Unsupervised Learning: Clustering

Vibhav Gogate
The University of Texas at Dallas

Slides adapted from Carlos Guestrin, Dan Klein & Luke Zettlemoyer

Machine Learning

Supervised Learning

Unsupervised Learning

Reinforcement Learning

Parametric

Non-parametric

Y Continuous

Y Discrete

Gaussians

Learned in closed form

Linear Functions

1. Learned in closed form
2. Using gradient descent

Decision Trees

Greedy search; pruning

Probability of class | features

1. Learn P(Y), P(X|Y); apply Bayes
2. Learn P(Y|X) w/ gradient descent

Non-probabilistic

Linear: perceptron gradient descent

Nonlinear: neural net: backprop

Support vector machines

Overview of Learning

Type of Supervision

(eg, Experience, Feedback)

What is Being Learned?

	Labeled Examples	Reward	Nothing
Discrete Function	Classification		Clustering
Continuous Function	Regression		
Policy	Apprenticeship Learning	Reinforcement Learning	

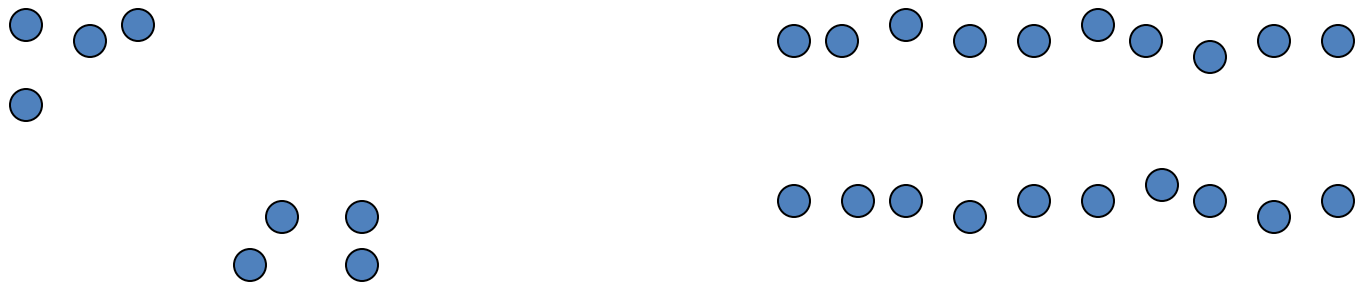
Clustering

Clustering systems:

- Unsupervised learning
- Requires data, but no labels
- Detect patterns e.g. in
  - Group emails or search results
  - Customer shopping patterns
  - Program executions (intrusion detection)
- Useful when don't know what you're looking for
- But: often get gibberish

Clustering

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



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

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

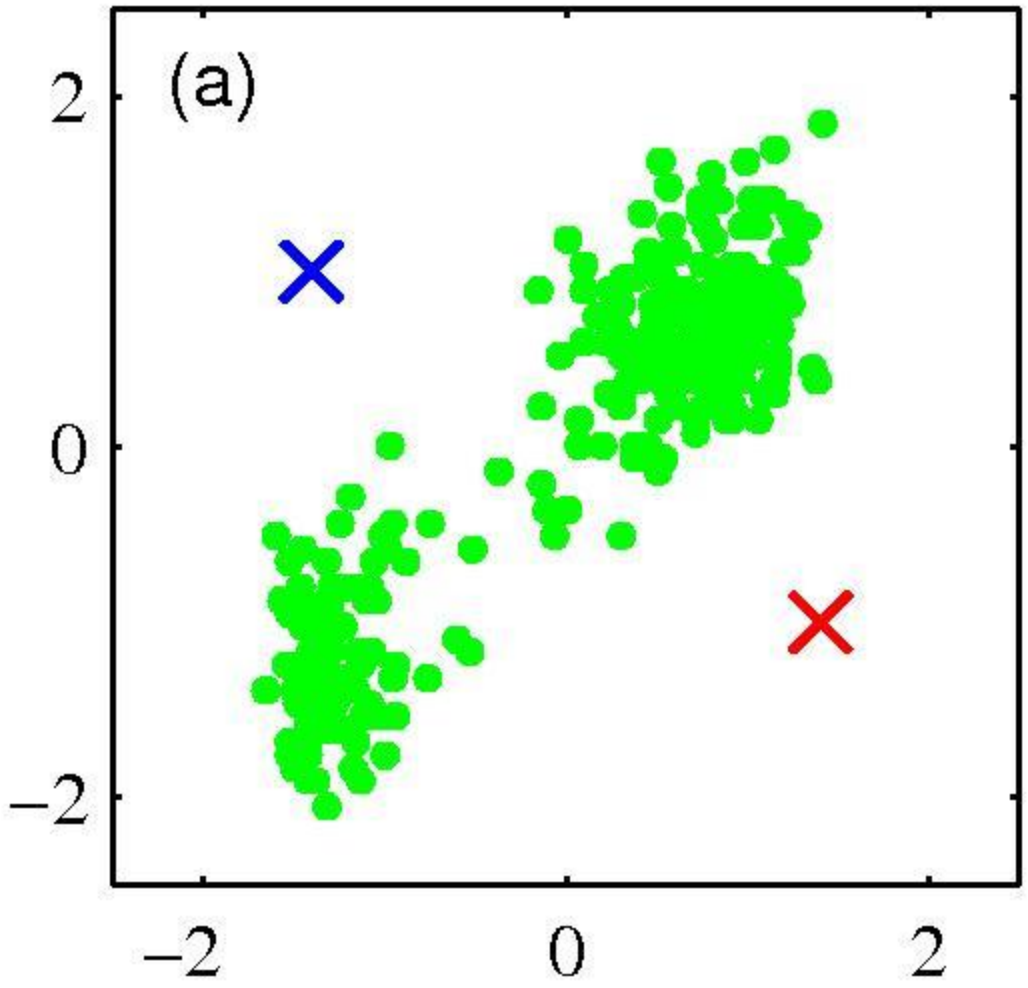
Outline

- K-means & Agglomerative Clustering
- Agglomerative Clustering
- Expectation Maximization (EM)

K-Means: Algorithm

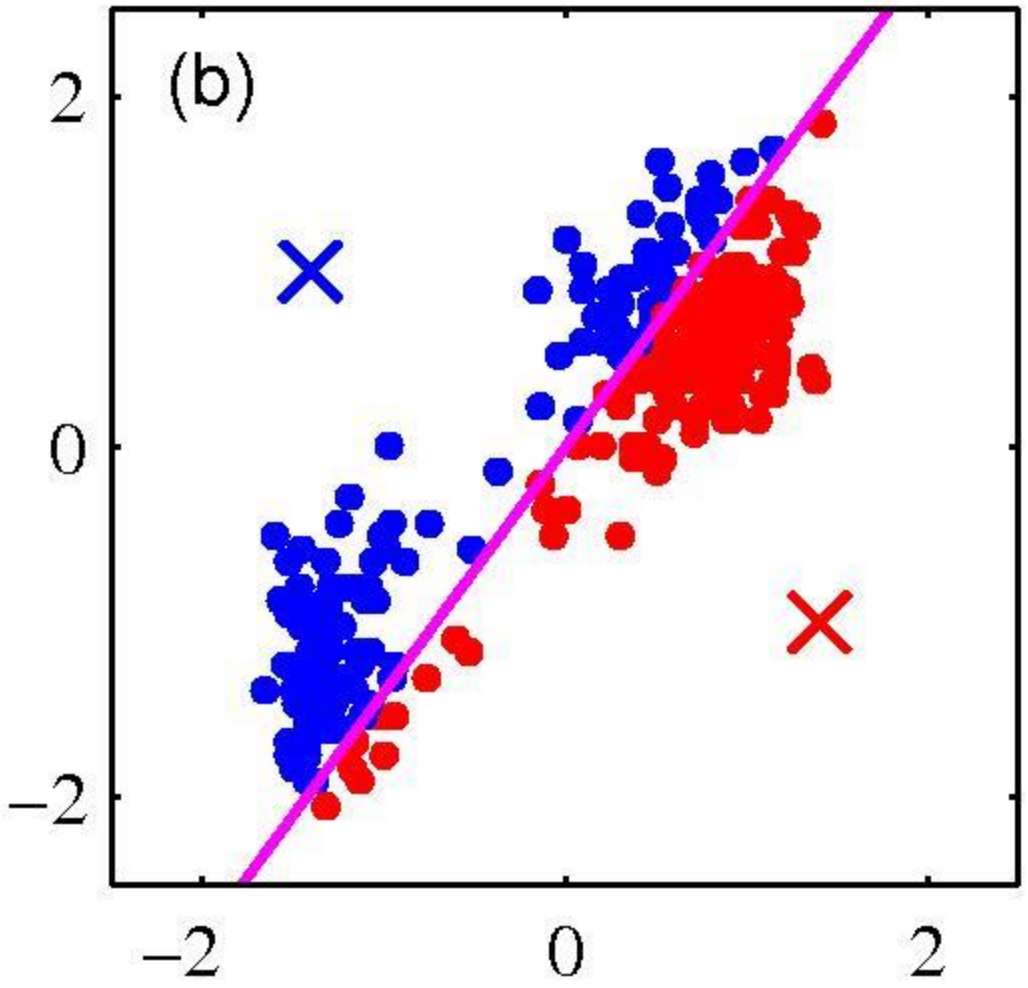
- An iterative clustering algorithm
  - Pick K random points as cluster centers (means)
  - Alternate:
    - Assign data instances to closest cluster center
    - Change the cluster center to the average of its assigned points
  - Stop when no points' assignments change

