Clustering

Image Segmentation


Image Segmentation

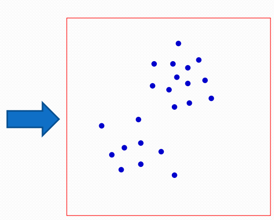

Pink/White pixel : Apple blossom

Orange pixel : Orange

Green pixel : leaf

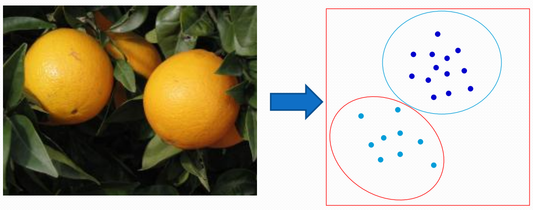
Image Segmentation


Pixels as features

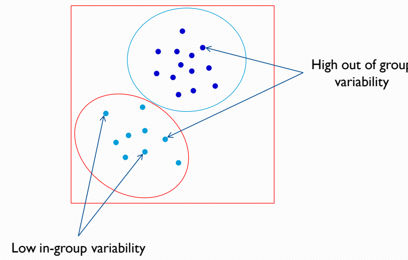



Principle of clustering:
Put things that are closer to each other (in feature space) into the same group

Pixels as features



But what is a 'good' cluster?

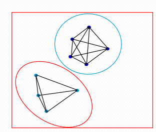


Compactness: Min(in group variability)

- Need a measure that shows how 'compact' our clusters are
- Distance based measures

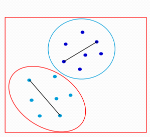
Distance-based Measures

- Total distance between each element in the cluster and every other element



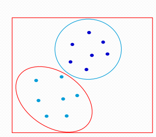
Distance-based Measures

- Distance between farthest points in cluster



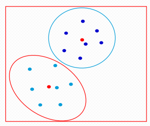
Distance-based Measures

- Total distance of every element in the cluster from the Centroid in the cluster



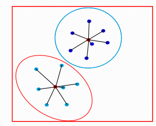
Distance-based Measures

- Total distance of every element in the cluster from the Centroid in the cluster



Distance-based Measures

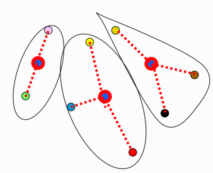
- Total distance of every element in the cluster from the Centroid in the cluster



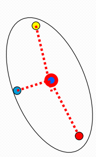
Finding clusters: K-means

K-means algorithm

- Minimizes scatter: Distance from centroid

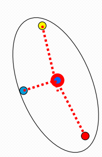


What is a 'Centroid'



$$m_{cluster} = \frac{1}{n} \sum_{i \in cluster} x_i$$

What is a 'Centroid'



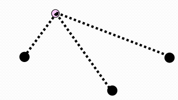
$$m_{cluster} = \frac{1}{\sum_{i \in cluster} w_i} \sum_{i \in cluster} w_i x_i$$

K–means

1. Initialize a set of centroids randomly

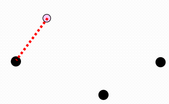
K–means

1. Initialize a set of centroids randomly
2. For each data point ***x***, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$



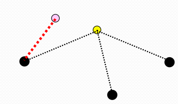
K–means

1. Initialize a set of centroids randomly
2. For each data point ***x***, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{cluster}$ is minimum



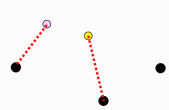
K–means

1. Initialize a set of centroids randomly
2. For each data point ***x***, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{cluster}$ is minimum



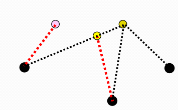
K–means

1. Initialize a set of centroids randomly
2. For each data point ***x***, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{cluster}$ is minimum



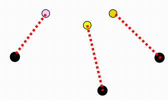
K–means

1. Initialize a set of centroids randomly
2. For each data point ***x***, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{cluster}$ is minimum



