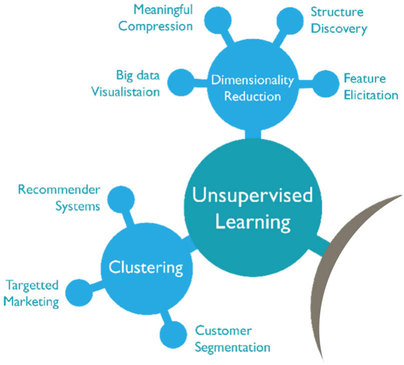
# Turing Machine and Deep Learning
# Lecture 2: Unsupervised Machine Learning

Satchit Chatterji
satchit.chatterji@gmail.com

MSc Artificial Intelligence
University of Amsterdam

26th May 2023

# Re: Unsupervised ML




## Given

Unlabelled, possibly unstructured data, notated simply as:

$$x_i \in \mathbb{R}^K, i \in \{1, ..., n\}$$

## Goal

Several, including:

- Dimensionality reduction
- Find patterns/structures/groups
- Model probability densities

e.g.: **PCA, K-means clustering, Gaussian mixture models**, autoencoders

# *Tangent:* Representations again...

## Given (Unsupervised Learning)

Unlabelled, possibly unstructured data, notated simply as:

$$x_i \in \mathbb{R}^K, i \in \{1, ..., n\}$$

## Given (Supervised Learning)

Input-output pairs of the form:

$$S = (x_i, y_i)$$

$$x_i \in \mathbb{R}^K, y_i \in \mathbb{R}^M, i = \{1, ..., n\}$$

# *Tangent:* Representations again...

### Given (Unsupervised Learning)

Unlabelled, possibly unstructured data, notated simply as:

$$x_i \in \mathbb{R}^K, i \in \{1, ..., n\}$$

### Given (Supervised Learning)

Input-output pairs of the form:

$$S = (x_i, y_i)$$

$$x_i \in \mathbb{R}^K, y_i \in \mathbb{R}^M, i = \{1, ..., n\}$$

**Q**: How do you represent non-numeric 'things' numerically?

# Vector Representations of Features (I)

*Numerical data*: Most often use continuous real vectors.
e.g. a house is $800m^2$, has 3 bedrooms, 2 bathrooms, costs €$6,600,000$.

$$x = [800, 3, 2, 6600000]^T$$

## Potential Issues

- Interpretability (What does 3.2 bedrooms mean?)
- Scale of features ($2 <<< 6600000$)
- Often questionable choice to represent ordered categorical quantities (e.g. if 1=low, 2=med, 3=high, is 'high-low=medium'?)

# Vector Representations of Features (II)

*Categorical data*: Most often use 'one-hot encodings', a vector of zeros, with a one in place of the index of a class label.
e.g. classes $= \{$'cat', 'dog', 'mouse', 'horse', 'chicken'$\}$. If we have a dog:

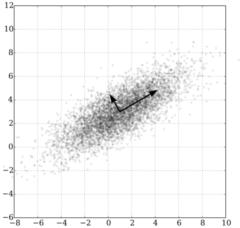
$$x = [0, 1, 0, 0, 0]^T$$

## Potential Issues

- Dimensionality Explosion ($x \in \mathbb{R}^K, K = |\text{classes}|$)
- Loss of comparative information between classes (e.g. order)
- Handling rare/unknown classes
- Sparsity (vector mostly zeros)

# Principal Component Analysis

# PCA Intro




## Major use cases

- Find the 'principal components' of the variation in the data (analysis).
- Project high-dimensional down data for visualization.
- Dimensionality reduction: for every data point $x_i \in \mathbb{R}^K$, find an approximate representation $\tilde{x}_i \in \mathbb{R}^{K'}$, $K' < K$.
- *Feature extraction* – removes redundancies and collinearity.

