Machine Learning: ITCS 6156/8156

Clustering: k-Means and HAC

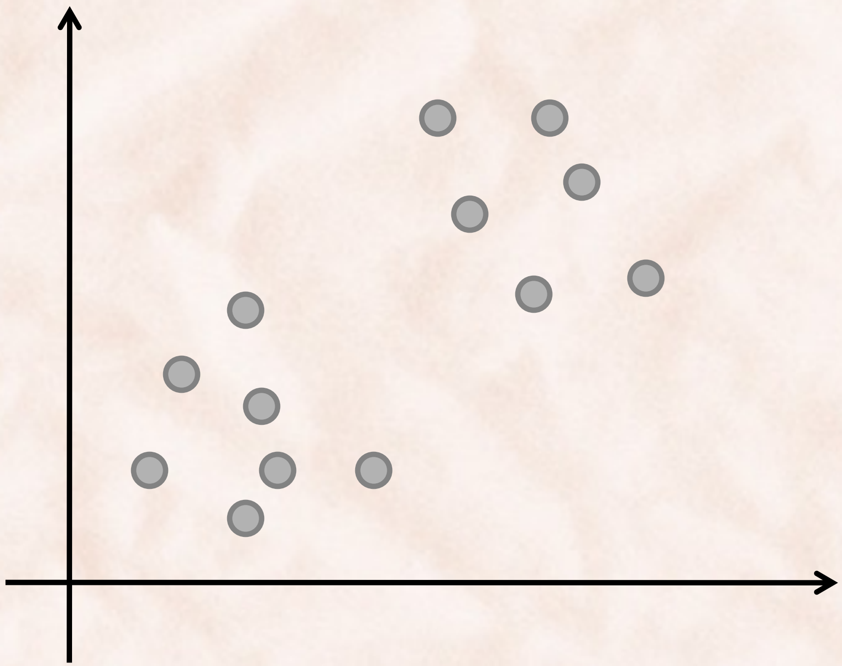
Razvan C. Bunescu

Department of Computer Science @ CCI

razvan.bunescu@uncc.edu

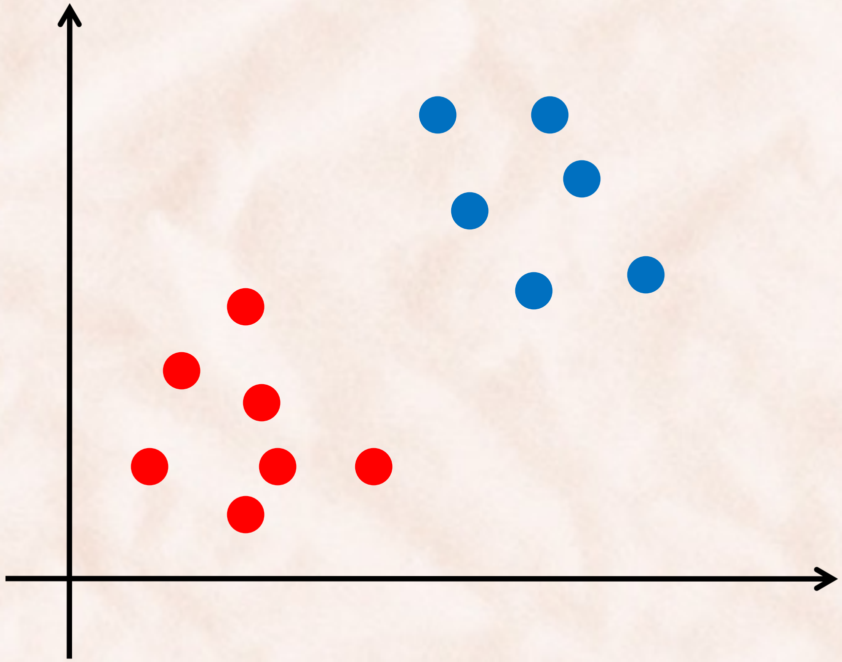
Unsupervised Learning: Clustering

- Partition unlabeled examples into disjoint clusters such that:
  - Examples in the same cluster are very similar.
  - Examples in different clusters are very different.



Unsupervised Learning: Clustering

- Partition unlabeled examples into disjoint clusters such that:
  - Examples in the same cluster are very similar.
  - Examples in different clusters are very different.



Divisive Clustering with k-Means

- The goal is to produce k clusters C $=\{C_1, C_2, \ldots, C_k\}$ such that instances are close to the cluster centroids:
  - The cluster centroid $\mathbf{m}_i$ is the mean of all instances in the cluster C_i.

- Optimization problem:

