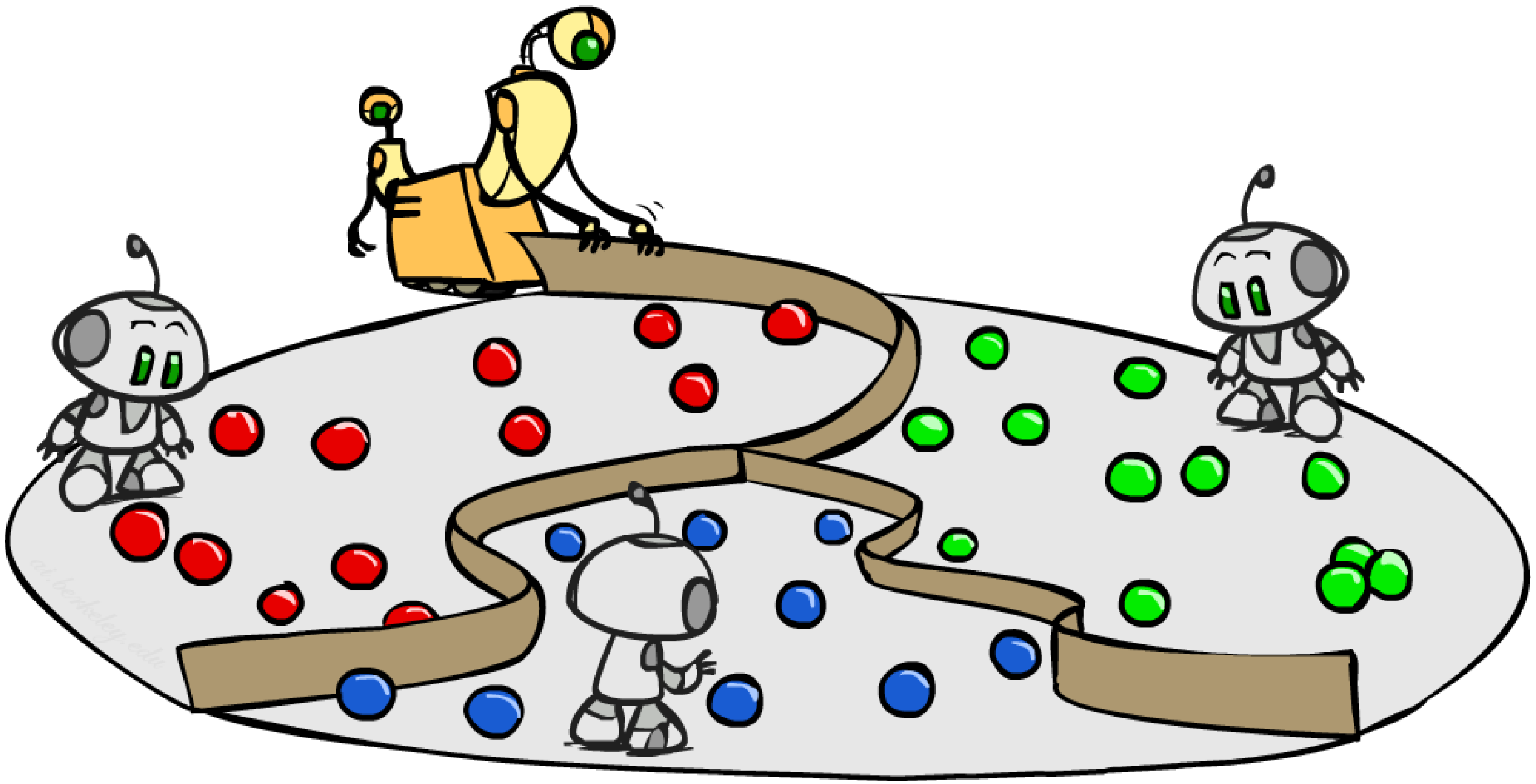
Unsupervised Machine Learning



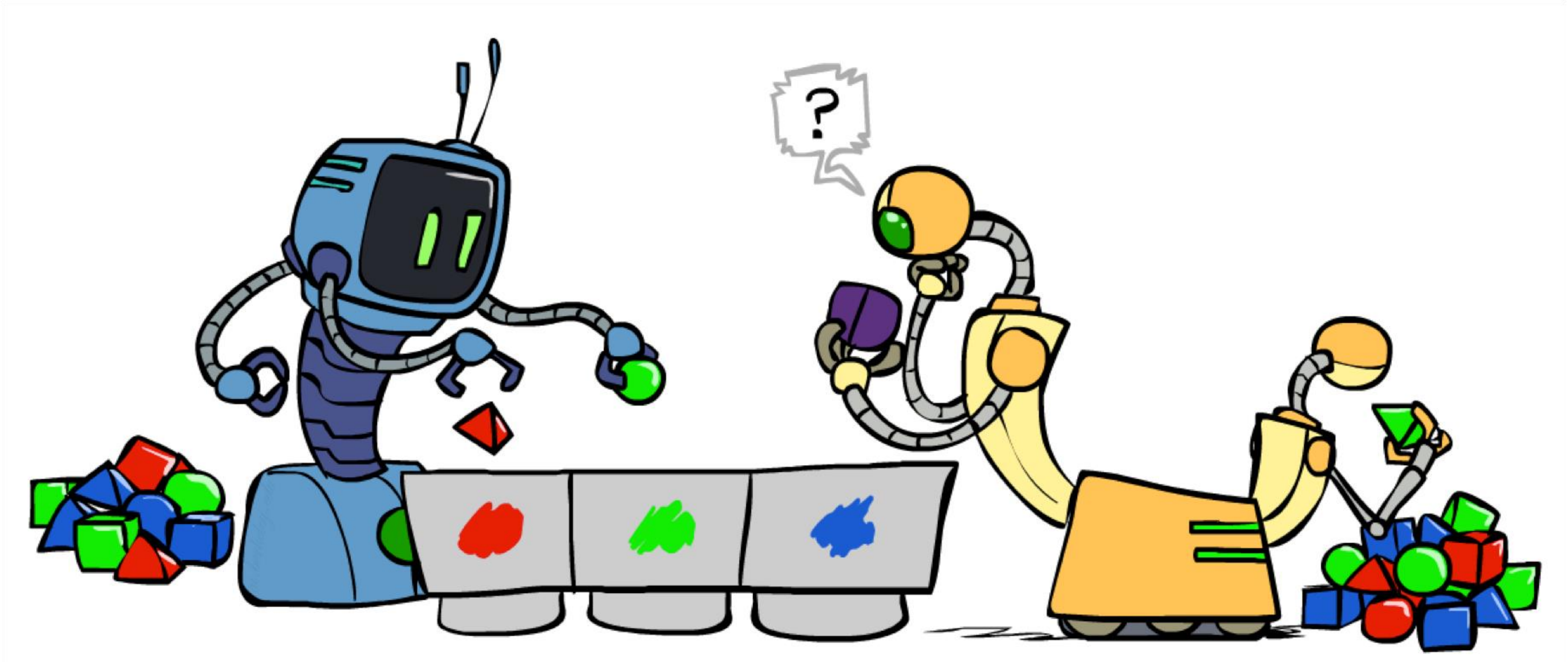
AIMA Chapter 20

[Adapted from slides by Dan Klein and Pieter Abbeel at UC Berkeley and from Daniel Weld, Carlos Guestrin, & Luke Zettlemoyer at U Washington]

Types of Learning

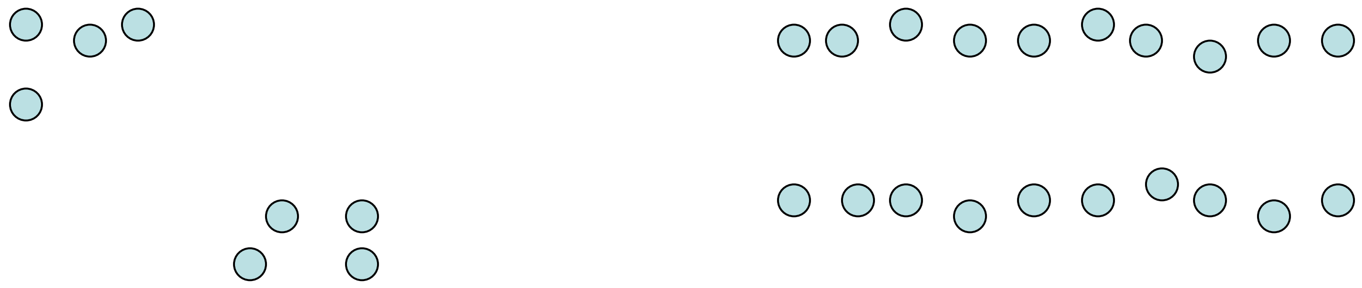
- Supervised learning
  - Training data includes desired outputs
- Unsupervised learning
  - Training data does not include desired outputs
- Semi-supervised learning
  - Training data includes a few desired outputs
- Reinforcement learning
  - Rewards from sequence of actions

Clustering



Clustering

- Basic idea: group together similar instances
- Example: 2D point patterns



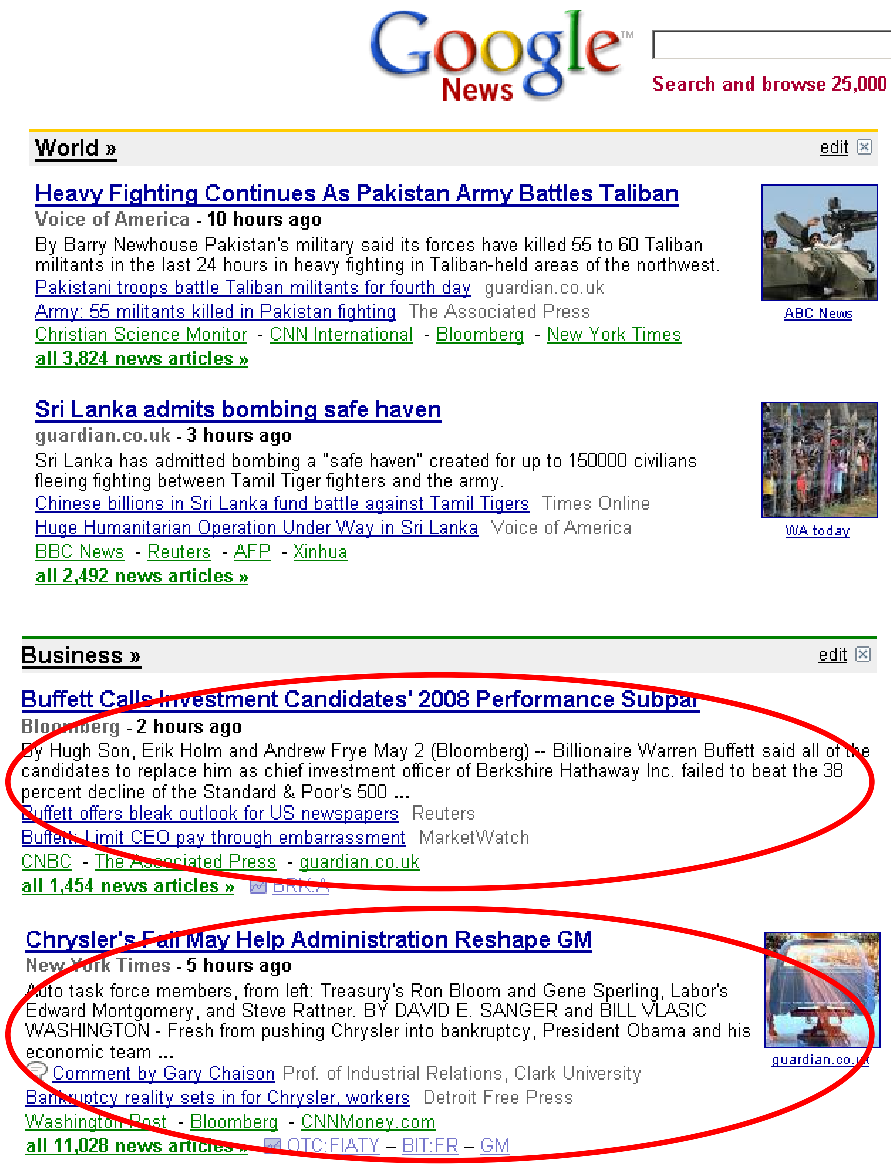
- What could "similar" mean?
  - One option: small (squared) Euclidean distance

$$\text{dist}(x, y) = (x-y)^\top (x-y) = \sum_i (x_i - y_i)^2$$

  - Many other options, often domain specific

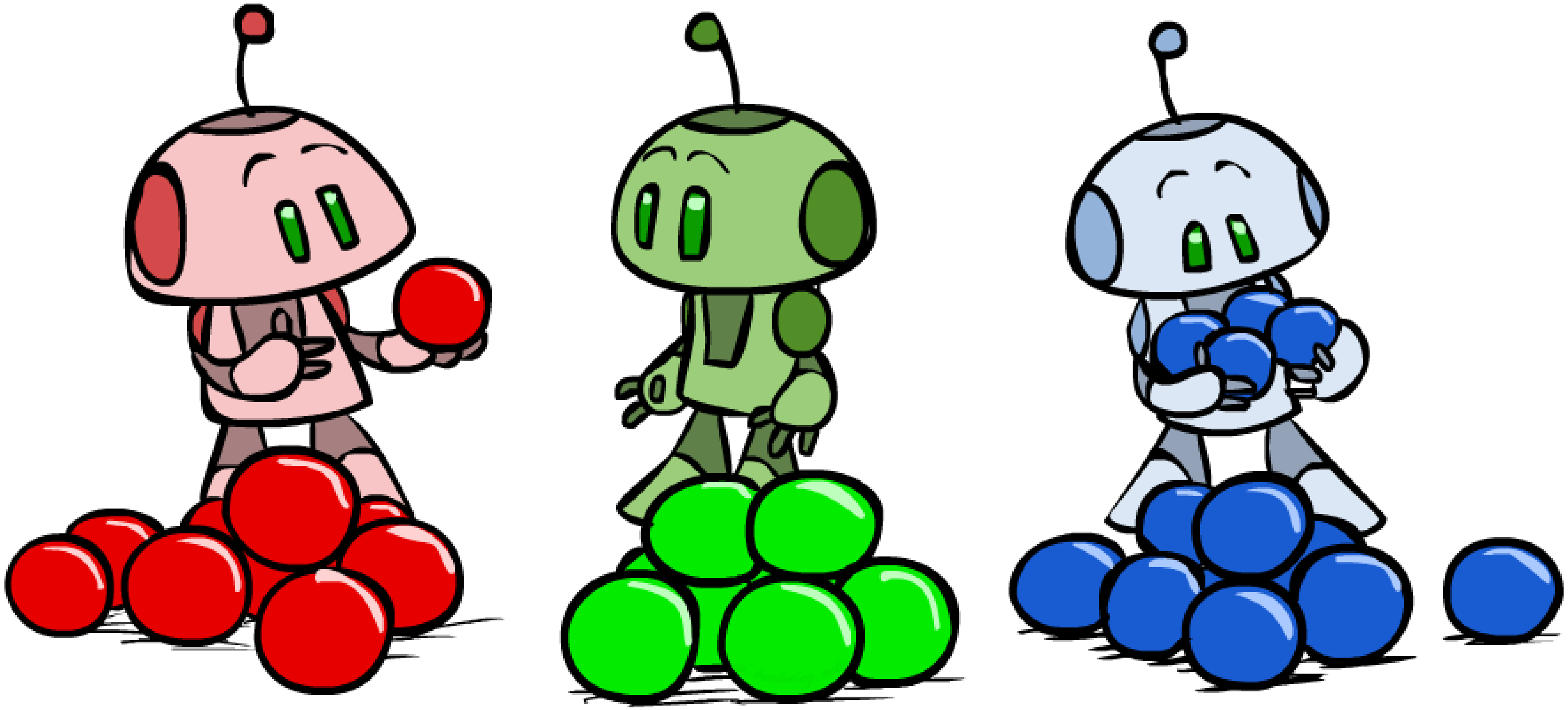
Clustering

- Applications
  - Group emails
  - Group search results
  - Find categories of customers
  - Detect anomalous program executions



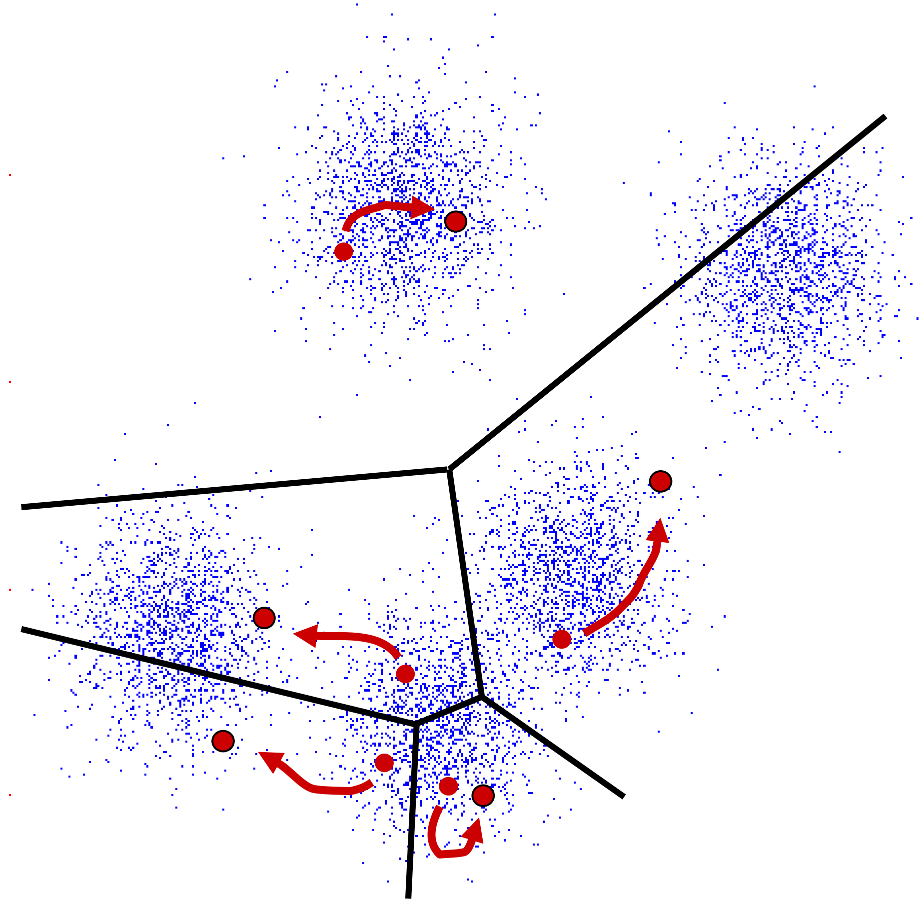
Story groupings: unsupervised clustering

K-Means



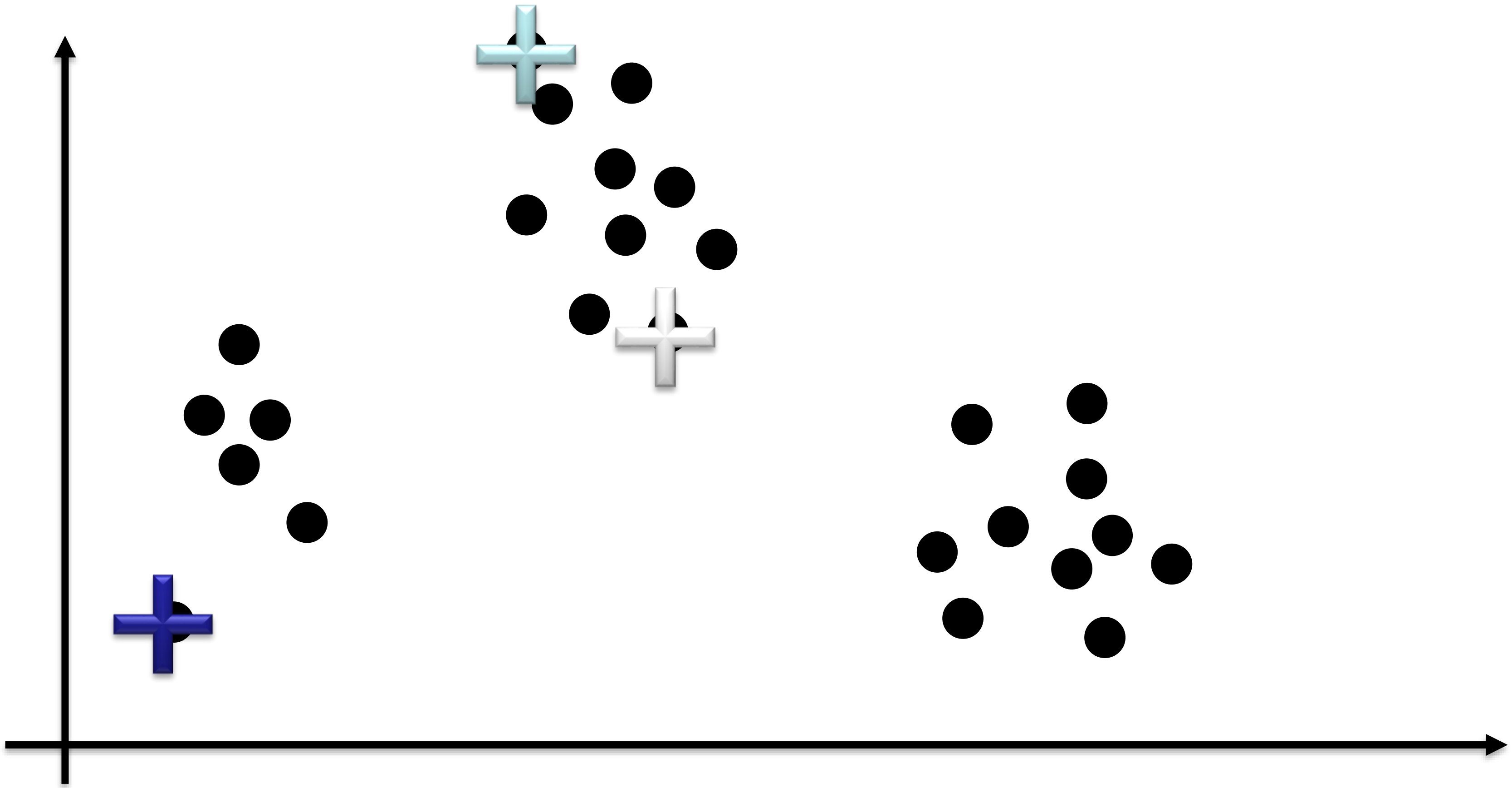
K-Means

- An iterative clustering algorithm
  - Pick K random points as cluster centers (means)
  - Alternate:
    - Assign data instances to closest mean
    - Assign each mean to the average of its assigned points
  - Stop when no points' assignments change



