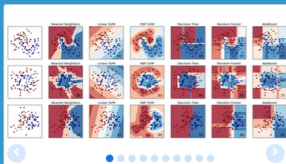


Unsupervised learning & Mixture models

Dustin Lang
McWilliams Postdoc Fellow
Carnegie Mellon University
visiting University of Waterloo

Local Group Astrostats / 2015-06-04

The Machine Learning Landscape



scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.

Algorithms: *SVM, nearest neighbors, random forest, ...* — Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: *SVR, ridge regression, Lasso, ...* — Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: *k-Means, spectral clustering, mean-shift, ...* — Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency

Algorithms: *PCA, feature selection, non-negative matrix factorization.* — Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning

Modules: *grid search, cross validation, metrics.* — Examples

Preprocessing

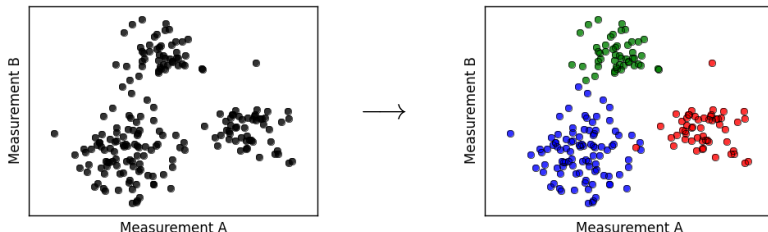
Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.

Modules: *preprocessing, feature extraction.* — Examples

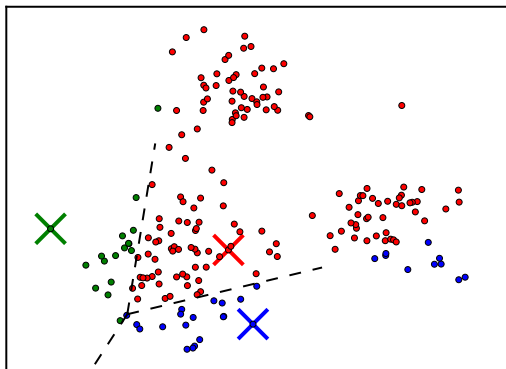
Unsupervised learning

- ▶ **No labels / no truth**: We are not given a target value we want to reconstruct
- ▶ Just given the data (usually assumed: low-dimensional measurements with equal noise)
- ▶ Main task: **Clustering**



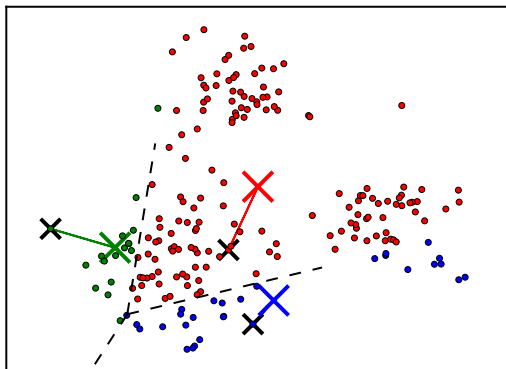
Clustering: the K -means algorithm

- ▶ Assume K clusters, each characterized by a centroid
- ▶ Start by randomly choosing K data points as centroids
- ▶ Iteratively:
 - ▶ Assign each data point to the nearest cluster
 - ▶ Compute new cluster center based on assigned data points



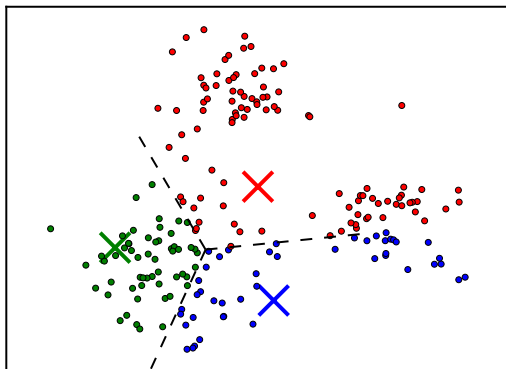
Clustering: the K -means algorithm

- ▶ Assume K **clusters**, each characterized by a **centroid**
- ▶ Start by randomly choosing K data points as centroids
- ▶ Iteratively:
 - ▶ **Assign** each data point to the **nearest** cluster
 - ▶ **Compute** new cluster center based on assigned data points