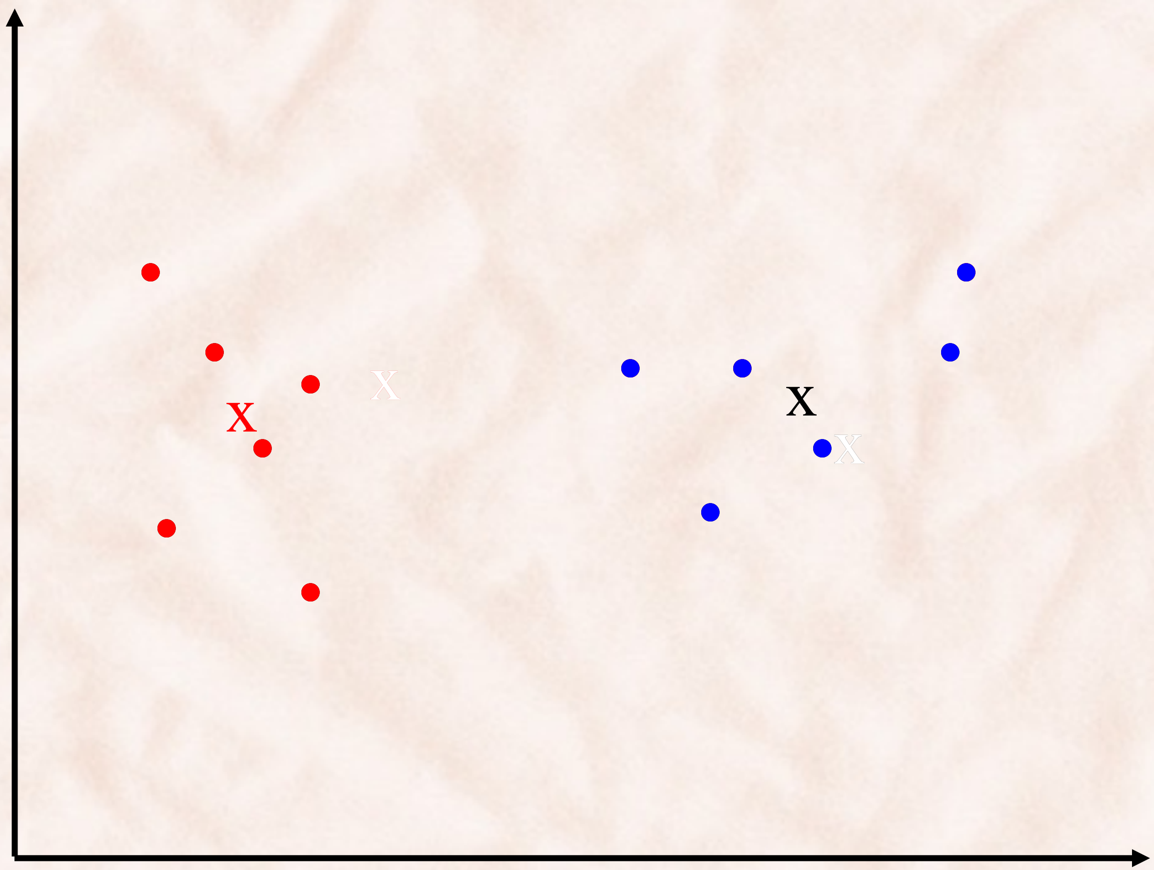
$$\hat{C} = \arg\min_{C} J(C)$$

$$J(C) = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \| \mathbf{x} - \mathbf{m}_i \|^2$$

The k-Means Algorithm

1. start with some seed centroids $\mathbf{m}_1^{(0)}, \mathbf{m}_2^{(0)}, \ldots, \mathbf{m}_k^{(0)}$
2. **set** $t \leftarrow 0$.
3. **while** not converged:
4. **for** each $\mathbf{x}$:
5. **set** $\mathbf{m}^{(t)}(\mathbf{x}) \leftarrow \arg\min_{\mathbf{m}_i^{(t)}} \left\| \mathbf{x} - \mathbf{m}_i^{(t)} \right\|$ ← [**E**] step
6. **set** $C_i^{(t+1)} \leftarrow \left\{ \mathbf{x} \mid \mathbf{m}^{(t)}(\mathbf{x}) = \mathbf{m}_i^{(t)} \right\}$
7. **set** $\mathbf{m}_i^{(t+1)} \leftarrow \frac{1}{\left|C_i^{(t+1)}\right|} \sum_{\mathbf{x} \in C_i^{(t+1)}} \mathbf{x}$ ← [**M**] step
8. **set** $t \leftarrow t + 1$

The k-Means Algorithm ($k = 2$)