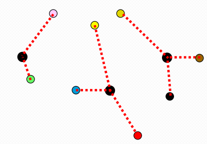
K–means

1. Initialize a set of centroids randomly
2. For each data point $\mathbf{x}$, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$



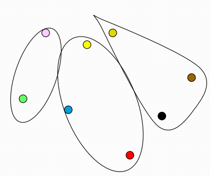
K–means

1. Initialize a set of centroids randomly
2. For each data point $\mathbf{x}$, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\mathbf{d}_{cluster}$ is minimum



K–means

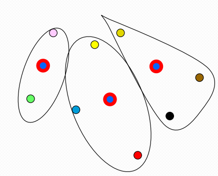
1. Initialize a set of centroids randomly
2. For each data point $\mathbf{x}$, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\mathbf{d}_{cluster}$ is minimum



K–means

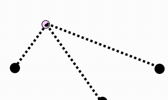
1. Initialize a set of centroids randomly
2. For each data point $\mathbf{x}$, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\mathbf{d}_{cluster}$ is minimum
4. When all data points are clustered, recompute centroids

$$m_{cluster} = \frac{1}{\sum_{i \in cluster} w_i} \sum_{i \in cluster} w_i x_i$$



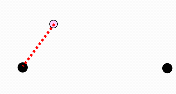
K–means

1. Initialize a set of centroids randomly
2. For each data point $\mathbf{x}$, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$



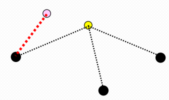
K–means

1. Initialize a set of centroids randomly
2. For each data point $\mathbf{x}$, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\mathbf{d}_{cluster}$ is minimum



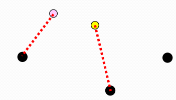
K–means

1. Initialize a set of centroids randomly
2. For each data point ***x***, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{cluster}$ is minimum



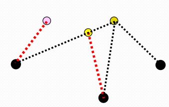
K–means

1. Initialize a set of centroids randomly
2. For each data point ***x***, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{cluster}$ is minimum



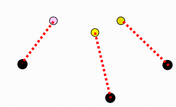
K–means

1. Initialize a set of centroids randomly
2. For each data point ***x***, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{cluster}$ is minimum



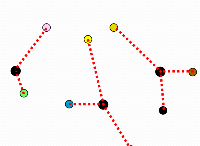
K–means

1. Initialize a set of centroids randomly
2. For each data point ***x***, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{cluster}$ is minimum



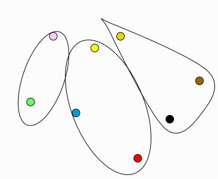
K–means

1. Initialize a set of centroids randomly
2. For each data point ***x***, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{cluster}$ is minimum



K–means

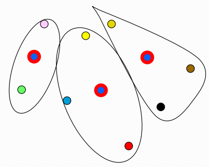
1. Initialize a set of centroids randomly
2. For each data point ***x***, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{cluster}$ is minimum



K–means

1. Initialize a set of centroids randomly
2. For each data point ***x***, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{cluster}$ is minimum
4. When all data points are clustered, recompute centroids

$$m_{cluster} = \frac{1}{\sum_{i \in cluster} w_i} \sum_{i \in cluster} w_i x_i$$

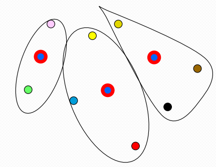


K–means

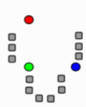
1. Initialize a set of centroids randomly
2. For each data point ***x***, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{cluster}$ is minimum
4. When all data points are clustered, recompute centroids

$$m_{cluster} = \frac{1}{\sum_{i \in cluster} w_i} \sum_{i \in cluster} w_i x_i$$