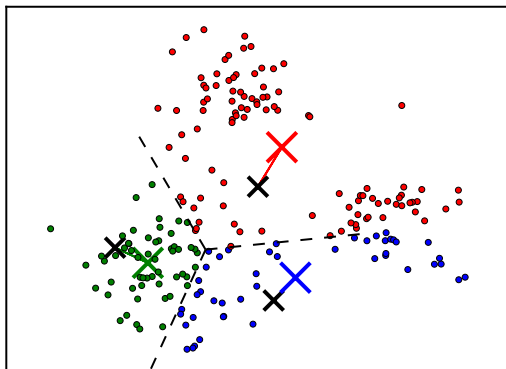
Clustering: the K -means algorithm

- ▶ Assume K clusters, each characterized by a centroid
- ▶ Start by randomly choosing K data points as centroids
- ▶ Iteratively:
 - ▶ Assign each data point to the nearest cluster
 - ▶ Compute new cluster center based on assigned data points



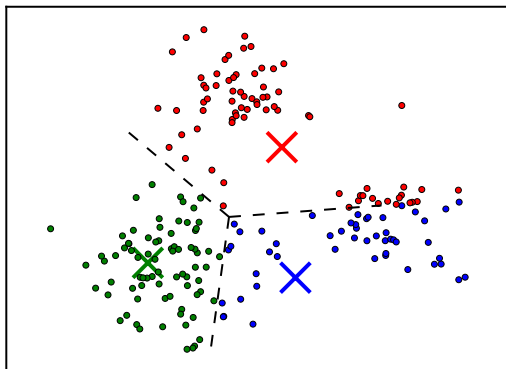
Clustering: the K -means algorithm

- ▶ Assume K clusters, each characterized by a centroid
- ▶ Start by randomly choosing K data points as centroids
- ▶ Iteratively:
 - ▶ Assign each data point to the nearest cluster
 - ▶ Compute new cluster center based on assigned data points



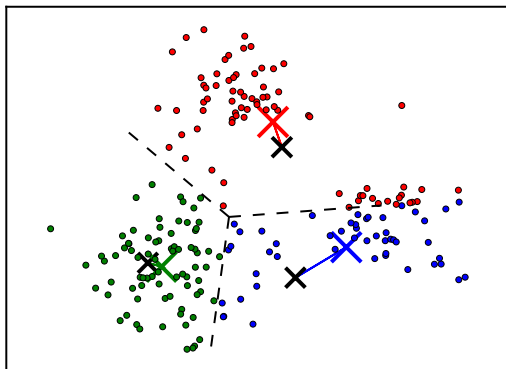
Clustering: the K -means algorithm

- ▶ Assume K clusters, each characterized by a centroid
- ▶ Start by randomly choosing K data points as centroids
- ▶ Iteratively:
 - ▶ Assign each data point to the nearest cluster
 - ▶ Compute new cluster center based on assigned data points



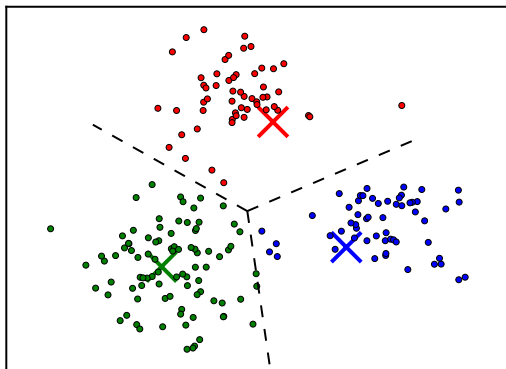
Clustering: the K -means algorithm

- ▶ Assume K **clusters**, each characterized by a **centroid**
- ▶ Start by randomly choosing K data points as centroids
- ▶ Iteratively:
 - ▶ **Assign** each data point to the **nearest** cluster
 - ▶ **Compute** new cluster center based on assigned data points



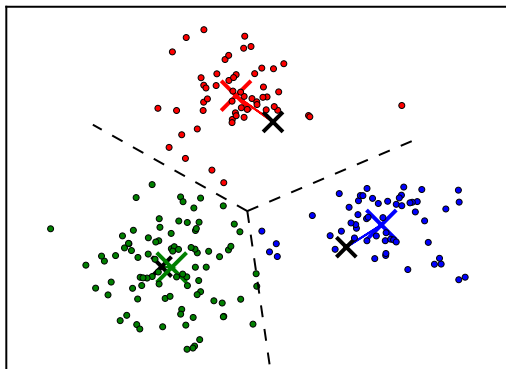
Clustering: the K -means algorithm

- ▶ Assume K clusters, each characterized by a centroid
- ▶ Start by randomly choosing K data points as centroids
- ▶ Iteratively:
 - ▶ Assign each data point to the nearest cluster
 - ▶ Compute new cluster center based on assigned data points



