30.08.2024

# Statistical Methods in AI (CS7.403)

## Lecture-8: Clustering (k-means, Gaussian Mixture Models)

Ravi Kiran (ravi.kiran@iiit.ac.in)

https://ravika.github.io

@vikataravi

Center for Visual Information Technology (CVIT)

IIIT Hyderabad

Repeat until convergence: {

For every $i$, set

$$c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2.$$

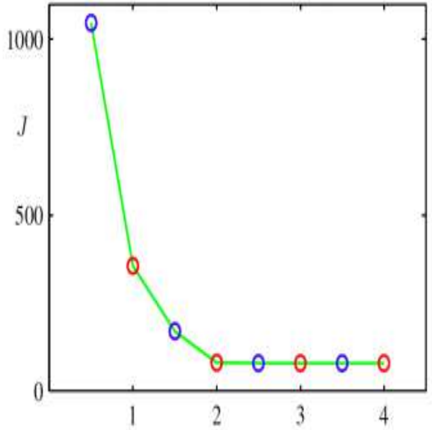
For each $j$, set

$$\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\}x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}.$$

}

$$\sum_{k=1}^{K}\sum_{i=1}^{n_k}\left\|x_{ki} - \mu_k\right\|^2$$

- Whenever an assignment is changed, the sum squared distances $J$ of data points from their assigned cluster centers is reduced.
- Whenever a cluster center is moved, $J$ is reduced.
- Test for convergence: If the assignments do not change in the assignment step, we have converged (to at least a local minimum).




- K-means cost function after each E step (blue) and M step (red). The algorithm has converged after the third M step

- The objective $J$ is non-convex (so coordinate descent on $J$ is not guaranteed to converge to the global minimum)

$$se = \sum_{k=1}^{K} \sum_{i=1}^{n_k} \left\| x_{ki} - \mu_k \right\|^2$$

1. Initialize **cluster centroids** $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$ randomly.
2. Repeat until convergence: {

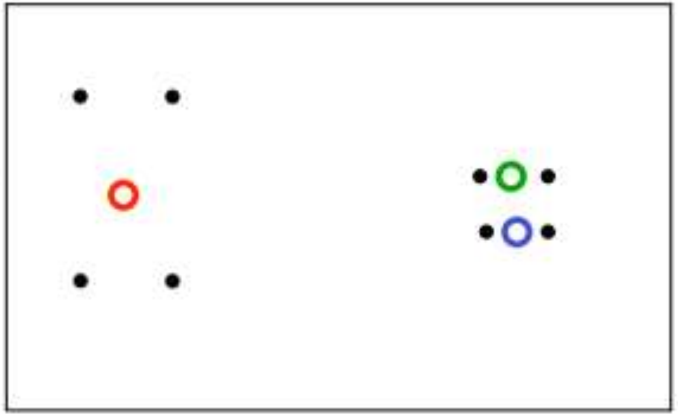
   For every $i$, set
   $$c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2.$$

   For each $j$, set
   $$\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}.$$

   }

- The objective $J$ is non-convex (so coordinate descent on $J$ is not guaranteed to converge to the global minimum)
- There is nothing to prevent k-means getting stuck at local minima.

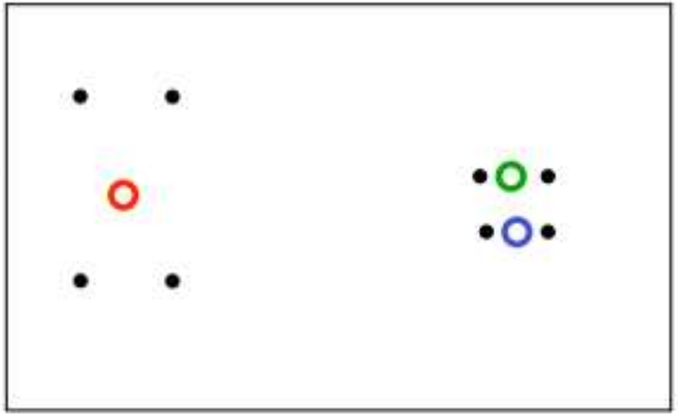
A bad local optimum



- The objective $J$ is non-convex (so coordinate descent on $J$ is not guaranteed to converge to the global minimum)
- There is nothing to prevent k-means getting stuck at local minima.
- We could try many random starting points