Pick seeds

Reassign clusters

Compute centroids

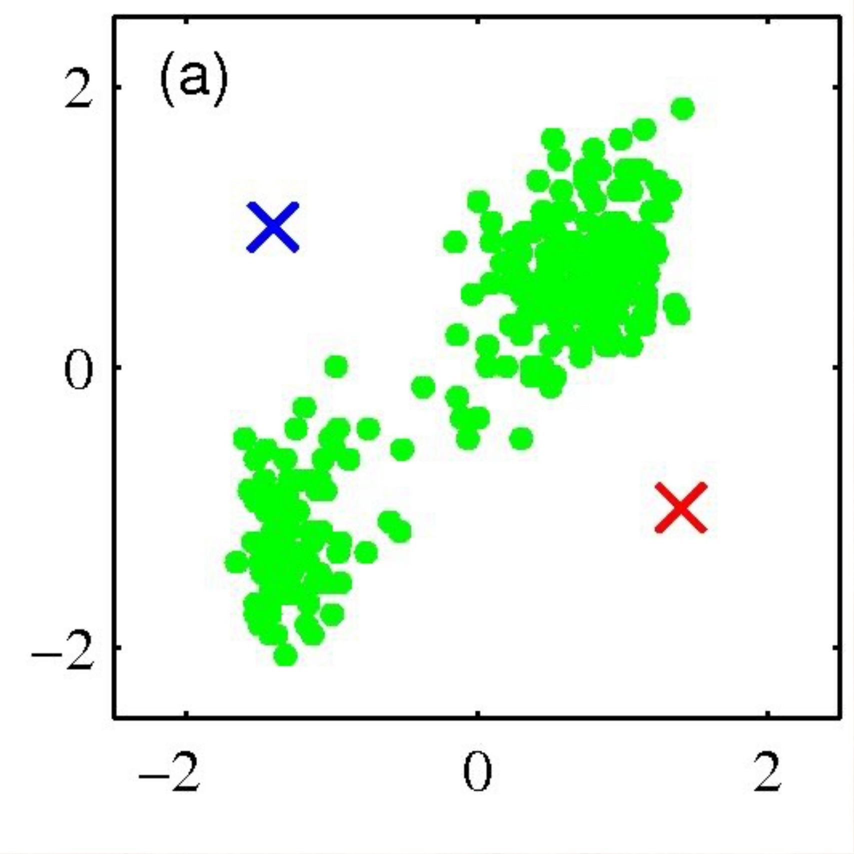
Reasssign clusters

Compute centroids

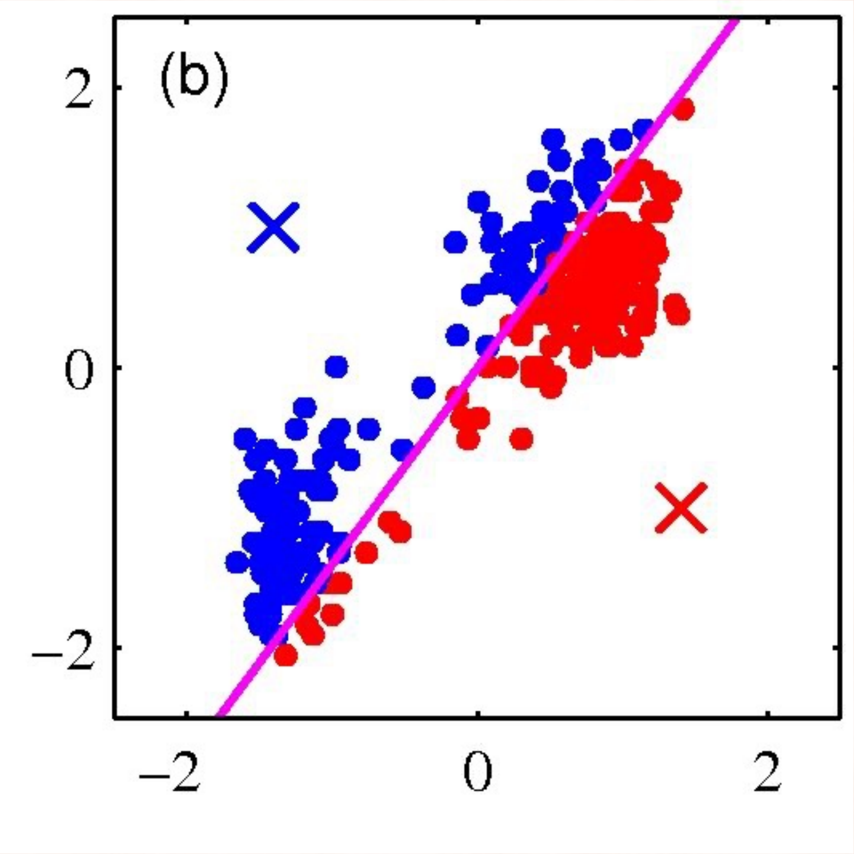
Reassign clusters

Converged!

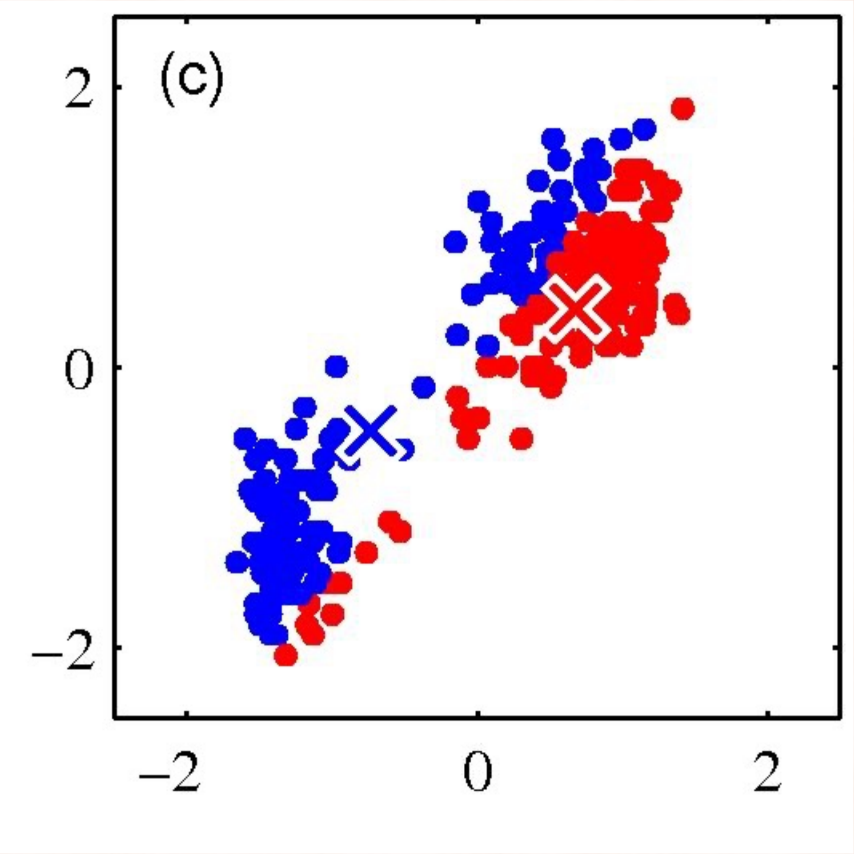
The k-Means Algorithm ($k = 2$)