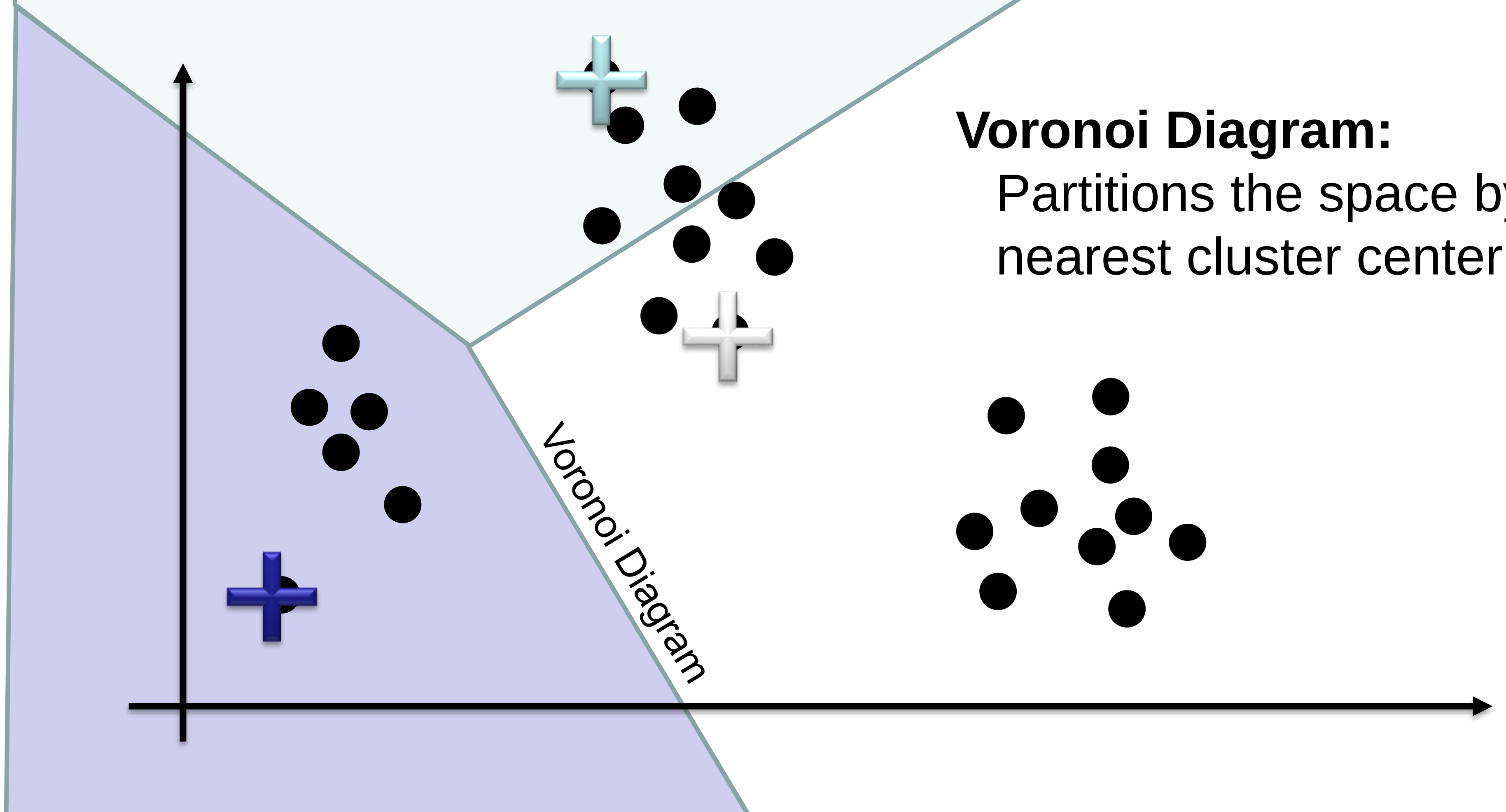
K-Means Example

- Pick an initial set of K points as cluster centers



K-Means Example

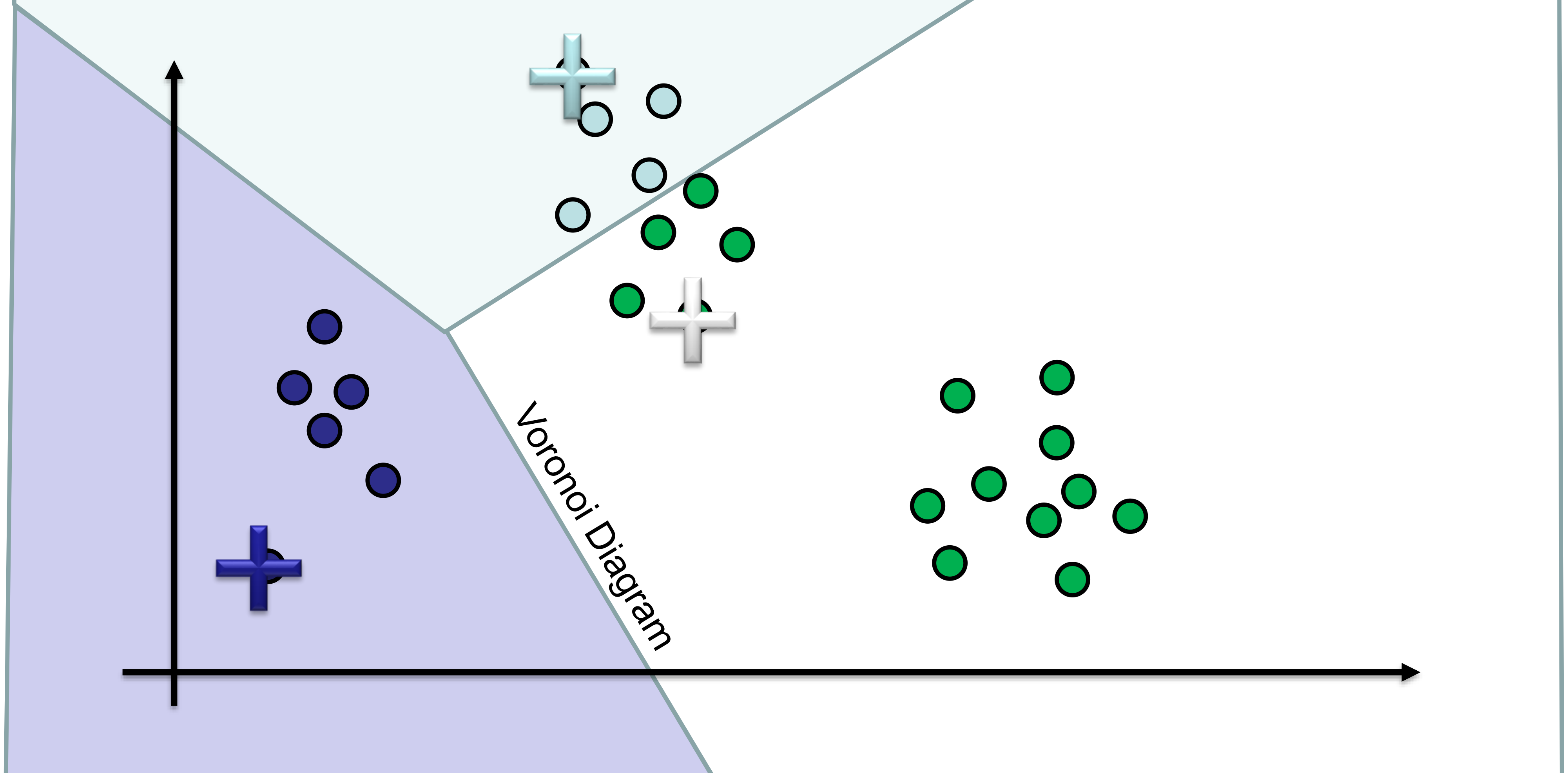
- For each data point find the nearest cluster center



Voronoi Diagram:
Partitions the space by nearest cluster center

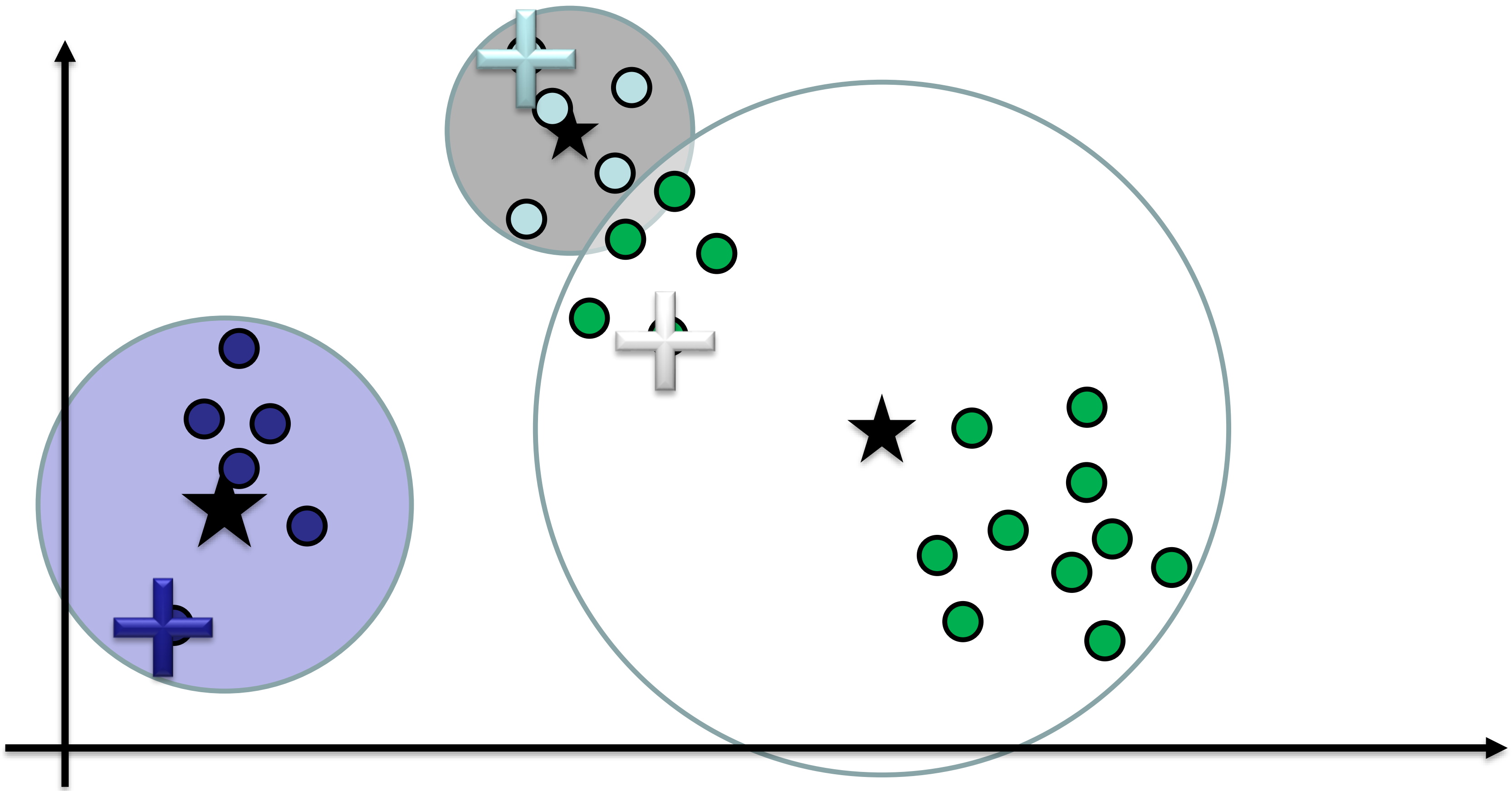
K-Means Example

- For each data point find the nearest cluster center



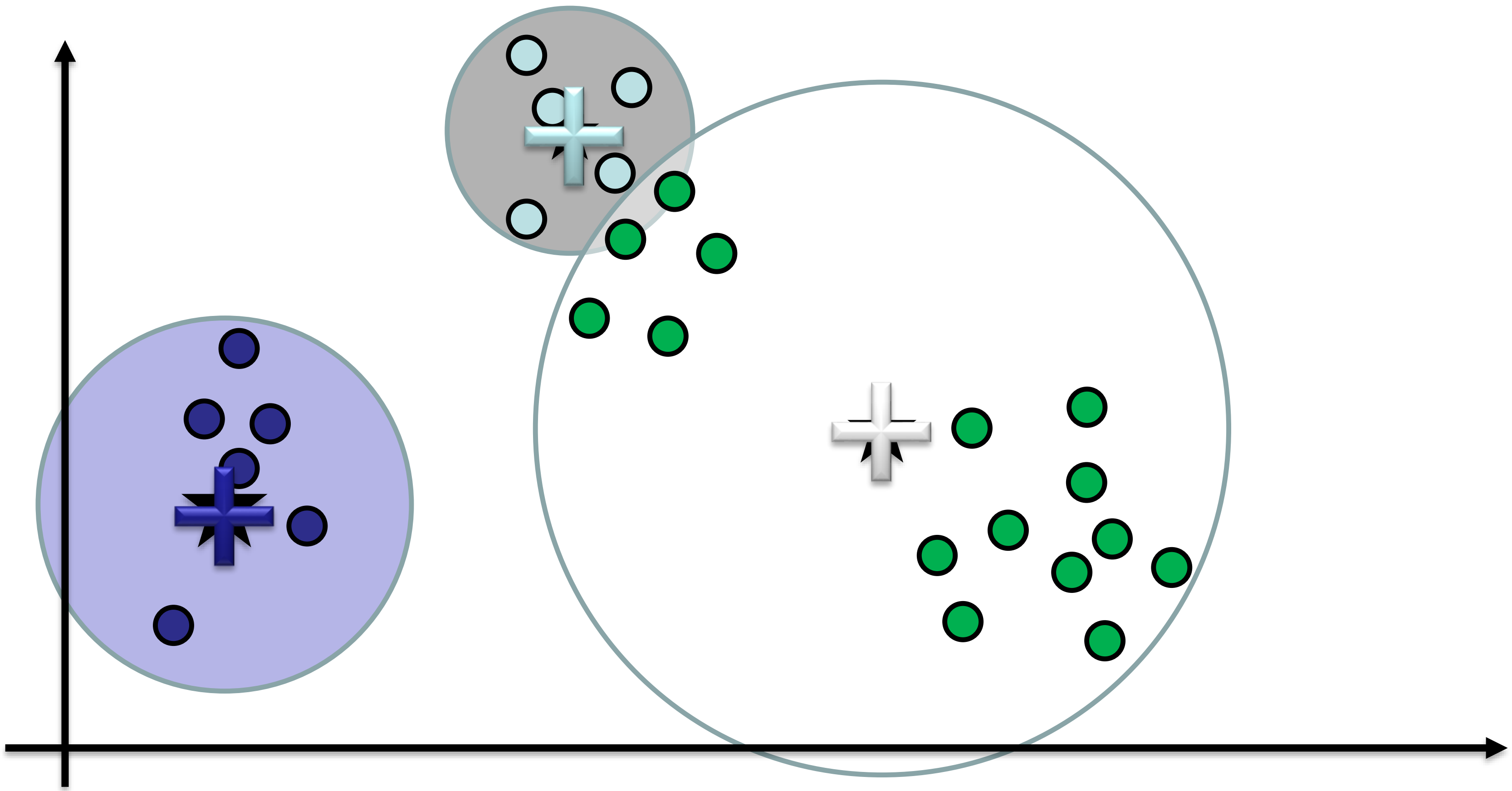
K-Means Example

- Compute mean of points in each "cluster"



