CS 2750 Machine Learning
Lecture 17a

Clustering

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Clustering

Groups together "similar" instances in the data sample

Basic clustering problem:

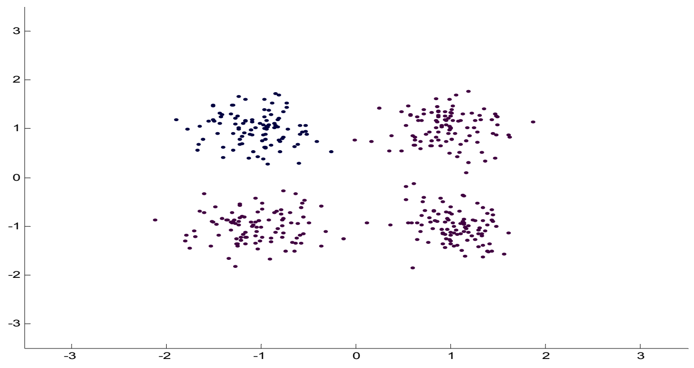
- distribute data into *k* different groups such that data points similar to each other are in the same group
- Similarity between data points is defined in terms of some distance metric (can be chosen)

Clustering is useful for:

- **Similarity/Dissimilarity analysis**
  Analyze what data points in the sample are close to each other
- **Dimensionality reduction**
  High dimensional data replaced with a group (cluster) label

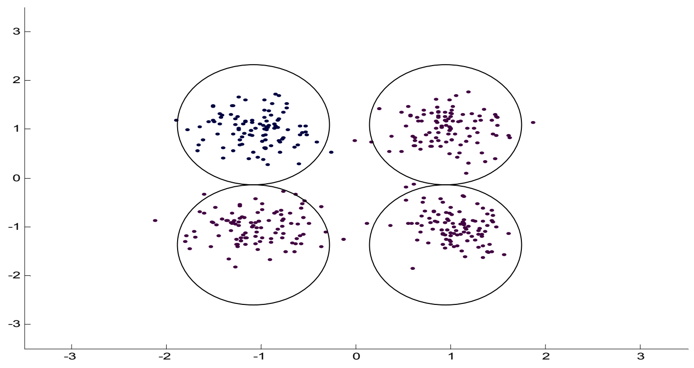
Clustering example

- We see data points and want to partition them into the groups
- Note that this is a problem different from density estimation !!!



Clustering example

- We see data points and want to partition them into the groups
- Points close to each other (e.g. in terms of Euclidean distance) in the same group



Clustering example

- A set of patient cases
- We want to partition them into the groups based on similarities

Patient #	Age	Sex	Heart Rate	Blood pressure …
Patient 1	55	M	85	125/80
Patient 2	62	M	87	130/85
Patient 3	67	F	80	126/86
Patient 4	65	F	90	130/90
Patient 5	70	M	84	135/85

Clustering algorithms

Partitioning algorithms:

- **K-means algorithm**
  - **suitable** only when data points have continuous values; groups are defined in terms of cluster centers (also called **means**).
  - refinement of the method to categorical values: **K-medoids**
- **Probabilistic methods (with EM)**
  - **Latent variable models**: class (cluster) is represented by a latent (hidden) variable value.
  - **Examples:** mixture of Gaussians, Naïve Bayes with a hidden class
- **Hierarchical methods**
  - **Agglomerative**
  - **Divisive**

K-means

K-Means algorithm:

Initialize randomly k values of means (centers)

Repeat two steps until no change in the means:

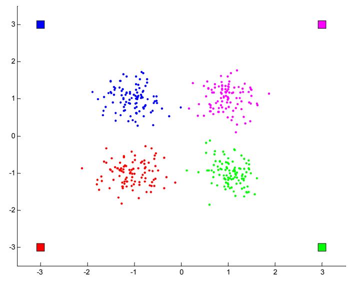
- Partition the data according to the current set of means (using the similarity measure)
- Move the means to the center of the data in the current partition

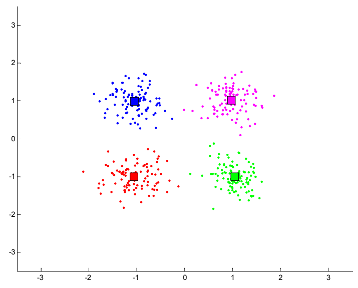
Stop when no change in the means

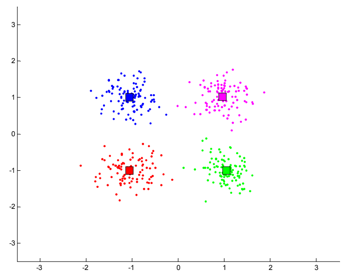
Properties:

- Minimizes the sum of squared center-point distances for all clusters
- The algorithm always converges (local optima).