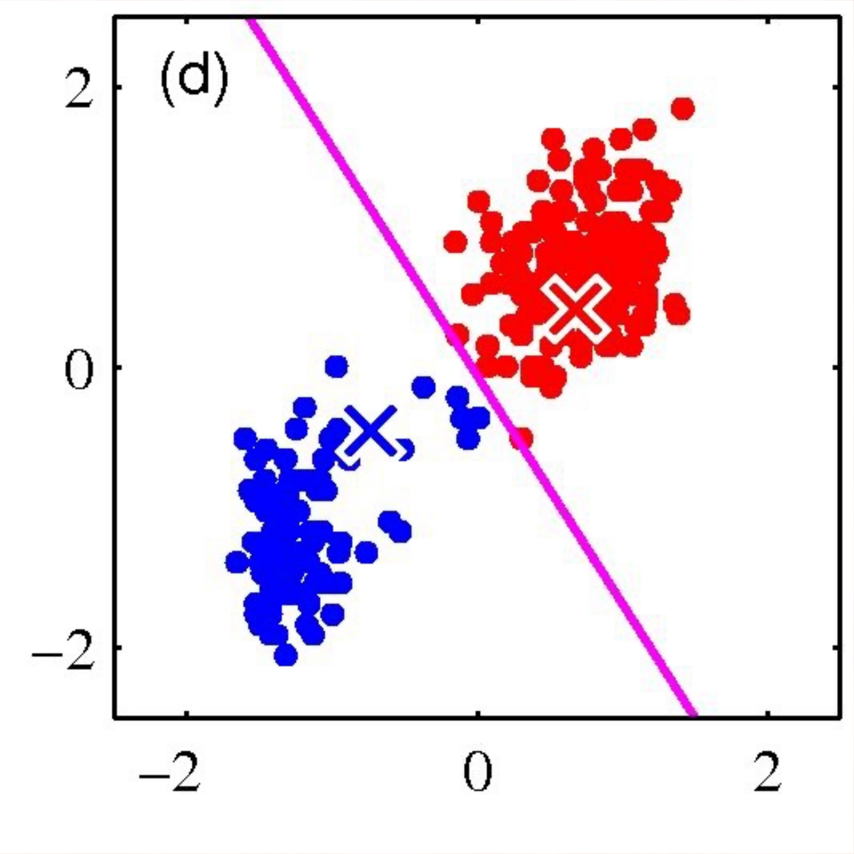
The k-Means Algorithm ($k = 2$)



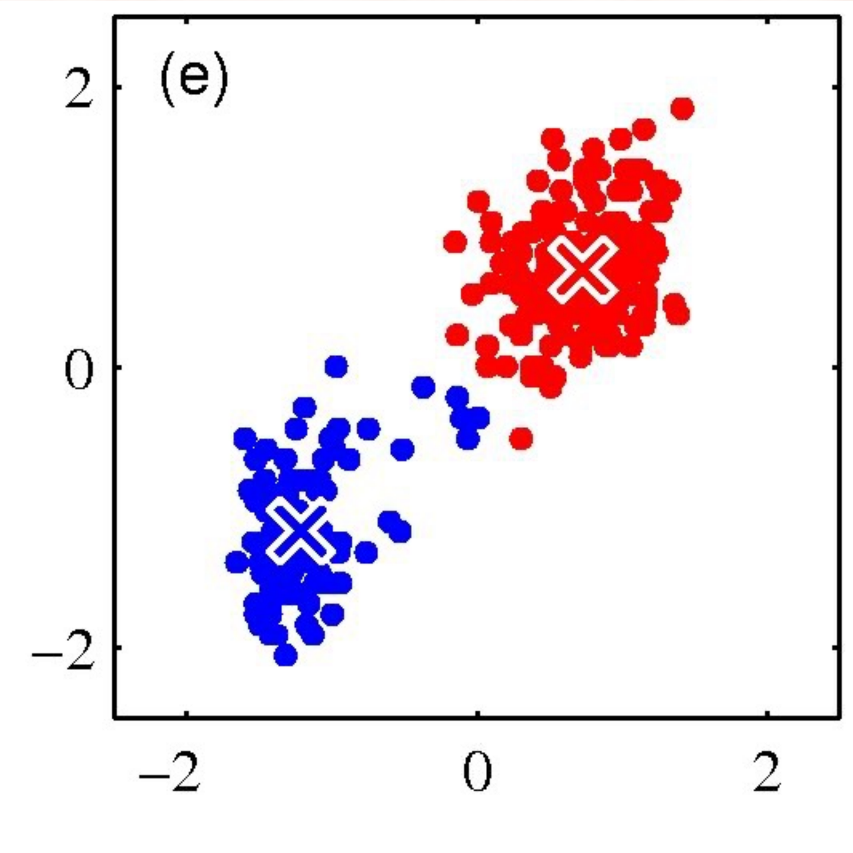
The k-Means Algorithm ($k = 2$)