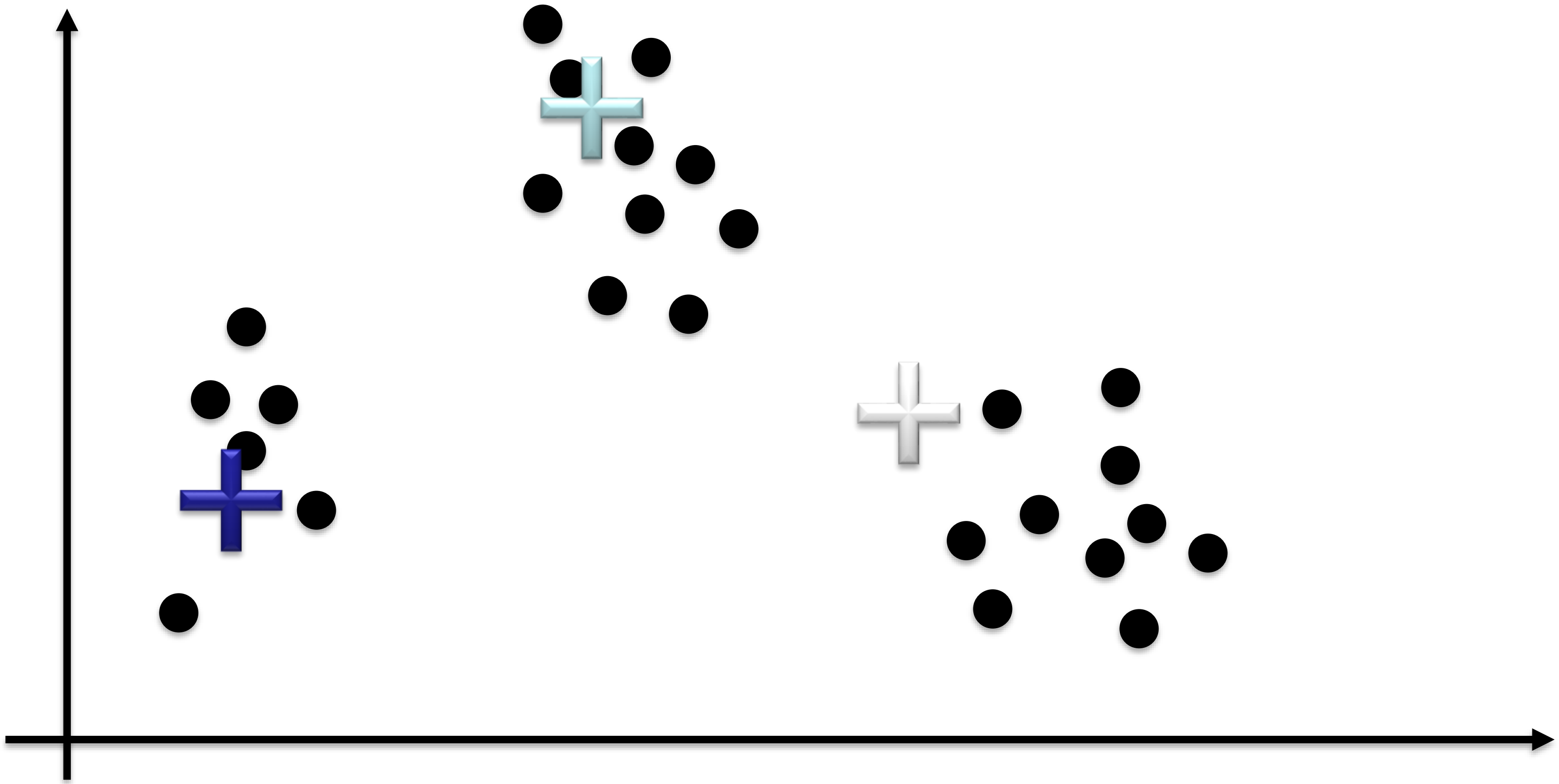
K-Means Example

- Adjust cluster centers to be the mean of the cluster



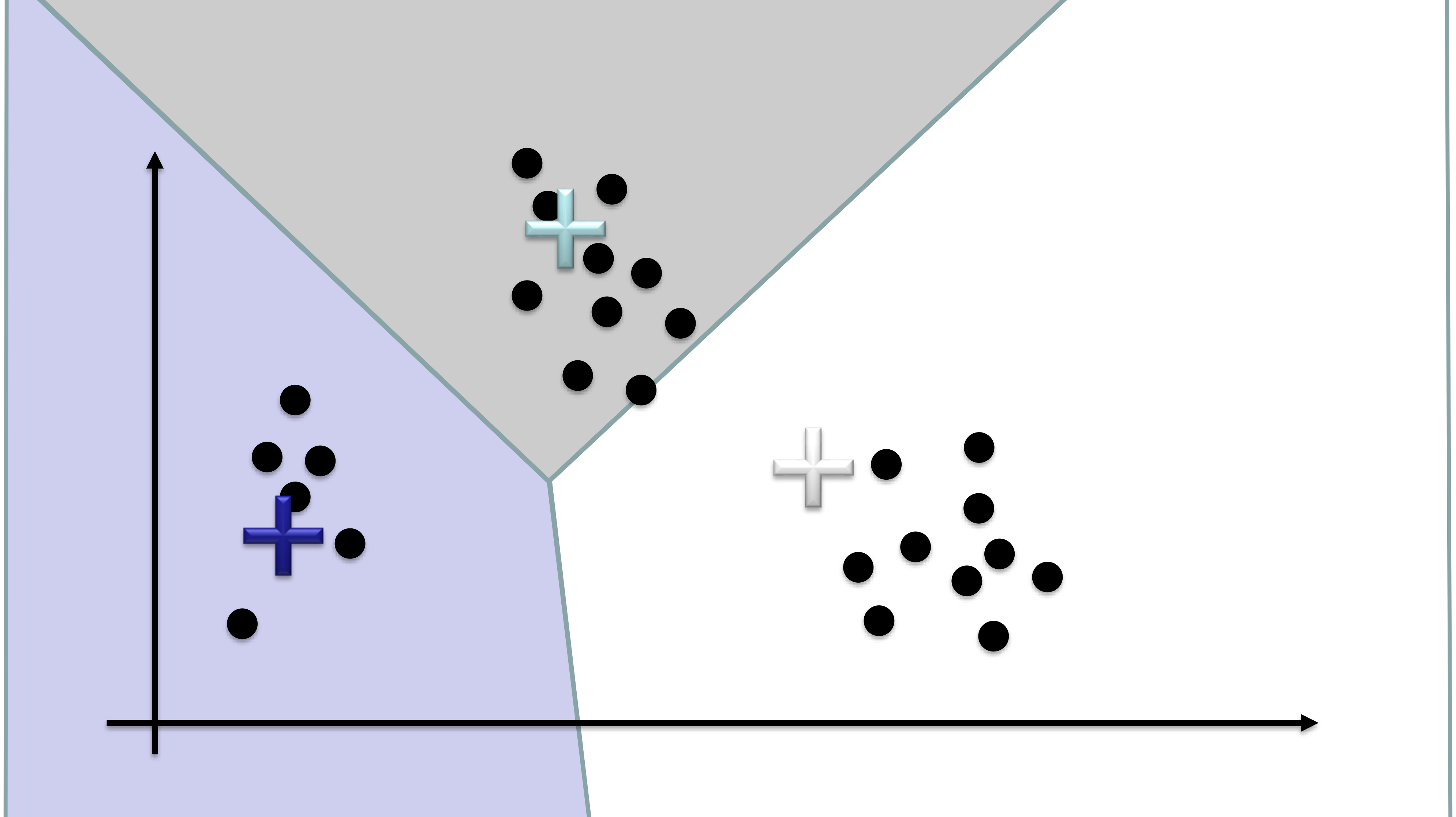
K-Means Example

- Improved?
- Repeat



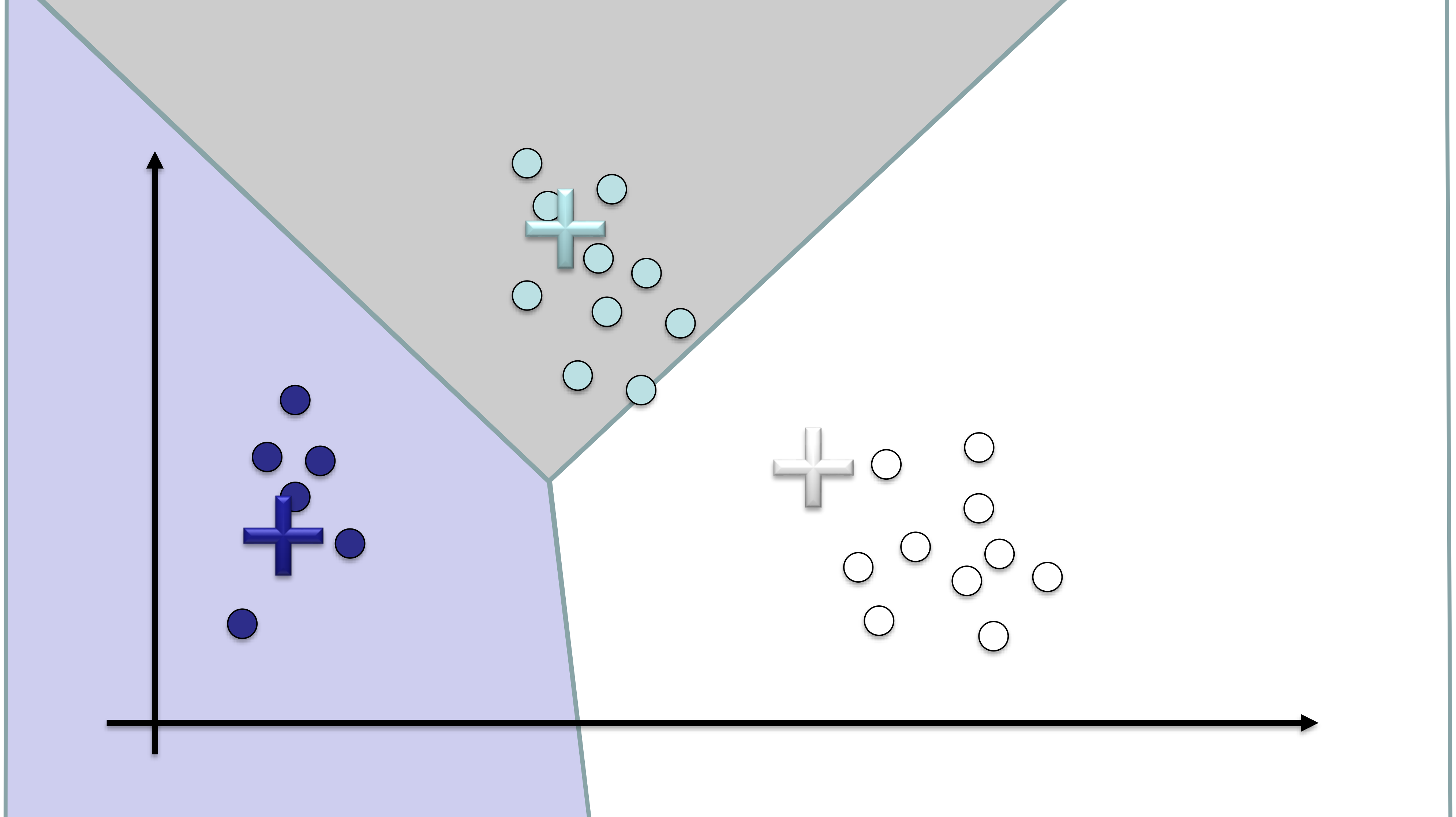
K-Means Example

- Assign Points



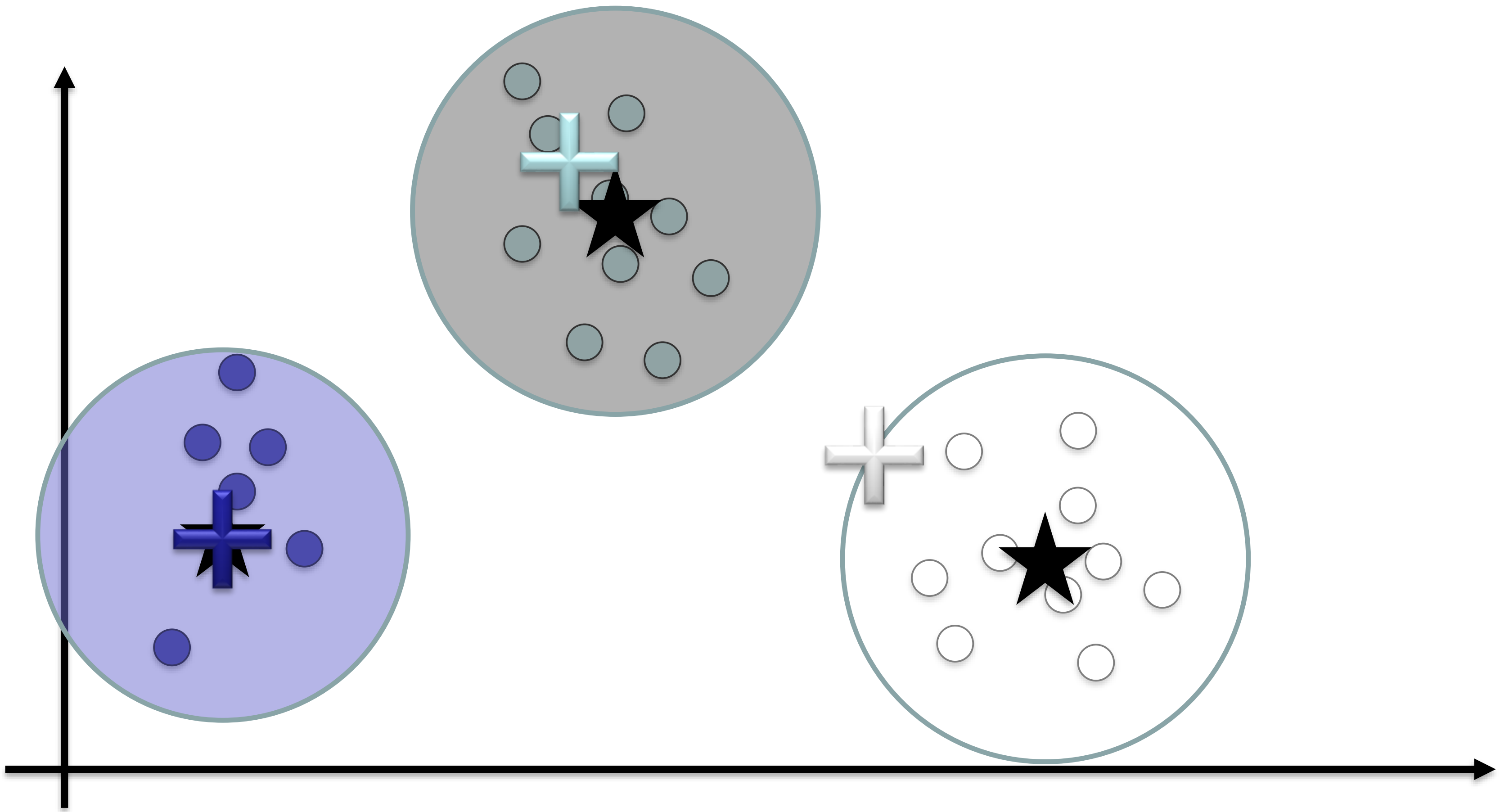
K-Means Example

- Assign Points



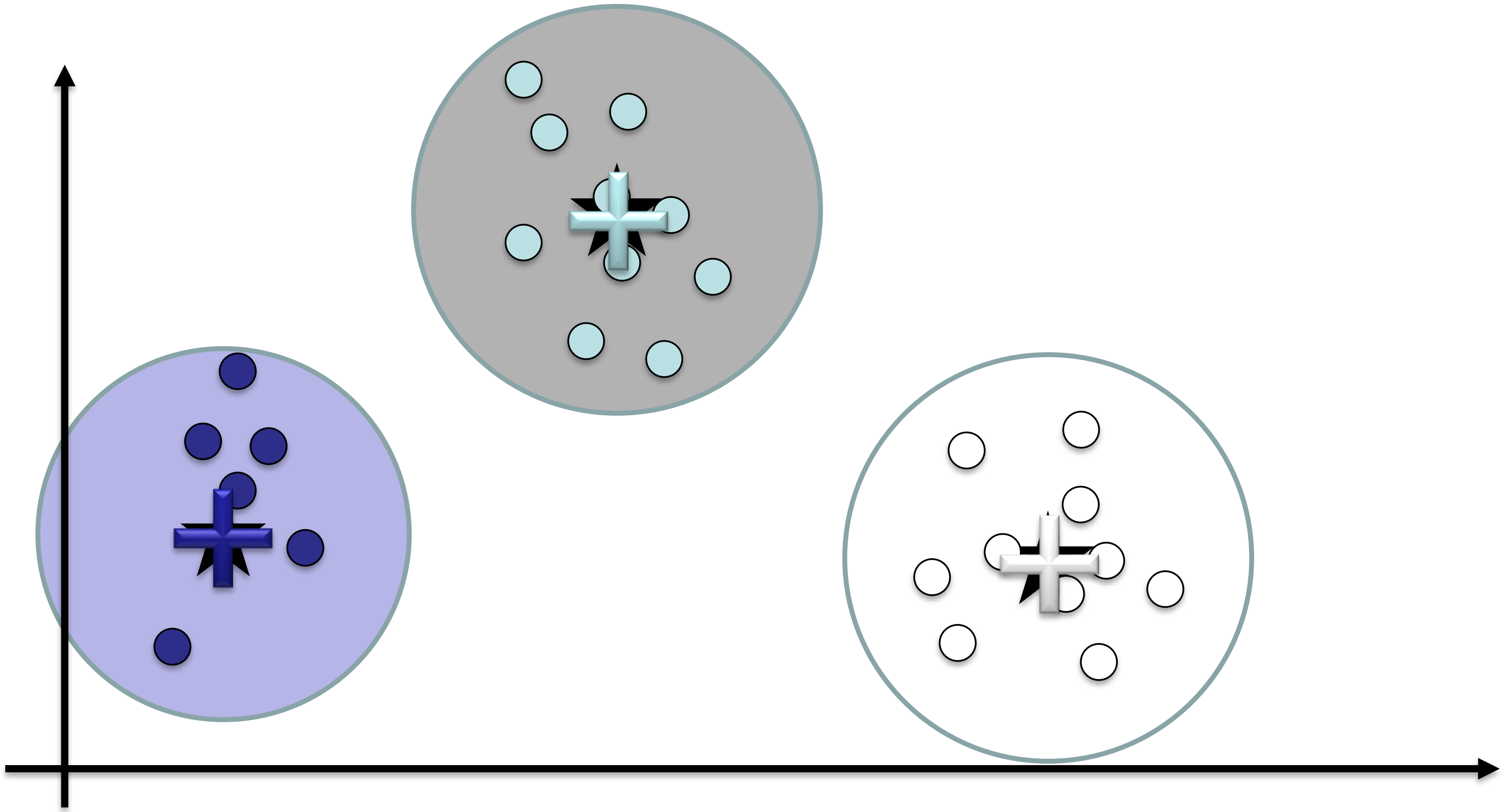
K-Means Example

- Compute cluster means



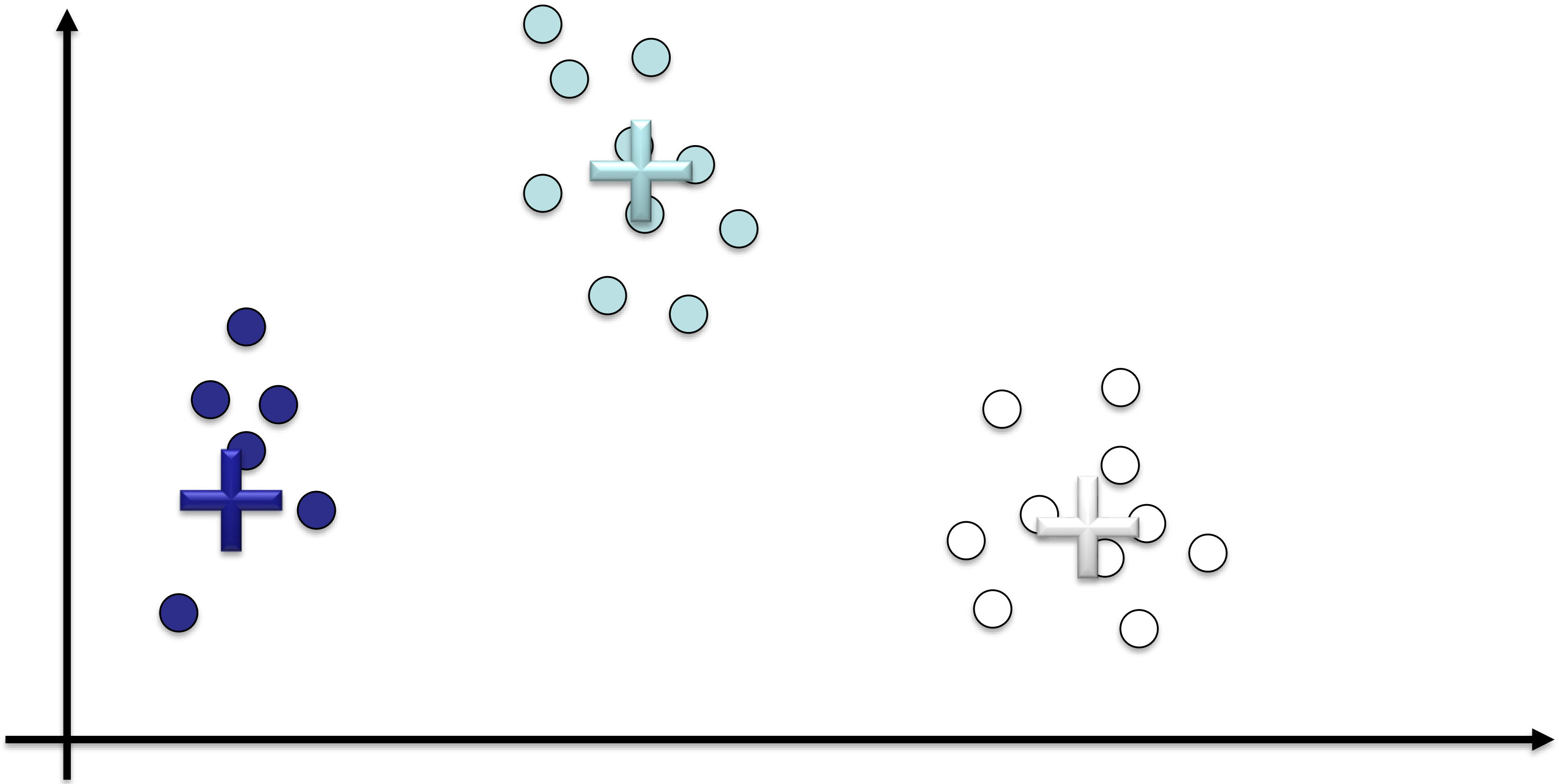
K-Means Example

- Update cluster centers



K-Means Example

- Repeat?
  - If nothing changes → Converged!