# PCA How-to

1. **Data preprocessing**: Normalizing to zero mean and unit variance.
2. **Compute Covariance Matrix**: if $X$ is the data matrix,

$$\Sigma = X^T X$$

3. **Eigendecomposition**:

$$\Sigma = VDV^{-1}$$

4. **Choose $\ell$ components**: Stack $\ell$ eigenvectors to form a projection matrix $P$.
5. **Project data**: Compute the lower-dimensional data matrix:

$$\tilde{X} = PX$$

# PCA How-to

1. **Data preprocessing**: Normalizing to zero mean and unit variance.
2. **Compute Covariance Matrix**: if $X$ is the data matrix,
$$\Sigma = X^T X$$
3. **Eigendecomposition**:
$$\Sigma = VDV^{-1}$$
4. **Choose $\ell$ components**: Stack $\ell$ eigenvectors to form a projection matrix $P$.
5. **Project data**: Compute the lower-dimensional data matrix:
$$\tilde{X} = PX$$

**Exercise:** What are the shapes of the matrices?

# K-Means Clustering

# K-Means Clustering – Basics

- 'Most' commonly used clustering method – find groups and structure within unlabelled data.
- Iterative algorithm – no closed form solution, converges towards a solution.

**Goal**

Partitioning a dataset into K clusters, where each data point belongs to the cluster with the nearest mean (centroid).

# K-Means: Algorithm

Called the *expectation-maximization (EM) algorithm*:

1. Randomly initialize $K$ centroids
2. Assign each data point to the nearest centroid
3. Recalculate the centroids based on the assigned data points
4. Repeat steps 2 and 3 until convergence!

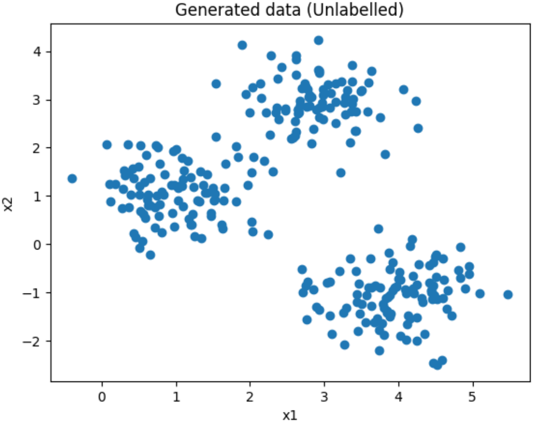
# K-Means: Algorithm

Called the *expectation-maximization (EM) algorithm*:

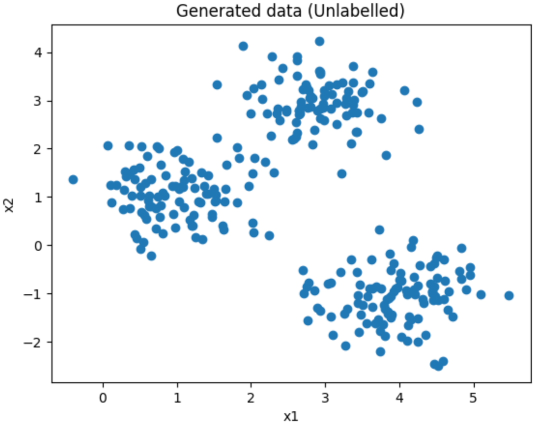
1. Randomly initialize $K$ centroids
2. Assign each data point to the nearest centroid
3. Recalculate the centroids based on the assigned data points
4. Repeat steps 2 and 3 until convergence!