K-Means example







K-means algorithm

- **Properties:**
  - converges to centers minimizing the sum of squared center-point distances (still local optima)
  - The result is sensitive to the initial means' values
- **Advantages:**
  - Simplicity
  - Generality – can work for more than one distance measure
- **Drawbacks:**
  - Can perform poorly with overlapping regions
  - Lack of robustness to outliers
  - Good for attributes (features) with continuous values
    - Allows us to compute cluster means

Probabilistic (EM-based) algorithms

- **Latent variable models**

  Examples: Naïve Bayes with hidden class

  Mixture of Gaussians
- **Partitioning:**
  - the data point belongs to the class with the highest posterior
- **Advantages:**
  - Good performance on overlapping regions
  - Robustness to outliers
  - Data attributes can have different types of values
- **Drawbacks:**
  - EM is computationally expensive and can take time to converge
  - Density model should be given in advance

Hierarchical clustering.

Uses an arbitrary similarity/dissimilarity measure.

Typical similarity measures *d(a,b)* :

Pure real-valued data-points:

– Euclidean, Manhattan, Minkowski distances

Pure binary values data:

– Number of matching values

Pure categorical data:

– Number of matching values

Combination of real-valued and categorical attributes

– A weighted sum approach

Hierarchical clustering.

Approach:

- **Compute dissimilarity matrix for all pairs of points**
  - uses standard or other distance measures
- **Construct clusters greedily:**
  - **Agglomerative approach**
    - Merge pair of clusters in a bottom-up fashion, starting from singleton clusters
  - **Divisive approach:**
    - Splits clusters in top-down fashion, starting from one complete cluster
- **Stop the greedy construction** when some criterion is satisfied
  - E.g. fixed number of clusters

Cluster merging

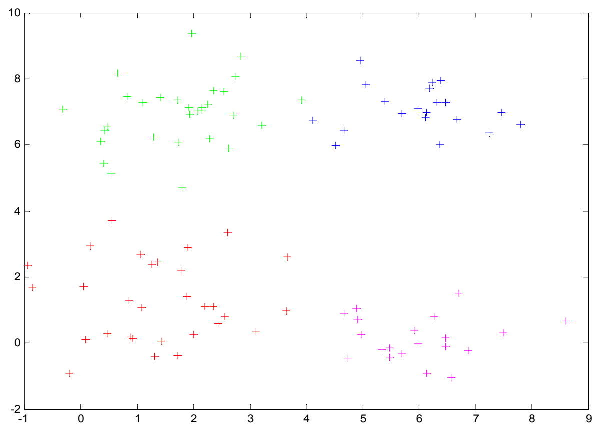
- **Construction of clusters through greedy agglomerative approach**
  - Merge pair of clusters in a bottom-up fashion, starting from singleton clusters
  - Merge clusters based on cluster distances. Defined in terms of point distances. **Examples:**

Min distance $$d_{\min}(C_i, C_j) = \min_{p \in C_i, q \in C_j} |p - q|$$

Max distance $$d_{\max}(C_i, C_j) = \max_{p \in C_i, q \in C_j} |p - q|$$

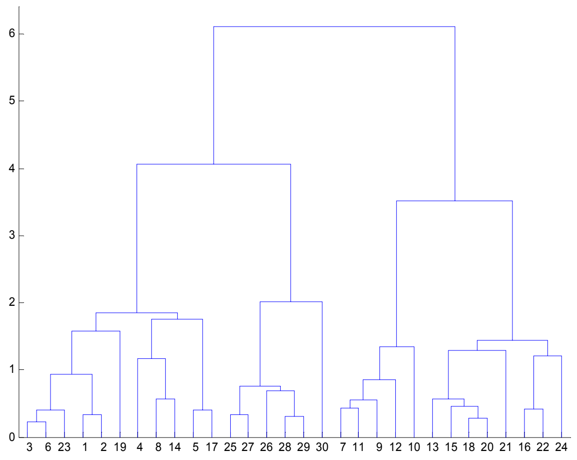
Mean distance $$d_{mean}(C_i, C_j) = \left| \frac{1}{|C_i|} \sum_i p_i - \frac{1}{|C_j|} \sum_j q_j \right|$$

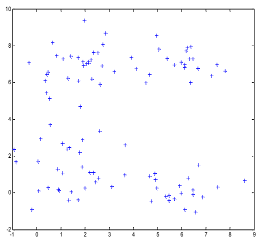
Hierarchical clustering example



Hierarchical clustering example

- **dendogram**





Hierarchical clustering

- **Advantage:**
  - Smaller computational cost; avoids scanning all possible clusterings
- **Disadvantage:**
  - Greedy choice fixes the order in which clusters are merged; cannot be repaired
  - **Partial solution:** combine hierarchical clustering with iterative algorithms like k-means