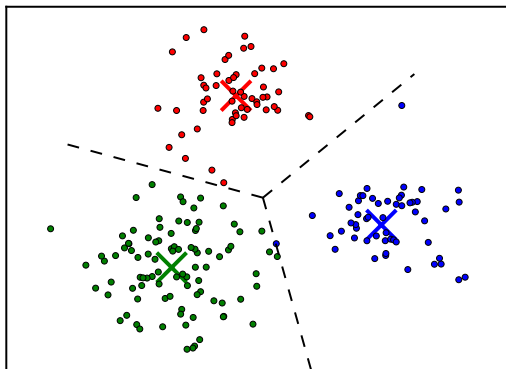
Clustering: the K -means algorithm

- ▶ Assume K **clusters**, each characterized by a **centroid**
- ▶ Start by randomly choosing K data points as centroids
- ▶ Iteratively:
 - ▶ **Assign** each data point to the **nearest** cluster
 - ▶ **Compute** new cluster center based on assigned data points



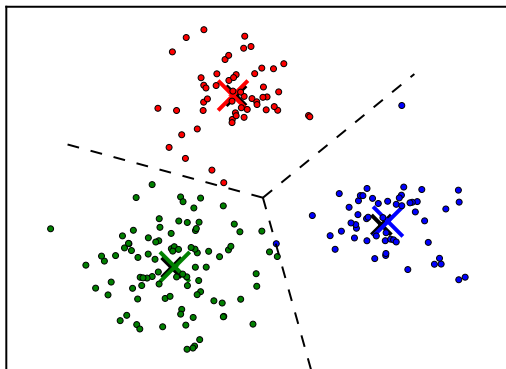
Clustering: the K -means algorithm

- ▶ Assume K clusters, each characterized by a centroid
- ▶ Start by randomly choosing K data points as centroids
- ▶ Iteratively:
 - ▶ Assign each data point to the nearest cluster
 - ▶ Compute new cluster center based on assigned data points



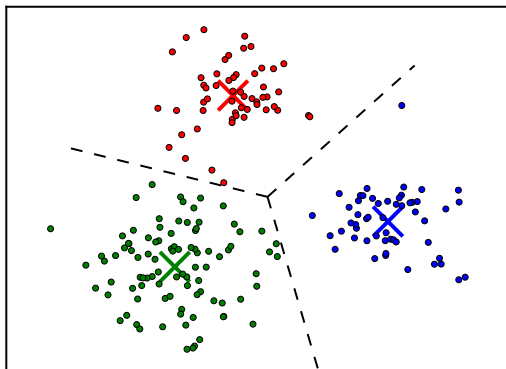
Clustering: the K -means algorithm

- ▶ Assume K **clusters**, each characterized by a **centroid**
- ▶ Start by randomly choosing K data points as centroids
- ▶ Iteratively:
 - ▶ **Assign** each data point to the **nearest** cluster
 - ▶ **Compute** new cluster center based on assigned data points

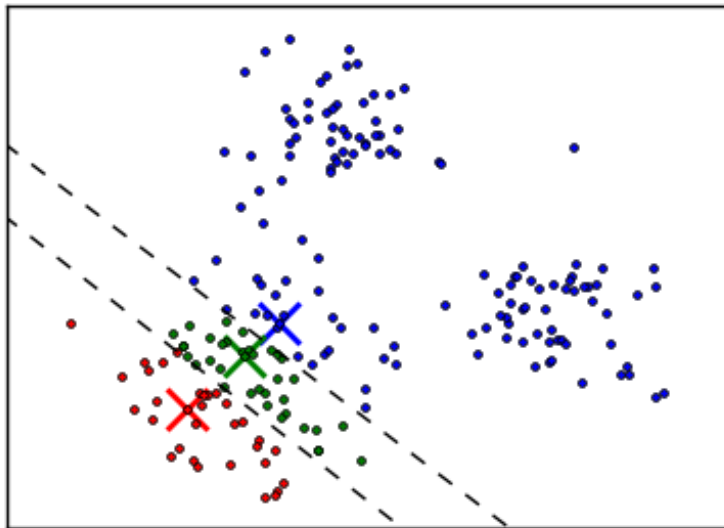


Clustering: the K -means algorithm

- ▶ Assume K clusters, each characterized by a centroid
- ▶ Start by randomly choosing K data points as centroids
- ▶ Iteratively:
 - ▶ Assign each data point to the nearest cluster
 - ▶ Compute new cluster center based on assigned data points