K-Means as Optimization

- Consider the total distance to the means:

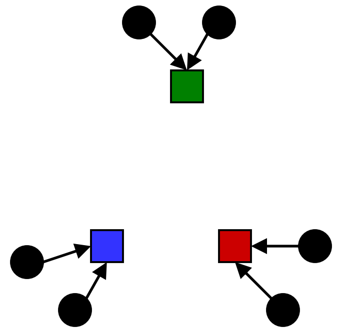
$$\phi(\{x_i\}, \{a_i\}, \{c_k\}) = \sum_i \text{dist}(x_i, c_{a_i})$$

points

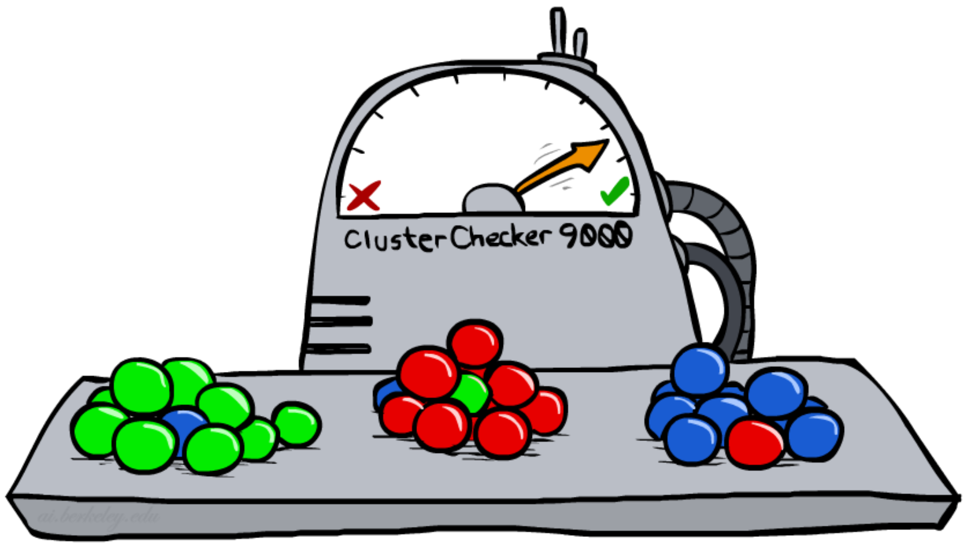
assignments

means

squared Euclidean distance



- Two stages each iteration:
  - Update assignments: fix means c, change assignments a
  - Update means: fix assignments a, change means c
- Each step cannot increase phi



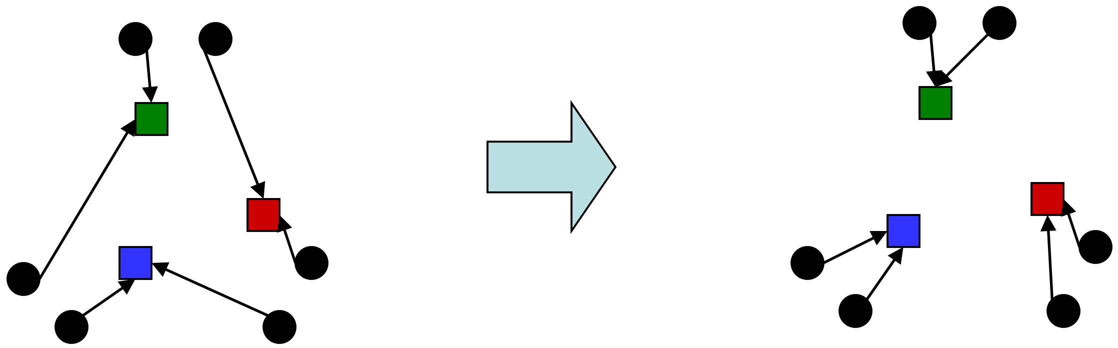
Phase I: Update Assignments

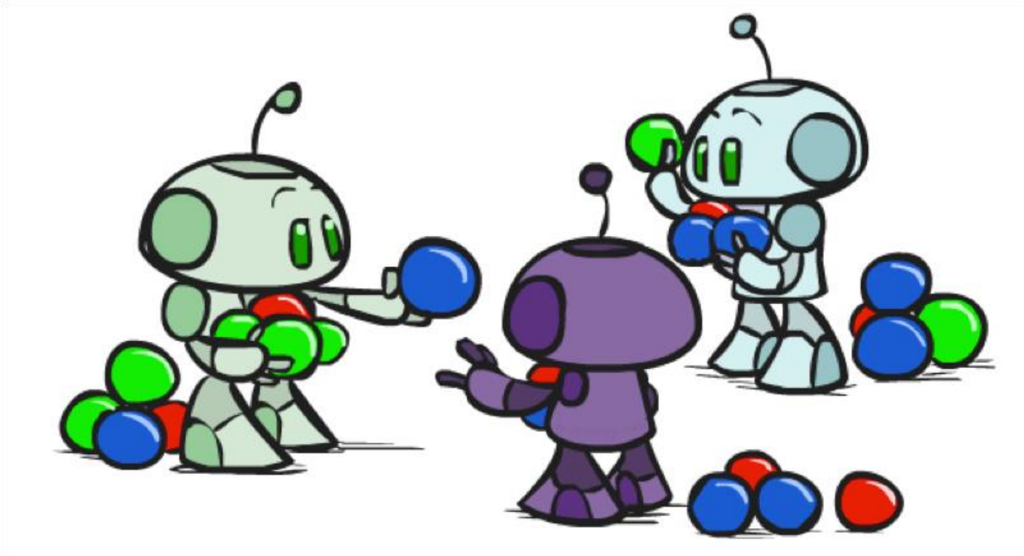
- For each point, re-assign to closest mean:

$$a_i = \underset{k}{\operatorname{argmin}} \operatorname{dist}(x_i, c_k)$$

- Cannot increase total distance phi!

$$\phi(\{x_i\}, \{a_i\}, \{c_k\}) = \sum_i \operatorname{dist}(x_i, c_{a_i})$$

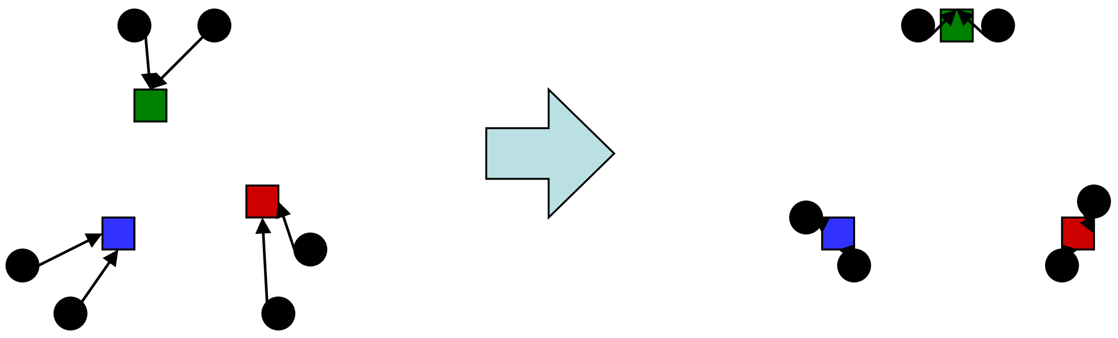




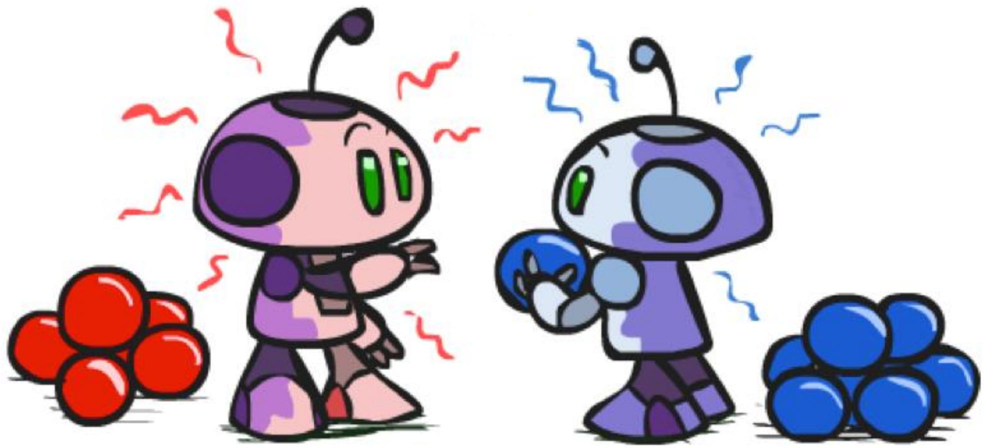
Phase II: Update Means

- Move each mean to the average of its assigned points:

$$c_k = \frac{1}{|\{i : a_i = k\}|} \sum_{i:a_i=k} x_i$$

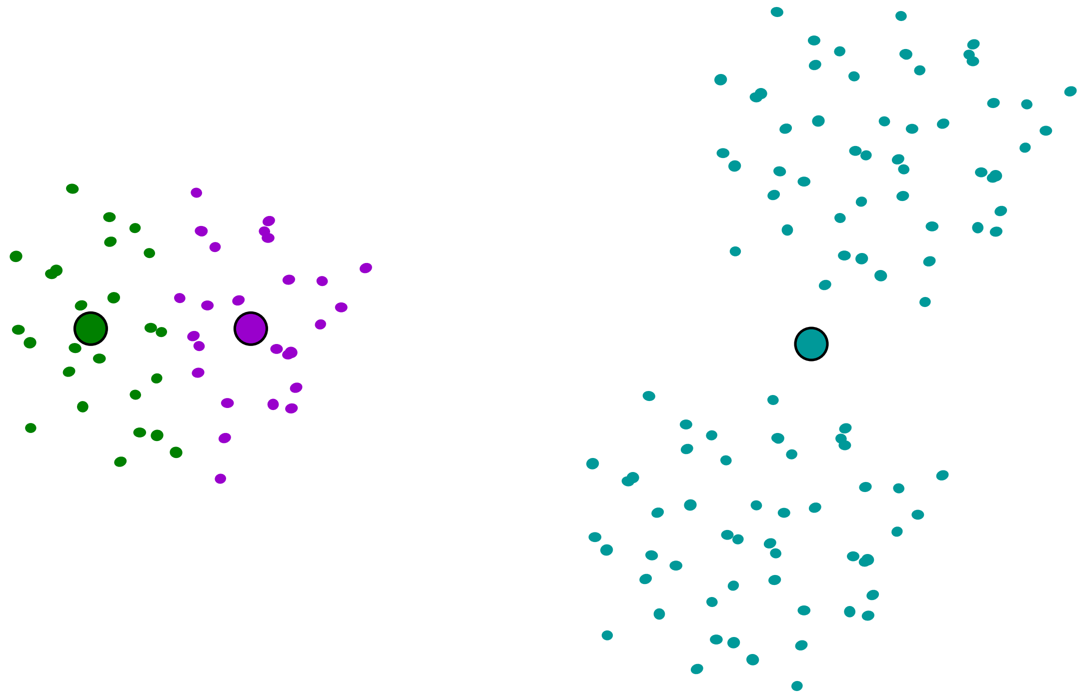


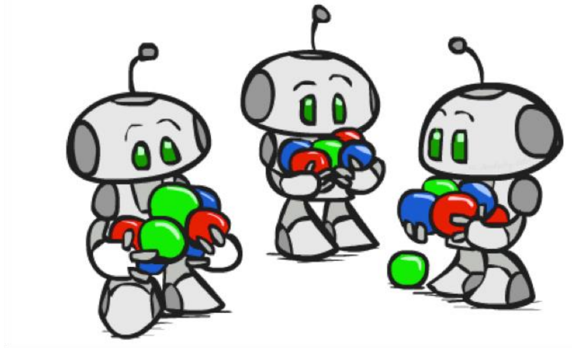
- Also cannot increase total distance
  - Fun fact: the point y with minimum squared Euclidean distance to a set of points {x} is their mean

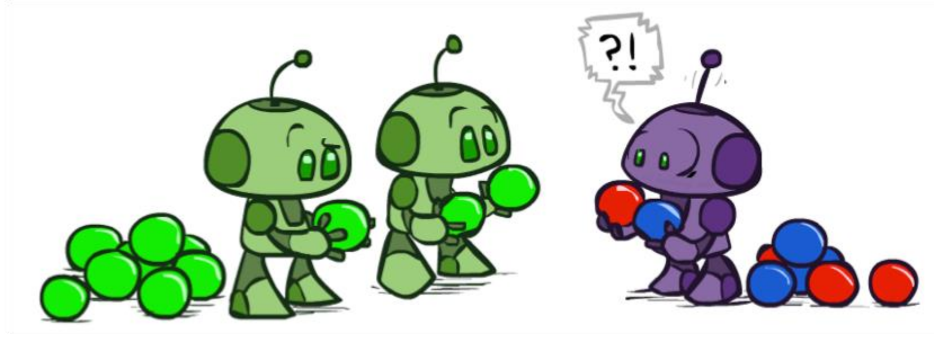


Initialization

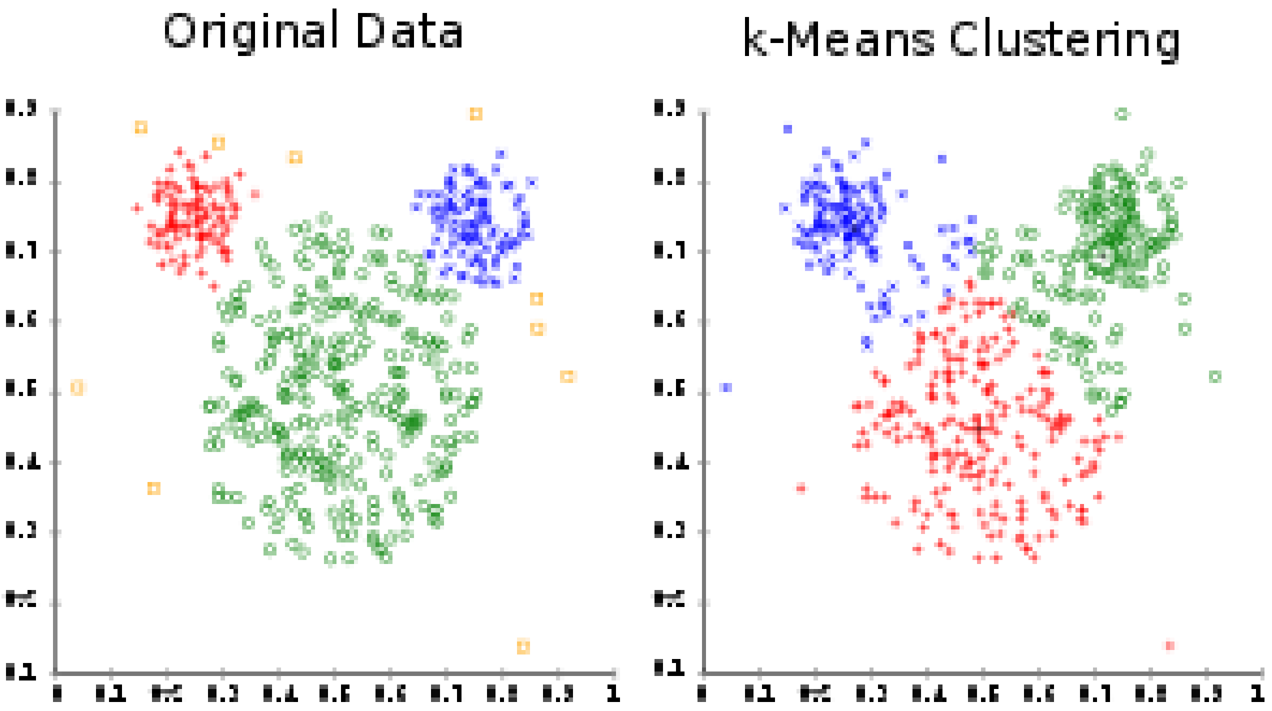
- K-means is non-deterministic
  - Requires initial means
  - It does matter what you pick!
  - What can go wrong?
    - Local optima



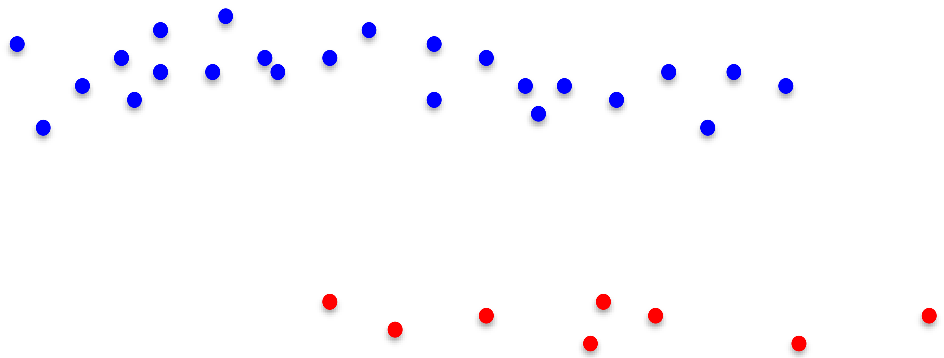




Inductive Bias



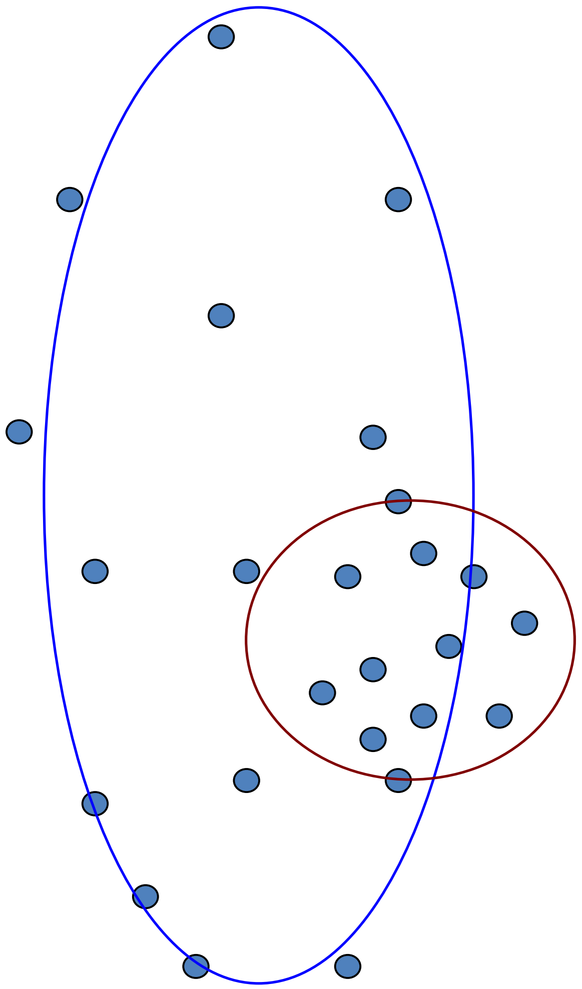
Equally Sized Clusters



Circular Clusters

Expectation-Maximization (EM)

Problems with k-means



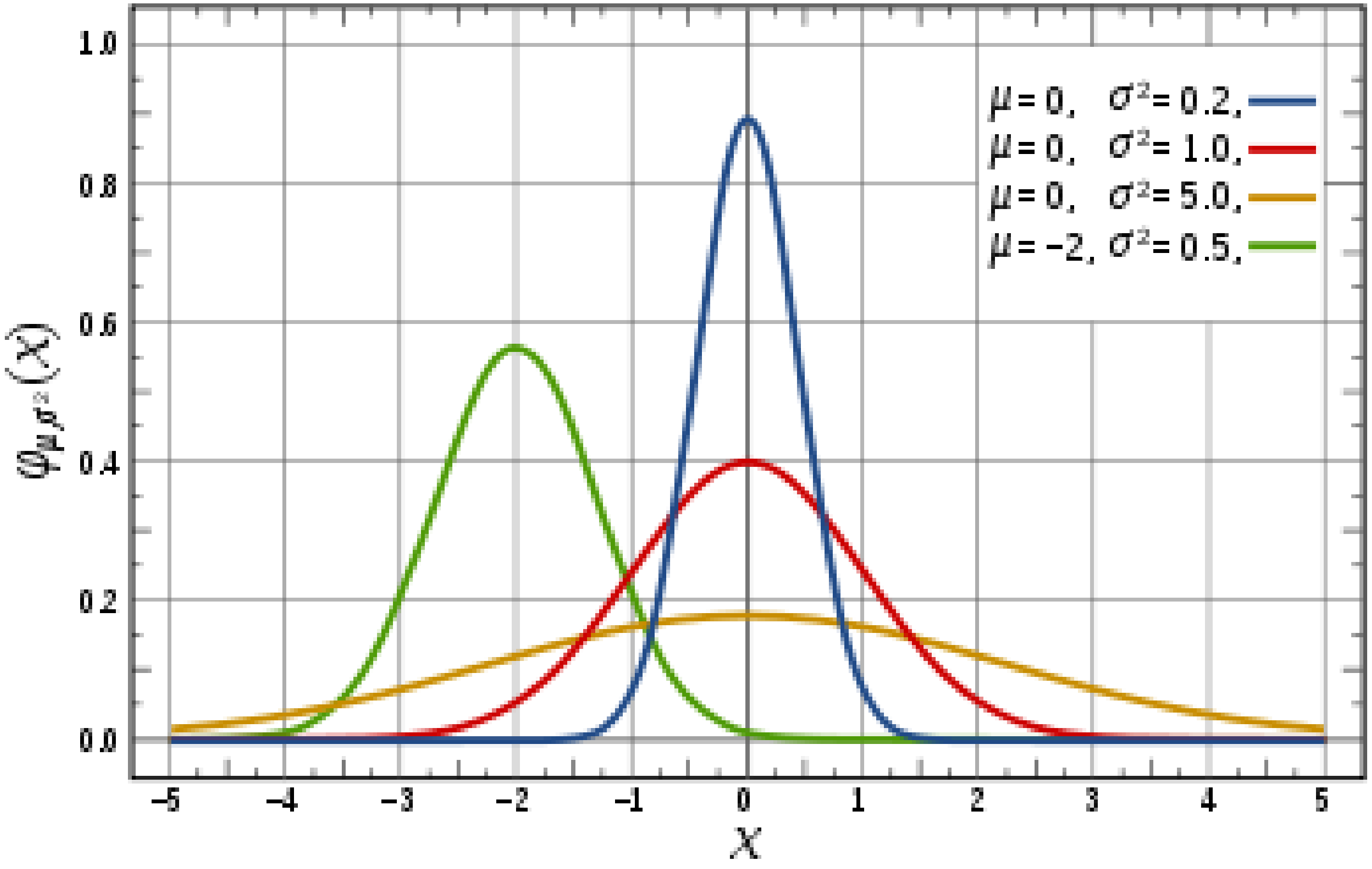
- Assigning data to closest centers
  - But some clusters may be “wider” than others
  - Distances can be deceiving!
- Hard Assignments
  - But clusters may overlap

Probabilistic Clustering

- Try a probabilistic model!
  - allows overlaps, clusters of different sizes/shapes, etc.