$$\sum_{k=1}^{K} \sum_{i=1}^{n_k} \left\| x_{ki} - \mu_k \right\|^2$$

A bad local optimum



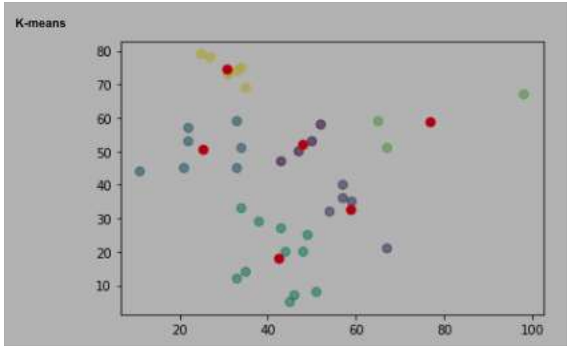
# K-means++: Improving K-means initialization

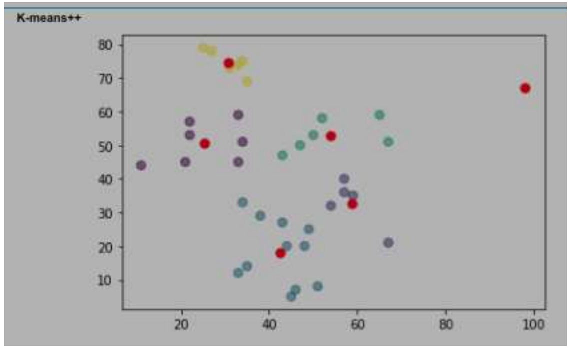
- Common way to improve k-means - smart initialization!
- General idea - try to get good coverage of the data.
- k-means++ algorithm:
  1. Pick the first center randomly
  2. For all points $\mathbf{x}^{(n)}$ set $d^{(n)}$ to be the distance to closest center.
  3. Pick the new center to be at $\mathbf{x}^{(n)}$ with probability proportional to $d^{(n)2}$
  4. Repeat steps 2+3 until you have k centers

https://iq.opengenus.org/k-means-vs-k-means-p/

# K-means++: Improving K-means initialization

- Common way to improve k-means - smart initialization!
- General idea - try to get good coverage of the data.
- k-means++ algorithm:
  1. Pick the first center randomly
  2. For all points $\mathbf{x}^{(n)}$ set $d^{(n)}$ to be the distance to closest center.
  3. Pick the new center to be at $\mathbf{x}^{(n)}$ with probability proportional to $d^{(n)2}$
  4. Repeat steps 2+3 until you have k centers