K-means clustering: Example



- Pick K random points as cluster centers (means)

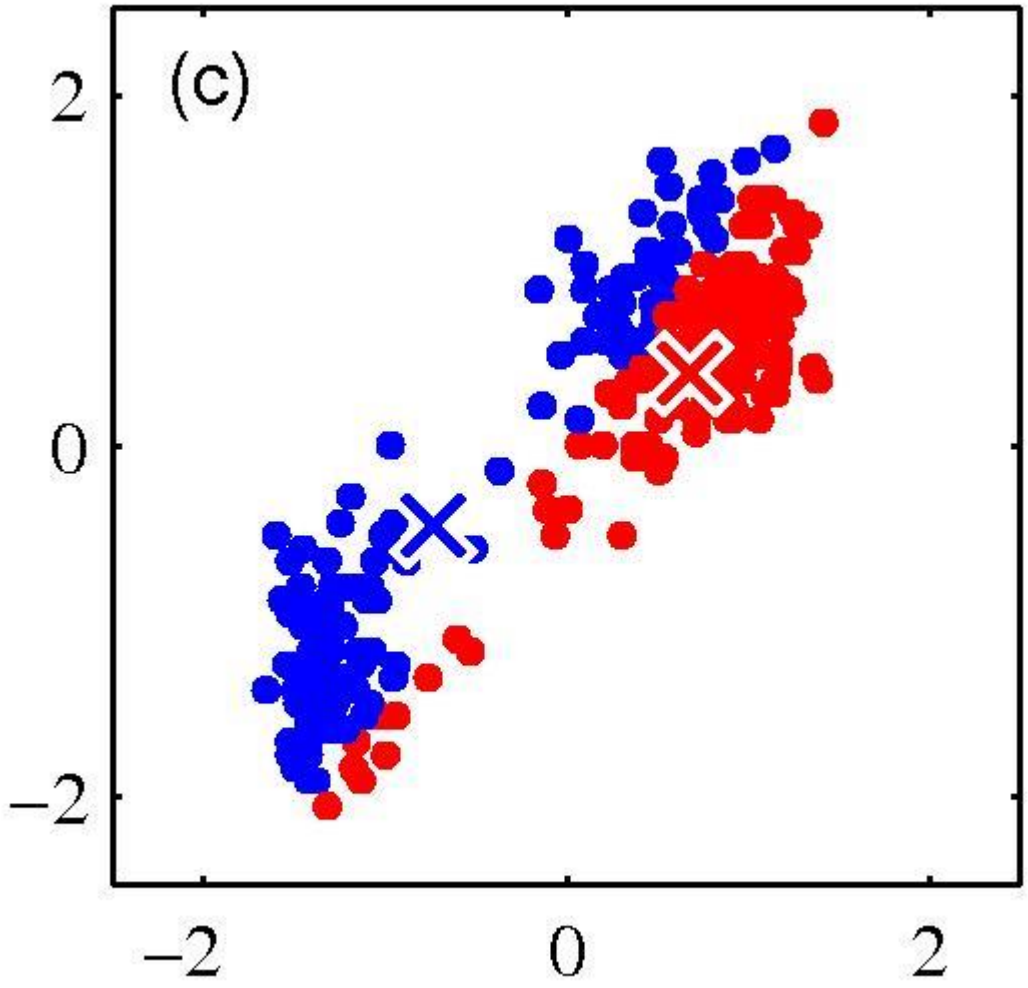
K-means clustering: Example



Iterative Step 1

- Assign data instances to closest cluster center

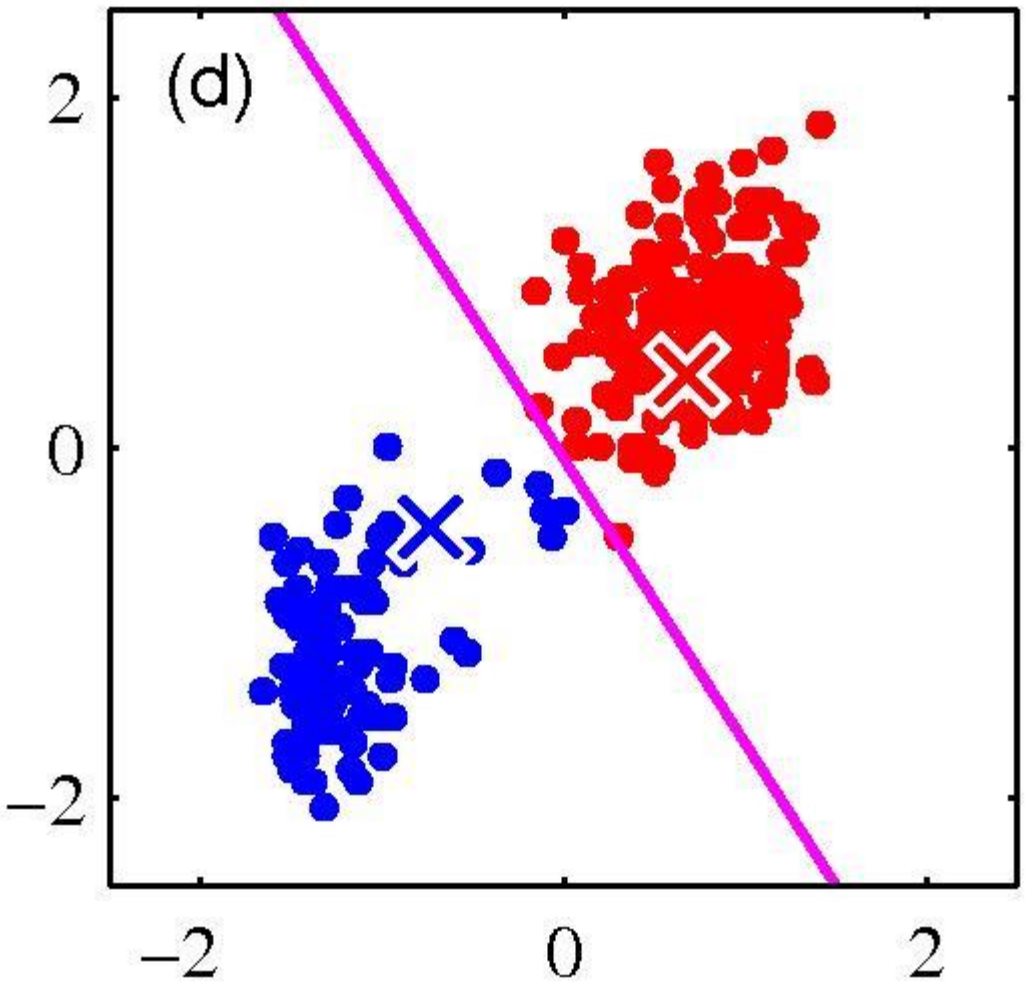
K-means clustering: Example



Iterative Step 2

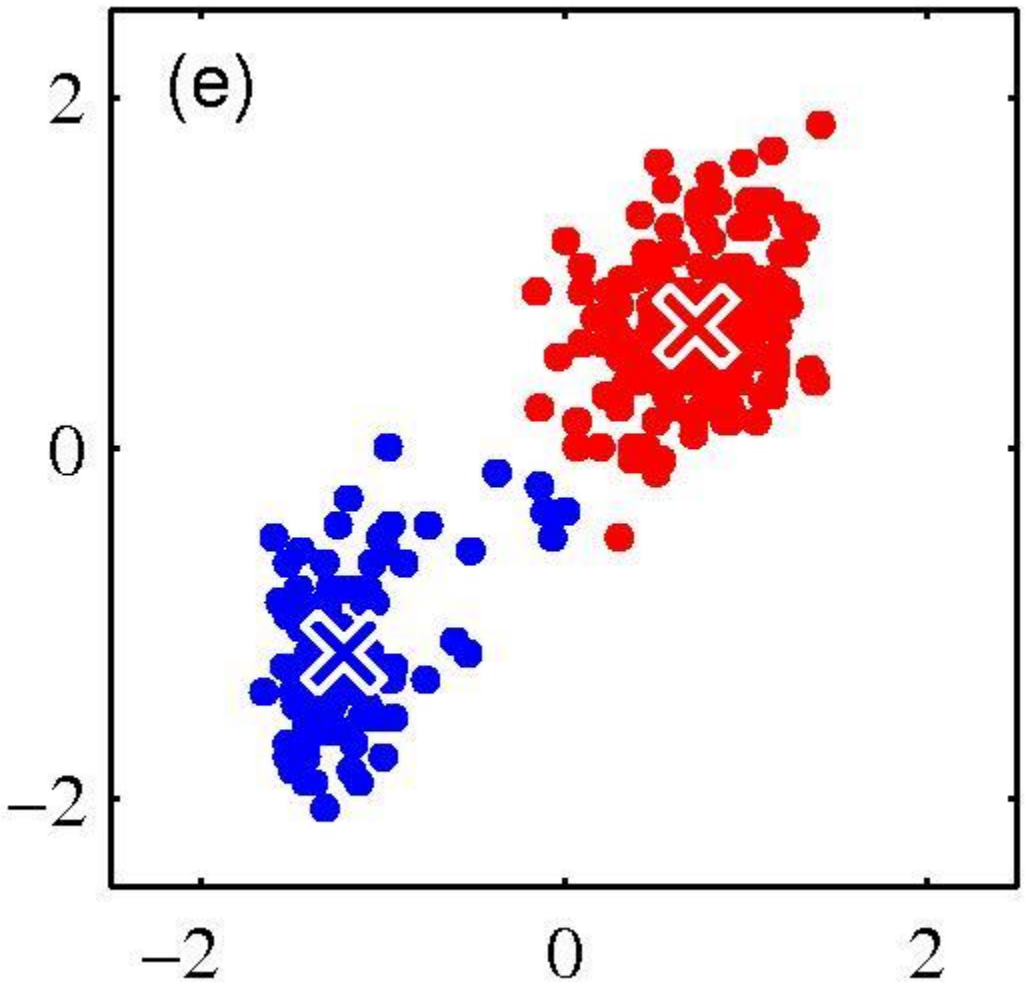
- Change the cluster center to the average of the assigned points