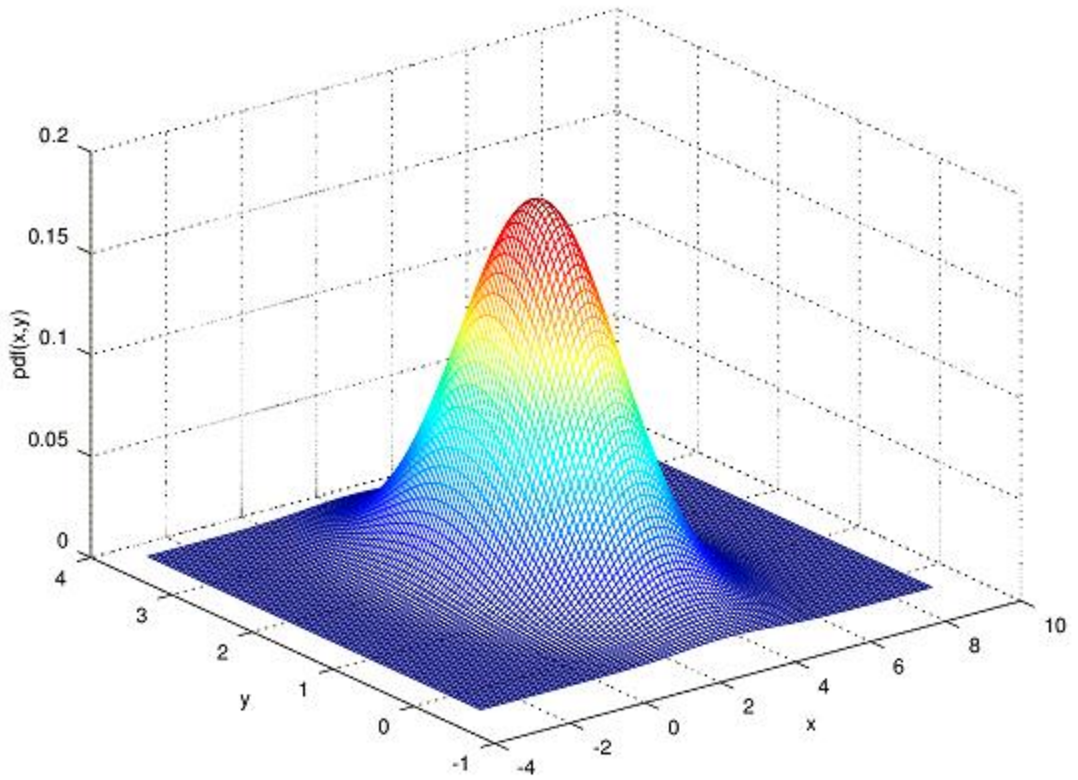
- Gaussian mixture model (GMM)
  - also called Mixture of Gaussians

Review: Gaussians



$$P(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

Multivariate Gaussians

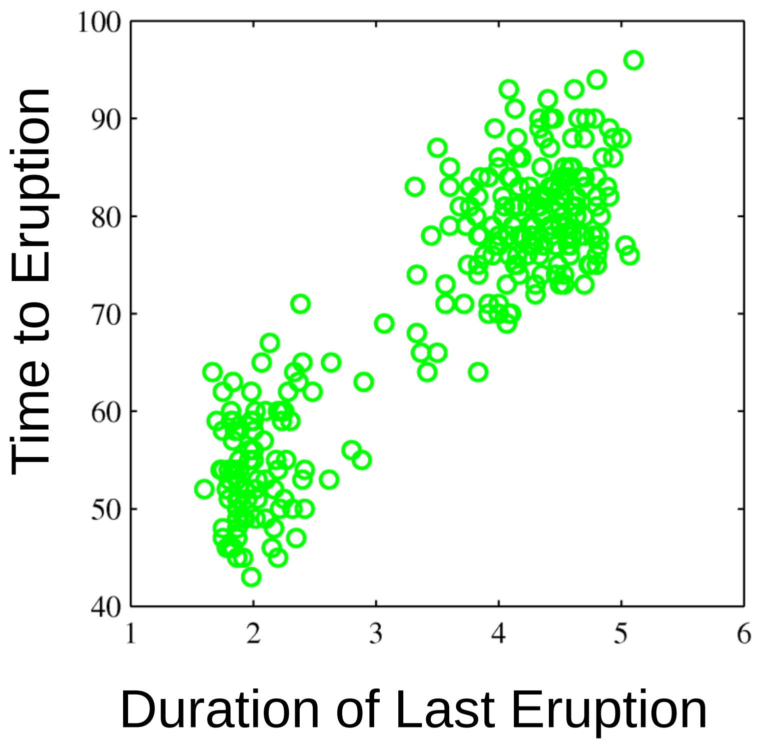


Covariance matrix Σ: degree to which x_i vary together

$$P(X=\mathbf{x}) = \frac{1}{(2\pi)^{m/2}\|\Sigma\|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right]$$

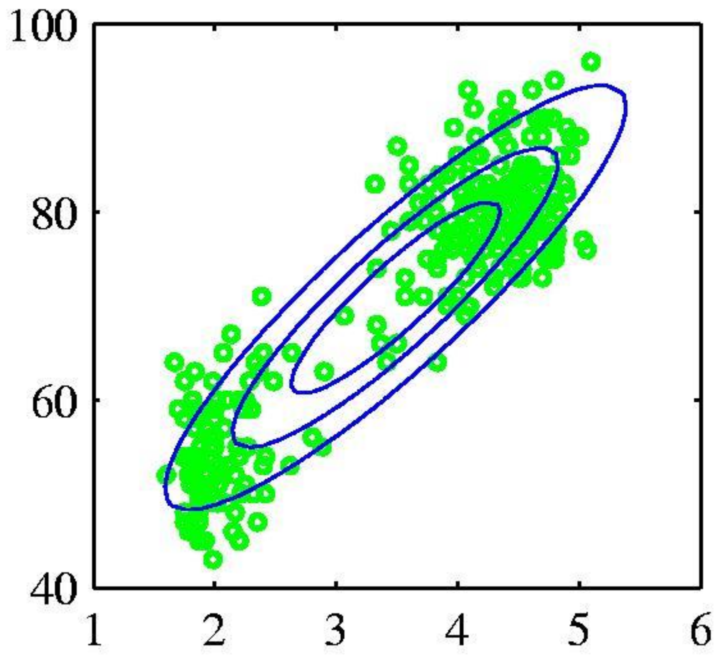
Mixtures of Gaussians

- Old Faithful Data Set

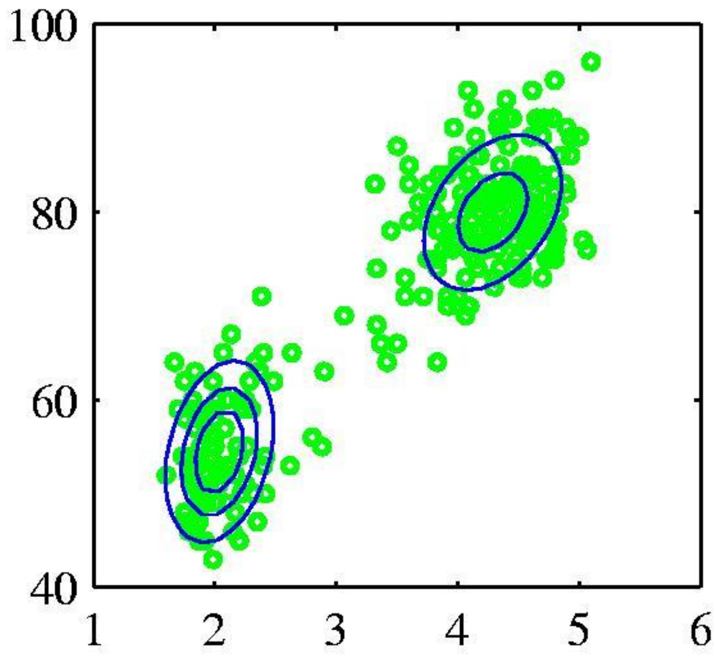


Mixtures of Gaussians

- Old Faithful Data Set



Single Gaussian



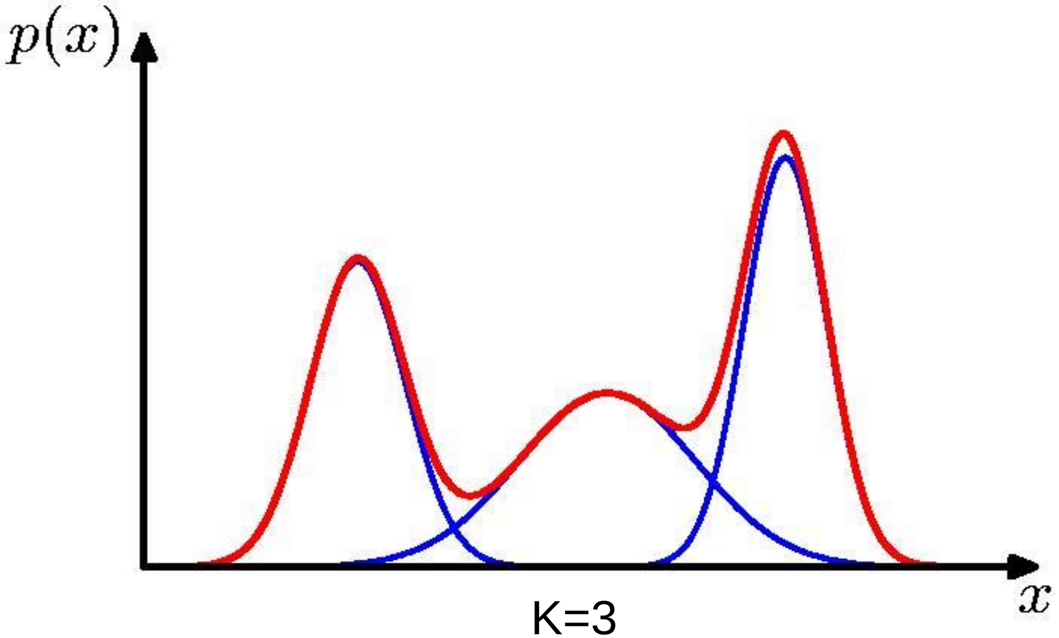
Mixture of two Gaussians

Mixtures of Gaussians

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

Mixing coefficient

$$\forall k : \pi_k \geqslant 0 \qquad \sum_{k=1}^{K} \pi_k = 1$$



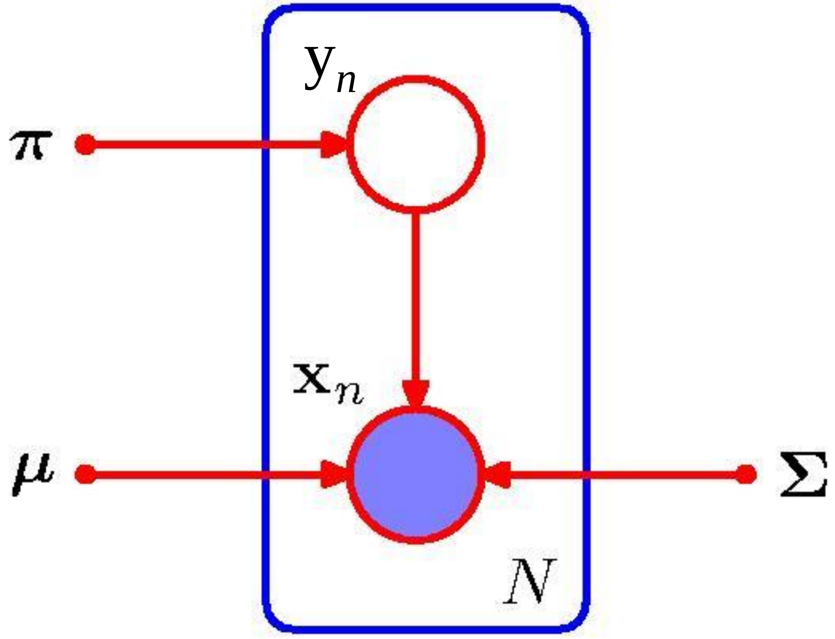
K=3

Gaussian mixture model

- P(Y): Distribution over *k* components (clusters)
- P(X|Y): Each component generates data from a **multivariate Gaussian** with mean μ_i and covariance matrix Σ_i

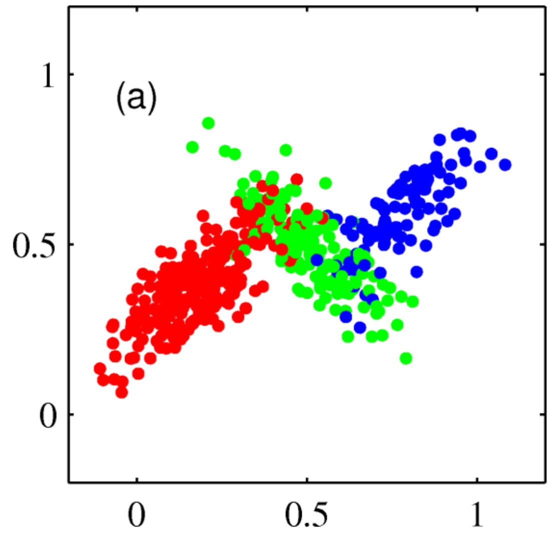
Each data point is sampled from a ***generative process***:

1. Choose component *i* with probability π_i
2. Generate data point from $N(\mathbf{x}|\mu_i, \Sigma_i)$



Supervised learning for GMM

- We observe both the data points and their labels (generated from which Gaussian components)
- How do we estimate parameters of GMM?



Supervised learning for GMM

- We observe both the data points and their labels (generated from which Gaussian components)
- How do we estimate parameters of GMM?
- Objective: maximize the likelihood

$$\prod_j P(y_j = i, \mathbf{x}_j) = \prod_j \pi_i N(\mathbf{x}_j | \mu_i, \Sigma_i)$$

- Closed form solution:
  - *m* data points. For component *i*, suppose we have *n* data points with label *i*.

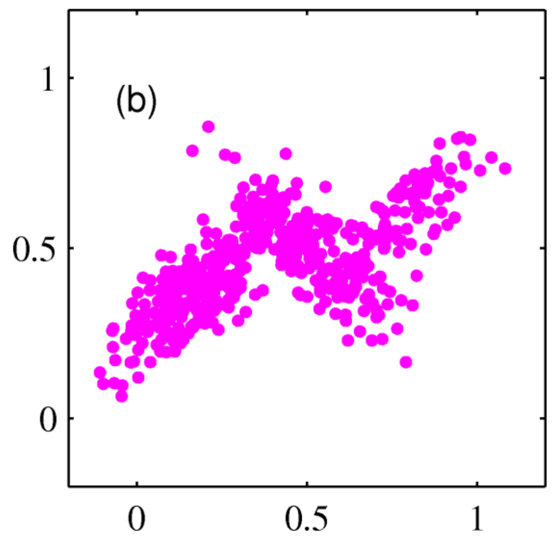
$$\mu_i = \frac{1}{n}\sum_{j=1}^{n} \mathbf{x}_j \qquad \Sigma_i = \frac{1}{n}\sum_{j=1}^{n} (\mathbf{x}_j - \mu_i)(\mathbf{x}_j - \mu_i)^T \qquad \pi_i = \frac{n}{m}$$

Unsupervised learning for GMM

- In clustering, we don't know the labels Y!
- Maximize marginal likelihood:

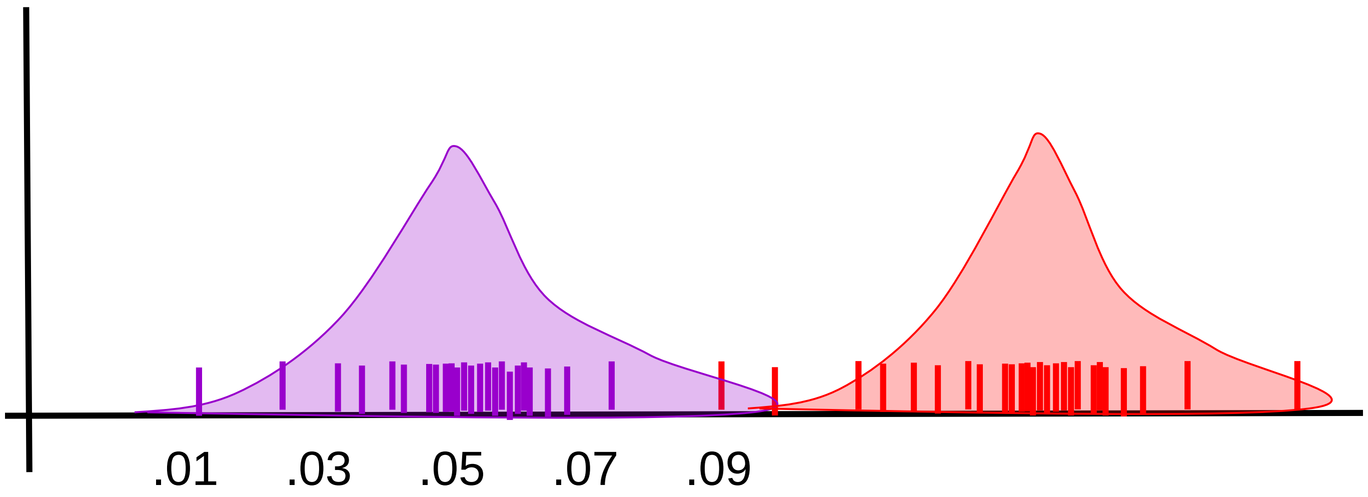
$$\prod_j P(\mathbf{x}_j) = \prod_j \sum_i P(y_j = i, \mathbf{x}_j) = \prod_j \sum_i \pi_i N(\mathbf{x}_j | \mu_i, \Sigma_i)$$

