Clustering, K-Means, EM Tutorial

Kamyar Ghasemipour

Parts taken from Shikhar Sharma, Wenjie Luo, and Boris Ivanovic's tutorial slides, as well as lecture notes

Organization:

- Clustering
  - Motivation
- K-Means
  - Review & Demo
- Gaussian Mixture Models
  - Review
- ~~EM Algorithm (time permitting)~~
  - ~~Free Energy Justification~~

Clustering

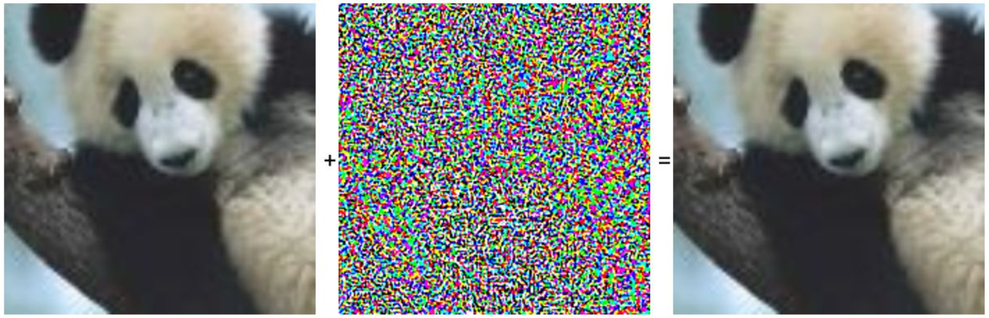
Clustering: Motivation

- Important assumption we make when doing any form of learning:

  “Similar data-points have similar behaviour”

- Eg. In the context of supervised learning

  “Similar inputs should lead to similar predictions”*



Original image classified as a panda with 60% confidence. | Tiny adversarial perturbation. | Imperceptibly modified image, classified as a gibbon with 99% confidence.

*sometimes our trained models don’t follow these assumption (cf. literature on adversarial examples)

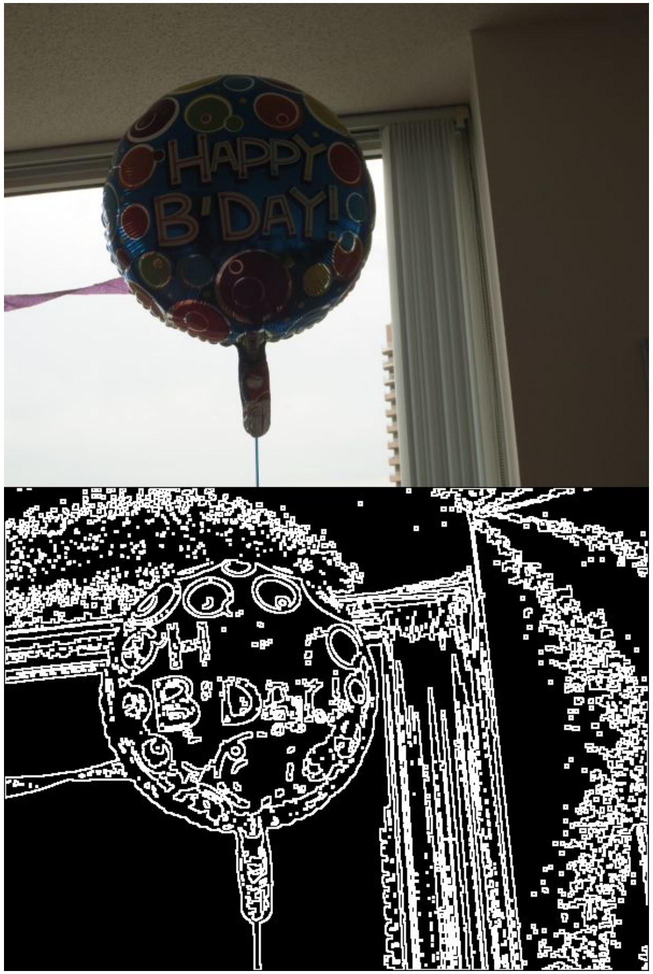
Clustering: Examples

- Discretizing colours for compression using a codebook

Figure from Bishop

Clustering: Examples

- Doing a very basic form of boundary detection
  - Discretize colours
  - Draw boundaries between colour groups