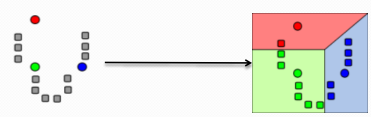
5. If not converged, go back to 2



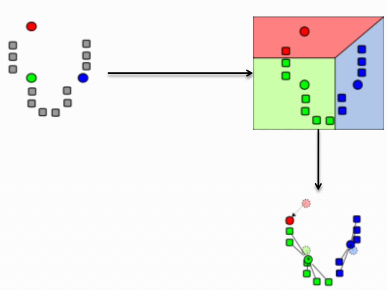
Another example



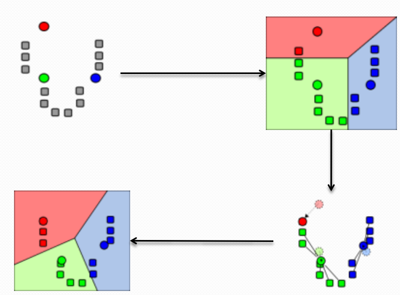
Another example



Another example



Another example



Going back to our first example

Going back to our first example

Going back to our first example

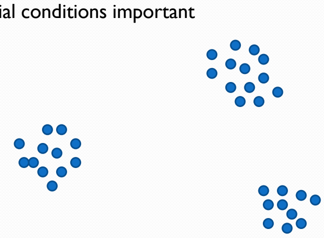
4 clusters

Going back to our first example

6 clusters

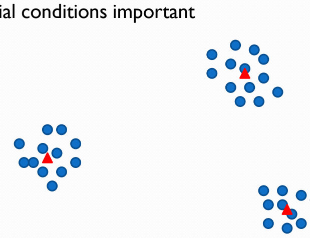
Problems with K-means

- Initial conditions important



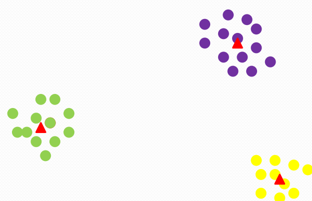
Problems with K-means

- Initial conditions important



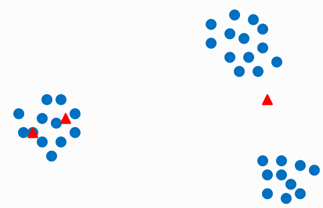
Problems with K-means

- Initial conditions important



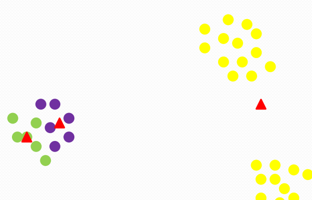
Problems with K-means

- Initial conditions important



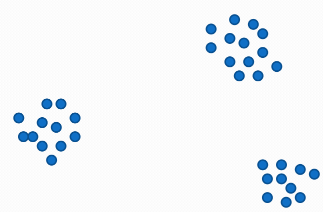
Problems with K-means

- Initial conditions important



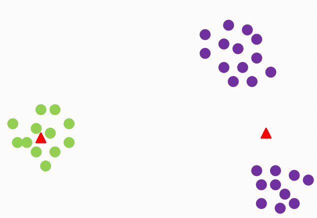
Problems with K-means

- What is K?



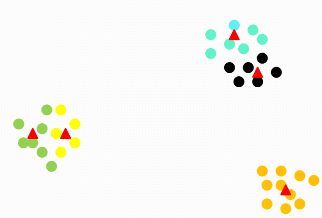
Problems with K-means

- K=2



Problems with K-means

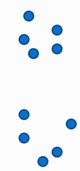
- K=5



Is there an optimal clustering method?

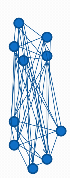
Optimal method: Exhaustive Enumeration

- Compute distances between every single pair of data points and cluster on that



Optimal method: Exhaustive Enumeration

- Compute distances between every single pair of data points and cluster on that



Optimal method: Exhaustive Enumeration

- Compute distances between every single pair of data points and cluster on that
- Very very computationally expensive
  - If M data points and we want N clusters:

$$\frac{1}{M!}\sum_{i=0}^{N}(-1)^i\binom{N}{i}(N-i)^M$$

- Compute goodness for every possible combination

Optimal method: Exhaustive Enumeration

- Compute distances between every single pair of data points and cluster on that
- Very very computationally expensive
  - If M data points and we want N clusters:

$$\frac{1}{M!}\sum_{i=0}^{N}(-1)^i\binom{N}{i}(N-i)^M$$

- Compute goodness for every possible combination

TOO SLOW !!!!