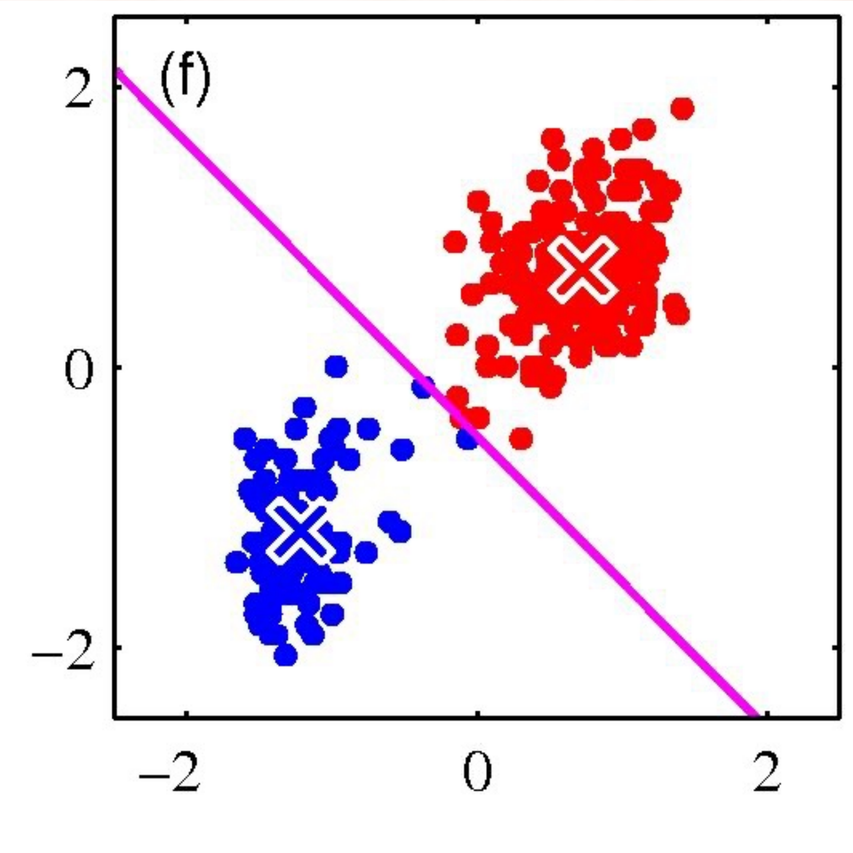
The k-Means Algorithm ($k = 2$)



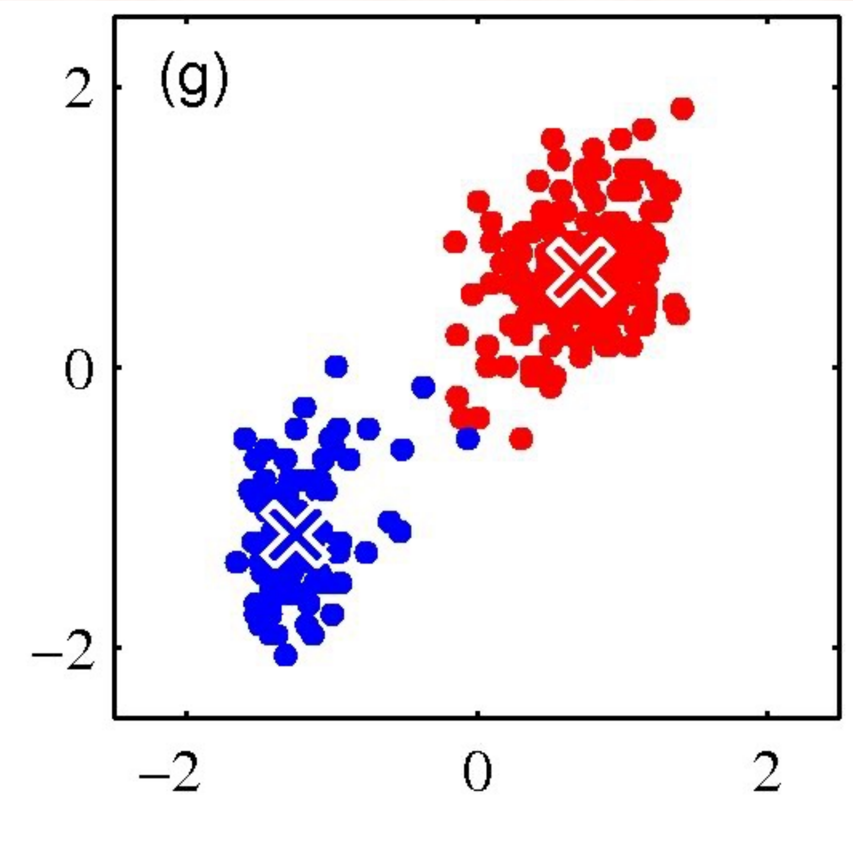
The *k*-Means Algorithm ($k = 2$)



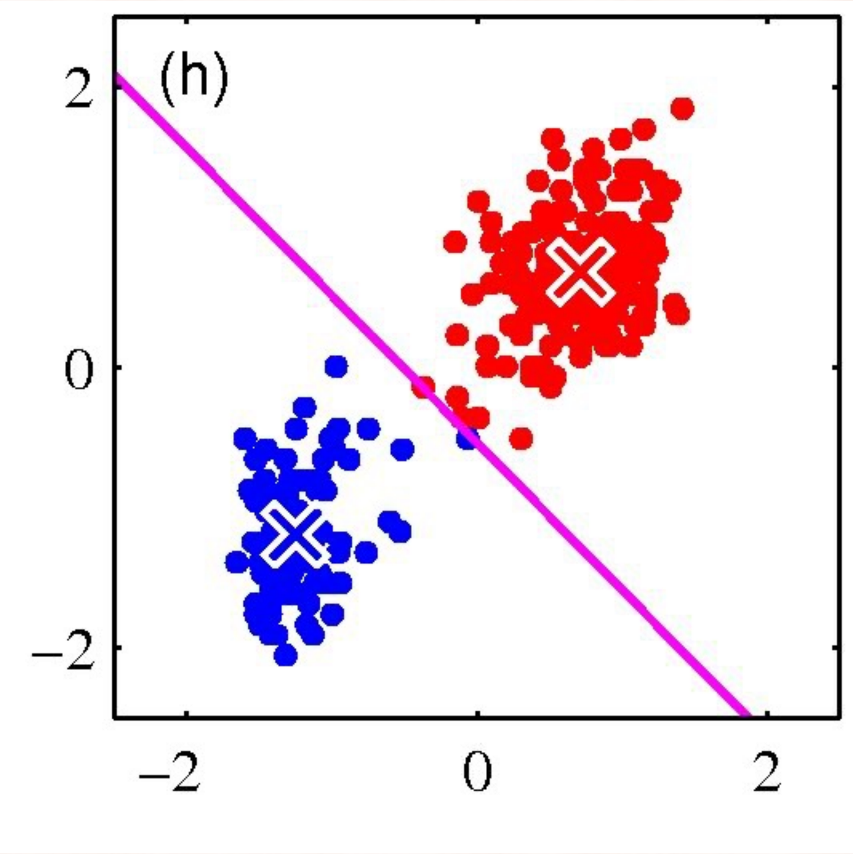
The k-Means Algorithm ($k = 2$)