? *'K', 'randomly initialize', 'centroids', 'nearest'/'distance', 'convergence'*

# K-Means: Visual

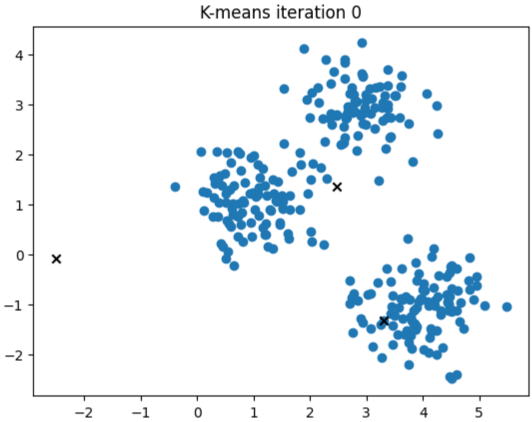


# K-Means: Visual



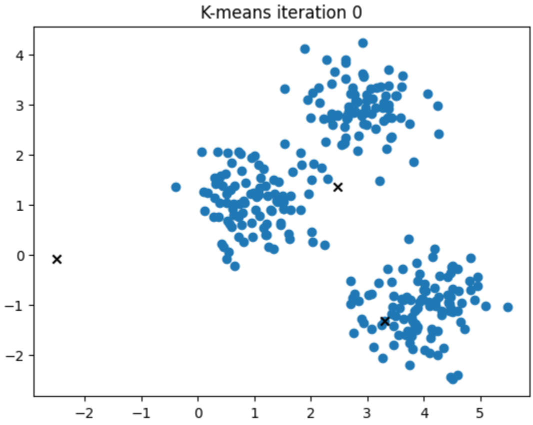

1. Randomly initialize $K$ centroids

# K-Means: Visual



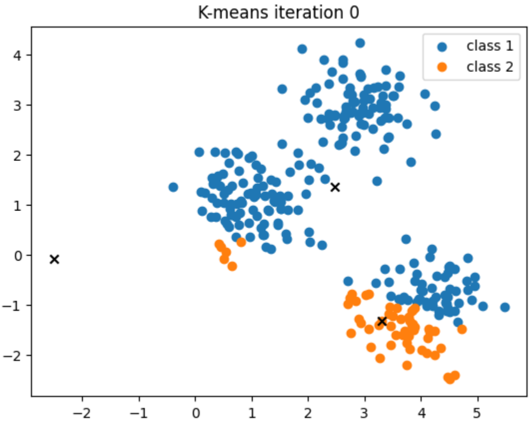

1. Randomly initialize $K = 3$ centroids

# K-Means: Visual



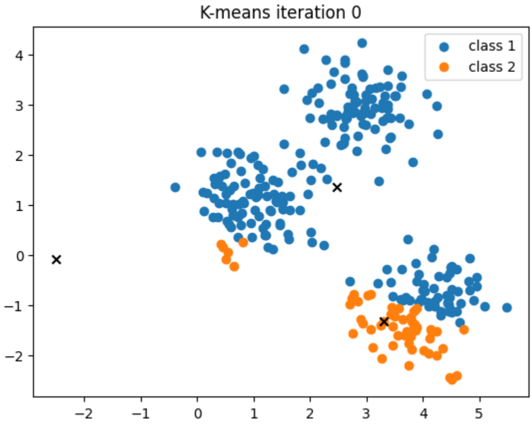

1. Randomly initialize $K = 3$ centroids
2. Assign each data point to the nearest centroid

# K-Means: Visual



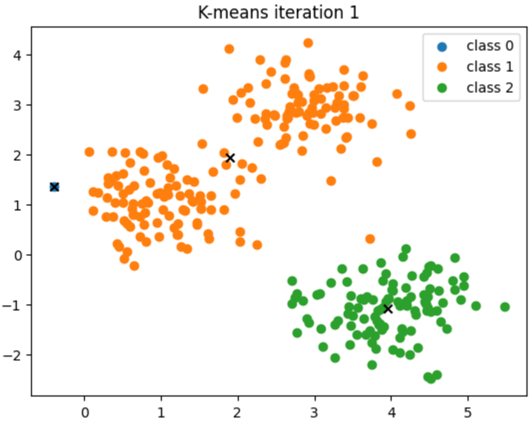

2. Assign each data point to the nearest centroid

# K-Means: Visual



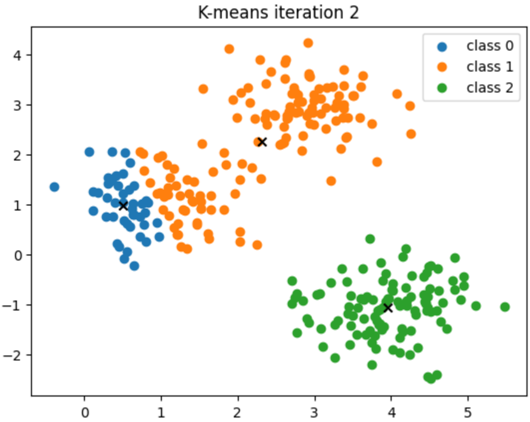

2. Assign each data point to the nearest centroid
3. Recalculate the centroids based on the assigned data points

# K-Means: Visual



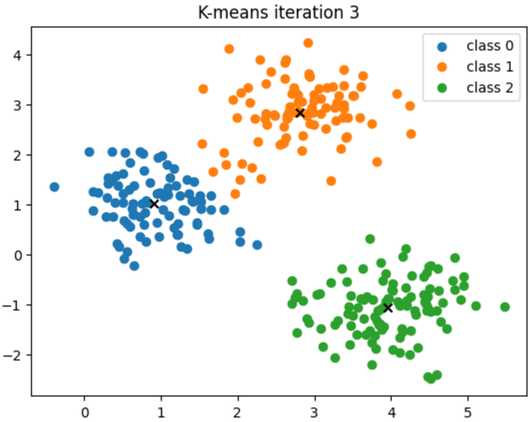

4. Repeat 2 and 3 until convergence!

# K-Means: Visual



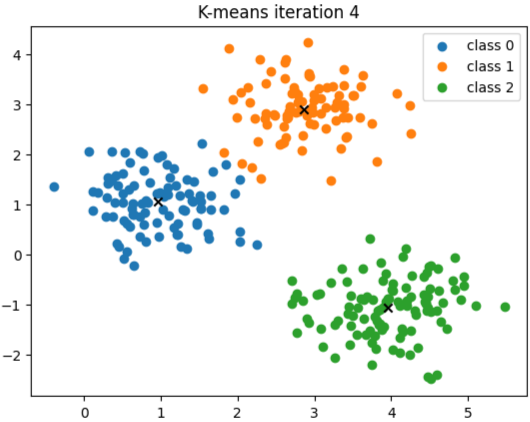

4. Repeat 2 and 3 until convergence!

# K-Means: Visual



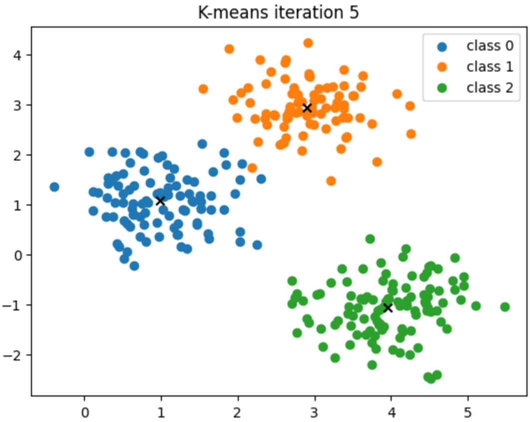