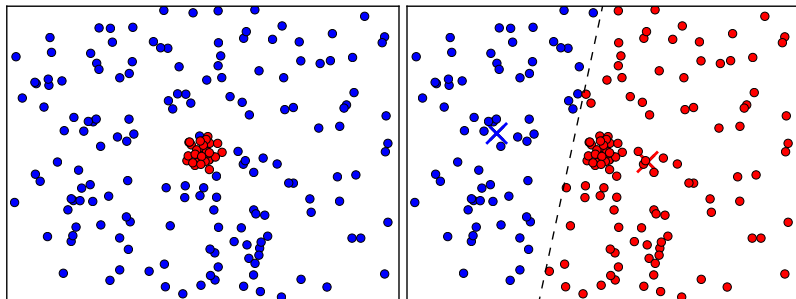
Clustering: the K -means algorithm



Problems with K -means

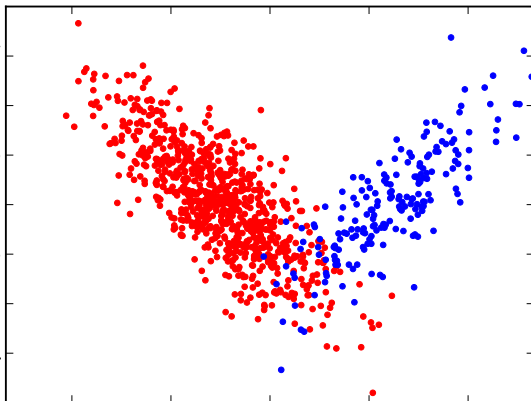
Clusters are defined by **centroids alone**

- ▶ clusters are separated by hyper-planes
- ▶ no covariances in the clusters
- ▶ no weights for clusters
- ▶ data points are hard-assigned to clusters
- ▶ noisy measurements not considered
- ▶ local optimization (multiple random initializations)



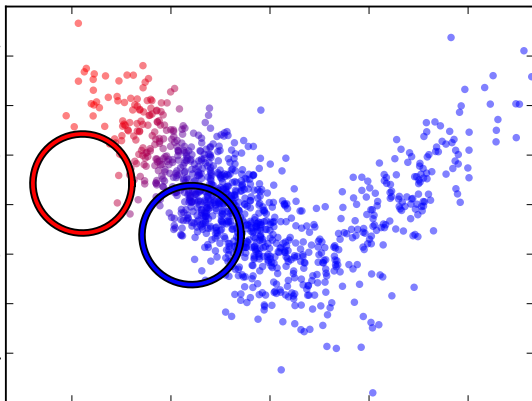
Gaussian Mixture Models

- ▶ Instead of finding just **centroids**, we want to find **weights** and Gaussians **means** and **covariances** of clusters
- ▶ Can be optimized efficiently using the **Expectation-Maximization** (EM) algorithm



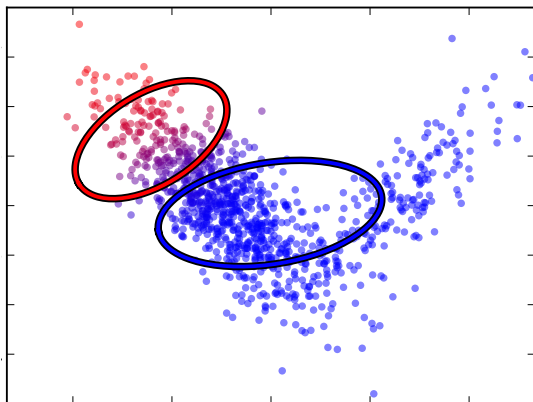
Gaussian Mixture Models

- ▶ Instead of finding just **centroids**, we want to find **weights** and Gaussians **means** and **covariances** of clusters
- ▶ Can be optimized efficiently using the **Expectation-Maximization** (EM) algorithm



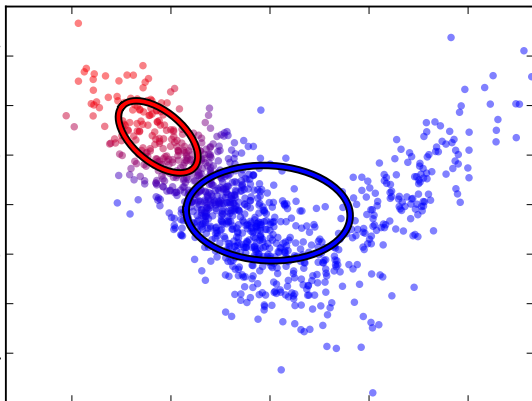
Gaussian Mixture Models

- ▶ Instead of finding just **centroids**, we want to find **weights** and Gaussians **means** and **covariances** of clusters
- ▶ Can be optimized efficiently using the **Expectation-Maximization** (EM) algorithm