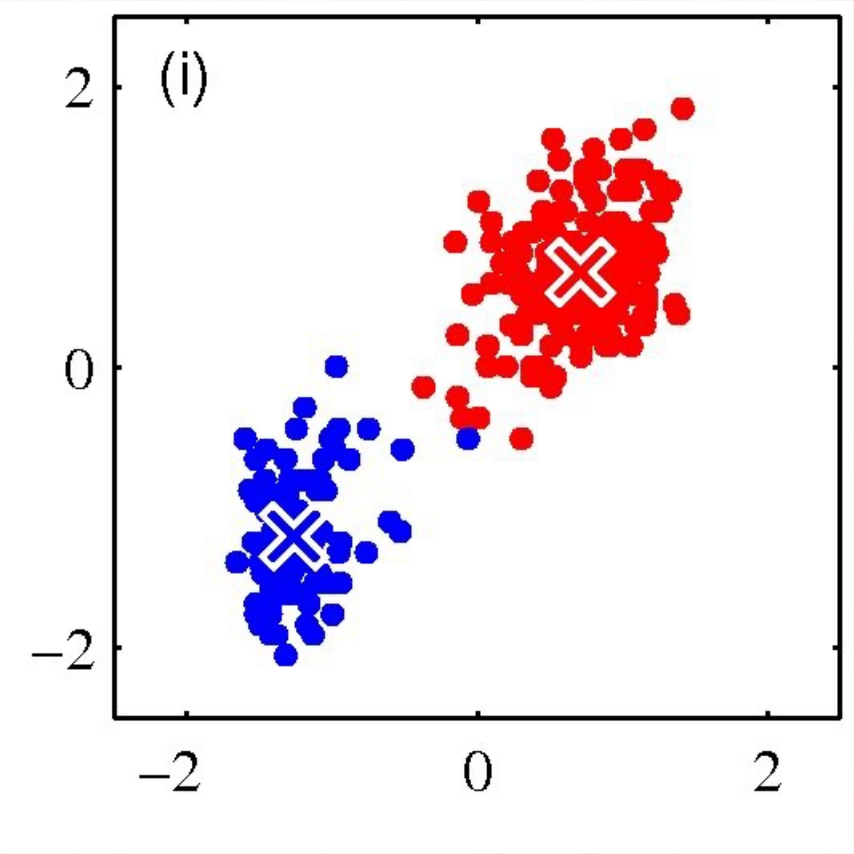
The k-Means Algorithm ($k = 2$)



The k-Means Algorithm ($k = 2$)



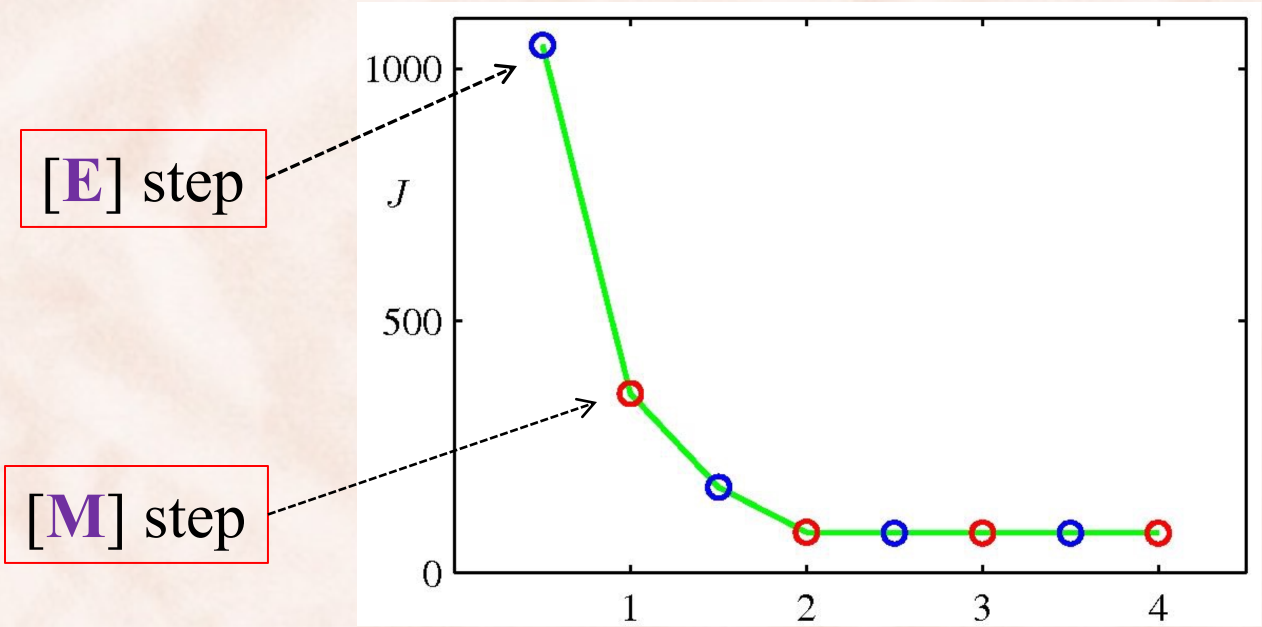
The k-Means Algorithm ($k = 2$)