- How do we optimize it?
  - No closed form solution

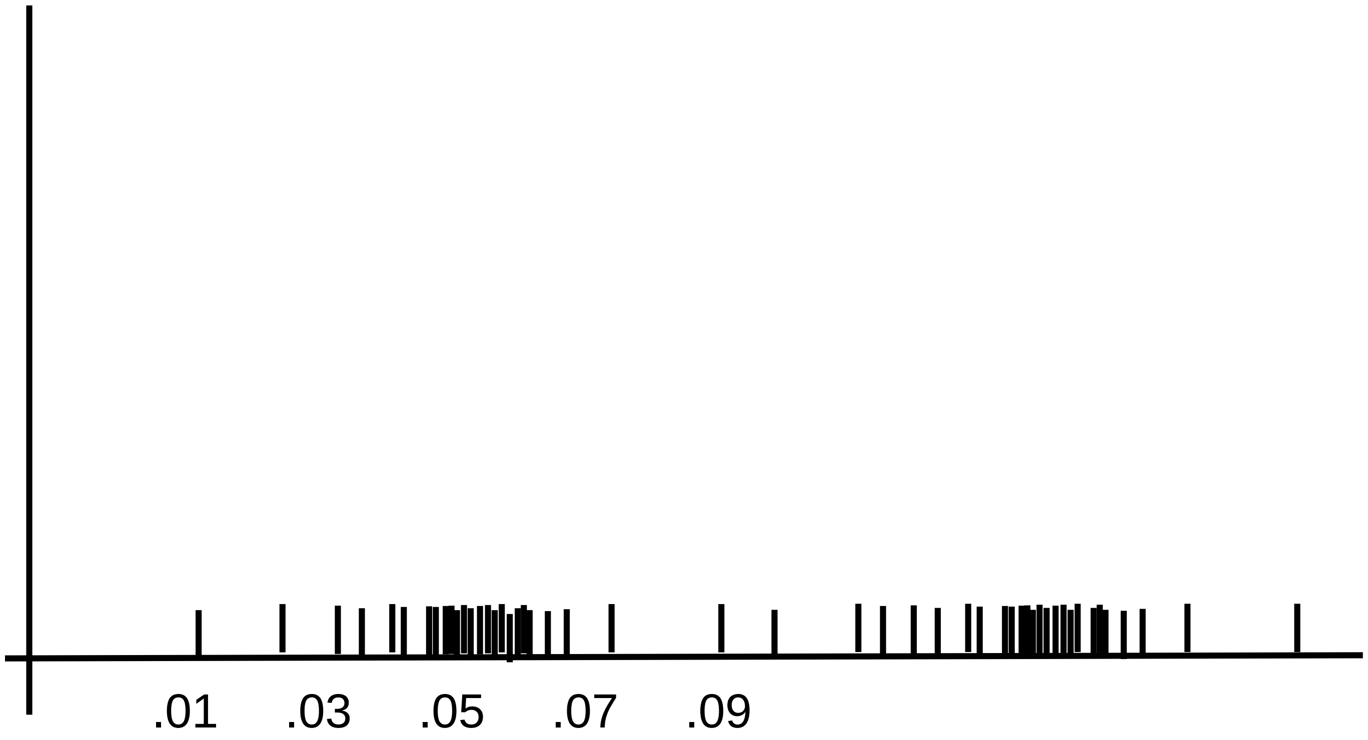


Simplest Example

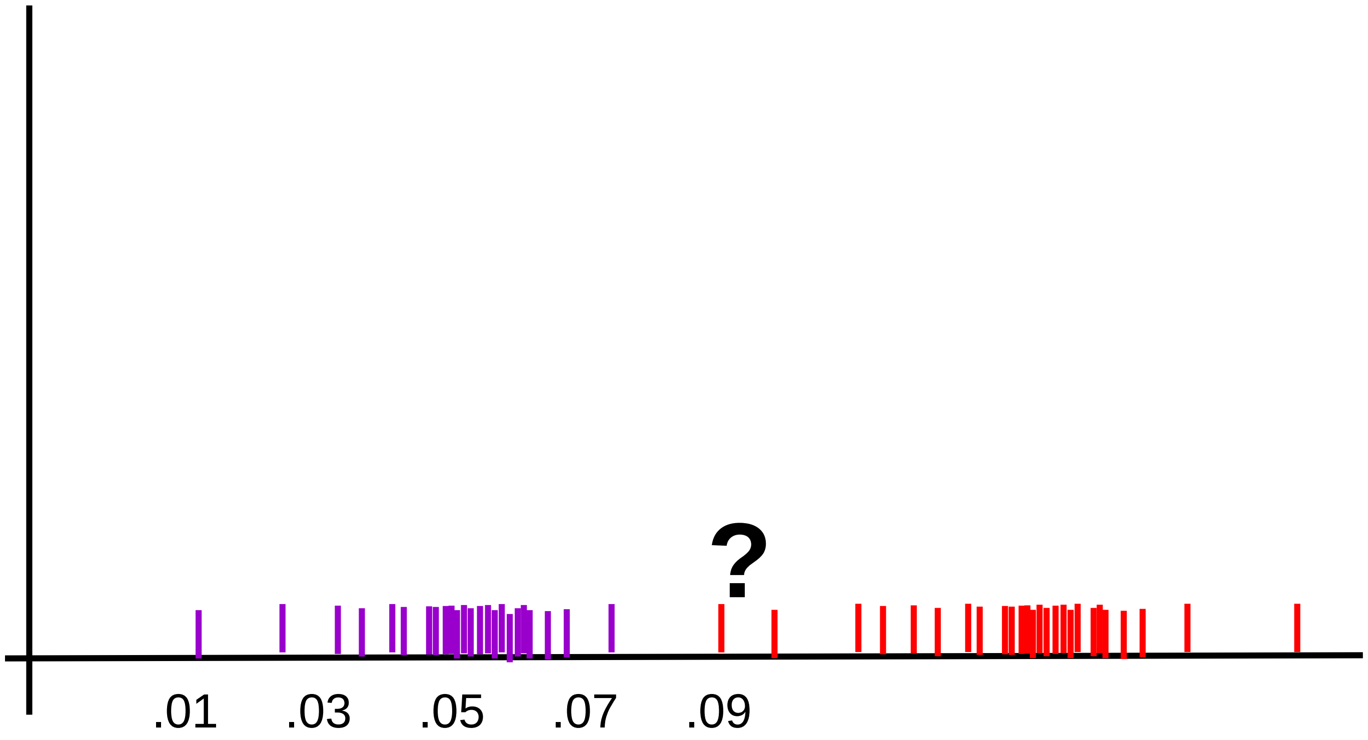
Mixture of two distributions



Input Looks Like



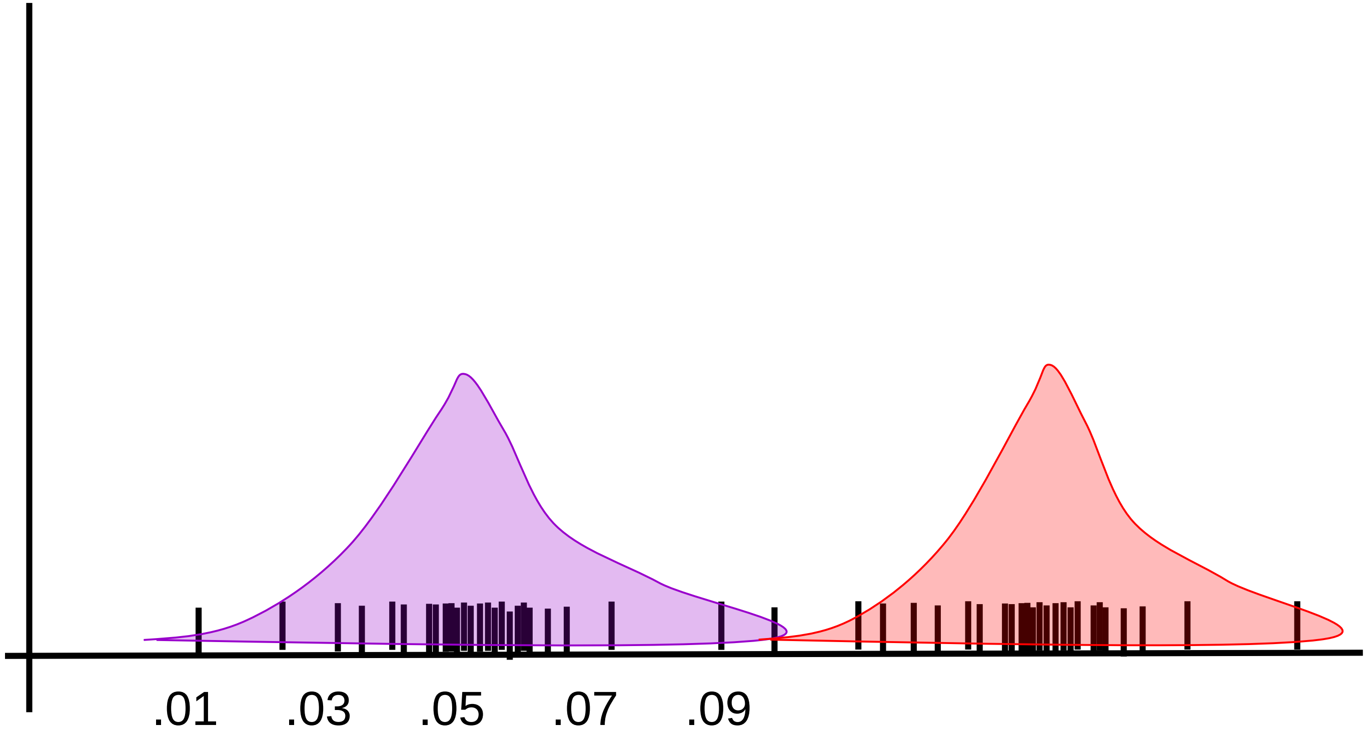
We Want to Predict



Chicken & Egg

Note that coloring instances would be easy

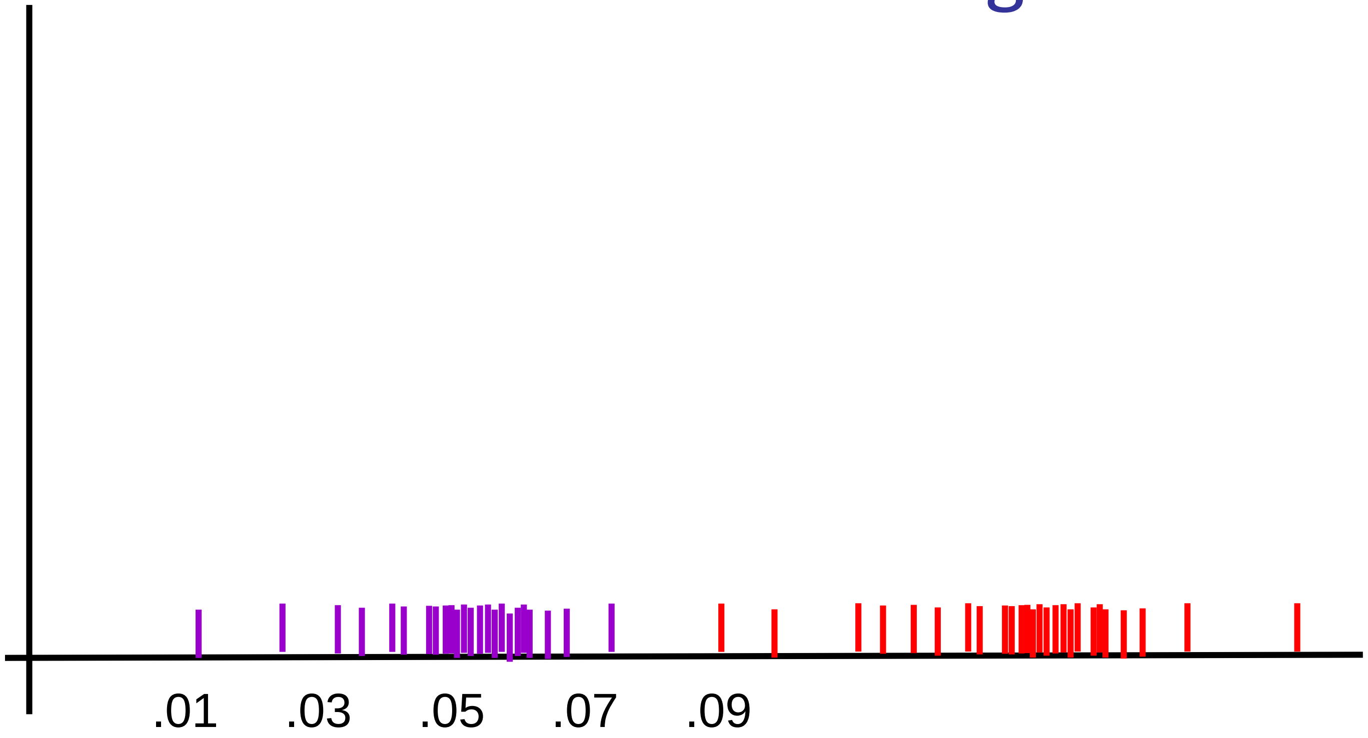
if we knew Gaussians….



Chicken & Egg

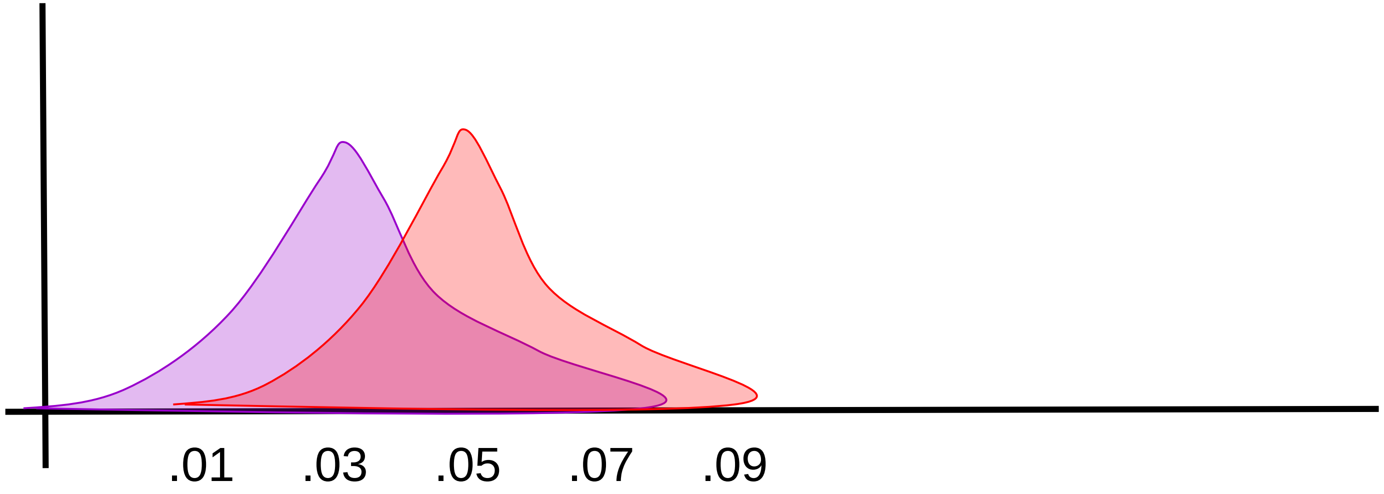
And finding the Gaussians would be easy

If we knew the coloring



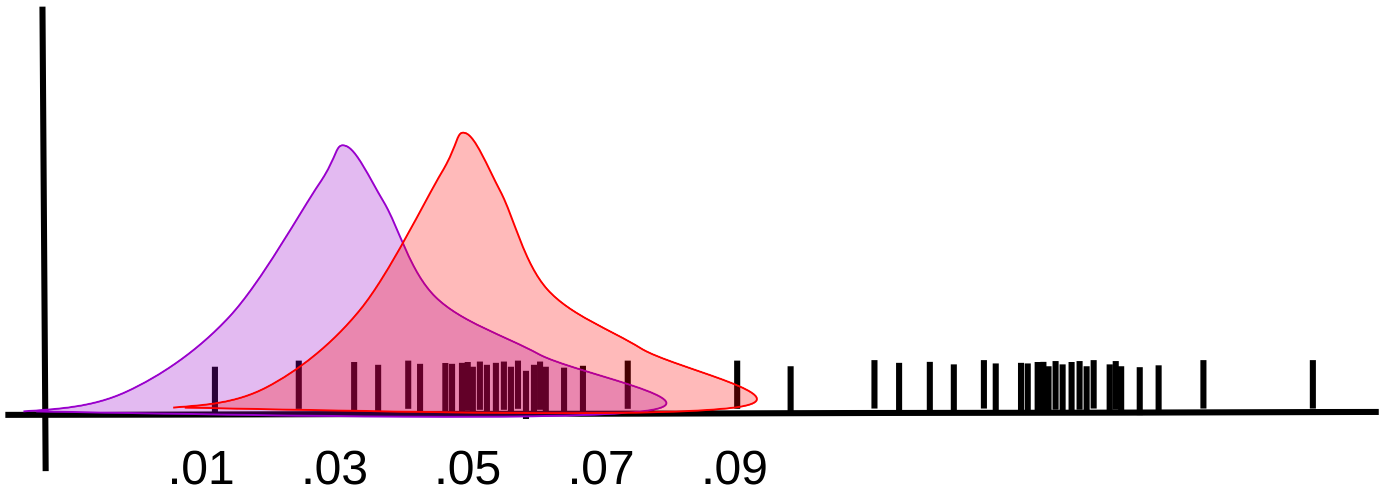
Expectation Maximization (EM)

- Pretend we do know the parameters
  - Initialize randomly



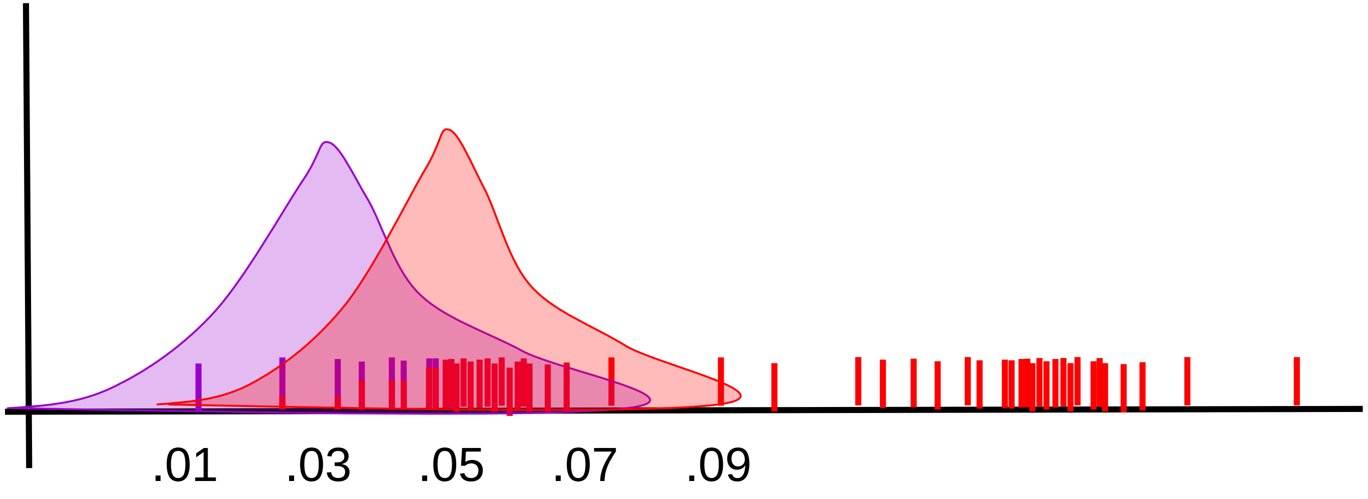
Expectation Maximization (EM)

- [E step] Compute probability of each instance having each possible label



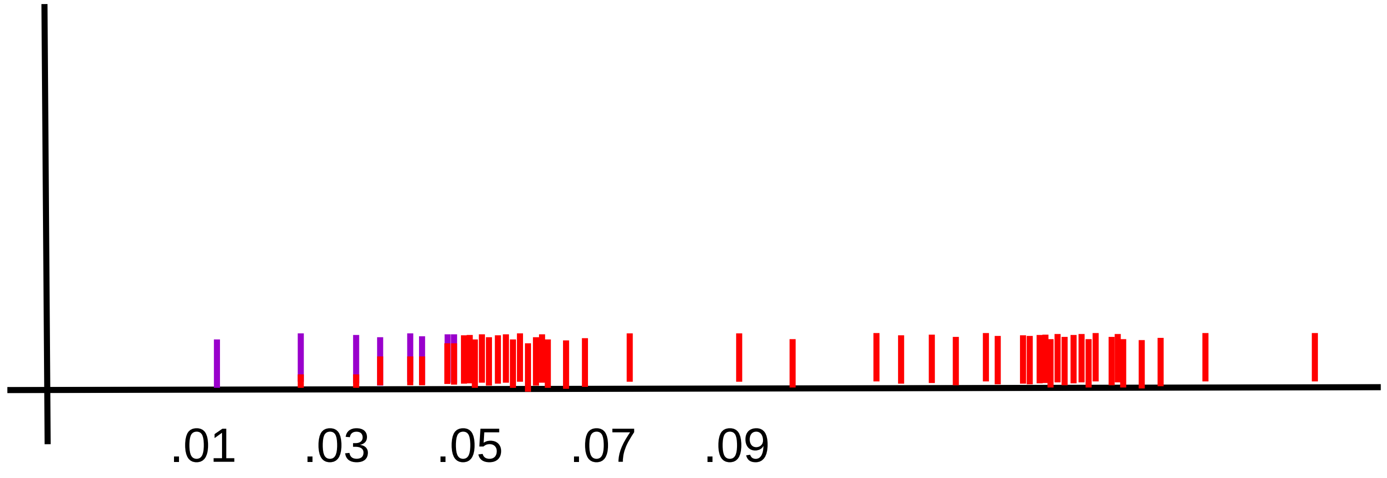
Expectation Maximization (EM)

- [E step] Compute probability of each instance having each possible label



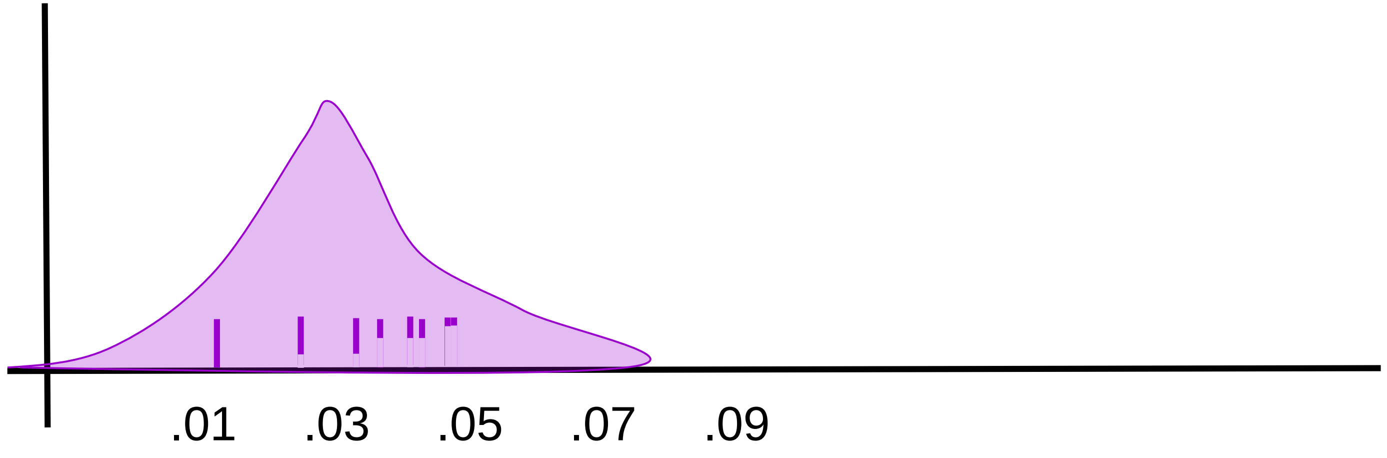
Expectation Maximization (EM)

- [E step] Compute probability of each instance having each possible label
- [M step] Treating each instance as fractionally having both labels, compute the new parameter values



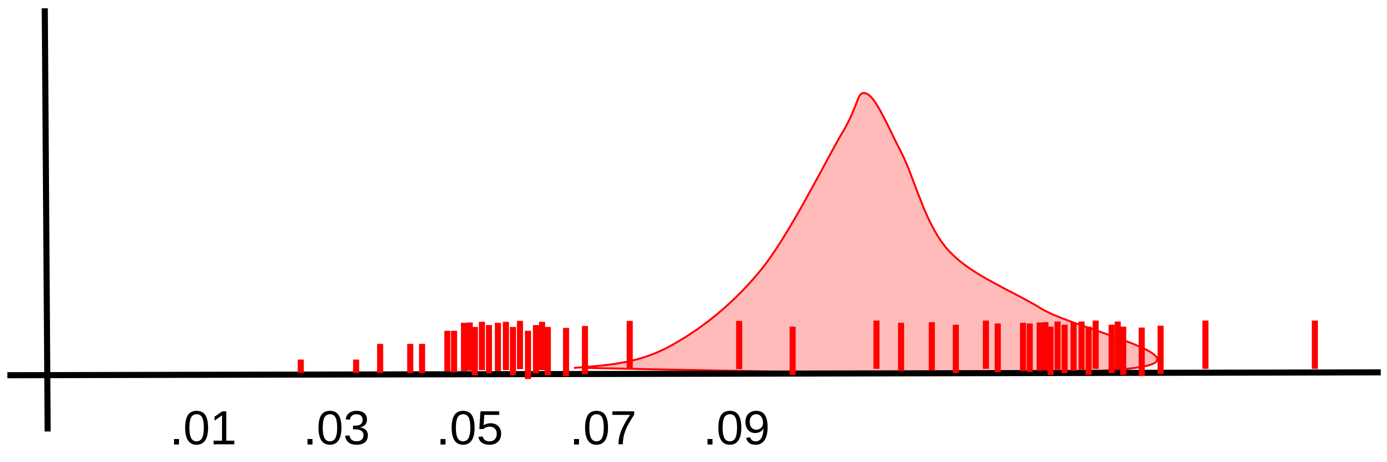
Expectation Maximization (EM)