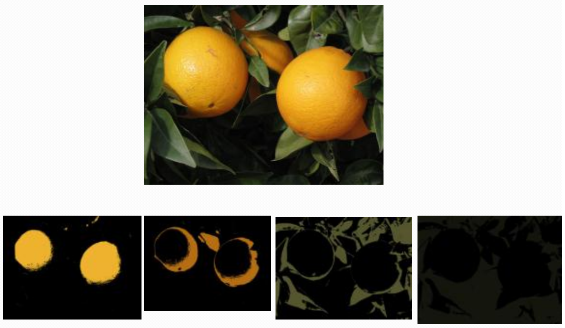
Optimal method: Exhaustive Enumeration

- Compute distances between every single pair of data points and cluster on that
- Very very computationally expensive
  - If M data points and we want N clusters:
- Compute goodness for every possible combination

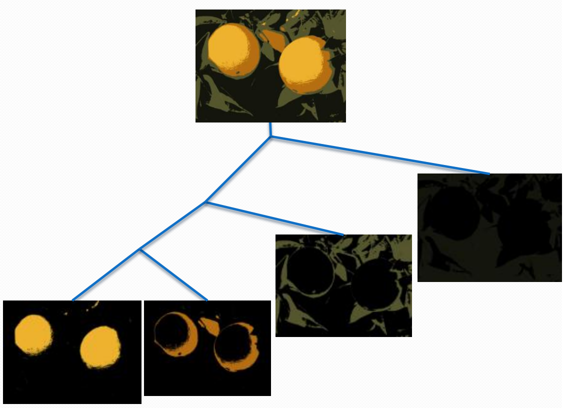
TOO SLOW !!!!

K-means: Fast but greedy

Going back to our first example



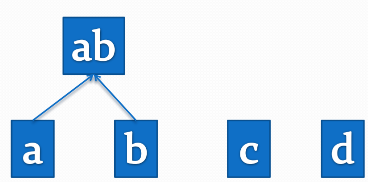
Hierarchical clustering



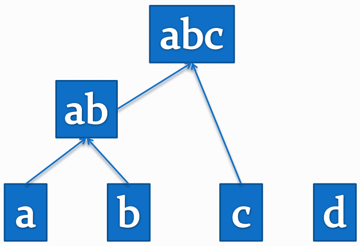
Hierarchical clustering: Bottom up

Bottom up clustering

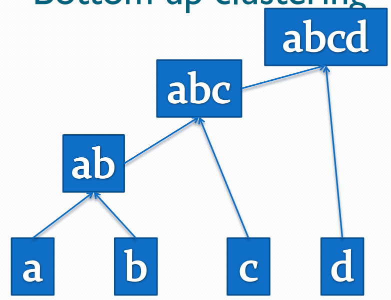
Bottom up clustering



Bottom up clustering

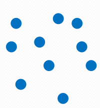


Bottom up clustering

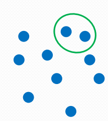


Bottom up clustering

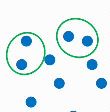
- Initially, every point is its own cluster



