Unsupervised Learning

- What’s unknown?
  - In the training set, which class does each sample belong to?
  - For the problem in general, how many classes (clusters) is appropriate?
### Clustering Algorithm

- **Agglomerative clustering**
  - **Step1**: assign each data point in the training set to a separate cluster
  - **Step2**: merge the two "closest" clusters
  - **Step3**: repeat step2 until you get the number of clusters you want or the appropriate cluster number

- The result is highly dependent on the measure of cluster distance

### Distance from a Point to a Cluster

- **Euclidean distance**  \(d_{eucl}(x, A) = \|x - \mu_s\|\)
- **City block distance**
- **Squared Mahalanobis distance**  
  \[d_{mah}(x, A) = (x - \mu_s)^T \Sigma_s^{-1} (x - \mu_s)\]

### Distance between Clusters

- **The centroid distance**  
  \[d_{mean}(A, B) = \|\mu_A - \mu_B\|\]
- **Nearest neighbor measure**  
  \[d_{min}(A, B) = \min_{a \in A} \min_{b \in B} d_{eucl}(a, b)\]
- **Furthest neighbor measure**  
  \[d_{max}(A, B) = \max_{a \in A} \max_{b \in B} d_{eucl}(a, b)\]
Example

\[ \begin{array}{c}
\text{dmin} \\
A \quad B \quad C
\end{array} \quad \begin{array}{c}
\text{dmax} \\
A \quad B \quad C
\end{array} \]

Minimum Spanning Tree

- Step 1: compute all edges in the graph
- Step 2: sort the edges by length
- Step 3: beginning with the shortest edge, for each edge between nodes \( u \) and \( v \), perform the following operations:
  - Step 3.1: \( A = \text{find}(u) \) (\( A \) is the cluster where \( u \) is in)
  - Step 3.2: \( B = \text{find}(v) \) (\( B \) is the cluster where \( v \) is in)
  - Step 3.3: if \( A \neq B \) \( C = \text{union}(A, B) \), and erase sets \( A \) and \( B \)

Comparison of Shape of Clusters

- \( \text{dmin} \) tends to choose clusters which are ??
- \( \text{dmax} \) tends to choose clusters which are ??
The k-means Algorithm

Step 1: Begin with an arbitrary assignment of samples to clusters or begin with an arbitrary set of cluster centers and assign samples to nearest clusters

Step 2: Compute the sample mean of each cluster

Step 3: Reassign each sample to the cluster with the nearest mean

Step 4: If the classification of all samples has not changed, stop; else go to step 2.

Winner-take-all Approach

- Begin with an arbitrary set of cluster centers \( \omega_i \).
- For each sample \( x_i \), find the nearest cluster center \( \omega_{i,j} \), which is called the winner.
- Modify \( \omega_{i,j} \) using \( \omega_{i,j}^{new} = \omega_{i,j}^{old} + \varepsilon (x_i - \omega_{i,j}^{old}) \)
  - \( \varepsilon \) is known as a "learning parameter".
  - Typical values of this parameter are small, on the order of 0.01.
Winner-take-all

Kohonen Feature Maps (NN)

- An extension of the winner-take-all algorithm. Also called self-organizing feature maps.
- A problem-dependent topological distance is assumed to exist between each pair of the cluster centers.
- When the winning cluster center is updated, so are its neighbors in the sense of this topological distance.

SOM – A Demo
**SOM – The Algorithm**

The winning cluster center and its neighbors are trained based on the following formula:

\[ \omega_i^{k+1} = \omega_i^k + \epsilon(k) \Phi(k) (x - \omega_i^k) \]

- \( \omega_i \) are the cluster centers
- \( g_n \) are the coordinate of the cluster centers
- \( x_{winner} \) is the coordinate of the winner
- \( \epsilon(k) \) is the learning rate
- \( \Phi(k) = \exp \left( - \frac{||x - x_{winner}||^2}{2\sigma^2} \right) \)

Learning Rate (as \( k \) inc, \( \epsilon \) dec, more stable)

**Mean Shift Clustering**

- Originally proposed in 1975 by Fukunaga and Hostetler for mode detection
- Cheng's generalization to solve clustering problems in 1995
- Non-parametric – no prior knowledge needed about number of clusters
- Key parameter: window size
- Challenging issue:
  - How to determine the right window size?
  - Slow convergence

Mean Shift Clustering (as \( k \) inc, \( \epsilon \) dec, more stable)

**SOM - An Example**

[Image of SOM example]

[Link to SOM example](http://www.ai-junkie.com/ann/som/som1.html)
Mean Shift Clustering

1. Initialization: Choose a window/kernel of size \( h \), e.g., a flat kernel, and apply the window on each data point, \( x \)
   
   \[
   K(x) = \begin{cases} 
   1 & \text{if } ||x|| \leq h \\
   0 & \text{if } ||x|| > h 
   \end{cases}
   \]

2. Mean calculation: Within each window centered at \( x \), compute the mean of data, where \( \Omega_k \) is the set of points enclosed within window \( h \)
   
   \[
   m(x) = \frac{\sum_{x \in \Omega_k} K(x - x) s}{\sum_{x \in \Omega_k} K(x - x)}
   \]

3. Mean shift: Shift the window to the mean, i.e., \( x = m(x) \), where the difference \( m(x) - x \) is referred to as the mean shift.

4. If \( ||m(x) - x|| > \epsilon \), go back to step 2.

Figure 8.14: Mean Shift Clustering

- Original
- RGB plot
- \( k=10, 24, 64 \) (kmeans)
- Init (random | uniform), \( k=6 \)
- Mean shift (h=30 \( \rightarrow \) 14 clusters)
- Mean shift (h=20 \( \rightarrow \) 35 clusters)