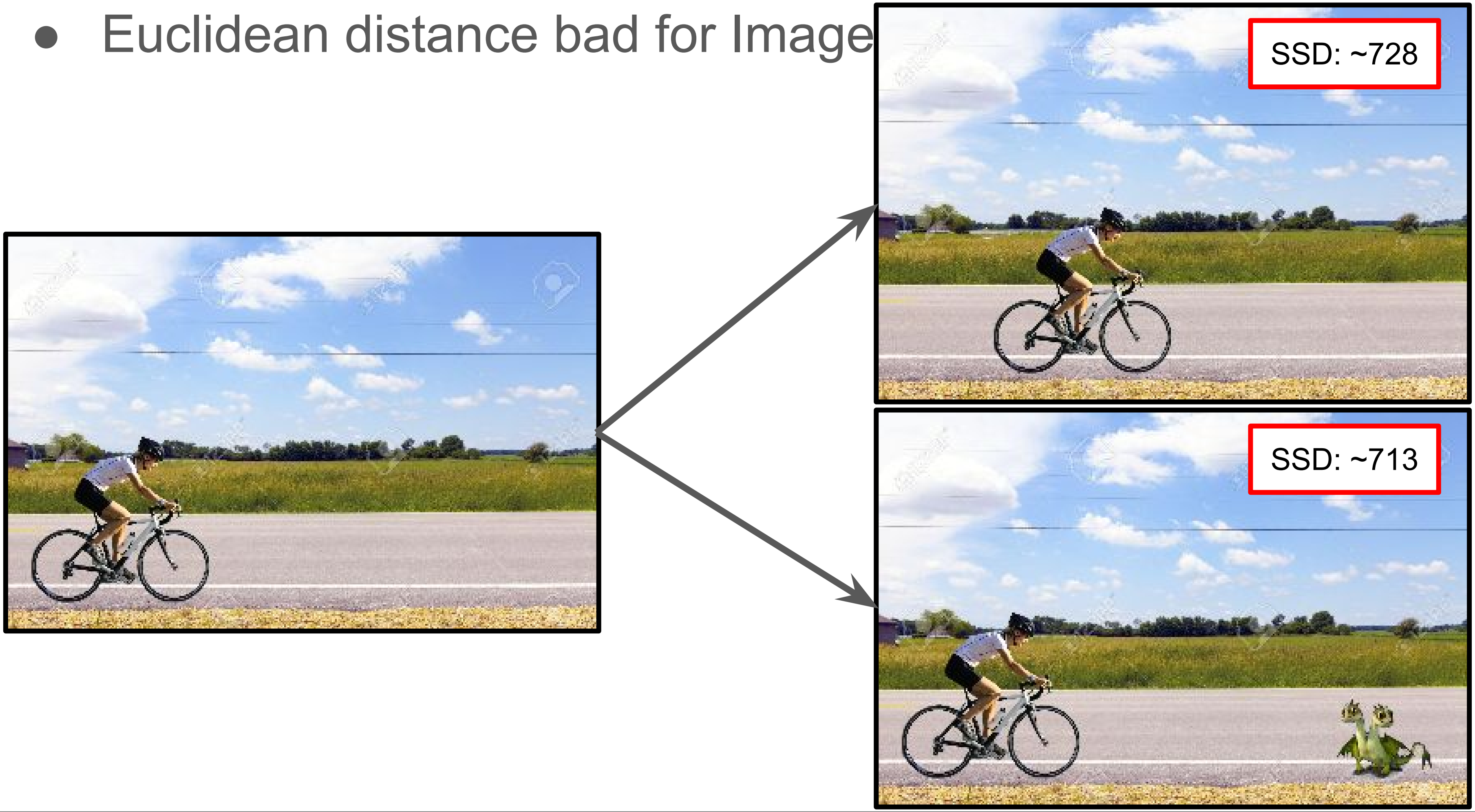


Clustering: Examples

- Like all unsupervised learning algorithms, clustering can be incorporated into the pipeline for training a supervised model
- We will go over an example of this very soon