4. Repeat 2 and 3 until convergence!

# K-Means: Visual



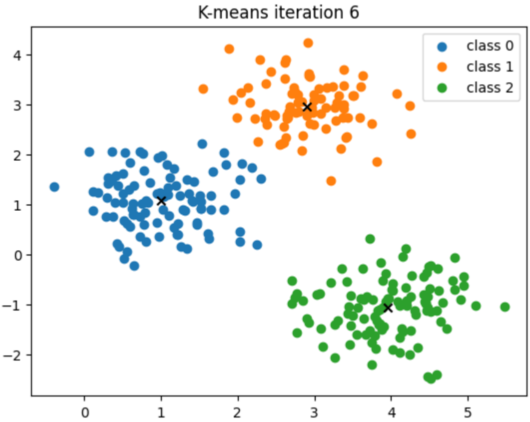

4. Repeat 2 and 3 until convergence!

# K-Means: Visual



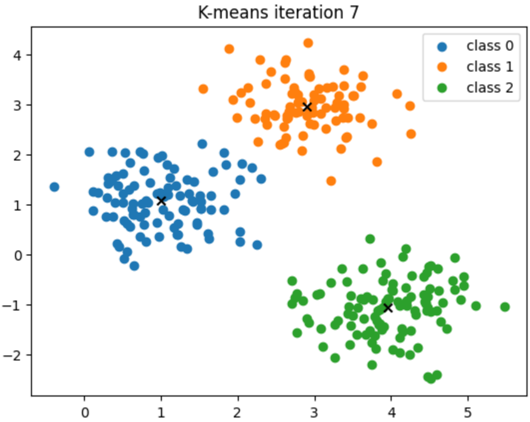

4. Repeat 2 and 3 until convergence!

# K-Means: Visual




4. Repeat 2 and 3 until convergence!

# K-Means: Visual




4. Repeat 2 and 3 until convergence!

# K-Means: Details of Example

- **K**: Hyperparameter: how many classes are you trying to find?
- **Randomly Initialize**: Classical K-means: randomly, e.g. uniformly between `min(data)` and `max(data)`.
- **Centroids**: The current center of each class's assigned data points
- **Nearest/Distance**: Squared Euclidean/L2-norm: $||x - c||_2^2$
- **Convergence**: When there is no change in the assignment of data points from one iteration to the next/when the centroids do not change.
- **Effective loss function**:

# K-Means: Details of Example

- **K**: Hyperparameter: how many classes are you trying to find?