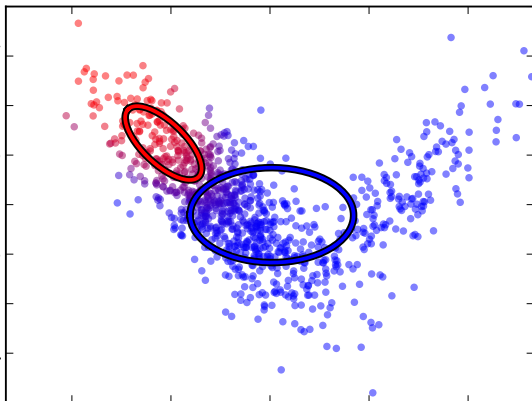
Gaussian Mixture Models

- ▶ Instead of finding just **centroids**, we want to find **weights** and Gaussians **means** and **covariances** of clusters
- ▶ Can be optimized efficiently using the **Expectation-Maximization** (EM) algorithm



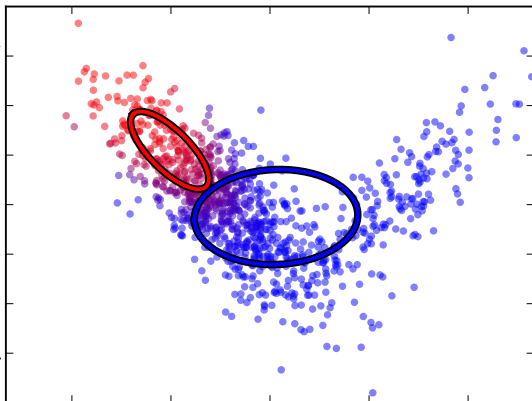
Gaussian Mixture Models

- ▶ Instead of finding just **centroids**, we want to find **weights** and Gaussians **means** and **covariances** of clusters
- ▶ Can be optimized efficiently using the **Expectation-Maximization** (EM) algorithm



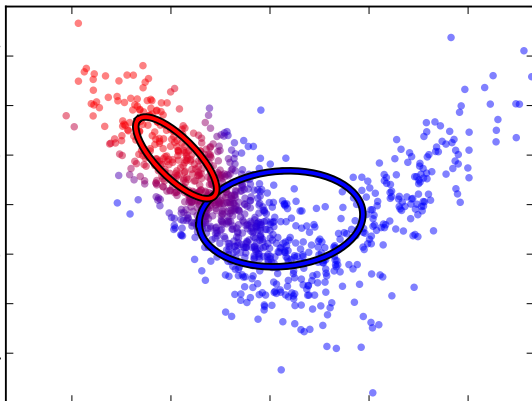
Gaussian Mixture Models

- ▶ Instead of finding just **centroids**, we want to find **weights** and Gaussians **means** and **covariances** of clusters
- ▶ Can be optimized efficiently using the **Expectation-Maximization** (EM) algorithm



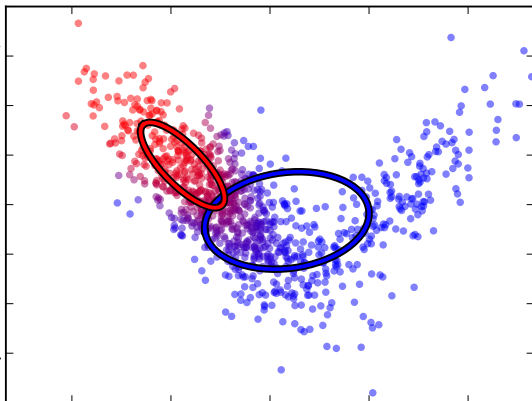
Gaussian Mixture Models

- ▶ Instead of finding just **centroids**, we want to find **weights** and Gaussians **means** and **covariances** of clusters
- ▶ Can be optimized efficiently using the **Expectation-Maximization** (EM) algorithm



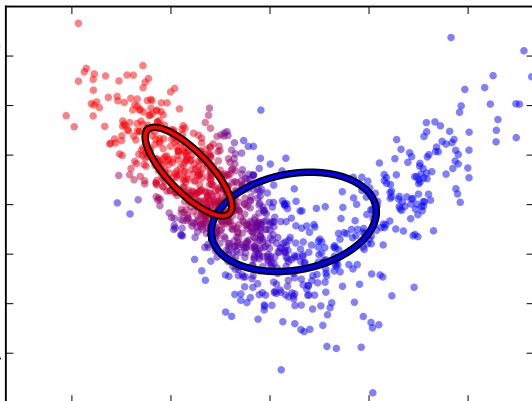
Gaussian Mixture Models

- ▶ Instead of finding just **centroids**, we want to find **weights** and Gaussians **means** and **covariances** of clusters
- ▶ Can be optimized efficiently using the **Expectation-Maximization** (EM) algorithm