Clustering: Challenges

- What is a good notion of “similarity”?
- Euclidean distance bad for Image



Clustering: Challenges

- The notion of similarity used can make the same algorithm behave in very different ways and can in some cases be a motivation for developing new algorithms (not necessarily just for clustering algorithms)
- Another question is how to compare different clustering algorithms
  - May have specific methods for making these decisions based on the clustering algorithms used
  - Can also use performance on down-the-line tasks as a proxy when choosing between different setups

Clustering: Some Specific Algorithms

- Today we shall review:
  - K-Means
  - Gaussian Mixture Models
- Hopefully there will be some time to go over EM as well

K-Means

K-Means: The Algorithm

1. Initialize K centroids
2. Iterate until convergence
   a. Assign each data-point to it’s closest centroid
   b. Move each centroid to the center of data-points assigned to it

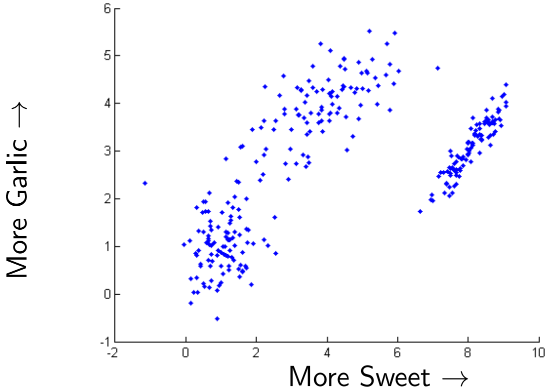
K-Means: A look at how it can be used

<< Slides from TA’s past >>

Tomato sauce

- A major tomato sauce company wants to tailor their brands to sauces to suit their customers
- They run a market survey where the test subject rates different sauces
- After some processing they get the following data
- Each point represents the preferred sauce characteristics of a specific person

Tomato sauce data



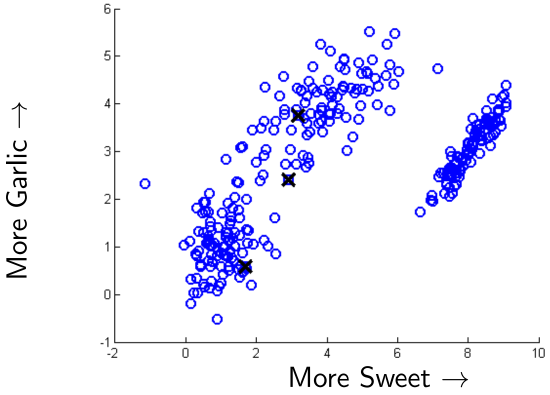
This tells us how much different customers like different flavors

Some natural questions

- How many different sauces should the company make?
- How sweet/garlicy should these sauces be?
- Idea: We will segment the consumers into groups (in this case 3), we will then find the best sauce for each group

