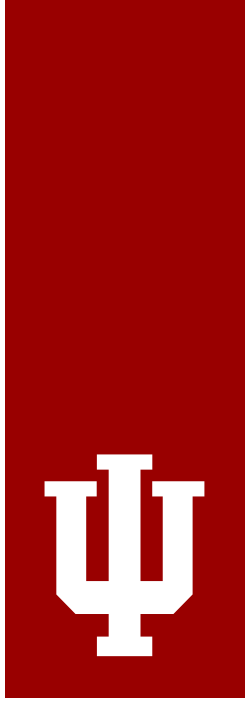
- Keyword spotting demo HMM versus Spatial-Temporal WTA Hashing





Reading

- Textbook 6.8 – 6.16
- Textbook 2.5.5
- Bishop, “Pattern Recognition and Machine Learning” Chapter 9



Thank You!