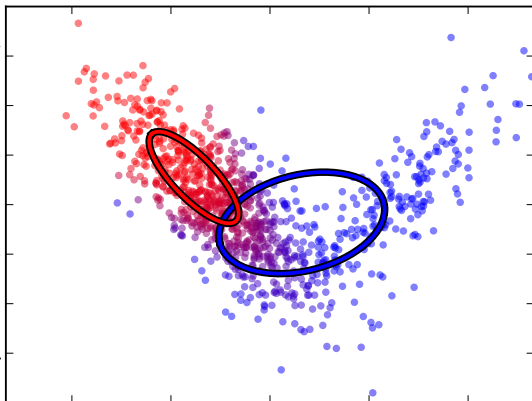
Gaussian Mixture Models

- ▶ Instead of finding just **centroids**, we want to find **weights** and Gaussians **means** and **covariances** of clusters
- ▶ Can be optimized efficiently using the **Expectation-Maximization** (EM) algorithm



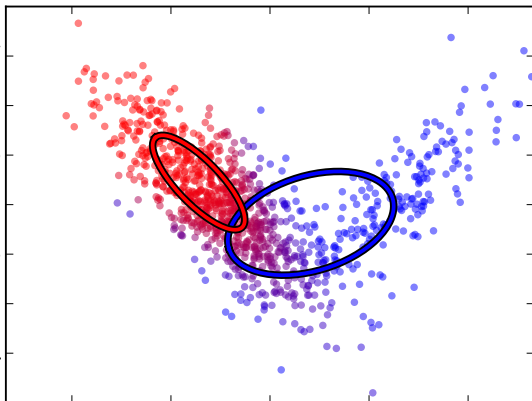
Gaussian Mixture Models

- ▶ Instead of finding just **centroids**, we want to find **weights** and Gaussians **means** and **covariances** of clusters
- ▶ Can be optimized efficiently using the **Expectation-Maximization** (EM) algorithm



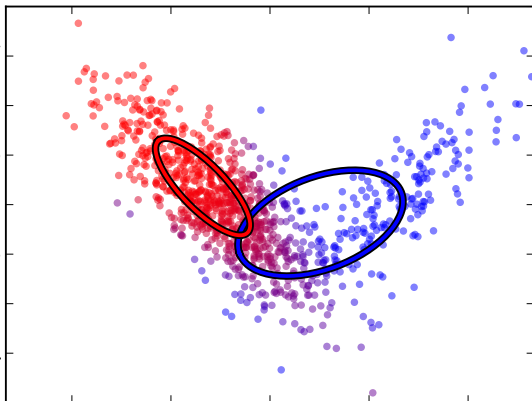
Gaussian Mixture Models

- ▶ Instead of finding just **centroids**, we want to find **weights** and Gaussians **means** and **covariances** of clusters
- ▶ Can be optimized efficiently using the **Expectation-Maximization** (EM) algorithm



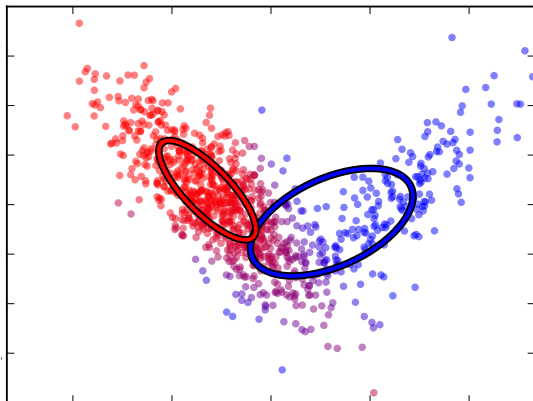
Gaussian Mixture Models

- ▶ Instead of finding just **centroids**, we want to find **weights** and Gaussians **means** and **covariances** of clusters
- ▶ Can be optimized efficiently using the **Expectation-Maximization** (EM) algorithm