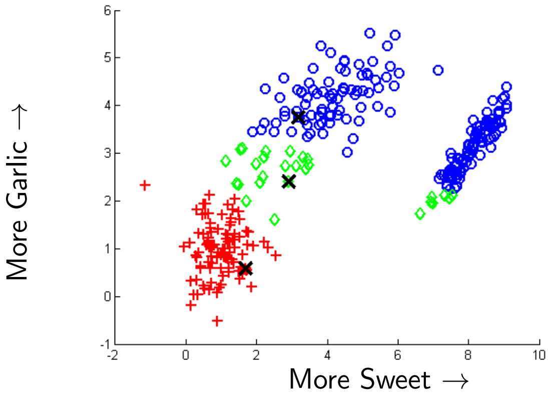
Approaching k-means

- Say I give you 3 sauces whose garlicy-ness and sweetness are marked by X

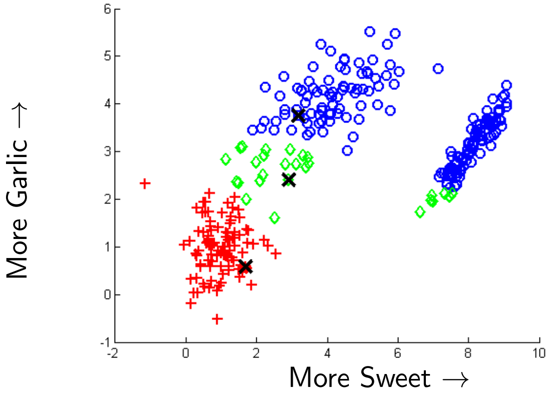


Approaching k-means

- We will group each customer by the sauce that most closely matches their taste

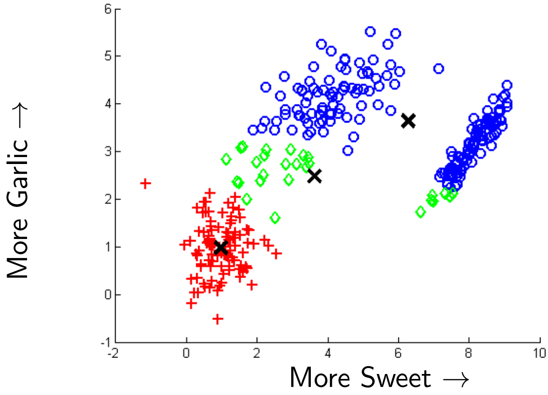


- Given this grouping, can we choose sauces that would make each group happier on average?



Approaching k-means

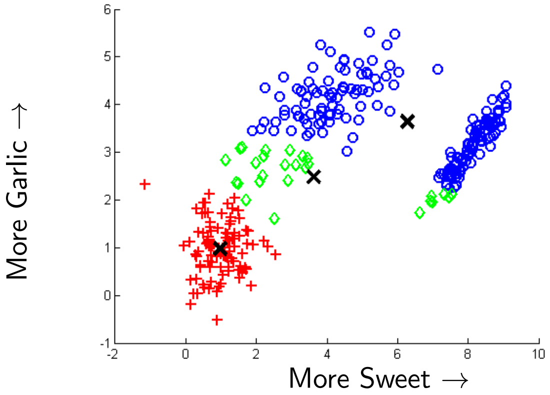
- Given this grouping, can we choose sauces that would make each group happier on average?



Yes !

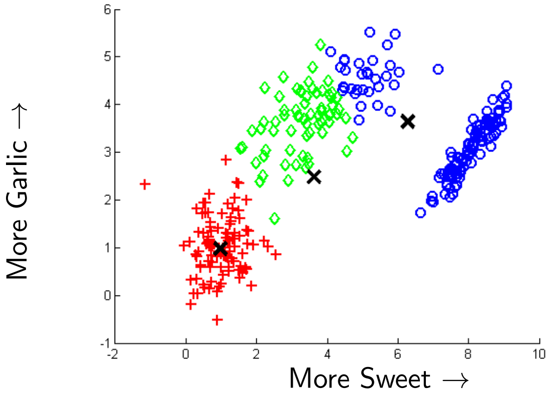
Approaching k-means

- Given these new sauces, we can regroup the customers



Approaching k-means

- Given these new sauces, we can regroup the customers



K-Means: Challenges

- How to initialize?
  - ~~You saw k-means++ in lecture slides~~
  - Can come up with other heuristics
- How do you choose K?
  - You may come up with criteria for the value of K based on:
    - Restrictions on the magnitude of K
      - Everyone can't have their own tomato sauce
    - Performance on some down-the-line task
      - If used for doing supervised learning later, must choose K such that you do not under/over fit