K-means clustering: Example

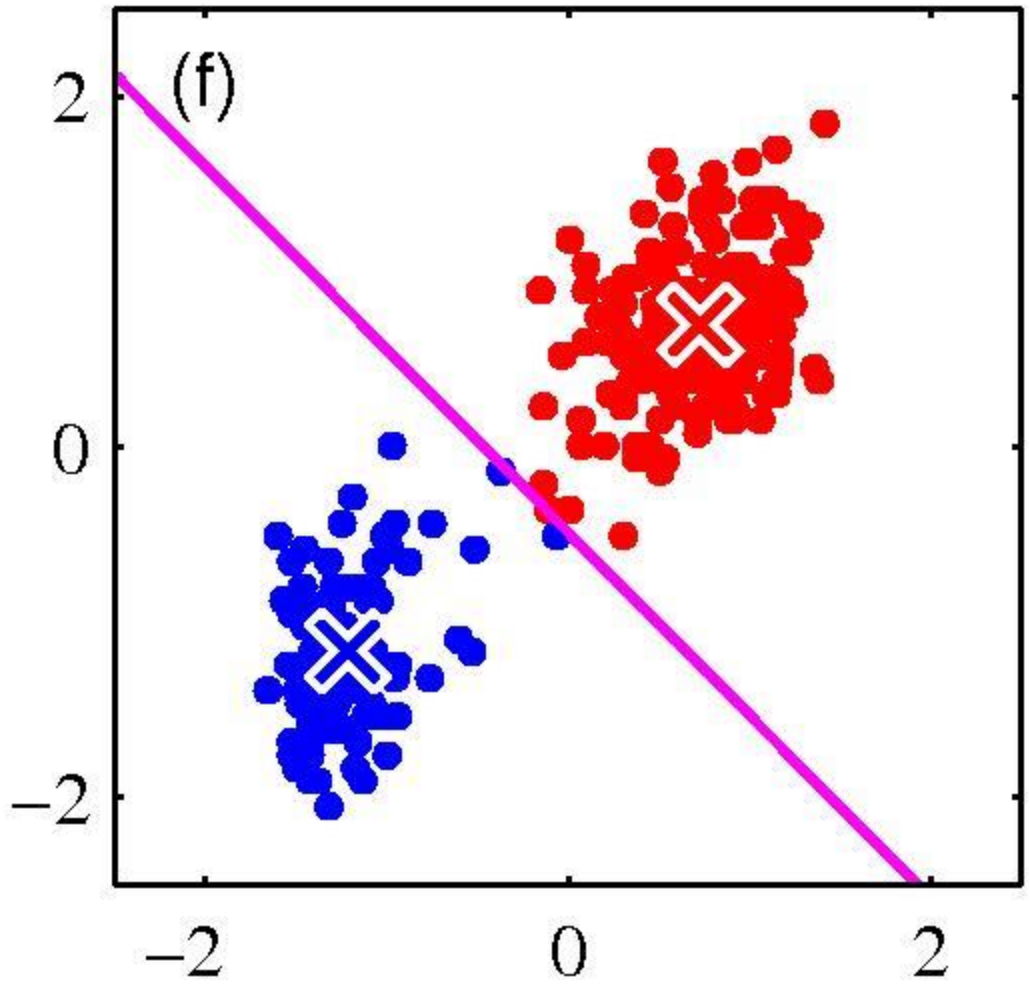


- Repeat until convergence

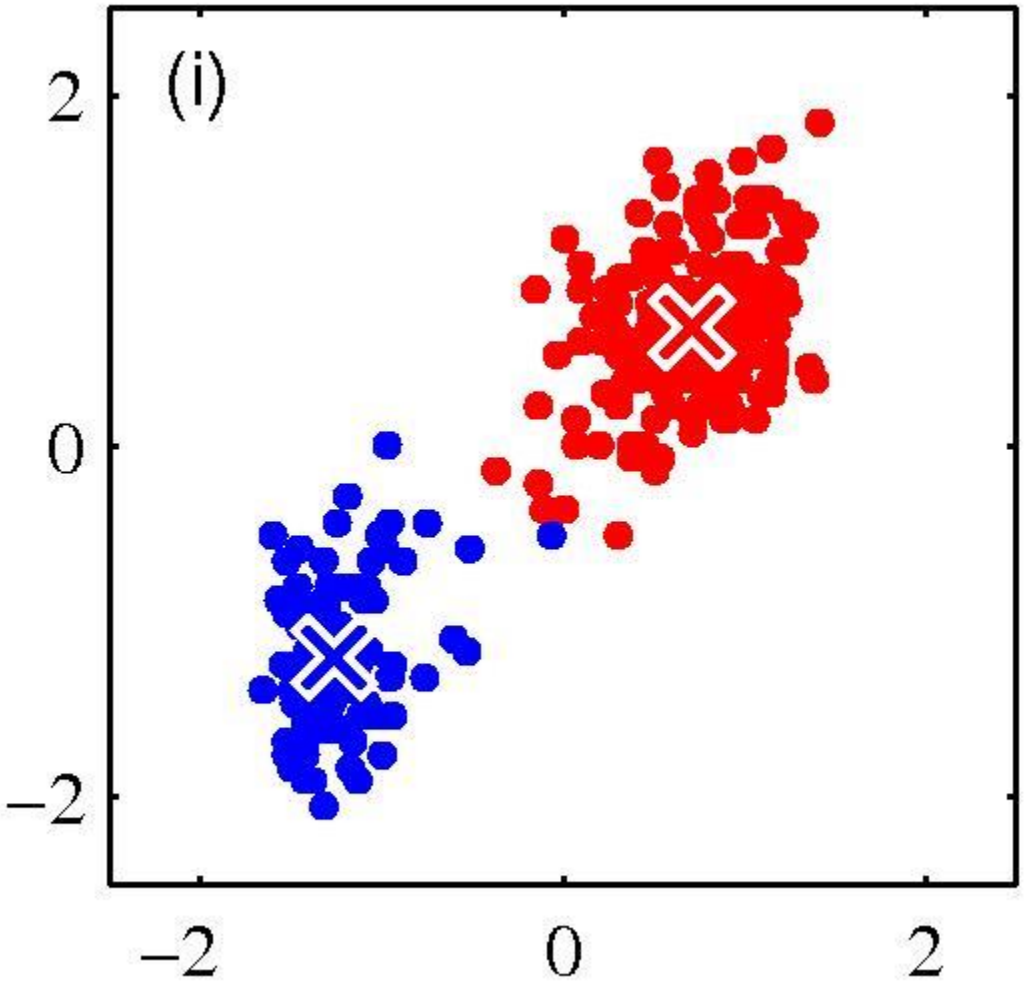
K-means clustering: Example



K-means clustering: Example



K-means clustering: Example



Example: K-Means for Segmentation

K=2

Goal of Segmentation is to partition an image into regions each of which has reasonably homogenous visual appearance.