- **Randomly Initialize**: Classical K-means: randomly, e.g. uniformly between `min(data)` and `max(data)`.
- **Centroids**: The current center of each class's assigned data points
- **Nearest/Distance**: Squared Euclidean/L2-norm: $||x - c||_2^2$
- **Convergence**: When there is no change in the assignment of data points from one iteration to the next/when the centroids do not change.
- **Effective loss function**: *within-cluster sum of squares*

$$WCSS = \sum_{i=1}^{n} \sum_{j=1}^{K} (x_i - c_j)^2$$

# K-Means: Advantages & Limitations

## Advantages

- Simple, efficient
- Scalable to large data
- Interpretable
- Widely used, understood and supported
- Many, many extensions

## Limitations

- Sensitive to initial conditions
- Choice of optimal $K$
- Sensitive to outliers
- Spherical clusters of equal variance
- Hard ('crisp') classifications

# K-Means: Advantages & Limitations

## Advantages

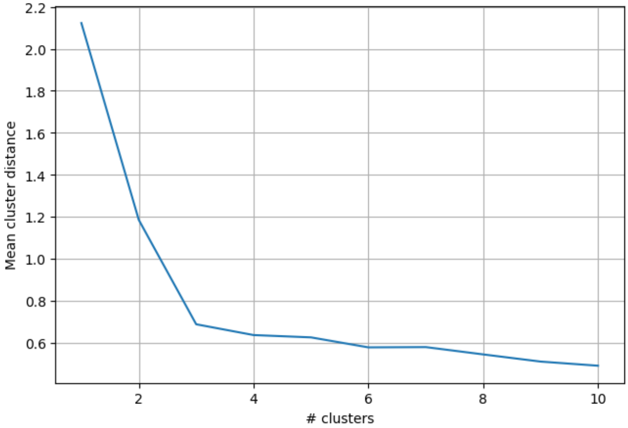
- Simple, efficient
- Scalable to large data
- Interpretable
- Widely used, understood and supported
- Many, many extensions