K-Means: Challenges

- K-Means algorithm converges to a local minimum:
  - Can try multiple random restarts
  - Other heuristics such as splitting discussed in lecture

- **Questions about K-Means?**

Gaussian Mixture Models

Generative Models

- One important class of methods in machine learning
- The goal is to define some parametric family of probability distributions and then maximize the likelihood of your data under this distribution by finding the best parameters

Gaussian Mixture Model (GMM)

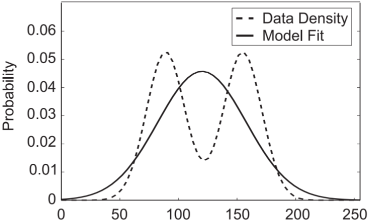
What is $p(\mathbf{x})$?

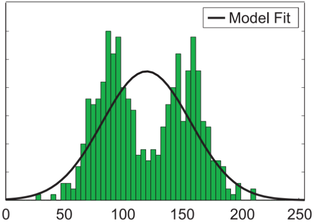
$$p(\mathbf{x}) = \sum_{k=1}^{K} p(z = k)p(\mathbf{x}|z = k) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \mathbf{I})$$

- This distribution is an example of a Gaussian Mixture Model (GMM), and π_k are known as the mixing coefficients
- In general, we would have different covariance for each cluster, i.e., $p(\mathbf{x} \,|\, z = k) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. For this lecture, we assume $\boldsymbol{\Sigma}_k = \mathbf{I}$ for simplicity.
- If we allow arbitrary covariance matrices, GMMs are **universal approximators of densities** (if you have enough Gaussians). Even diagonal GMMs are universal approximators.

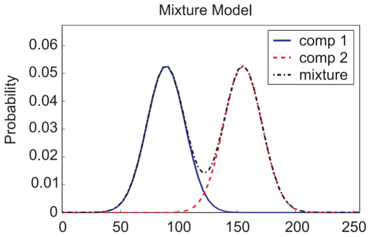
Visualizing a Mixture of Gaussians – 1D Gaussians

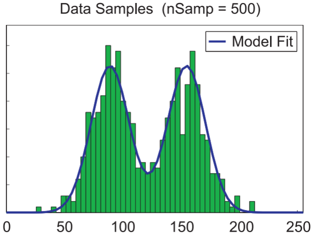
- If you fit one Gaussian distribution to data:





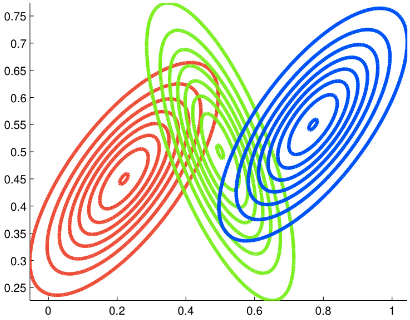
- Now, we are trying to fit a GMM with $K = 2$:

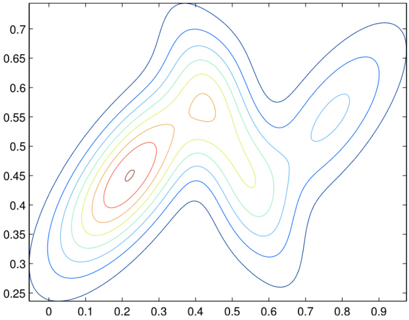


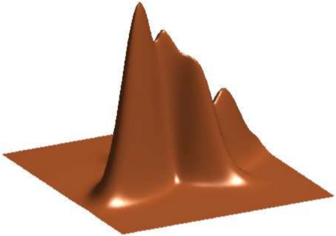


[Slide credit: K. Kutulakos]

Visualizing a Mixture of Gaussians – 2D Gaussians







Fitting GMMs: Maximum Likelihood

Maximum likelihood objective:

$$\log p(\mathcal{D}) = \sum_{n=1}^{N} \log p(\mathbf{x}^{(n)}) = \sum_{n=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}^{(n)} | \boldsymbol{\mu}_k, \mathbf{I}) \right)$$

- How would you optimize this w.r.t. parameters $\{\pi_k, \boldsymbol{\mu}_k\}$?
  - No closed-form solution when we set derivatives to 0
  - Difficult because sum inside the log
- One option: gradient ascent. Can we do better?
- Can we have a closed-form update?