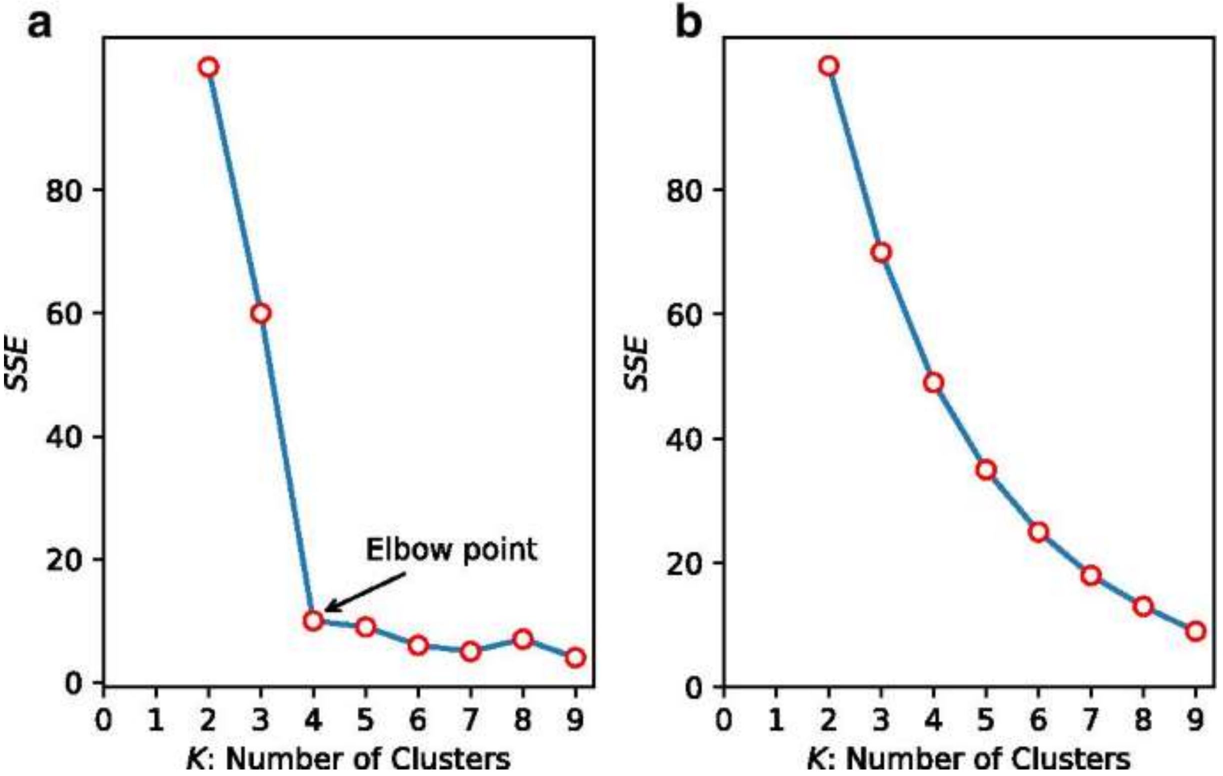



https://iq.opengenus.org/k-means-vs-k-means-p/

# How to choose k ?

$$se = \sum_{k=1}^{K} \sum_{i=1}^{n_k} \left\| x_{ki} - \mu_k \right\|^2$$



# Regularization

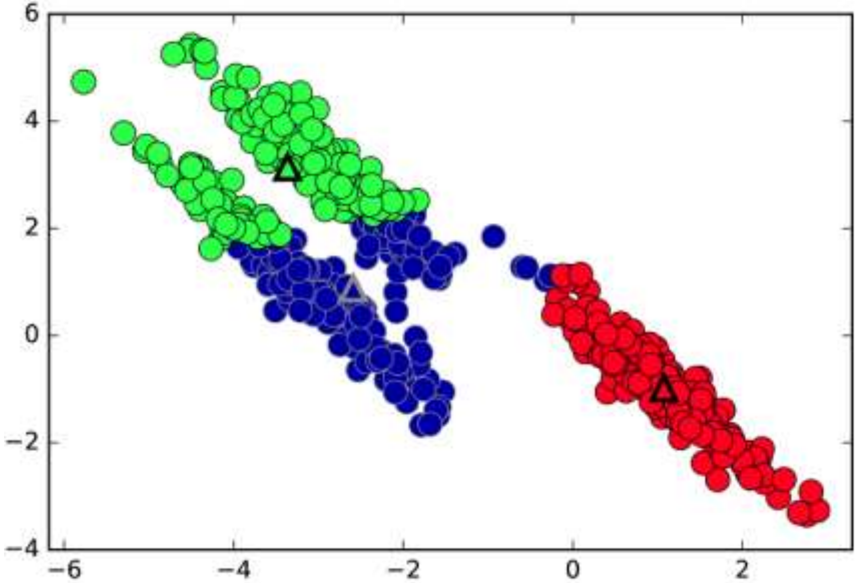
- Penalize “overly” large or “overly” small clusters

# K-mediods

- Squared Euclidean distance loss function of K-means not robust.
- Use L1 loss function $J = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \|x_i - \mu_k\|_1$ instead of squared Euclidean distance.
- Use an iterative procedure as before.
  - Prototype is the median of the points assigned to a cluster.