## Limitations

- Sensitive to initial conditions
- **Choice of optimal $K$**
- **Sensitive to outliers**
- **Spherical clusters of equal variance**
- **Hard ('crisp') classifications**

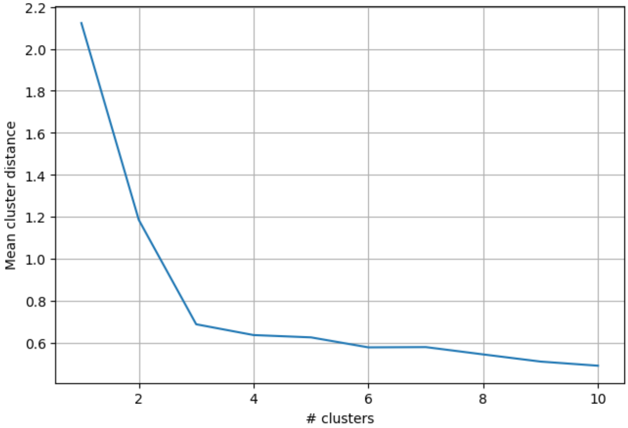
# K-Means: Choice of $K$

The *Elbow method*:



# K-Means: Choice of $K$

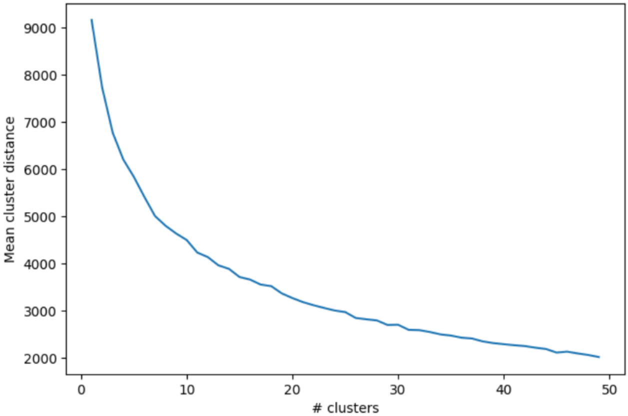
The *Elbow method*:




Maybe 3 clusters?

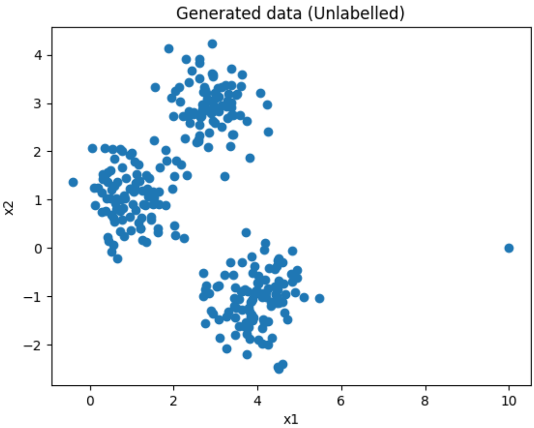
# K-Means: Choice of $K$

Where is the elbow?



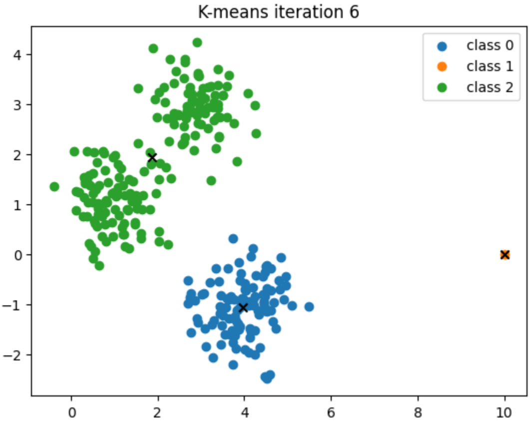
# K-Means: Outliers

What will happen here?