- [E step] Compute probability of each instance having each possible label
- [M step] Treating each instance as fractionally having both labels, compute the new parameter values



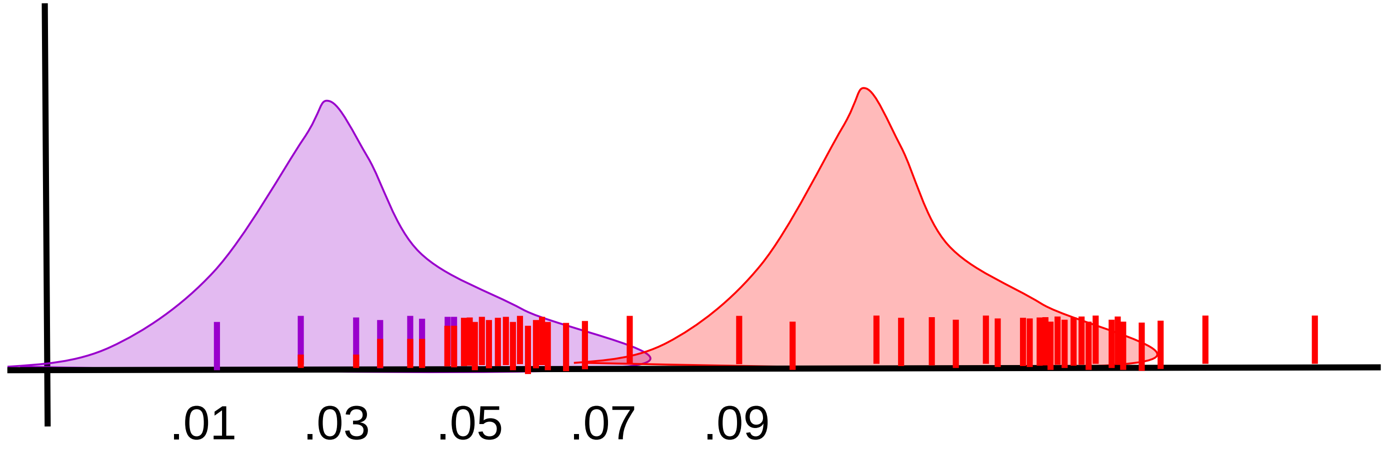
Expectation Maximization (EM)

- [E step] Compute probability of each instance having each possible label
- [M step] Treating each instance as fractionally having both labels, compute the new parameter values



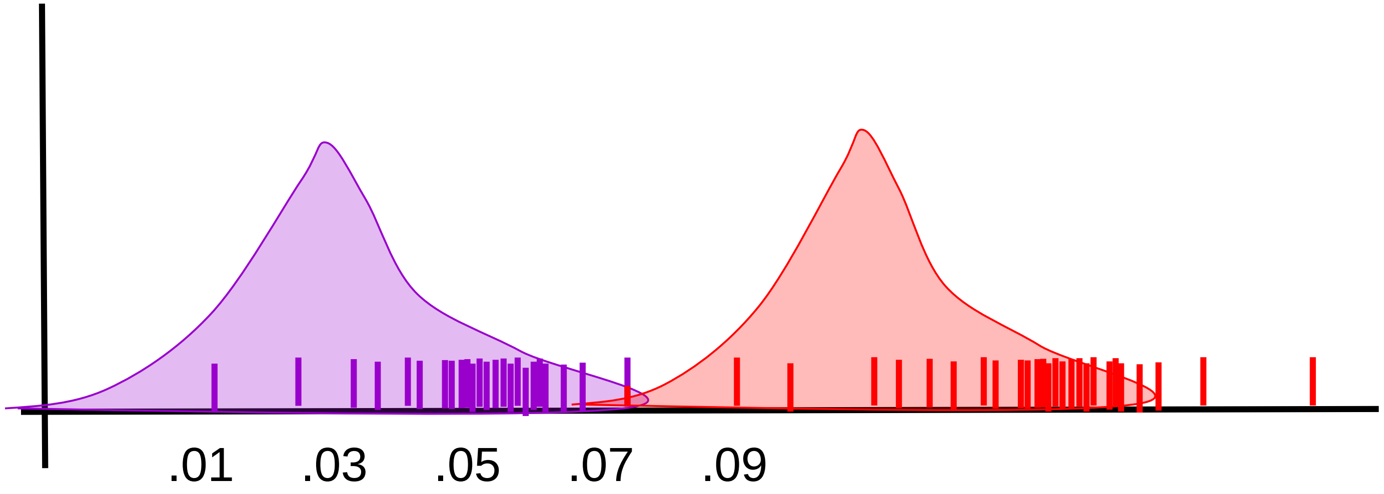
Expectation Maximization (EM)

- [E step] Compute probability of each instance having each possible label
- [M step] Treating each instance as fractionally having both labels, compute the new parameter values



Expectation Maximization (EM)

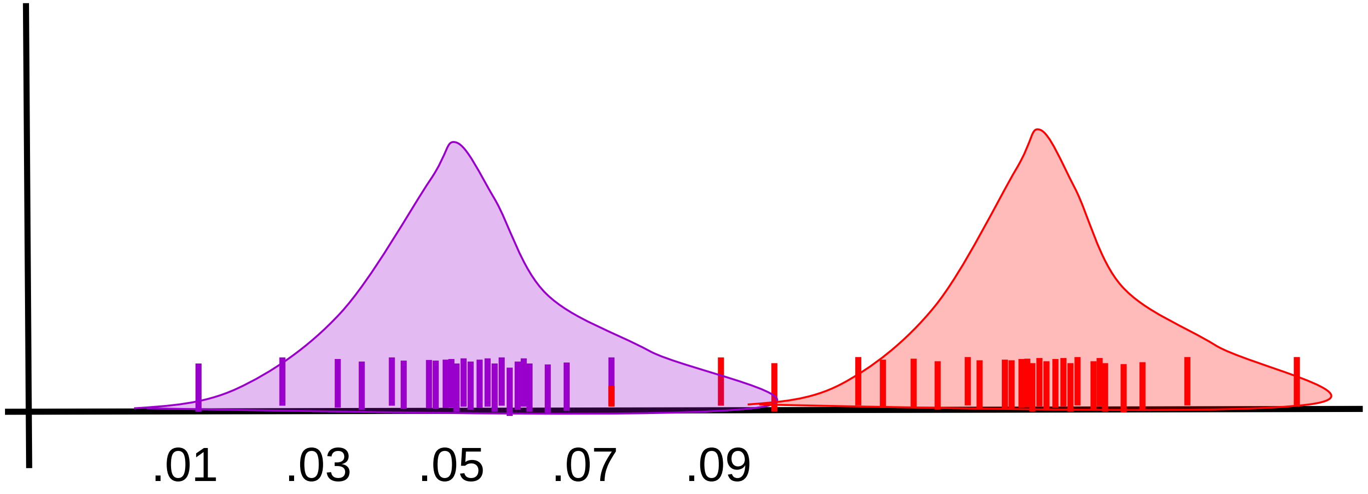
- Repeat E-step



Expectation Maximization (EM)

- Repeat E-step
- Repeat M-step

… until convergence



Expectation Maximization (EM)

- Pick K random cluster models (Gaussians)
- Alternate:
  - Assign data instances proportionately to different models
  - Revise each cluster model based on its (proportionately) assigned points
- Stop when no changes

EM for GMM

Iterate: On the *t*'th iteration let our estimates be

$\theta^{(t)} = \{ \mu_1^{(t)}, \mu_2^{(t)} \ldots \mu_k^{(t)}, \Sigma_1^{(t)}, \Sigma_2^{(t)} \ldots \Sigma_k^{(t)}, \pi_1^{(t)}, \pi_2^{(t)} \ldots \pi_k^{(t)} \}$

E-step

Compute label distribution of each data point

$$\mathrm{P}\left(y_j = i \,\middle|\, x_j, \theta^{(t)}\right) \propto \pi_i^{(t)} N\left(x_j \,\middle|\, \mu_i^{(t)}, \Sigma_i^{(t)}\right)$$

Just evaluate a Gaussian at x_j

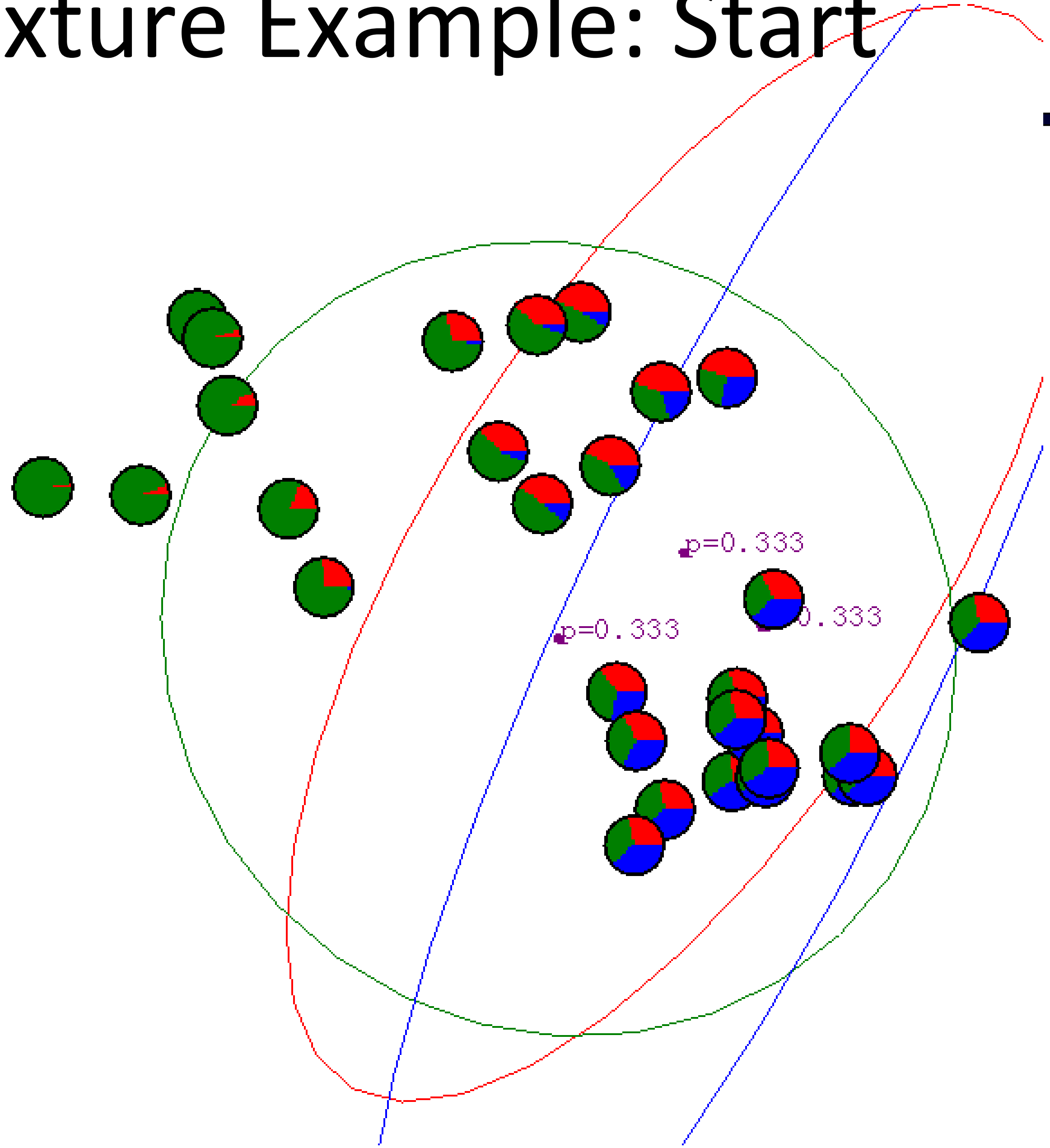
M-step

Compute weighted MLE of parameters given label distributions

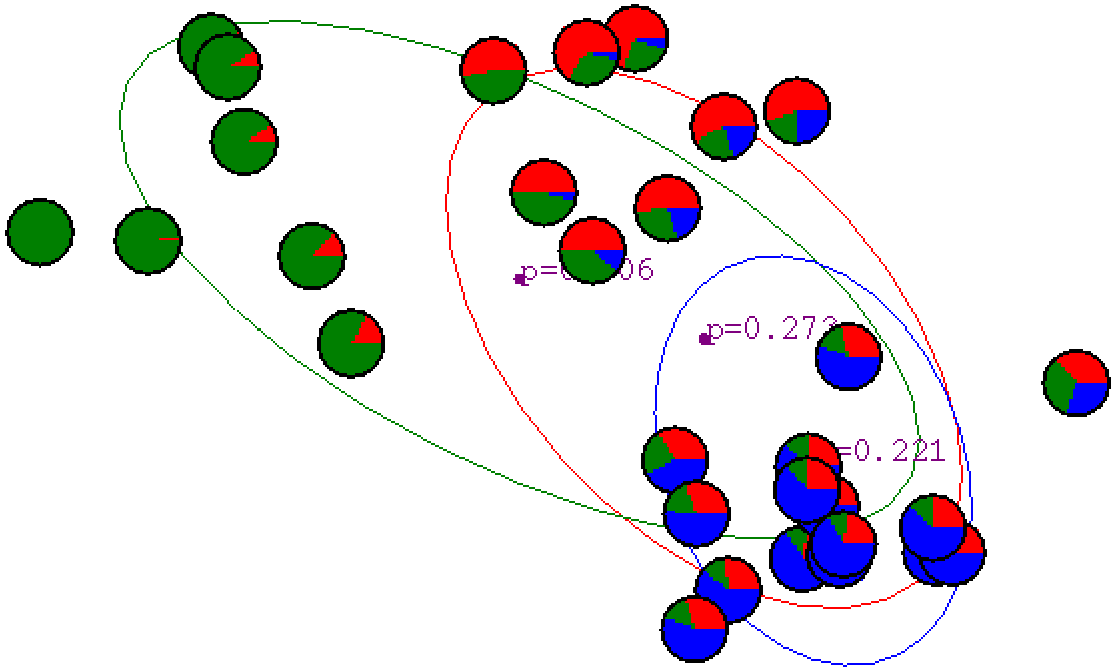
$$\mu_i^{(t+1)} = \frac{\sum_j \mathrm{P}\left(y_j = i \,\middle|\, x_j, \theta^{(t)}\right) x_j}{\sum_{j'} \mathrm{P}\left(y_{j'} = i \,\middle|\, x_{j'}, \theta^{(t)}\right)} \qquad \Sigma_i^{(t+1)} = \frac{\sum_j \mathrm{P}\left(y_j = i \,\middle|\, x_j, \theta^{(t)}\right) \left[x_j - \mu_i^{(t+1)}\right]\left[x_j - \mu_i^{(t+1)}\right]^T}{\sum_{j'} \mathrm{P}\left(y_{j'} = i \,\middle|\, x_{j'}, \theta^{(t)}\right)} \qquad \pi_i^{(t+1)} = \frac{\sum_j \mathrm{P}\left(y_j = i \,\middle|\, x_j, \theta^{(t)}\right)}{m}$$