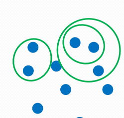
Bottom up clustering



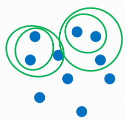
Bottom up clustering



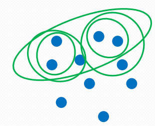
Bottom up clustering



Bottom up clustering

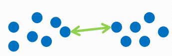


Bottom up clustering



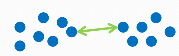
Notes about bottom up clustering

- Single Link: Nearest neighbor distance

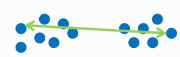


Notes about bottom up clustering

- Single Link: Nearest neighbor distance

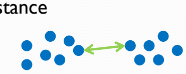


- Complete link: Farthest neighbor distance

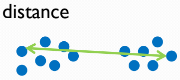


Notes about bottom up clustering

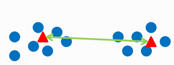
- Single Link: Nearest neighbor distance



- Complete link: Farthest neighbor distance



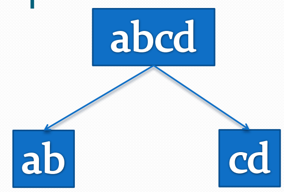
- Centroid: Distance between centroids



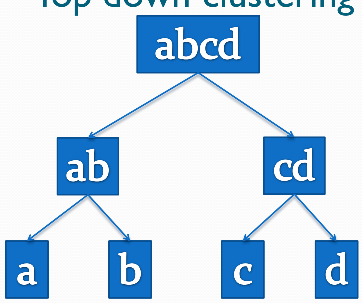
Hierarchical clustering: Top Down

Top down clustering

Top down clustering

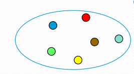


Top down clustering



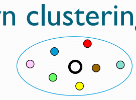
K-Means for Top–Down clustering

1. Start with one cluster



K-Means for Top–Down clustering

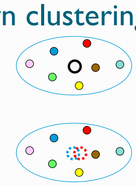
1. Start with one cluster



2. Split each cluster into two:
   - Perturb centroid of cluster slightly (by < 5%) to generate two centroids

K-Means for Top–Down clustering

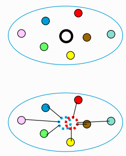
1. Start with one cluster



2. Split each cluster into two:
   - Perturb centroid of cluster slightly (by < 5%) to generate two centroids

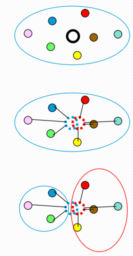
K-Means for Top–Down clustering

1. Start with one cluster
2. Split each cluster into two:
   - Perturb centroid of cluster slightly (by < 5%) to generate two centroids
3. Initialize K means with new set of centroids



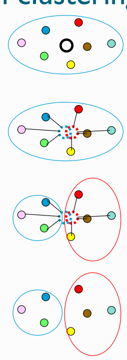
K-Means for Top–Down clustering

1. Start with one cluster
2. Split each cluster into two:
   - Perturb centroid of cluster slightly (by < 5%) to generate two centroids
3. Initialize K means with new set of centroids
4. Iterate Kmeans until convergence



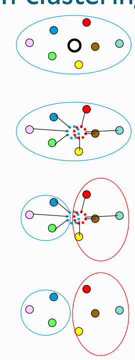
K-Means for Top–Down clustering

1. Start with one cluster
2. Split each cluster into two:
   - Perturb centroid of cluster slightly (by < 5%) to generate two centroids
3. Initialize K means with new set of centroids
4. Iterate Kmeans until convergence



K-Means for Top–Down clustering

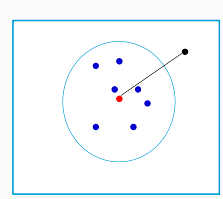
1. Start with one cluster
2. Split each cluster into two:
   - Perturb centroid of cluster slightly (by < 5%) to generate two centroids
3. Initialize K means with new set of centroids
4. Iterate Kmeans until convergence
5. If the desired number of clusters is not obtained, return to 2



When is a data point in a cluster?