Maximum Likelihood

- **Observation: if we knew** $z^{(n)}$ for every $\mathbf{x}^{(n)}$, (i.e. our dataset was $\mathcal{D}_{\text{complete}} = \{(z^{(n)}, \mathbf{x}^{(n)})\}_{n=1}^{N}$) the maximum likelihood problem is easy:

$$\begin{aligned}
\log p(\mathcal{D}_{\text{complete}}) &= \sum_{n=1}^{N} \log p(z^{(n)}, \mathbf{x}^{(n)}) \\
&= \sum_{n=1}^{N} \log p(\mathbf{x}^{(n)} | z^{(n)}) + \log p(z^{(n)}) \\
&= \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{I}\{z^{(n)} = k\} \left(\log \mathcal{N}(\mathbf{x}^{(n)} | \boldsymbol{\mu}_k, \mathbf{I}) + \log \pi_k \right)
\end{aligned}$$

Maximum Likelihood

$$\log p(\mathcal{D}_{\text{complete}}) = \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{I}\{z^{(n)} = k\} \left(\log \mathcal{N}(\mathbf{x}^{(n)} | \boldsymbol{\mu}_k, \mathbf{I}) + \log \pi_k \right)$$

- We have been optimizing something similar for Naive bayes classifiers
- By maximizing $\log p(\mathcal{D}_{\text{complete}})$, we would get this:

$$\begin{aligned} \hat{\boldsymbol{\mu}}_k &= \frac{\sum_{n=1}^{N} \mathbb{I}\{z^{(n)} = k\}\, \mathbf{x}^{(n)}}{\sum_{n=1}^{N} \mathbb{I}\{z^{(n)} = k\}} = \text{class means} \\ \hat{\pi}_k &= \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}\{z^{(n)} = k\} = \text{class proportions} \end{aligned}$$

Maximum Likelihood

- We haven't observed the cluster assignments $z^{(n)}$, but we can compute $p(z^{(n)}|\mathbf{x}^{(n)})$ using Bayes rule
- Conditional probability (using Bayes rule) of z given $\mathbf{x}$

$$
\begin{aligned}
p(z=k|\mathbf{x}) &= \frac{p(z=k)p(\mathbf{x}|z=k)}{p(\mathbf{x})} \\
&= \frac{p(z=k)p(\mathbf{x}|z=k)}{\sum_{j=1}^{K} p(z=j)p(\mathbf{x}|z=j)} \\
&= \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \mathbf{I})}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \mathbf{I})}
\end{aligned}
$$

Maximum Likelihood

$$\log p(\mathcal{D}_{\text{complete}}) = \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{I}\{z^{(n)} = k\}(\log \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \mathbf{I}) + \log \pi_k)$$

- We don't know the cluster assignments $\mathbb{I}\{z^{(n)} = k\}$ (they are our latent variables), but we know their expectation $\mathbb{E}[\mathbb{I}\{z^{(n)} = k\} \mid \mathbf{x}^{(n)}] = p(z^{(n)} = k|\mathbf{x}^{(n)})$.
- If we plug in $r_k^{(n)} = p(z^{(n)} = k|\mathbf{x}^{(n)})$ for $\mathbb{I}\{z^{(n)} = k\}$, we get:

$$\sum_{n=1}^{N} \sum_{k=1}^{K} r_k^{(n)} (\log \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \mathbf{I}) + \log \pi_k)$$