m = #training examples

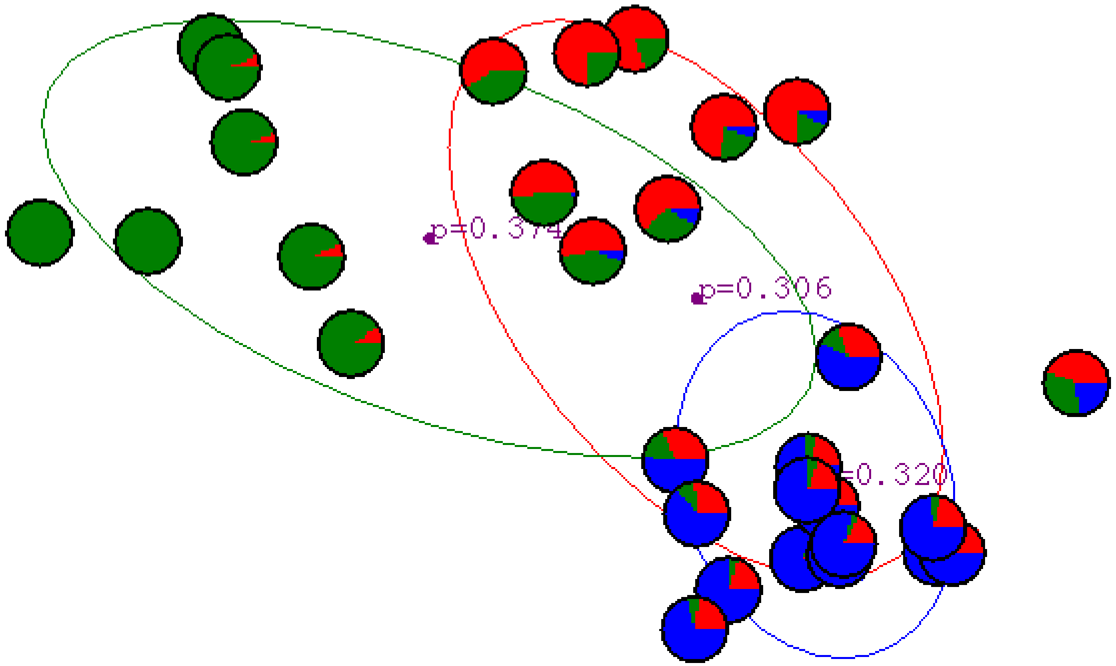
Gaussian Mixture Example: Start



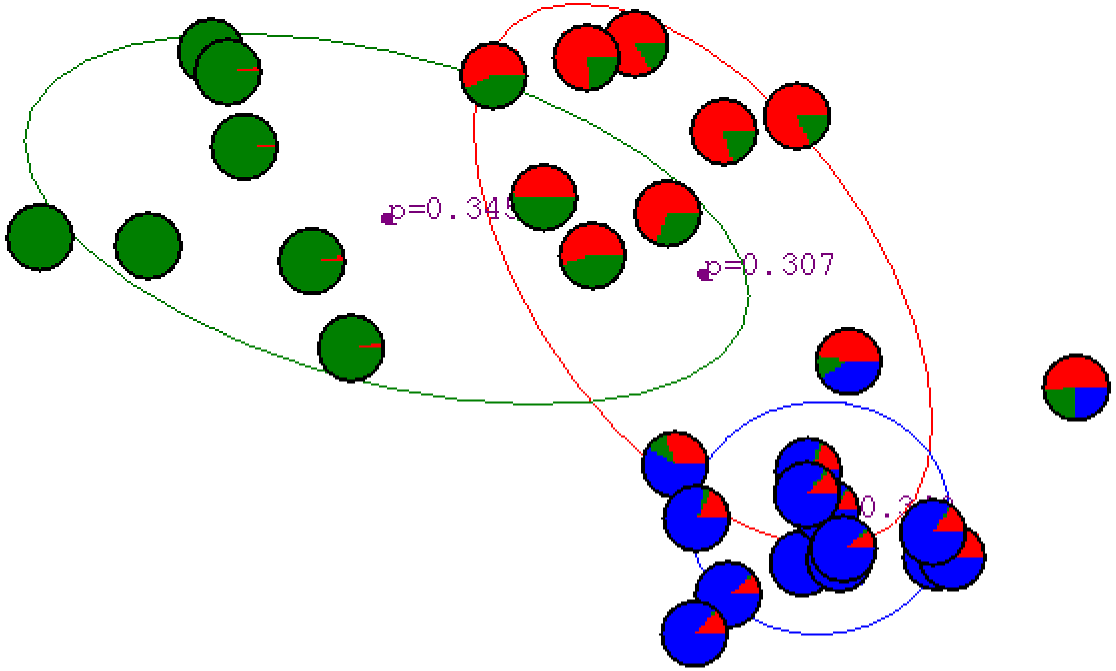
After first iteration



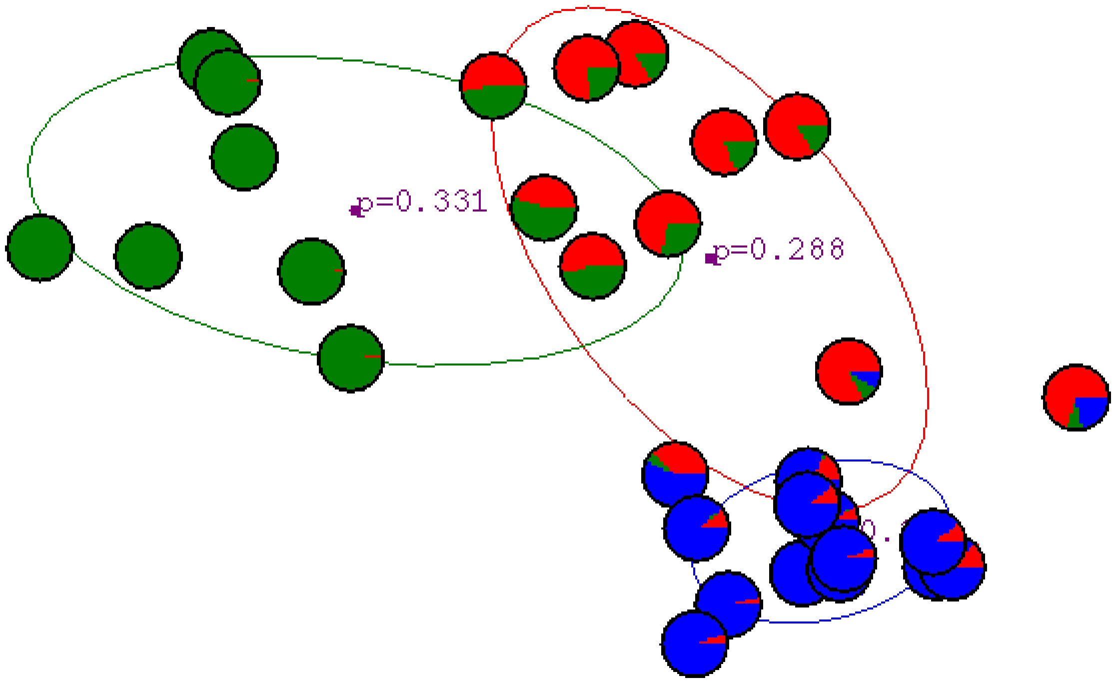
After 2nd iteration



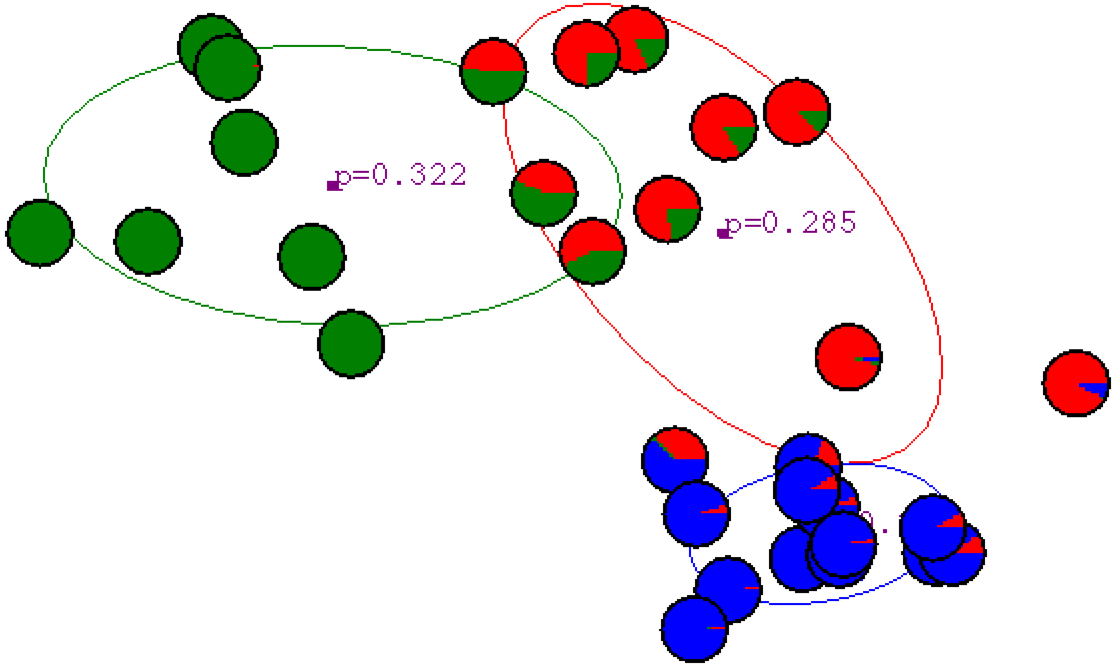
After 3rd iteration



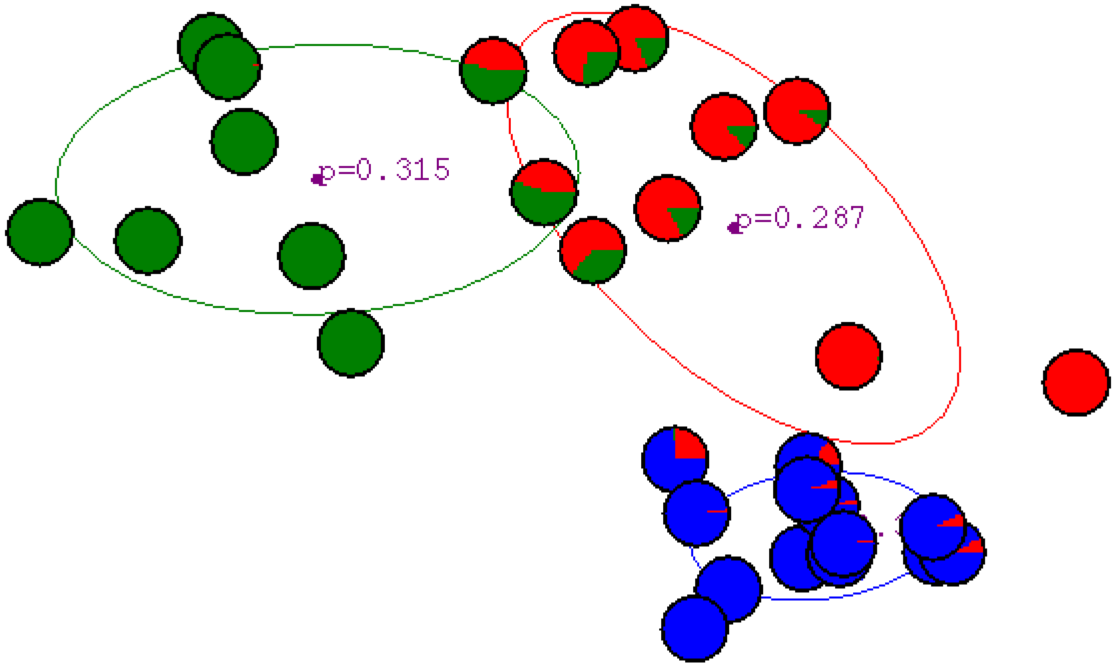
After 4th iteration



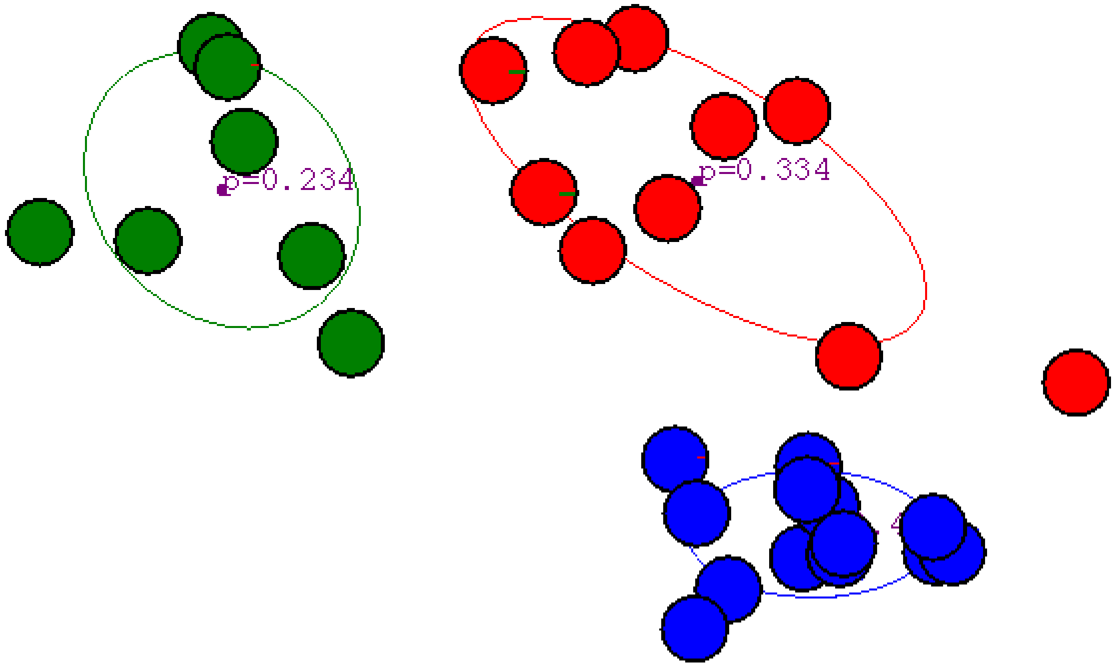
After 5th iteration



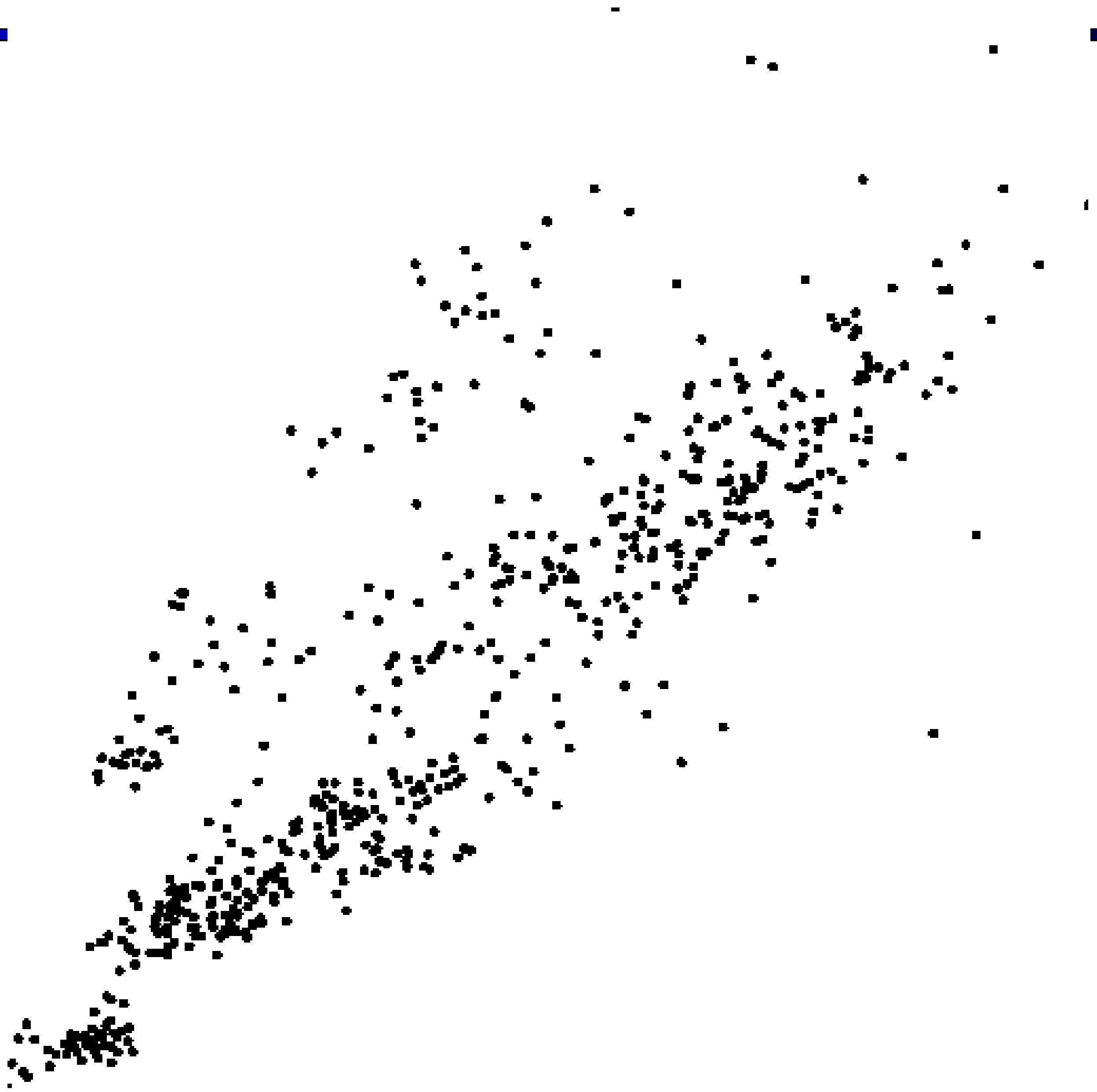
After 6th iteration



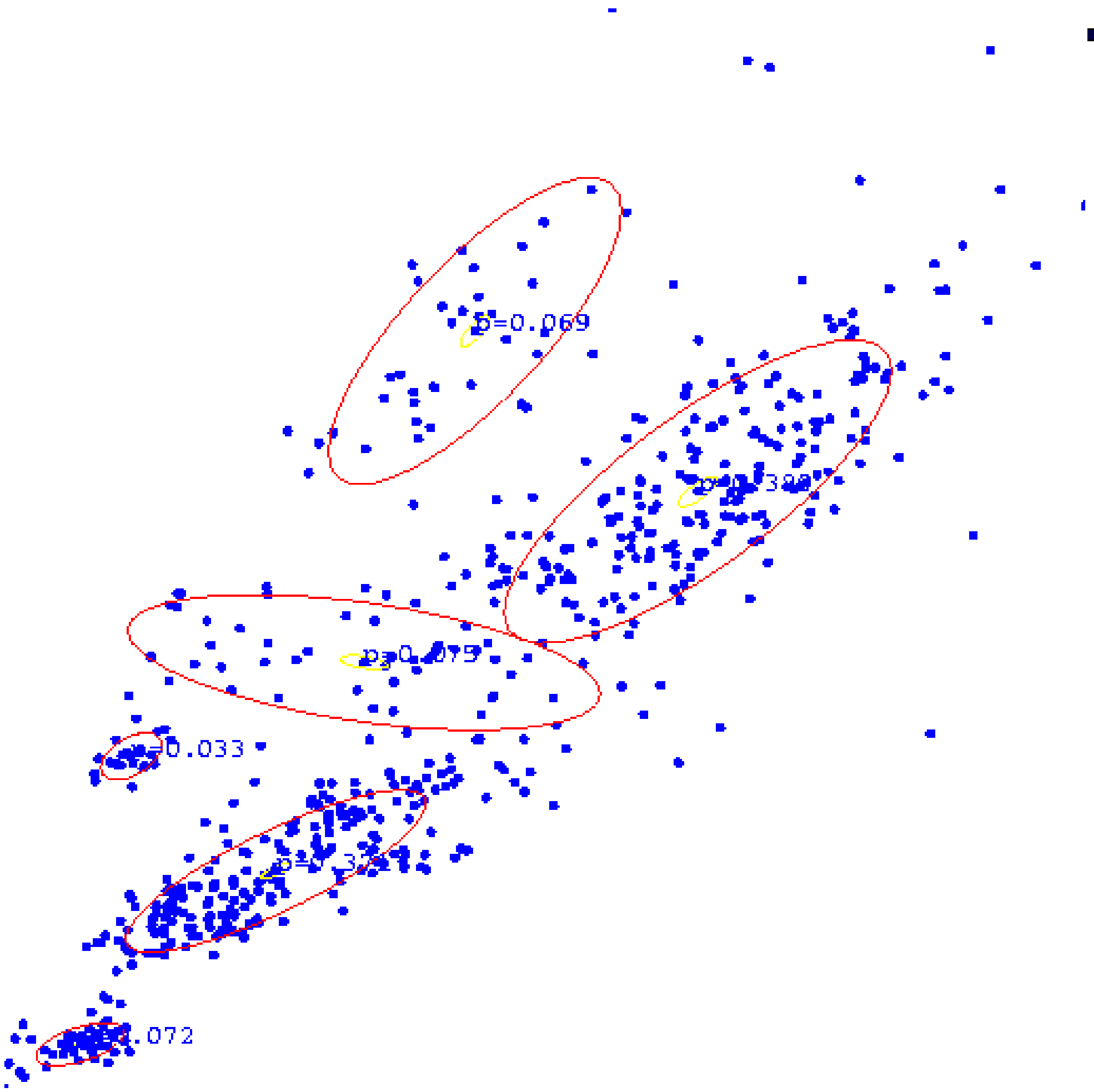
After 20th iteration



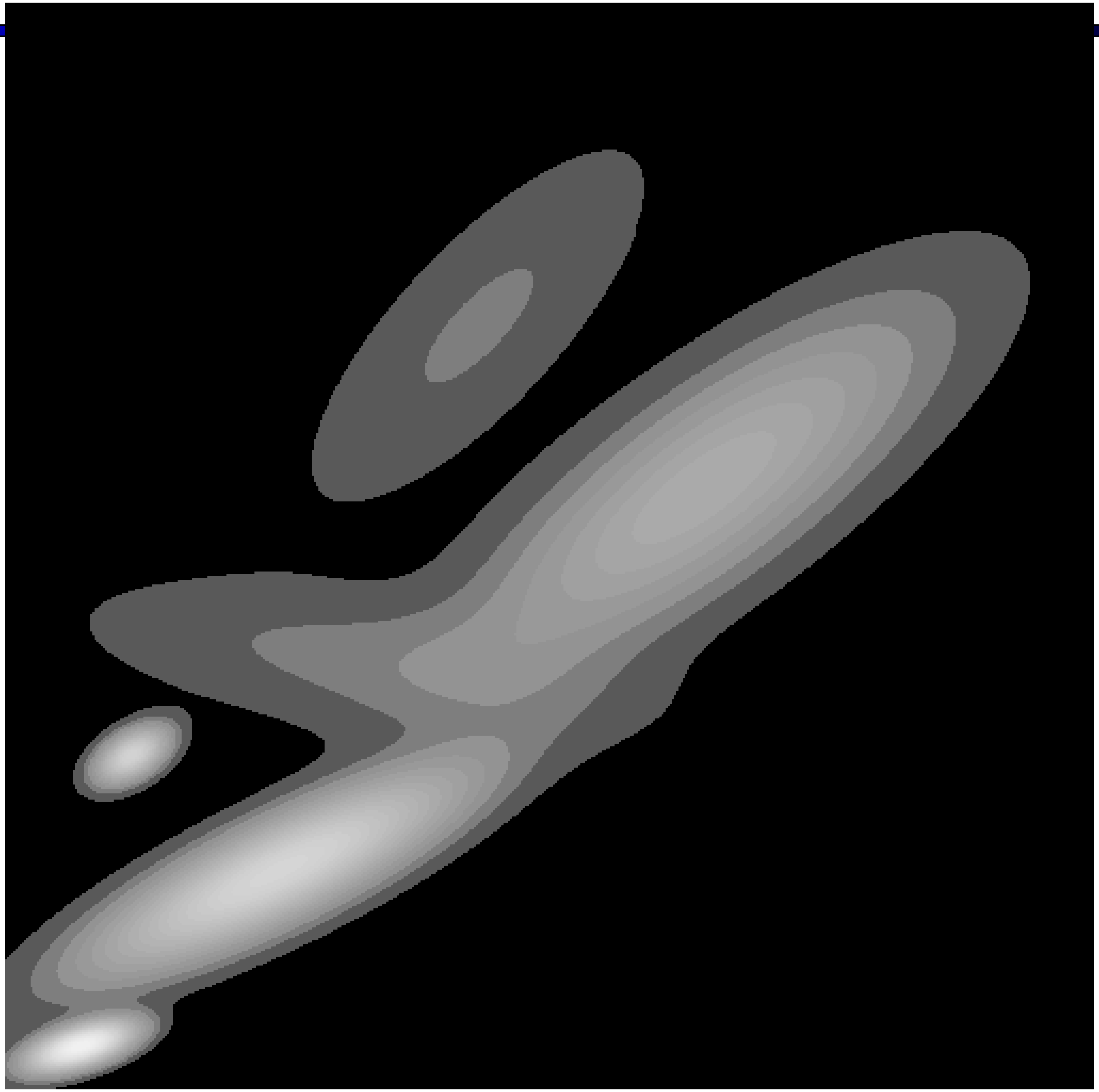
Some Bio Assay data



GMM clustering of the assay data



Resulting Density Estimator



EM and K-means

- EM degrades to k-means if we assume
  - All the Gaussians are spherical and have identical weights and covariances
    - i.e., the only parameters are the means
  - The label distributions computed at E-step are point-estimations
    - i.e., hard-assignments of data points to Gaussians
    - Alternatively, assume the variances are close to zero

EM in General

- Can be used to learn any model with hidden variables (missing data)
- Alternate:
  - Compute distributions over hidden variables based on current parameter values
  - Compute new parameter values to maximize expected log likelihood based on distributions over hidden variables
- Stop when no changes

EM for HMMs

- [E step] Compute the distributions of hidden states given each training instance
  - Infeasible to enumerate. But can compute expected counts of transitions and emissions using the forward and backward algorithms.
- [M step] Update the parameters to maximize expected log likelihood based on distributions over hidden states
  - Closed-form solution: simply normalize the expected counts of transitions and emissions
- Known as Baum–Welch algorithm

Math Behind EM

- EM is coordinate ascent on F(θ, Q)

$$\ell(\theta : \mathcal{D}) \underset{\text{Jensen's inequality}}{\geq} F(\theta, Q) = \sum_{j=1}^{m} \sum_{\mathbf{z}} Q(\mathbf{z} \mid \mathbf{x}_j) \log \frac{P(\mathbf{z}, \mathbf{x}_j \mid \theta)}{Q(\mathbf{z} \mid \mathbf{x}_j)}$$

  - E-step fixes θ and optimizes Q
  - M-step fixes Q and optimizes θ
- Convergence of EM
  - Neither E-step nor M-step decreases F(θ, Q)

Summary

- Clustering
  - Group together similar instances
- K-means
  - Assign data instances to closest mean
  - Assign each mean to the average of its assigned points
- EM
  - Assign data instances proportionately to different Gaussian models
  - Revise each model based on its (proportionately) assigned points