Original

Example: K-Means for Segmentation

K=2

K=3

Original

Example: K-Means for Segmentation

K=2

K=3

K=10

Original

4%

8%

17%

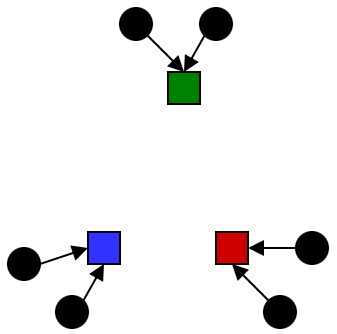
K-Means as Optimization

- 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

- Two stages each iteration:
  - Update assignments: fix means c, change assignments a
  - Update means: fix assignments a, change means c



- Co-ordinate Gradient Descent
- Will it converge?
  - Yes!, if you can argue that each update can't increase Φ

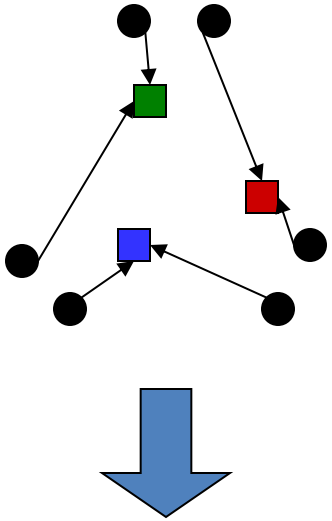
Phase I: Update Assignments

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

$$a_i = \operatorname*{argmin}_k \mathsf{dist}(x_i, c_k)$$

- Can only decrease total distance phi!

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



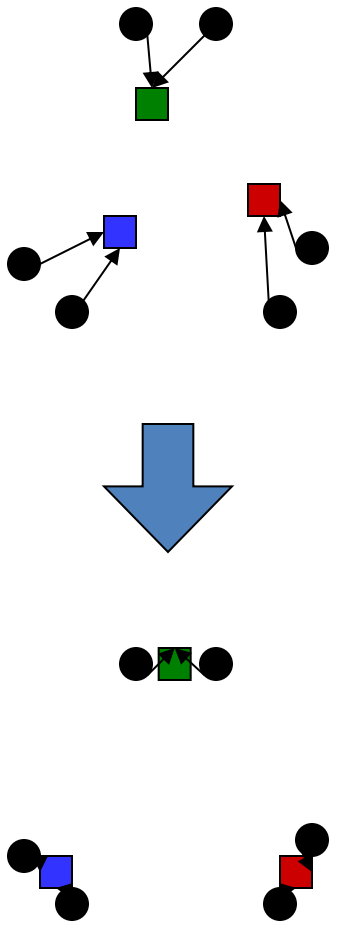
Phase II: Update Means

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

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

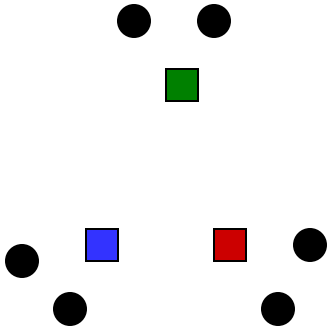
- Also can only decrease total distance… (Why?)

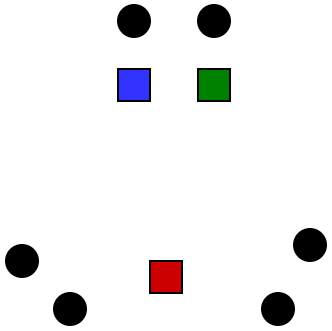
- Fun fact: the point y with minimum squared Euclidean distance to a set of points {x} is their mean



Initialization

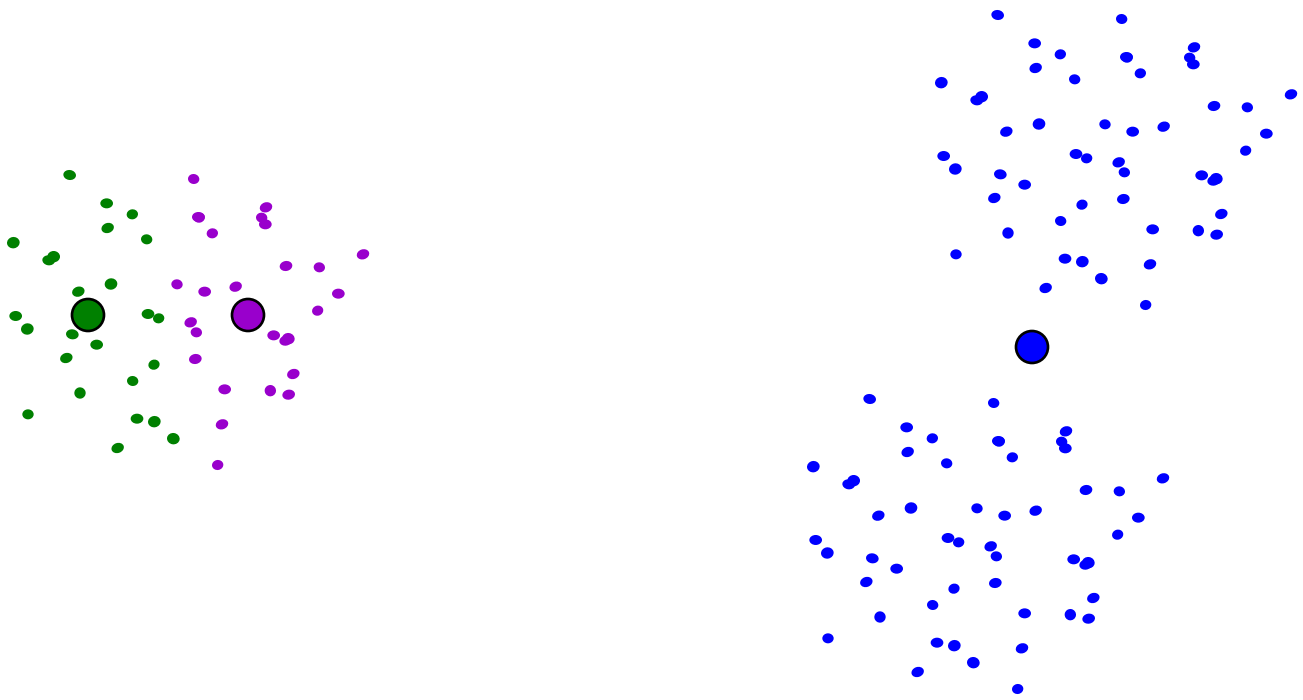
- K-means is non-deterministic
  - Requires initial means
  - It does matter what you pick!
  - What can go wrong?
  - Various schemes for preventing this kind of thing: variance-based split / merge, initialization heuristics





K-Means Getting Stuck

A local optimum:

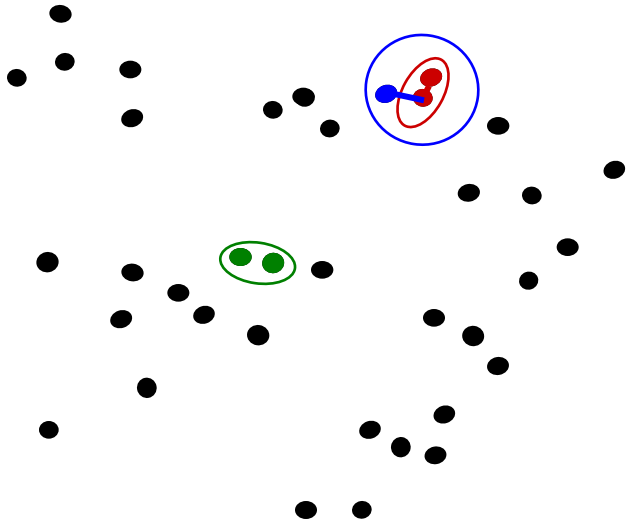


K-Means Questions

- Will K-means converge?
  - To a global optimum?
- Will it always find the true patterns in the data?
  - If the patterns are very very clear?
- Runtime?
- Do people ever use it?
- How many clusters to pick?

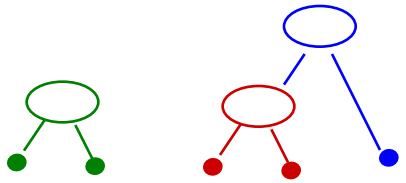
Agglomerative Clustering

- Agglomerative clustering:
  - First merge very similar instances
  - Incrementally build larger clusters out of smaller clusters



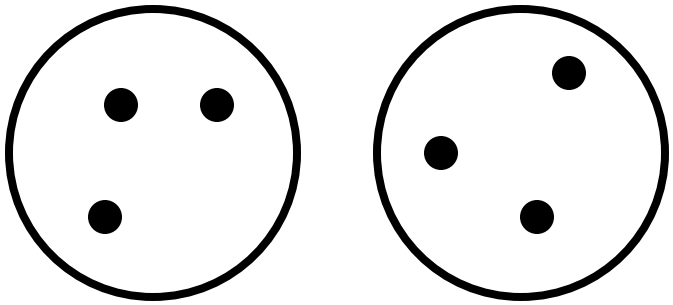
- Algorithm:
  - Maintain a set of clusters
  - Initially, each instance in its own cluster
  - Repeat:
    - Pick the two closest clusters
    - Merge them into a new cluster
    - Stop when there's only one cluster left

- Produces not one clustering, but a family of clusterings represented by a dendrogram



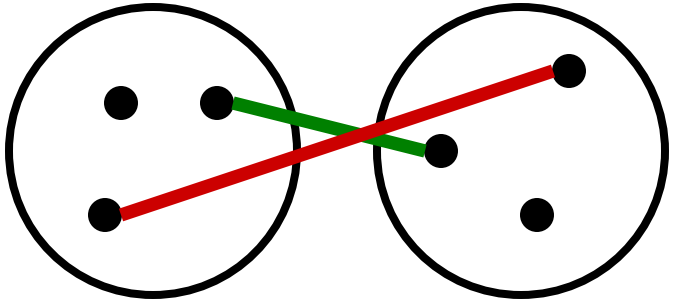
Agglomerative Clustering

- How should we define “closest” for clusters with multiple elements?