CS 2750 Machine Learning
Lecture 17b

Non-parametric density estimation

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Density estimation

- **Parametric density estimation method**
  - the form of the density is known
  - We need to estimate a fixed set of parameters
  - **Examples:** Gaussian distribution, Exponential distribution

- **Non-parametric density estimation**
  - The form of the density is not known
  - All examples are used in the estimate
    - every example acts as a parameter
  - The representation grows with the number of examples *N*
  - **Examples:** histogram, k-nearest neighbor

Density estimations

- **Semi-parametric methods**
  - A compromise between **parametric** and **non-parametric** techniques
  - The form of the density is restricted but there is some flexibility
  - Representation (parameters) does not grow with the number of examples in the data N
  - **Example:** Mixture of Gaussians with k mixtures

- **Already covered:** parametric and semi-parametric
- Next focus: **non-parametric methods**

Non-parametric density estimation

- **Data:** N samples from underlying distribution of $\mathbf{x}$
- **Objective: estimate of** p(**x**)

- Let R be a region (subspace) of the space of $\mathbf{x}$
- The probability of a point $\mathbf{x}$ in the region R can be estimated as:

$$p(\mathbf{x}) = \frac{1}{V}\frac{K}{N}$$

$$p(\mathbf{x}) = \frac{1}{\text{volume of the region}} \times \frac{\text{\# of examples in the same region } R \text{ as } x}{\text{\# of examples}}$$

- The density condition $\int p(x)\,dx = 1$

Non-parametric density estimation

- **We want an estimate:**

$$p(\mathbf{x}) = \frac{1}{V}\frac{K}{N}$$

- **Two options we have here:**
  - Fix V around **x** and count the number of examples in the data falling into the volume

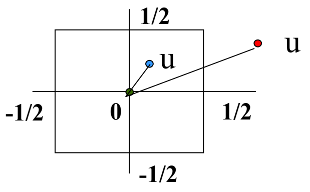
    Examples: Parzen windows, kernel regression
  - Fix K and compute the volume around it

    Example: K-nearest neighbors

Parzen windows: Naïve estimator

- Assume that a region is defined using a d-dimensional hypercube with h_n being the length of its edge
- We can define a **window function**:

$$w(\mathbf{u}) = \begin{cases} 1 & \text{if } \forall\ j = 1,\ldots d \ \ |u_j| \le 1/2 \\ 0 & \text{otherwise} \end{cases}$$



- Then the probability of x can be estimated as:

$$p(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{V_N} w\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_N}\right)$$

Parzen windows: Naïve estimator

- Then the probability of x can be estimated as:

$$p(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{V_N}w\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_N}\right)$$

- For $N \to \infty,\ h_N \to 0$, $p(\mathbf{x})$ converges to the correct value

$$p(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{V_N}w\left(\frac{x-x_i}{h_N}\right)$$

$$\overset{N\to\infty}{\to} \frac{1}{V_N}P(x_j - h_N/2 < x_j < x_j + h_N/2\ ;\forall\ j) \quad \overset{h_N\to 0}{\to} \ p(\mathbf{x})$$

Parzen windows: Kernel regression

- **Disadvantage** of the naïve window function:
  – Density function estimate exhibits discontinuities
- **Remedy:**
  – Instead of a naïve window function use a smooth switching on the boundary
  – Use symmetrical distribution, e.g. Gaussian, and wrap it around every training example
  – Compute a kind of similarity distance between point u and a point in the training set

$$p(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N}K(\mathbf{x},\mathbf{x}_i)$$

$K(\mathbf{x},\mathbf{x}')$ - kernel function

- Method: **kernel regression**

Parzen windows: Gaussian kernel

- **Probability estimate through kernel regression**
  - Gaussian kernels

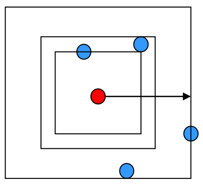
$$p(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} K(\mathbf{x}, \mathbf{x}_i)$$

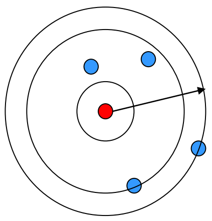
$$= \frac{1}{N}\sum_{i=1}^{N} \frac{1}{(2\pi)^{d/2}\left|\sigma^2\mathbf{I}\right|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mathbf{x}_i)^T \sigma^2\mathbf{I}(\mathbf{x}-\mathbf{x}_i)\right]$$

- **Advantage:** much smoother density estimate as compared to the naïve Parzen window approach
- Other possible symmetrical kernels: Epanechnikov kernel

K-nearest neighbor

- The **problem** with the Parzen window approach
  - How to choose the size of the window?
- **Idea:**
  - Make the size of the window (region) vary based on the data in the neighborhood of $\boldsymbol{x}$
  - Grow the window till k nearest neighbors are captured





$$p(\mathbf{x}) = \frac{1}{N}\frac{K}{V_{k,x}}$$ $V_{k,x}$ - changes from point to point