The *k*-Means Algorithm

- The objective function monotonically decreases at every iteration:

$$J^{(t)} \geq J^{(t+1)}$$



The *k*-Means Algorithm

- Optimization problem is NP-hard:
  - Results depend on seed selection.
  - Improve performance by providing *must-link* and/or *cannot-link* constraints $\Rightarrow$ semi-supervised clustering.

- Time complexity for each iteration is O(*knm*):
  - number of clusters is *k*.
  - feature vectors have dimensionality *m*.
  - total number of instances is *n*.

The *k*-Means Algorithm

1. start with some seed centroids $\mathbf{m}_1^{(0)}, \mathbf{m}_2^{(0)}, \ldots, \mathbf{m}_k^{(0)}$
2. **set** $t \leftarrow 0$.
3. **while** not converged:
4. **for** each **x**:
5. **set** $\mathbf{m}^{(t)}(\mathbf{x}) \leftarrow \arg\min_{\mathbf{m}_i^{(t)}} \left\| \mathbf{x} - \mathbf{m}_i^{(t)} \right\|$ ← [**E**] step
6. **set** $C_i^{(t+1)} \leftarrow \left\{ \mathbf{x} \mid \mathbf{m}^{(t)}(\mathbf{x}) = \mathbf{m}_i^{(t)} \right\}$
7. **set** $\mathbf{m}_i^{(t+1)} \leftarrow \frac{1}{\left|C_i^{(t+1)}\right|} \sum_{\mathbf{x} \in C_i^{(t+1)}} \mathbf{x}$ ← [**M**] step
8. **set** $t \leftarrow t + 1$

The *k*-Medoids Algorithm

1. start with some random seed centroids $\mathbf{m}_1^{(0)}, \mathbf{m}_2^{(0)}, \dots, \mathbf{m}_k^{(0)}$
2. **set** $t \leftarrow 0$.
3. **while** not converged:
4. **for** each $\mathbf{x}$:
5. **set** $\mathbf{m}^{(t)}(\mathbf{x}) \leftarrow \arg\min_{\mathbf{m}_i^{(t)}} d\left(\mathbf{x} - \mathbf{m}_i^{(t)}\right)$ ←------------ [E] step
6. **set** $C_i^{(t+1)} \leftarrow \left\{ \mathbf{x} \mid \mathbf{m}^{(t)}(\mathbf{x}) = \mathbf{m}_i^{(t)} \right\}$
7. **set** $\mathbf{m}_i^{(t+1)} \leftarrow \arg \min_{\mathbf{x} \in C_i^{(t+1)}} \sum_{\mathbf{y} \in C_i^{(t+1)}} d(\mathbf{x}, \mathbf{y})$ ←------------ [M] step
8. **set** $t \leftarrow t + 1$

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.

- **Soft clustering** gives probabilities that an instance belongs to each of a set of clusters.
- Each instance is assigned a probability distribution across a set of discovered categories.

Soft Clustering with EM

- Soft version of *k*-means.
- Assumes a probabilistic model of categories that allows computing $P(c_i \mid \mathbf{x})$ for each category, c_i, for a given example $\mathbf{x}$.
  - For text, typically assume a naïve-Bayes category model.
    - Parameters $\theta = \{P(c_i), P(w_j \mid c_i) \mid i \in \{1,\ldots k\}, j \in \{1,\ldots,|V|\}\}$

Soft Clustering with EM

- Iterative method for learning probabilistic categorization model from unsupervised data.
- Initially assume random assignment of examples to categories.
- Learn an initial probabilistic model by estimating model parameters θ from this randomly labeled data.
- Iterate following two steps until convergence:
  – Expectation (E-step): Compute $P(c_i \mid \mathbf{x})$ for each example given the current model, and probabilistically re-label the examples based on these posterior probability estimates.
  – Maximization (M-step): Re-estimate the model parameters, θ, from the probabilistically re-labeled data.

Learning with Probabilistic Labels

- Instead of training data labeled with "hard" category labels, training data is labeled with "soft" probabilistic category labels.
- When estimating model parameters θ from training data, weight counts by the corresponding probability of the given category label.
- For example, if $P(c_1 \mid \mathbf{x}) = 0.8$ and $P(c_2 \mid \mathbf{x}) = 0.2$, each word w_j in **x** contributes only 0.8 towards the counts n_1 and n_{1j}, and 0.2 towards the counts n_2 and n_{2j}.

Naïve Bayes EM

1. Randomly assign examples probabilistic category labels.
2. Use standard naïve-Bayes training to learn a probabilistic model with parameters θ from the labeled data.
3. Until convergence or until maximum number of iterations reached:
   - E-Step: Use the naïve Bayes model θ to compute $P(c_i \mid \mathbf{x})$ for each category and example, and re-label each example using these probability values as soft category labels.
   - M-Step: Use standard naïve-Bayes training to re-estimate the parameters θ using these new probabilistic category labels.