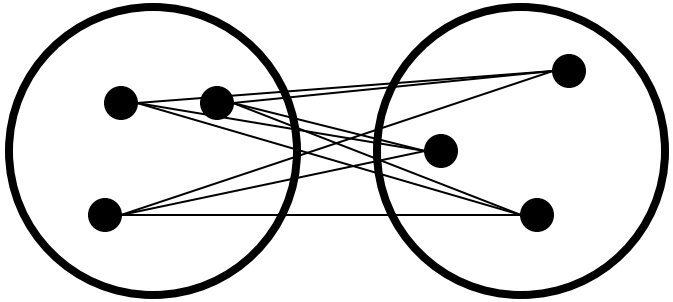


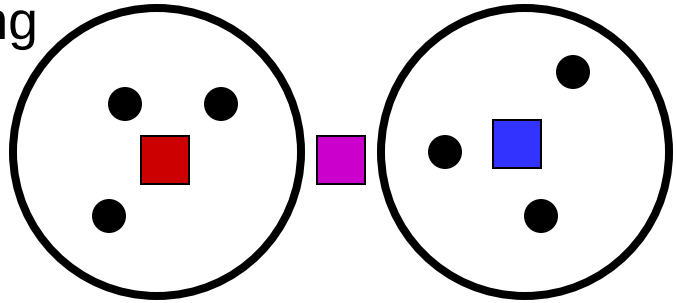
Agglomerative Clustering

- How should we define “closest” for clusters with multiple elements?



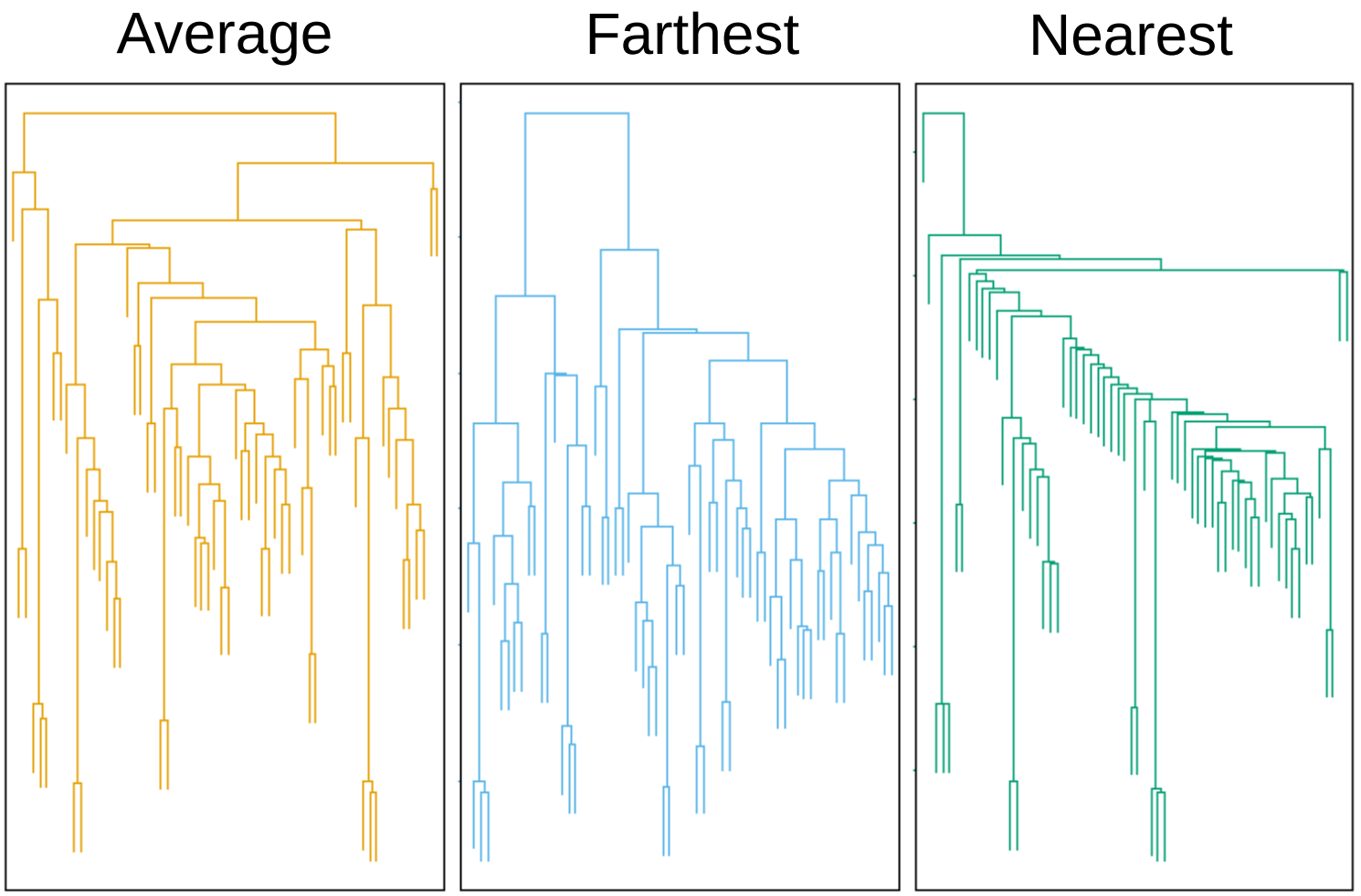
- Many options:
  - Closest pair (single-link clustering)
  - Farthest pair (complete-link clustering)
  - Average of all pairs
  - Ward’s method (min variance, like k-means)
    - Find pair of clusters that leads to minimum increase in total within cluster distance after merging





- Different choices create different clustering behaviors

Clustering Behavior



Mouse tumor data from [Hastie]

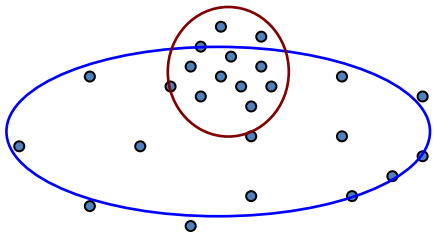
Agglomerative Clustering Questions

- Will agglomerative clustering converge?
  - To a global optimum?
- Will it always find the true patterns in the data?
- Do people ever use it?
- How many clusters to pick?

EM: Soft Clustering

- Clustering typically assumes that each instance is given a “hard” assignment to exactly one cluster.
- Does not allow uncertainty in class membership or for an instance to belong to more than one cluster.
  - Problematic because data points that lie roughly midway between cluster centers are assigned to one cluster
- *Soft clustering* gives probabilities that an instance belongs to each of a set of clusters.

Probabilistic Clustering



- Try a probabilistic model!
  - allows overlaps, clusters of different size, etc.
- Can tell a *generative story* for data
  - P(X|Z) P(Z)
- Challenge: we need to estimate model parameters without labeled Zs

Z	X_1	X_2
??	0.1	2.1
??	0.5	-1.1
??	0.0	3.0
??	-0.1	-2.0
??	0.2	1.5
…	…	…

Finite Mixture Models

$\underline{x}_i$ is a d-dimensional vector

- Given a dataset: $D = \{\underline{x}_1, \ldots, \underline{x}_N\}$
- **Mixture model**: $\Theta = \{\alpha_1, \ldots, \alpha_K, \theta_1, \ldots, \theta_K\}$

$$p(\underline{x}|\Theta) = \sum_{k=1}^{K} \alpha_k p_k(\underline{x}|z_k, \theta_k)$$

The $p_k(\underline{x}|z_k, \theta_k)$ are *mixture components*, $1 \leq k \leq K$

$z = (z_1, \ldots, z_K)$ is a vector of K binary indicator variables

Note: only one of them equals 1 at any given point. Each point is assumed to be generated from exactly one mixture component!

Mixture Weights. $\alpha_k = p(z_k)$ $\sum_{k=1}^{K} \alpha_k = 1$

Finite Mixture Model: Probabilistic View

the "membership weight" of data point $\underline{x}_i$ in cluster k, given parameters Θ

$$w_{ik} = p(z_{ik} = 1 | \underline{x}_i, \Theta) = \frac{p_k(\underline{x}_i | z_k, \theta_k) \cdot \alpha_k}{\sum_{m=1}^{K} p_m(\underline{x}_i | z_m, \theta_m) \cdot \alpha_m}$$

- The membership weight express our uncertainty about which of the "K" components generated the vector $\underline{x}_i$.