# K-Means: Outliers

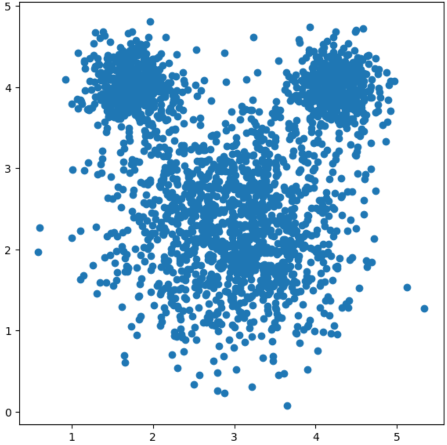
What will happen here?




**Remember**: Try and remove outliers (within reason) before any serious ML training.

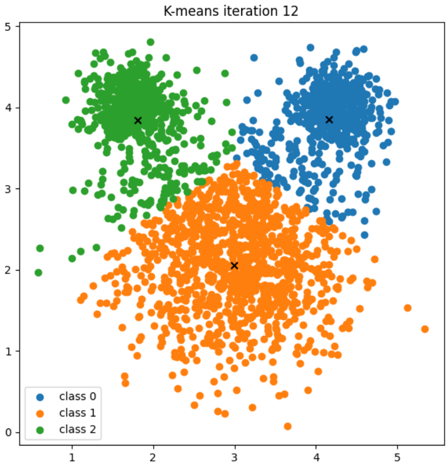
# K-Means: Spherical, Equal Variance Clusters

The "Micky Mouse" dataset



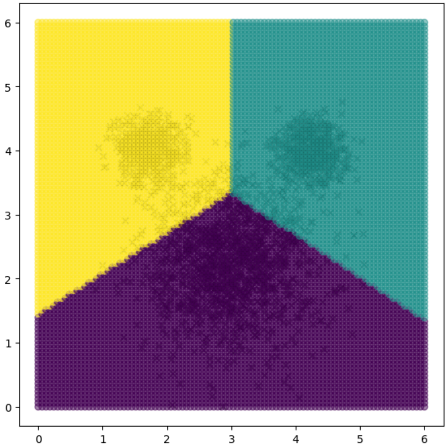
# K-Means: Spherical, Equal Variance Clusters

The "Micky Mouse" dataset



# K-Means: Spherical, Equal Variance Clusters

The 'hard' decision boundaries



# Gaussian Mixture Models

# GMMS – Basic Idea

Start with K-Means: parameterised by **mean** centroids

- Issue 1: Spherical clusters of equal variance
- Issue 2: Hard ('crisp') classifications
- *Solution*: Model the data in terms of a mixture/summation/superposition of *normal distributions*.

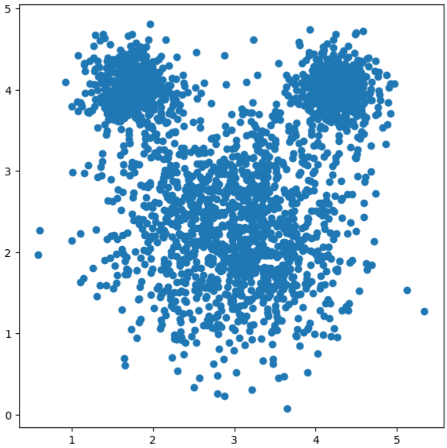
Core idea: observing each data point can be conditioned on a class multivariate normal distribution:

$$p(x|C_i) = \mathcal{N}(\hat{\mu}_i, \Sigma_i)$$

Train with the EM algorithm, but it's a bit more complicated this time.