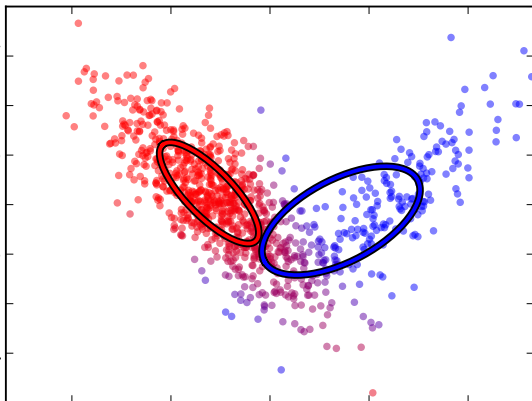
Gaussian Mixture Models

- ▶ Instead of finding just **centroids**, we want to find **weights** and Gaussians **means** and **covariances** of clusters
- ▶ Can be optimized efficiently using the **Expectation-Maximization** (EM) algorithm



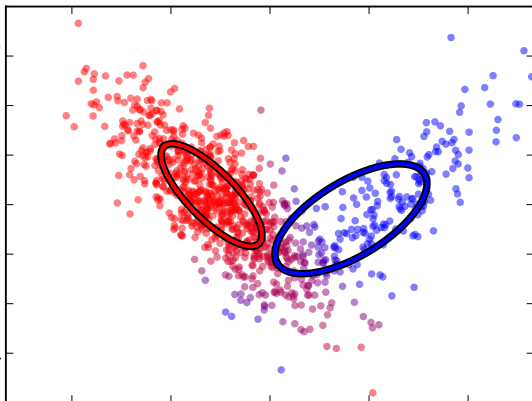
Gaussian Mixture Models

- ▶ Instead of finding just **centroids**, we want to find **weights** and Gaussians **means** and **covariances** of clusters
- ▶ Can be optimized efficiently using the **Expectation-Maximization** (EM) algorithm



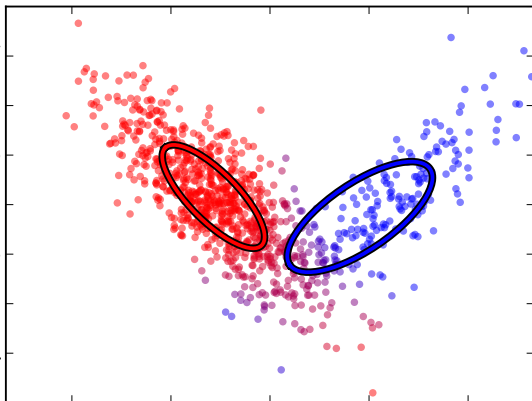
Gaussian Mixture Models

- ▶ Instead of finding just **centroids**, we want to find **weights** and Gaussians **means** and **covariances** of clusters
- ▶ Can be optimized efficiently using the **Expectation-Maximization** (EM) algorithm



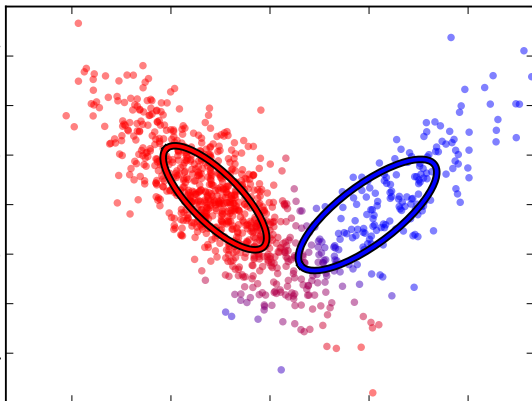
Gaussian Mixture Models

- ▶ Instead of finding just **centroids**, we want to find **weights** and Gaussians **means** and **covariances** of clusters
- ▶ Can be optimized efficiently using the **Expectation-Maximization** (EM) algorithm



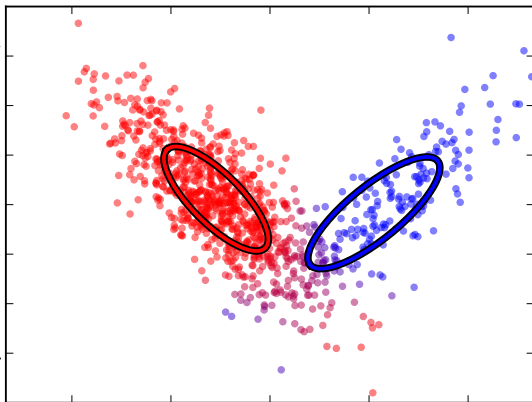
Gaussian Mixture Models

- ▶ Instead of finding just **centroids**, we want to find **weights** and Gaussians **means** and **covariances** of clusters
- ▶ Can be optimized efficiently using the **Expectation-Maximization** (EM) algorithm