Hierarchical Agglomerative Clustering (HAC)

- Start out with *n* clusters, one example per cluster.
- At each step merge the *nearest* two clusters.
- Stop when there is only one cluster left, or:
  - there are only *k* clusters left.
  - distance is above a threshold τ.

- History of clustering decision can be represented as a binary tree.

The HAC Algorithm

1. **let** $C_i = \{\mathbf{x}_i\}$, for $i \in 1 \ldots n$
2. **let** $C = \{C_i\}$, for $i \in 1 \ldots n$
3. **while** $|C| > 1$:
4. **set** $\langle C_i, C_j \rangle = \arg\min_{C_k \neq C_l} d(C_k, C_l)$
5. **replace** C_i , C_j in C with $C_i \cup C_j$

Q: How do we compute the distance d between two clusters?

Distance Measures

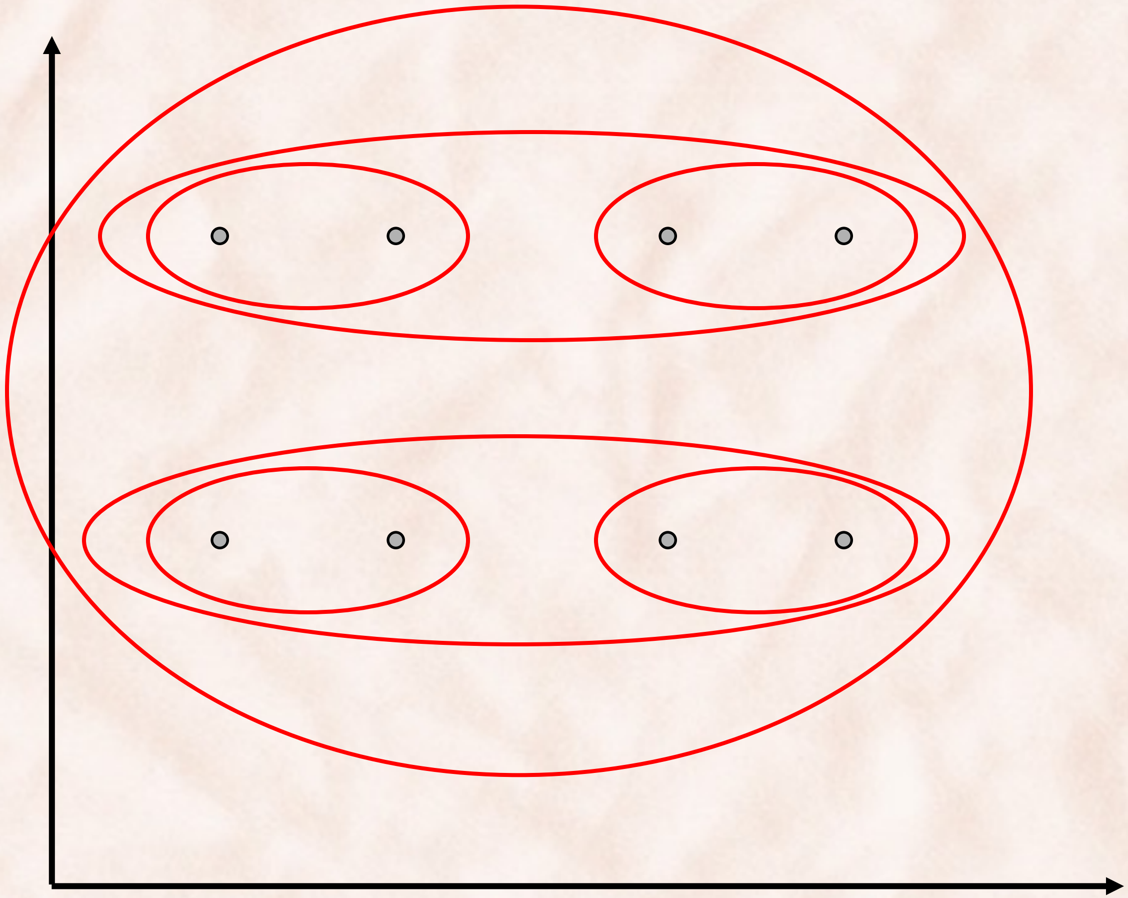
- Assume a distance function between any two instances:
  - Euclidean distance ||x−y||
- Single Link: $d(C_i, C_j) = \min_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} \|\mathbf{x} - \mathbf{y}\|$
- Complete Link: $d(C_i, C_j) = \max_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} \|\mathbf{x} - \mathbf{y}\|$
- Group Average: $d(C_i, C_j) = \frac{1}{|C_i| * |C_j|} \sum_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} \|\mathbf{x} - \mathbf{y}\|$
- Centroid Distance: $d(C_i, C_j) = \|\mathbf{m}_i - \mathbf{m}_j\|$

Single Link (Nearest Neighbor)

- Distance function $d(C_i, C_j) = \min_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} \|\mathbf{x} - \mathbf{y}\|$

- It favors elongated clusters.
- Equivalent with Kruskal’s MST algorithm.

Single Link



Complete Link (Farthest Neighbor)

- Distance function $d(C_i, C_j) = \max_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} \|\mathbf{x} - \mathbf{y}\|$

- It favors tight, spherical clusters.

- $d(C_i,C_j)$ is the *diameter* of the cluster $C_i \cup C_j$.

Complete Link