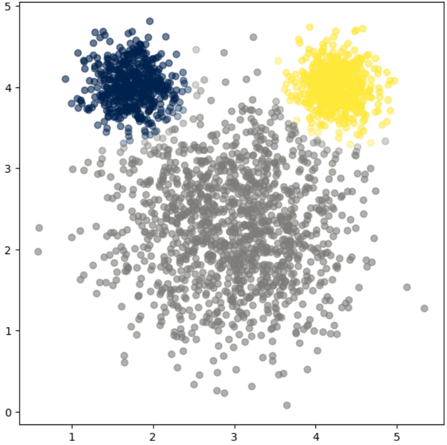
# GMMs: Micky Mouse dataset

The “Micky Mouse” dataset



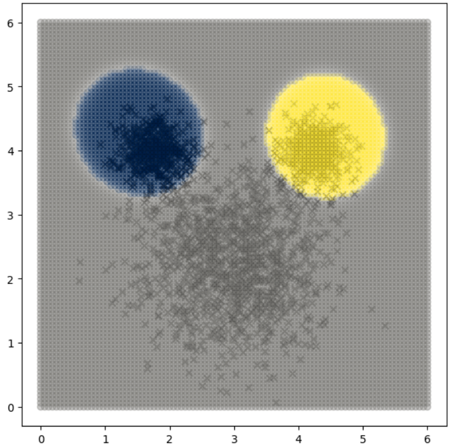
# GMMs: Micky Mouse dataset

The "Micky Mouse" dataset



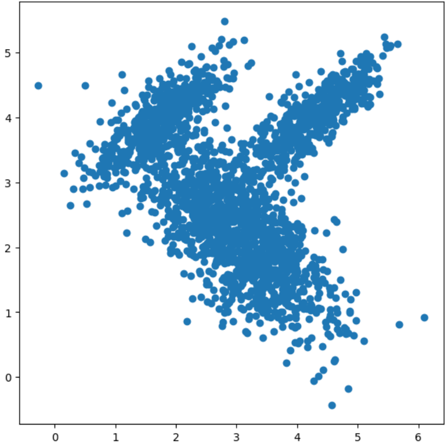
# GMMs: Micky Mouse dataset

The "soft" decision boundaries:



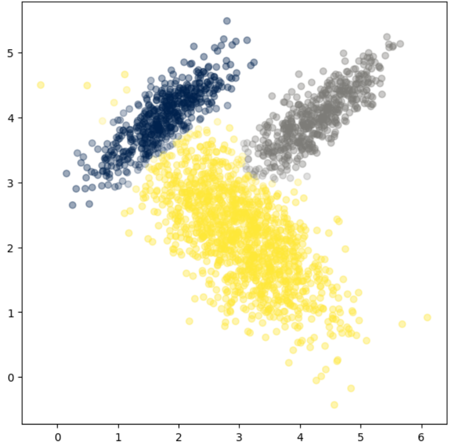
# GMMs: Stretched-out Micky Mouse dataset

The "Micky Mouse" dataset, but with someone who is bad at photoshop



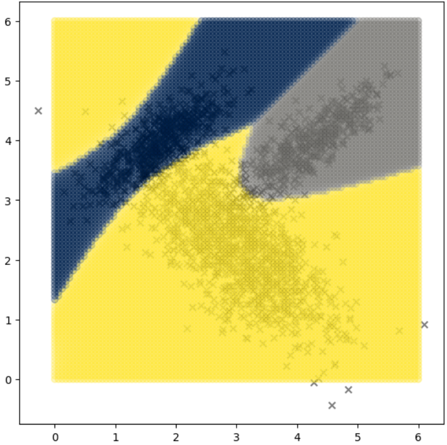
# GMMs: Stretched-out Micky Mouse dataset

The “Micky Mouse” dataset, but with independent covariances



# GMMs: Stretched-out Micky Mouse dataset

The "soft" decision boundaries, more flexible than K-Means, probabilistic:



# GMM: Advantages & Limitations

## Advantages

- Flexibility in modelling
- Soft assignment (probabalistic)
- Widely used and understood
- A *generative* model

## Limitations

- Sensitive to initial conditions
- Choice of optimal $K$
- Sensitive to outliers
- Assumes normal distributions
- Takes longer to train than K-Means

# Image credits

Slide 7: https://en.wikipedia.org/wiki/Principal_component_analysis