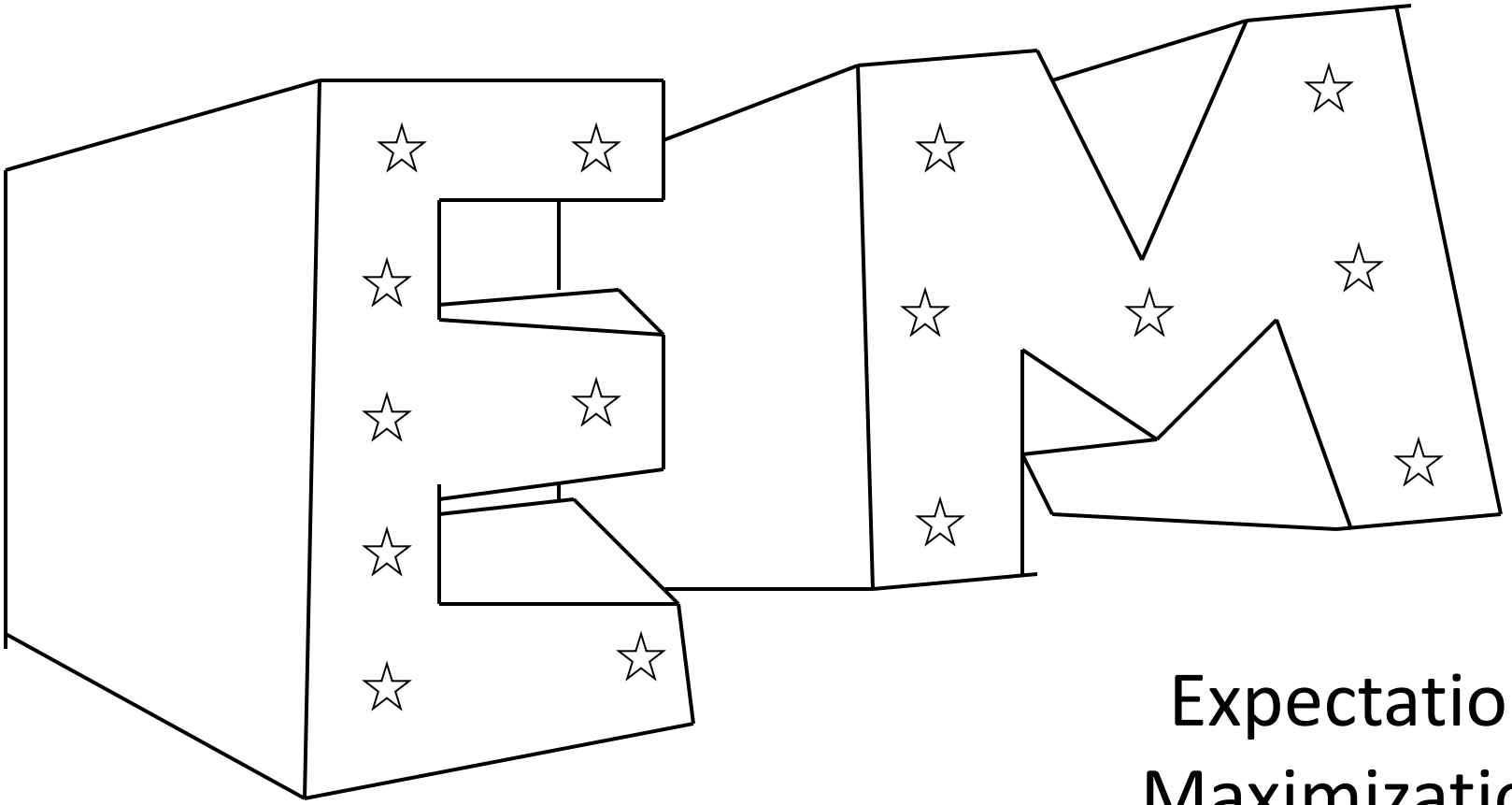
Gaussian Mixture Models (GMMs)

$$p_k(\underline{x}|\theta_k) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(\underline{x}-\underline{\mu}_k)^t \Sigma_k^{-1}(\underline{x}-\underline{\mu}_k)}$$

- We can define a GMM by making each “k-th” component a Gaussian density with parameters:

$$\theta_k = \{\underline{\mu}_k, \Sigma_k\}$$

Question: How to learn these parameters from data?



Expectation
Maximization

EM algorithm: Key Idea

- Start with random parameters
- Find a class for each example (E-step)
  - Since we are using probabilistic classification, each example will be given a vector of probabilities
- Now we have a supervised learning problem. Estimate the parameters of the model using the maximum likelihood method (M-step)
- Iterate between the E-step and M-step until convergence

EM: Two Easy Steps

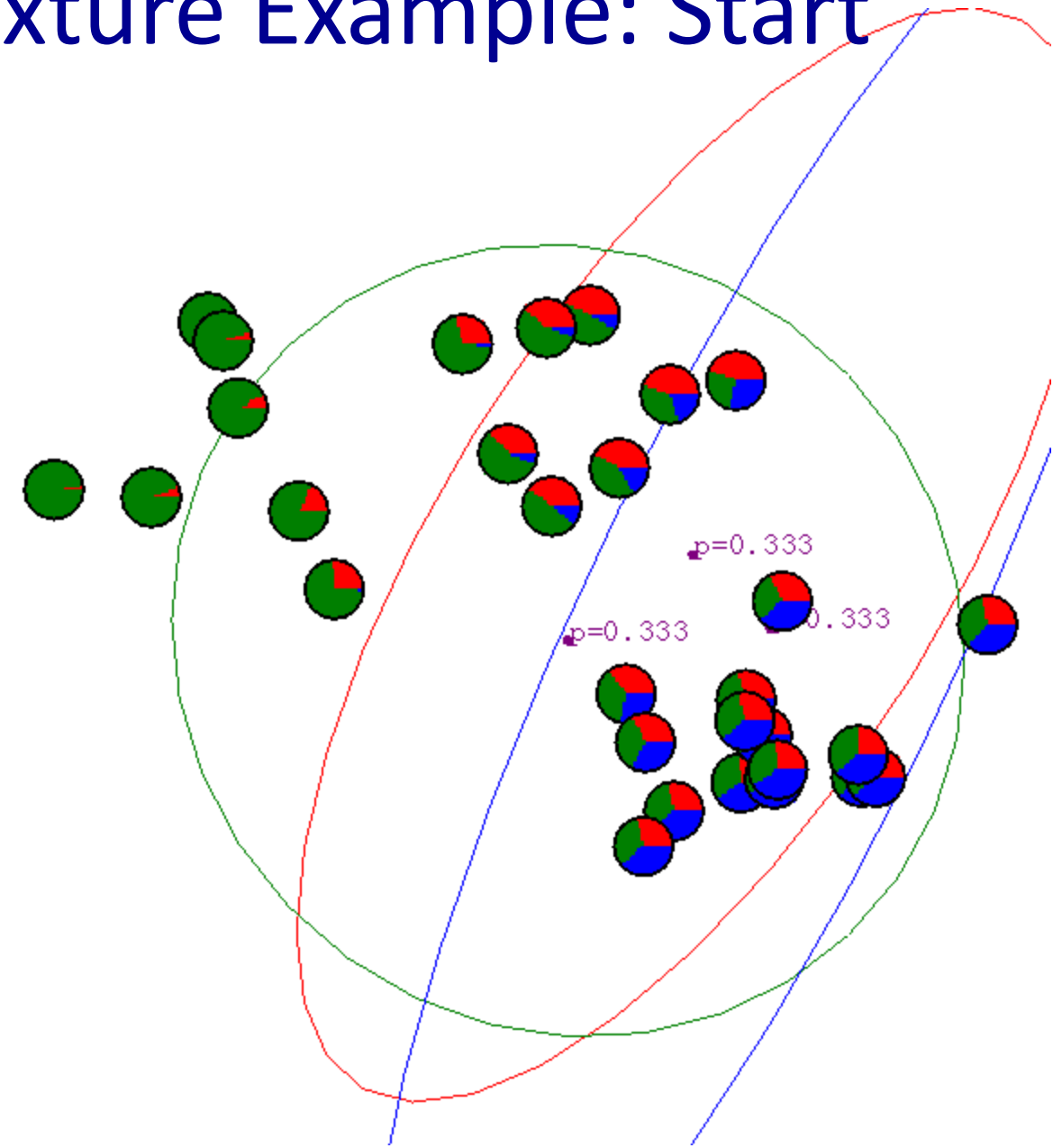
- E-step: (Yields a N x K matrix)
  - Compute w_{ik} for all data points indexed by "i" and all mixture components indexed by "k."
- M-step:
  - Use the membership weights and data to compute the new parameters