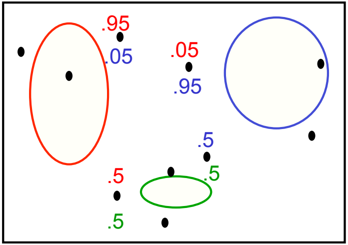
- This is still easy to optimize! Solution is similar to what we have seen:

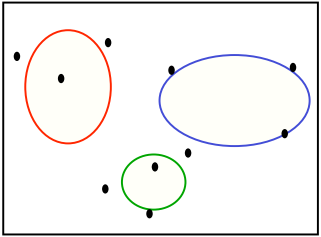
$$\hat{\boldsymbol{\mu}}_k = \frac{\sum_{n=1}^{N} r_k^{(n)} \mathbf{x}^{(n)}}{\sum_{n=1}^{N} r_k^{(n)}} \qquad \hat{\pi}_k = \frac{\sum_{n=1}^{N} r_k^{(n)}}{N}$$

- Note: this only works if we treat $r_k^{(n)} = \frac{\pi_k \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \mathbf{I})}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_j, \mathbf{I})}$ as fixed.

How Can We Fit a Mixture of Gaussians?

- This motivates the Expectation-Maximization algorithm, which alternates between two steps:
  1. E-step: Compute the posterior probabilities $r_k^{(n)} = p(z^{(n)} = k | \mathbf{x}^{(n)})$ given our current model, i.e., how much do we think a cluster is responsible for generating a datapoint.
  2. M-step: Use the equations on the last slide to update the parameters, assuming $r_k^{(n)}$ are held fixed – change the parameters of each Gaussian to maximize the probability that it would generate the data it is currently responsible for.





EM Algorithm for GMM

- Initialize the means $\hat{\boldsymbol{\mu}}_k$ and mixing coefficients $\hat{\pi}_k$
- Iterate until convergence:
  - E-step: Evaluate the responsibilities $r_k^{(n)}$ given current parameters

$$r_k^{(n)} = p(z^{(n)} = k | \mathbf{x}^{(n)}) = \frac{\hat{\pi}_k \mathcal{N}(\mathbf{x}^{(n)} | \hat{\boldsymbol{\mu}}_k, \mathbf{I})}{\sum_{j=1}^K \hat{\pi}_j \mathcal{N}(\mathbf{x}^{(n)} | \hat{\boldsymbol{\mu}}_j, \mathbf{I})} = \frac{\hat{\pi}_k \exp\{-\frac{1}{2}\|\mathbf{x}^{(n)} - \hat{\boldsymbol{\mu}}_k\|^2\}}{\sum_{j=1}^K \hat{\pi}_j \exp\{-\frac{1}{2}\|\mathbf{x}^{(n)} - \hat{\boldsymbol{\mu}}_j\|^2\}}$$

  - M-step: Re-estimate the parameters given current responsibilities

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{N_k} \sum_{n=1}^N r_k^{(n)} \mathbf{x}^{(n)}$$

$$\hat{\pi}_k = \frac{N_k}{N} \quad \text{with} \quad N_k = \sum_{n=1}^N r_k^{(n)}$$

  - Evaluate log likelihood and check for convergence

$$\log p(\mathcal{D}) = \sum_{n=1}^N \log \left(\sum_{k=1}^K \hat{\pi}_k \mathcal{N}(\mathbf{x}^{(n)} | \hat{\boldsymbol{\mu}}_k, \mathbf{I}) \right)$$

Gaussian Mixture Models: Connection to K-Means

- You saw soft K-means in lecture
- If you look at the update equations (and maybe some back of the envelope calculations) you will see that the update rule for soft k-means is the same as the GMMs where each Gaussian is spherical (0 mean, Identity covariance matrix)

Gaussian Mixture Models: Miscellany

- Can try initializing the centers with the k-means algorithm
- Your models will train a lot fast if you use diagonal covariance matrices (but it might not necessarily be a good idea)