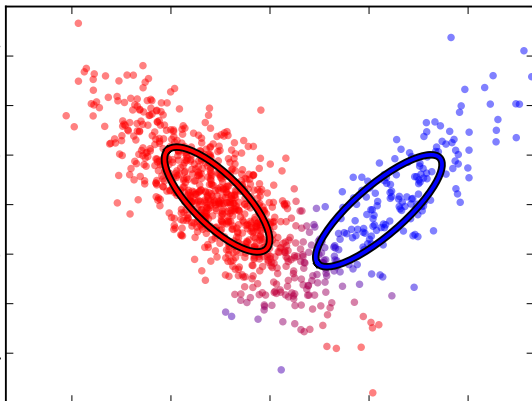
Gaussian Mixture Models

- ▶ Instead of finding just **centroids**, we want to find **weights** and Gaussians **means** and **covariances** of clusters
- ▶ Can be optimized efficiently using the **Expectation-Maximization** (EM) algorithm



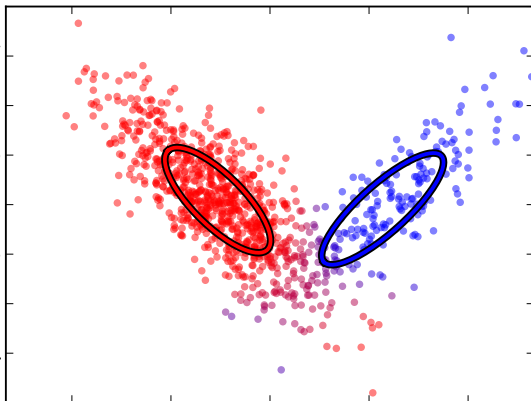
Gaussian Mixture Models

- ▶ Instead of finding just **centroids**, we want to find **weights** and Gaussians **means** and **covariances** of clusters
- ▶ Can be optimized efficiently using the **Expectation-Maximization** (EM) algorithm



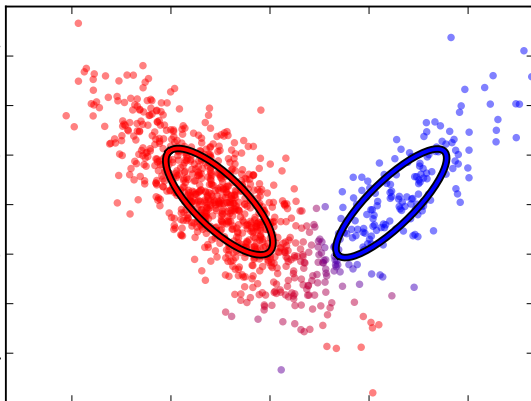
Gaussian Mixture Models

- ▶ Instead of finding just **centroids**, we want to find **weights** and Gaussians **means** and **covariances** of clusters
- ▶ Can be optimized efficiently using the **Expectation-Maximization** (EM) algorithm



Gaussian Mixture Models

- ▶ Instead of finding just **centroids**, we want to find **weights** and Gaussians **means** and **covariances** of clusters
- ▶ Can be optimized efficiently using the **Expectation-Maximization** (EM) algorithm



Gaussian Mixture Models / Expectation-Maximization

E-M proceeds by:

- ▶ **E**: compute probability of drawing each **data point** i from each **cluster** or **component** k :

$$z_{i,k} = a_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k)$$

where a_k is the cluster **weight**, and μ_k, Σ_k are its mean and covariance.

- ▶ **M**: compute new component weights and parameters, weighting by *relative* component-weights

Gaussian Mixture Models / Expectation-Maximization

- ▶ E-M “update equations” (M step) for Gaussian mixtures:

$$z'_{i,k} = \frac{z_{i,k}}{\sum_k z_{i,k}} \quad (1)$$

$$a'_k = \sum_k z'_{i,k} \quad (2)$$

$$\mu'_{i,k} = \frac{1}{a'_k} z'_{i,k} x_i \quad (3)$$

$$\Sigma'_{i,k} = \frac{1}{a'_k} z'_{i,k} (x_i - \mu_k)(x_i - \mu_k)^T \quad (4)$$

Mixture Models / Foreground-Background Models

- ▶ Often, **junk** or **interlopers** can get into your sample (of galaxies, stars, etc)
- ▶ If not dealt with, can strongly skew results (outliers)
- ▶ **Model** the objects you don't care about (background) **as well as** the ones you do care about (foreground)
- ▶ Background model can be a regular Gaussian component, or a flat (uniform) probability distribution

Problems with Gaussian Mixture Models

- ▶ Choosing the number of components
- ▶ Measurement noise not considered
- ▶ Local optimization