$$N_k = \sum_{i=1}^{N} w_{ik} \qquad \alpha_k^{new} = \frac{N_k}{N}$$

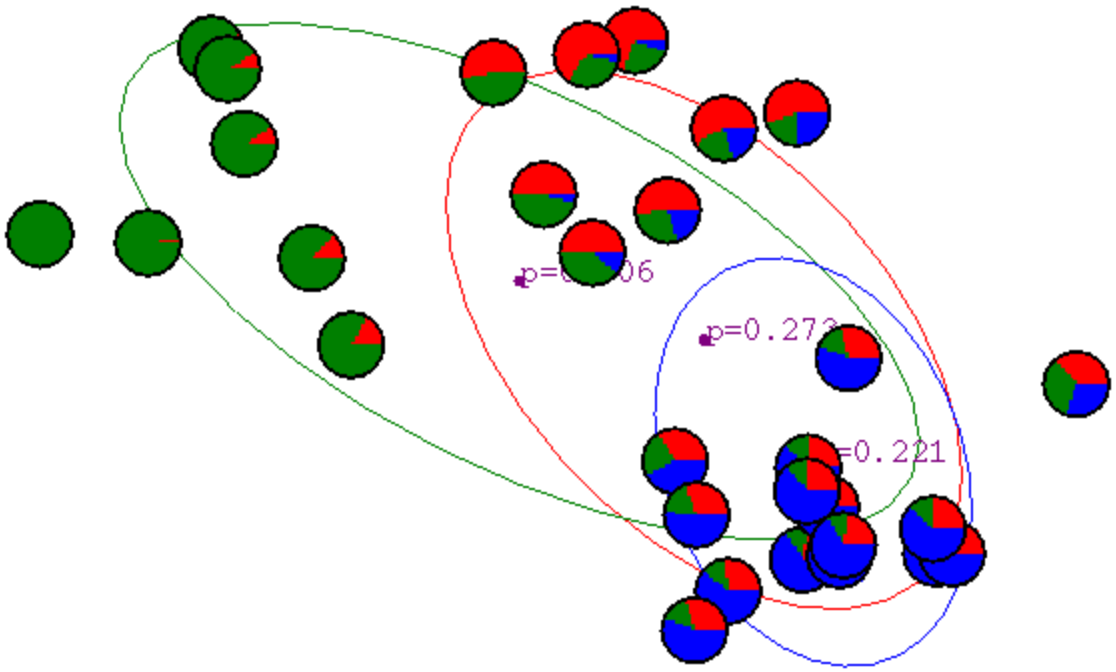
$$\underline{\mu}_k^{new} = \left(\frac{1}{N_k}\right) \sum_{i=1}^{N} w_{ik} \cdot \underline{x}_i$$

$$\Sigma_k^{new} = \left(\frac{1}{N_k}\right) \sum_{i=1}^{N} w_{ik} \cdot (\underline{x}_i - \underline{\mu}_k^{new})(\underline{x}_i - \underline{\mu}_k^{new})^t$$