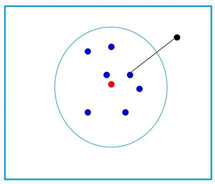
Distance from cluster

- Euclidean distance from centroid



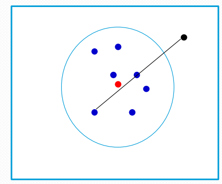
Distance from cluster

- Distance from the closest point



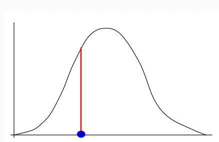
Distance from cluster

- Distance from the farthest point



Distance from cluster

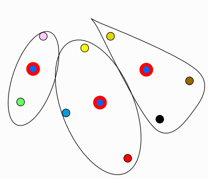
- Probability of data measured on cluster distribution



A closer look at 'Distance'

K–means

1. Initialize a set of centroids randomly
2. For each data point $\boldsymbol{x}$, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{cluster}$ is minimum
4. When all data points are clustered, recompute centroids
   $$m_{cluster} = \frac{1}{\sum_{i \in cluster} w_i} \sum_{i \in cluster} w_i x_i$$
5. If not converged, go back to 2



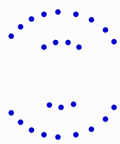
A closer look at 'Distance'

- Original algorithm uses L2 norm and weight=1

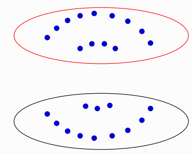
$$\mathbf{distance}_{cluster}(x, m_{cluster}) = \| x - m_{cluster} \|_2 \qquad m_{cluster} = \frac{1}{N_{cluster}} \sum_{i \in cluster} x_i$$

- This is an instance of generalized EM
- The algorithm is not guaranteed to converge for other distance metrics

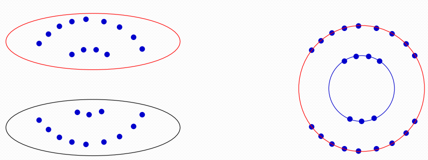
Problems with Euclidean distance



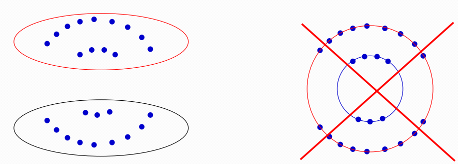
Problems with Euclidean distance



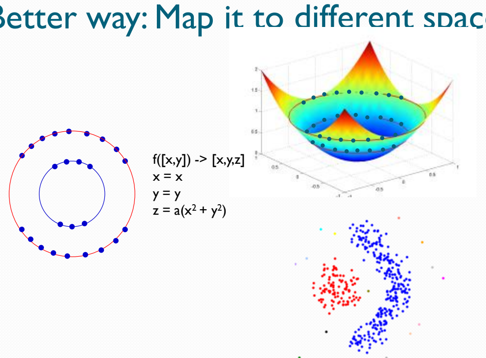
Problems with Euclidean distance



Problems with Euclidean distance



Better way: Map it to different space



The Kernel trick

The Kernel trick

- Transform data to higher dimensional space (even infinite!)
  - $\mathbf{z} = \Phi(\mathbf{x})$

The Kernel trick

- Transform data to higher dimensional space (even infinite!)
  - $\mathbf{z} = \Phi(\mathbf{x})$
- Compute distance in higher dimensional space
  - $d(\mathbf{x}_1, \mathbf{x}_2) = ||\mathbf{z}_1 - \mathbf{z}_2||^2 = ||\Phi(\mathbf{x}_1) - \Phi(\mathbf{x}_2)||^2$

The cool part

- Distance in low dimensional space:
  - $||\mathbf{x}_1 - \mathbf{x}_2||^2 = (\mathbf{x}_1 - \mathbf{x}_2)^T(\mathbf{x}_1 - \mathbf{x}_2) = \mathbf{x}_1.\mathbf{x}_1 + \mathbf{x}_2.\mathbf{x}_2 - 2\,\mathbf{x}_1.\mathbf{x}_2$