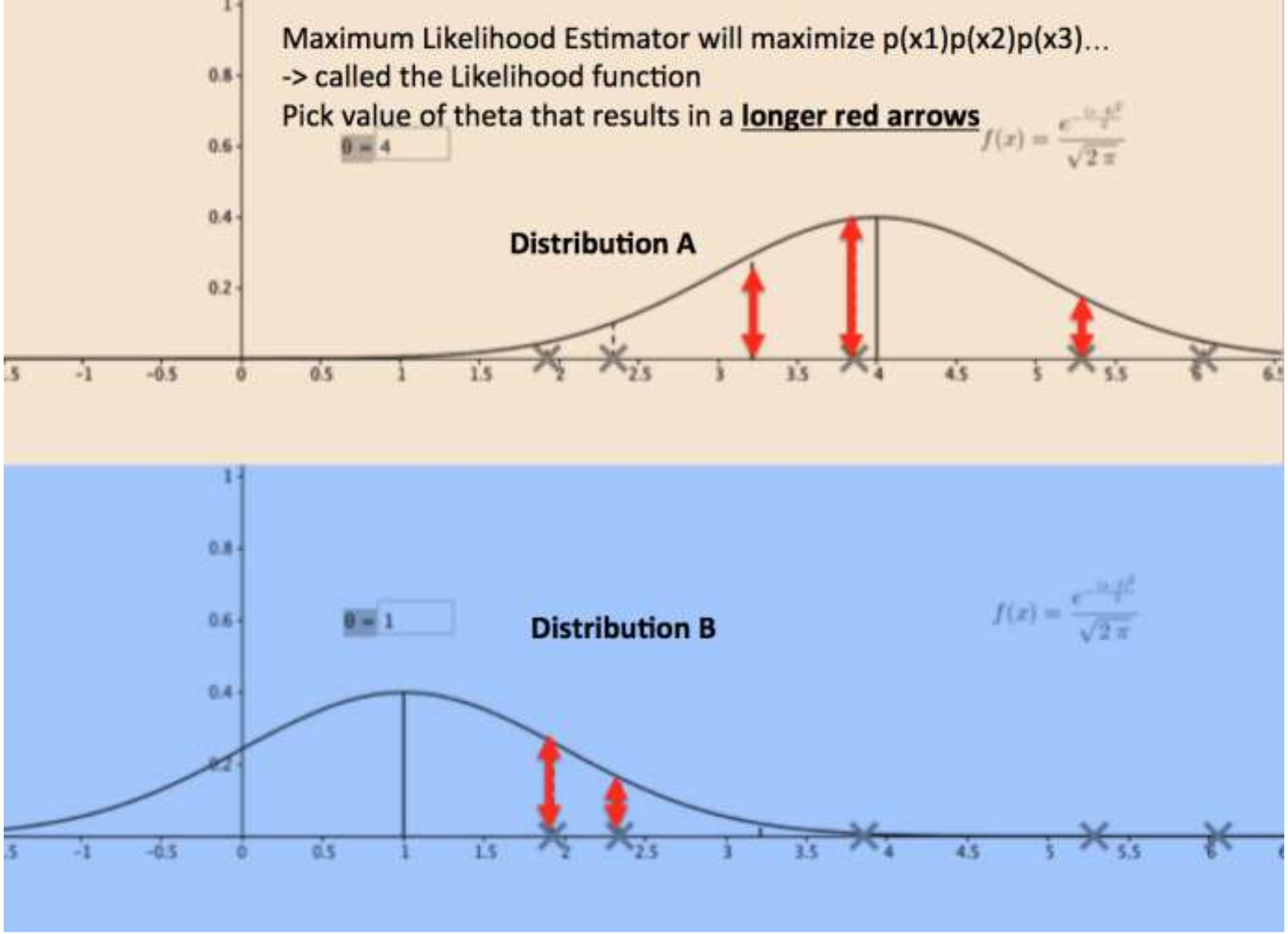
# K-means: Additional issues

- ‘Hard’ assignments
- Euclidean ➔ Favours ‘spherical’ clusters of equal ‘contribution’
- Sensitive to initialization
- Sensitive to outliers