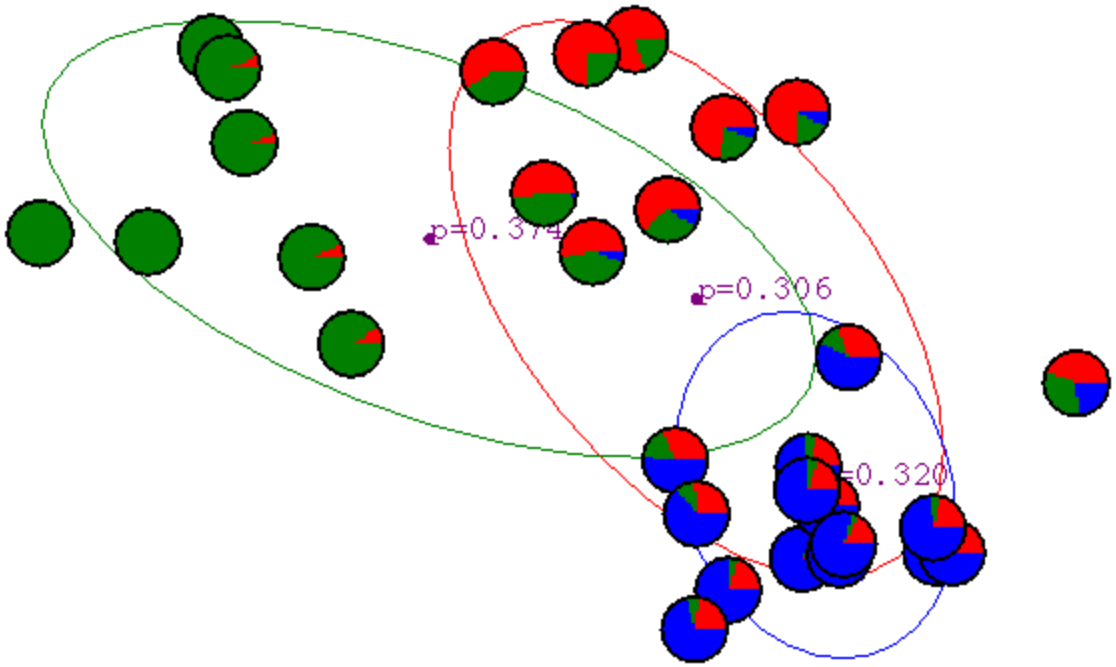
Gaussian Mixture Example: Start



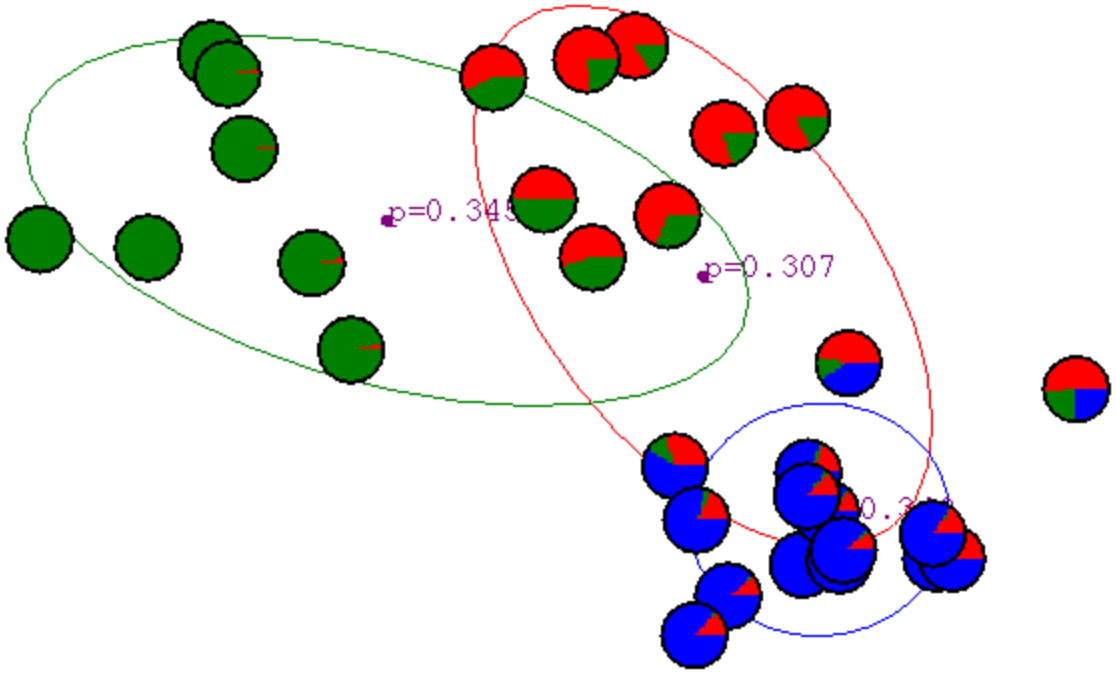
After first iteration



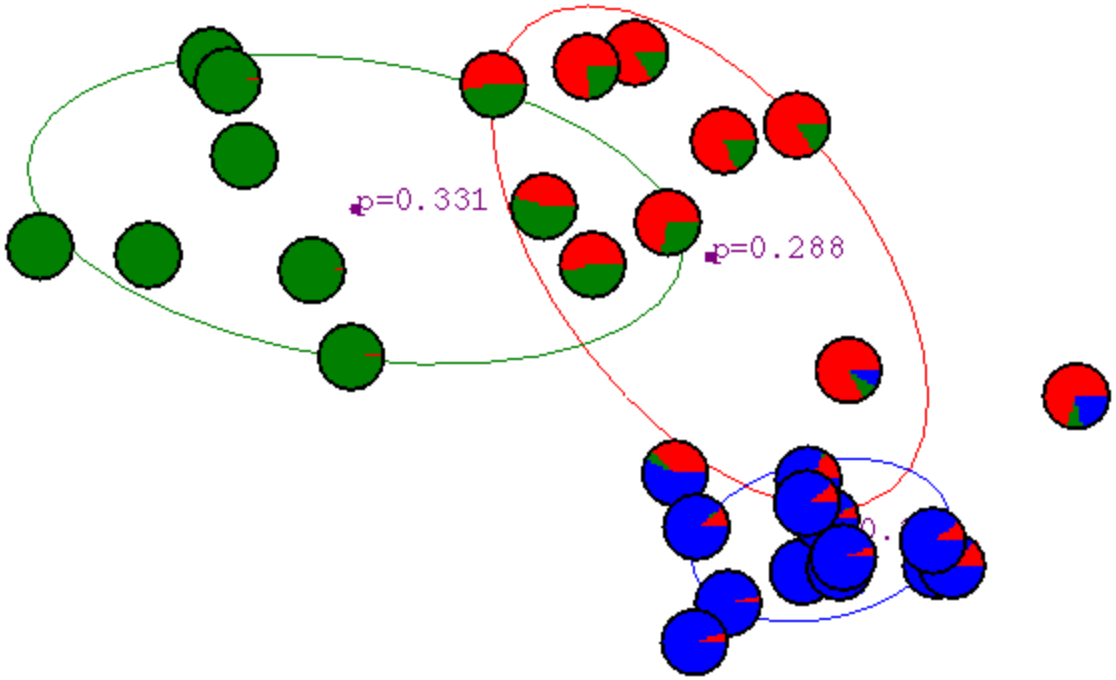
After 2nd iteration



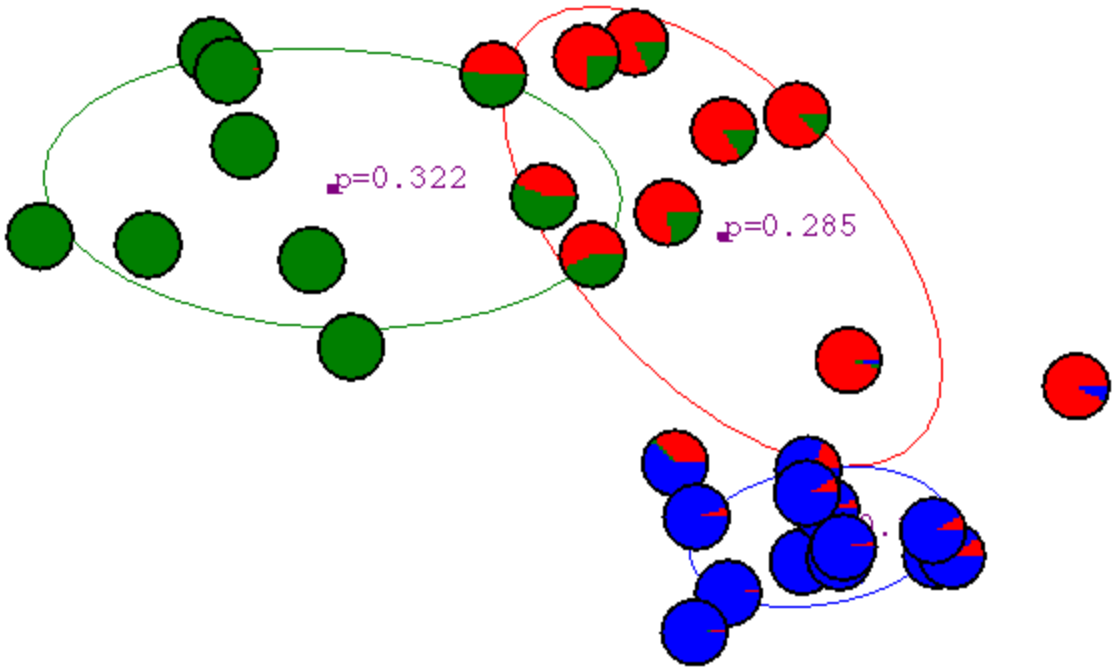
After 3rd iteration



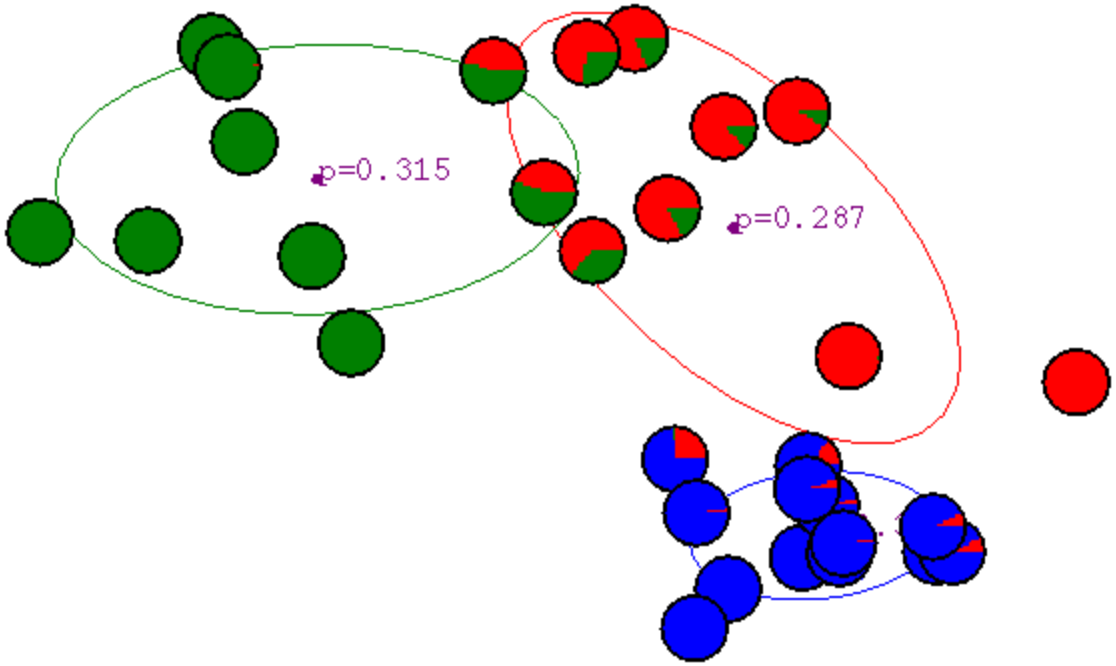
After 4th iteration



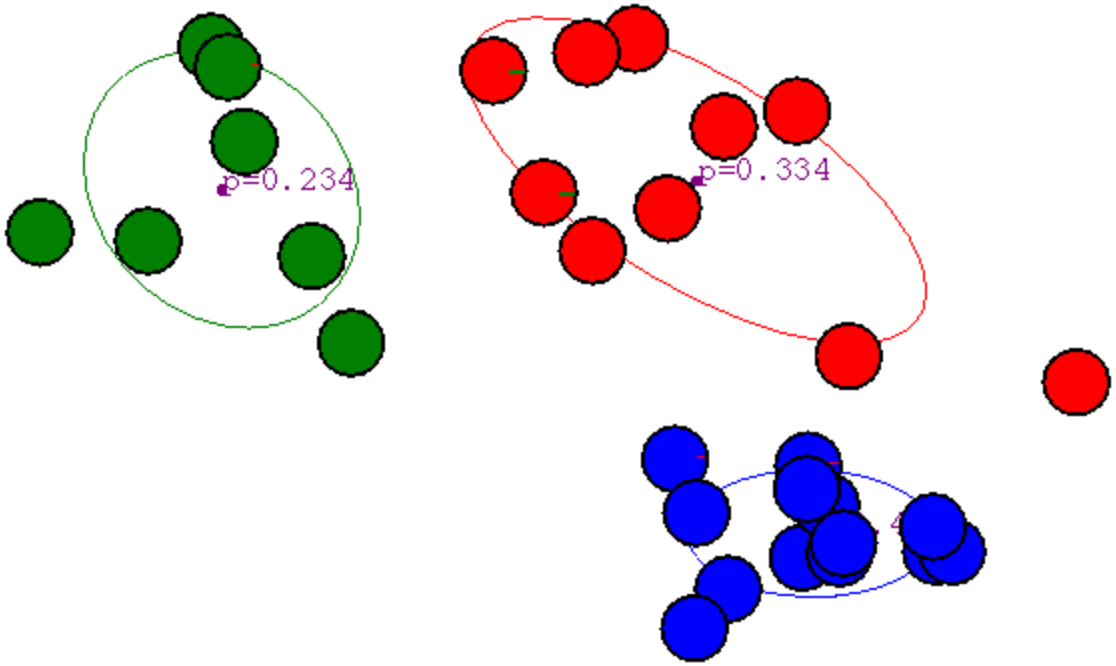
After 5th iteration



After 6th iteration



After 20th iteration



Properties of EM

- EM converges to a local minima
  - This is because each iteration improves the log-likelihood
  - Proof same as K-means
    - E-step can never decrease likelihood
    - M-step can never decrease likelihood
- If we make hard assignments instead of soft ones. Algorithm is equivalent to K-means!

What you should know

- K-means for clustering:
  - algorithm
  - converges because it's coordinate ascent
- Know what agglomerative clustering is
- EM for mixture of Gaussians:
- Remember, E.M. can get stuck in local minima,
  - And empirically it ***DOES!***