The cool part

- Distance in low dimensional space:
  - $||\mathbf{x}_1 - \mathbf{x}_2||^2 = (\mathbf{x}_1 - \mathbf{x}_2)^T(\mathbf{x}_1 - \mathbf{x}_2) = \mathbf{x}_1.\mathbf{x}_1 + \mathbf{x}_2.\mathbf{x}_2 - 2\,\mathbf{x}_1.\mathbf{x}_2$
- Distance in high dimensional space:
  - $d(\mathbf{x}_1, \mathbf{x}_2) = ||\Phi(\mathbf{x}_1) - \Phi(\mathbf{x}_2)||^2$
    $= \Phi(\mathbf{x}_1).\,\Phi(\mathbf{x}_1) + \Phi(\mathbf{x}_2).\,\Phi(\mathbf{x}_2) - 2\,\Phi(\mathbf{x}_1).\,\Phi(\mathbf{x}_2)$
- Note: Every term involves dot products!

Kernel function

- Kernel function is just
  - $K(\mathbf{x}_1,\mathbf{x}_2) = \Phi(\mathbf{x}_1).\,\Phi(\mathbf{x}_2)$
- Going back to our distance function in the high dimensional space:
  - $d(\mathbf{x}_1, \mathbf{x}_2) = ||\Phi(\mathbf{x}_1) - \Phi(\mathbf{x}_2)||^2$
    $= \Phi(\mathbf{x}_1).\,\Phi(\mathbf{x}_1) + \Phi(\mathbf{x}_2).\,\Phi(\mathbf{x}_2) - 2\,\Phi(\mathbf{x}_1).\,\Phi(\mathbf{x}_2)$
    $= K(\mathbf{x}_1,\mathbf{x}_1) + K(\mathbf{x}_2,\mathbf{x}_2) - 2K(\mathbf{x}_1,\mathbf{x}_2)$
- Kernel functions are more efficient than dot products

Typical Kernel Functions

- Linear: $K(\mathbf{x},\mathbf{y}) = \mathbf{x}^T\mathbf{y} + c$
- Polynomial $K(\mathbf{x},\mathbf{y}) = (a\mathbf{x}^T\mathbf{y} + c)^n$
- Gaussian: $K(\mathbf{x},\mathbf{y}) = \exp(-||\mathbf{x}-\mathbf{y}||^2/\sigma^2)$
- Exponential: $K(\mathbf{x},\mathbf{y}) = \exp(-||\mathbf{x}-\mathbf{y}||/\lambda)$
- Several others
  - Choosing the right Kernel with the right parameters for your problem is an art

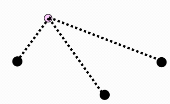
Kernel K-means

Kernel K–means

1. Initialize a set of centroids randomly

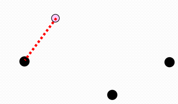
Kernel K–means

1. Initialize a set of centroids randomly
2. For each data point **x**, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$



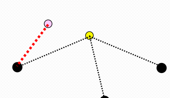
Kernel K–means

1. Initialize a set of centroids randomly
2. For each data point **x**, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{\text{cluster}}$ is minimum



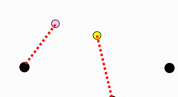
Kernel K–means

1. Initialize a set of centroids randomly
2. For each data point **x**, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{\text{cluster}}$ is minimum



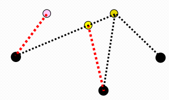
Kernel K–means

1. Initialize a set of centroids randomly
2. For each data point **x**, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{\text{cluster}}$ is minimum



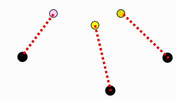
Kernel K–means

1. Initialize a set of centroids randomly
2. For each data point **x**, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{cluster}$ is minimum



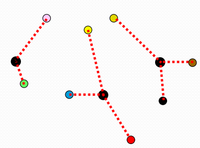
Kernel K–means

1. Initialize a set of centroids randomly
2. For each data point **x**, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{cluster}$ is minimum