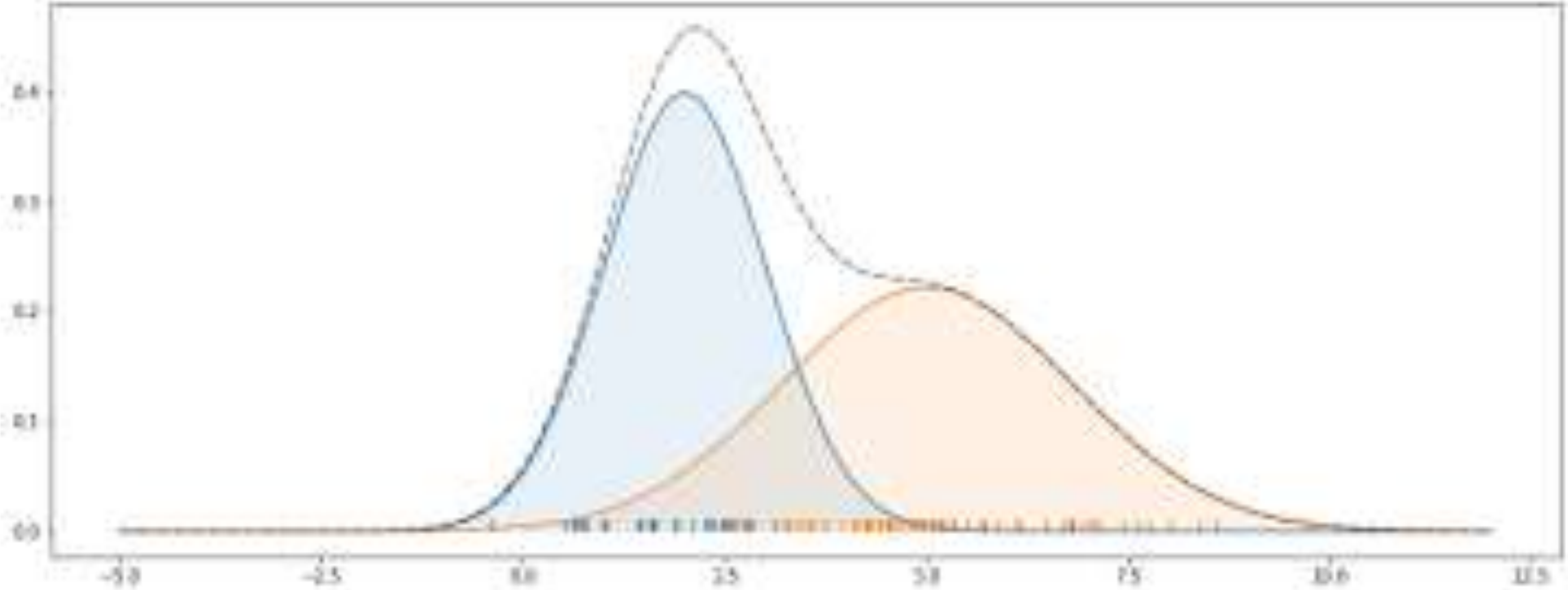



https://amueller.github.io/aml/03-unsupervised-learning/02-clustering-mixture-models.html

# Maximum Likelihood Estimation




https://pmirla.github.io/2016/07/20/maximum-likelihood-explanation.html


0.4
0.3
0.2
0.1
0.0
-5.0
-2.5
0.0
2.5
5.0
7.5
10.0
12.5

# Mixture Models

- Data distribution $p(x)$ assumed to be a weighted sum of $K$ distributions

$$p(x) = \sum_{k=1}^{K} \pi_k p(x|\boldsymbol{\theta}_k)$$

  where $\pi_k$'s are the mixing weights: $\sum_{k=1}^{K} \pi_k = 1, \quad \pi_k \geq 0$ (intuitively, $\pi_k$ is the proportion of data generated by the $k$-th distribution)
- Each component distribution $p(x|\boldsymbol{\theta}_k)$ represents a "cluster" in the data

https://www.cse.iitk.ac.in/users/piyush/courses/pml_winter16/slides_lec7.pdf

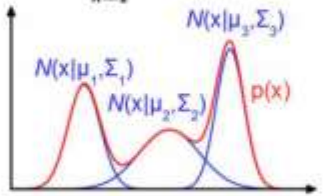
# Mixture Models

- Data distribution $p(x)$ assumed to be a weighted sum of $K$ distributions

$$p(x) = \sum_{k=1}^{K} \pi_k p(x|\boldsymbol{\theta}_k)$$

  where $\pi_k$'s are the mixing weights: $\sum_{k=1}^{K} \pi_k = 1, \quad \pi_k \geq 0$ (intuitively, $\pi_k$ is the proportion of data generated by the $k$-th distribution)
- Each component distribution $p(x|\boldsymbol{\theta}_k)$ represents a "cluster" in the data
- **Gaussian Mixture Model (GMM):** component distributions are Gaussians

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




https://www.cse.iitk.ac.in/users/piyush/courses/pml_winter16/slides_lec7.pdf

# Mixture Models

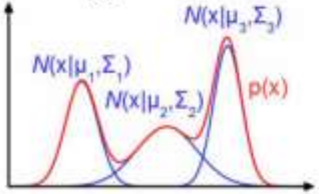
- Data distribution $p(x)$ assumed to be a weighted sum of $K$ distributions

$$p(x) = \sum_{k=1}^{K} \pi_k p(x|\boldsymbol{\theta}_k)$$

  where $\pi_k$'s are the mixing weights: $\sum_{k=1}^{K} \pi_k = 1, \quad \pi_k \geq 0$ (intuitively, $\pi_k$ is the proportion of data generated by the $k$-th distribution)