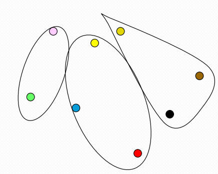
Kernel K–means

1. Initialize a set of centroids randomly
2. For each data point **x**, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{cluster}$ is minimum



Kernel K–means

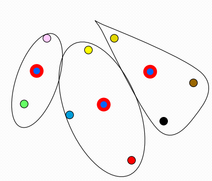
1. Initialize a set of centroids randomly
2. For each data point **x**, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{cluster}$ is minimum



Kernel K–means

1. Initialize a set of centroids randomly
2. For each data point **x**, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{cluster}$ is minimum
4. When all data points are clustered, recompute centroids

$$m_{cluster} = \frac{1}{\sum_{i \in cluster} w_i} \sum_{i \in cluster} w_i x_i$$

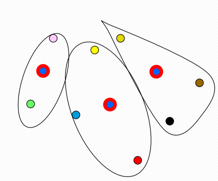


Kernel K–means

1. Initialize a set of centroids randomly
2. For each data point **x**, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{cluster}$ is minimum
4. When all data points are clustered, recompute centroids

$$m_{cluster} = \frac{1}{\sum_{i \in cluster} w_i} \sum_{i \in cluster} w_i x_i$$

5. If not converged, go back to 2

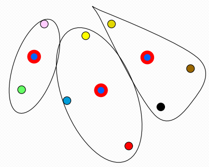


Kernel K–means

1. Initialize a set of centroids randomly
2. For each data point $\boldsymbol{x}$, find the distance from the centroid for each cluster
   - $d_{cluster} = \textbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{cluster}$ is minimum
4. When all data points are clustered, recompute centroids

$$m_{cluster} = \frac{1}{\sum_{i \in cluster} w_i} \sum_{i \in cluster} w_i x_i$$

5. If not converged, go back to 2



Distance metric

$$d(x, cluster) = \| \Phi(x) - m_{cluster} \|^2 = \left(\Phi(x) - C \sum_{i \in cluster} w_i \Phi(x_i) \right)^T \left(\Phi(x) - C \sum_{i \in cluster} w_i \Phi(x_i) \right)$$

Distance metric

$$d(x, cluster) = \| \Phi(x) - m_{cluster} \|^2 = \left(\Phi(x) - C \sum_{i \in cluster} w_i \Phi(x_i) \right)^T \left(\Phi(x) - C \sum_{i \in cluster} w_i \Phi(x_i) \right)$$

$$= \left(\Phi(x)^T \Phi(x) - 2C \sum_{i \in cluster} w_i \Phi(x)^T \Phi(x_i) + C^2 \sum_{i \in cluster} \sum_{j \in cluster} w_i w_j \Phi(x_i)^T \Phi(x_j) \right)$$

Distance metric

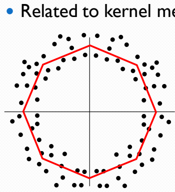
$$d(x, cluster) = \| \Phi(x) - m_{cluster} \|^2 = \left(\Phi(x) - C \sum_{i \in cluster} w_i \Phi(x_i) \right)^T \left(\Phi(x) - C \sum_{i \in cluster} w_i \Phi(x_i) \right)$$

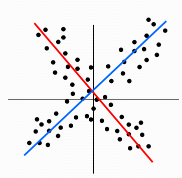
$$= \left(\Phi(x)^T \Phi(x) - 2C \sum_{i \in cluster} w_i \Phi(x)^T \Phi(x_i) + C^2 \sum_{i \in cluster} \sum_{j \in cluster} w_i w_j \Phi(x_i)^T \Phi(x_j) \right)$$

$$= K(x,x) - 2C \sum_{i \in cluster} w_i K(x, x_i) + C^2 \sum_{i \in cluster} \sum_{j \in cluster} w_i w_j K(x_i, x_j)$$

Other clustering methods

- Regression based clustering
- Find a regression representing each cluster
- Associate each point to the cluster with the best regression
  - Related to kernel methods





Questions?