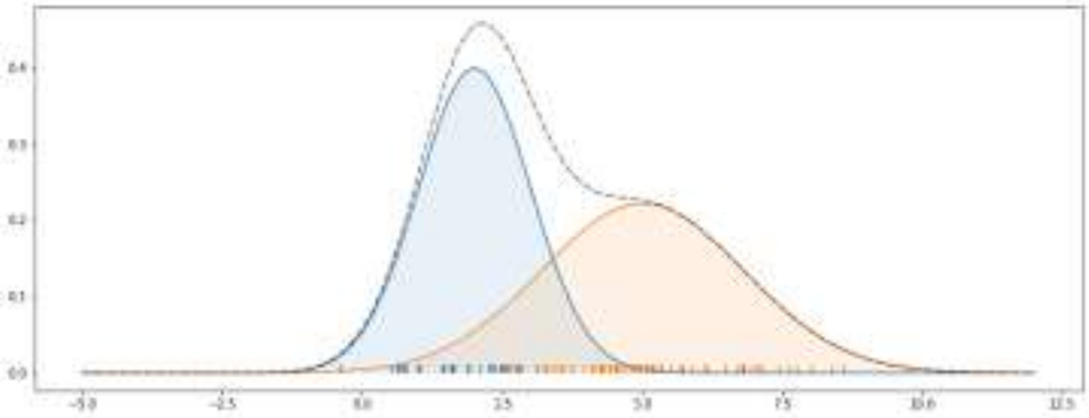
- Each component distribution $p(x|\boldsymbol{\theta}_k)$ represents a "cluster" in the data
- **Gaussian Mixture Model (GMM):** component distributions are Gaussians

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




- Mixture models used in many data modeling problems, e.g.,
  - Unsupervised Learning: Clustering (+density estimation)
  - Supervised Learning: Mixture of Experts models

# Mixture Models



A GMM represents a **distribution** as

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

with $\pi_k$ the mixing coefficients, where:

$$\sum_{k=1}^{K} \pi_k = 1 \quad \text{and} \quad \pi_k \geq 0 \quad \forall k$$

# Mixture Models

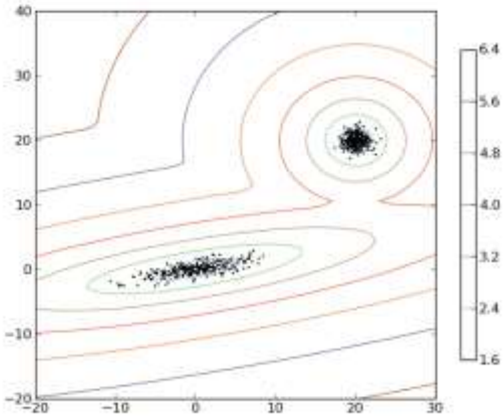
Most common mixture model: Gaussian mixture model (GMM)

- A GMM represents a **distribution** as

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

with $\pi_k$ the mixing coefficients, where:

$$\sum_{k=1}^{K} \pi_k = 1 \quad \text{and} \quad \pi_k \geq 0 \quad \forall k$$




http://scikit-learn.sourceforge.net/0.5/auto_examples/gmm/plot_gmm_pdf.html

- Can think of the data $\{\boldsymbol{x}_1, \boldsymbol{x}_n, \ldots, \boldsymbol{x}_N\}$ using a "generative story"
  - For each example $\boldsymbol{x}_n$, first choose its cluster assignment $z_n \in \{1, 2, \ldots, K\}$ as

$$z_n \sim \text{Multinoulli}(\pi_1, \pi_2, \ldots, \pi_K)$$ aka "categorical"

  - Now generate $\boldsymbol{x}$ from the Gaussian with id $z_n$

$$\boldsymbol{x}_n | z_n \sim \mathcal{N}(\boldsymbol{\mu}_{z_n}, \boldsymbol{\Sigma}_{z_n})$$

# Resources

- Textbook
  - PRML (Bishop) – Chapter 9: 9.1,9.2,9.3.2
  - Pattern Classification (Duda, Hart, Stork)
    - 10.4.3,10.6.1,10.7.1,10.7.2,10.8,10.10
- Videos
  - https://www.youtube.com/watch?v=REypj2sy_5U&list=PLBv09BD7ez_4e9LtmK626Evn1ion6ynrt
  - https://www.youtube.com/watch?v=rVfZHWTwXSA
- Blog posts/Lecture Notes
  - https://www.cse.iitk.ac.in/users/piyush/courses/pml_winter16/slides_lec7.pdf
  - https://see.stanford.edu/materials/aimlcs229/cs229-notes8.pdf
  - https://www.cs.toronto.edu/~jlucas/teaching/csc411/lectures/lec15_16_handout.pdf
  - https://mbernste.github.io/posts/gmm_em/
  - https://www.ritchievink.com/blog/2019/05/24/algorithm-breakdown-expectation-maximization/