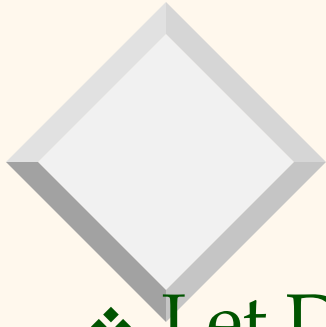


Clustering: Learning Goals

1. Unsupervised vs supervised learning
2. Partitioning vs hierarchical methods
3. Distance functions
4. K-means algorithm
5. K-medoids algorithm
 - (Russell, xxx; Hastie, Ch 14.3)



Unsupervised Learning

- ❖ Let D be our dataset consisting of data objects $\{x_1, \dots, x_d\}$
- ❖ *Clustering* seeks to answer the question:
What are sub-classes of objects in the dataset?
 - i.e., Identify classes and put class labels on data objects
- ❖ *Classification* answers the next question:
How are the sub-classes different from each other?
 - i.e., Identify characteristics that discriminate objects with different labels
- ❖ The latter is called *supervised* learning, requiring prior labeling of data



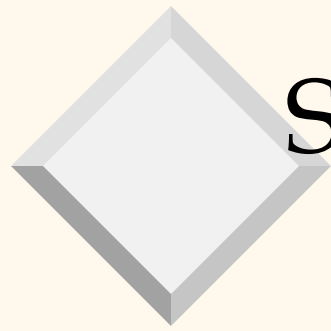
Everyday Examples

- ❖ E.g., shirt manufacturing
 - Armani has the data on tailored dress shirts of thousands of customers, e.g., neck, sleeve, chest, etc.
 - Now wants to create five standard-sized shirts to sell: {XS, S, M, L, XL}
 - What should the dimensions be for these sizes?
- ❖ E.g., how many classes of Facebook users? And what are those?
- ❖ E.g., how many classes of smart phone customers?
- ❖ E.g., how many classes of hockey goalies are there?



Partitioning Methods

- ❖ In the Armani shirt example, the number of classes to be created is given
- ❖ For other applications, we may want to identify a “natural” number of classes
- ❖ The **Partitioning** problem:
 - Given a positive integer m , put all the data objects into m clusters/groups, so that “similar” objects are in the same cluster and “dissimilar” objects are in different cluster



Simple 2D Example

A(0,0) ; B(4,0); C(4,3); D(6,3); E(6,5); F(10,6);
G(11,7); H(9,8)

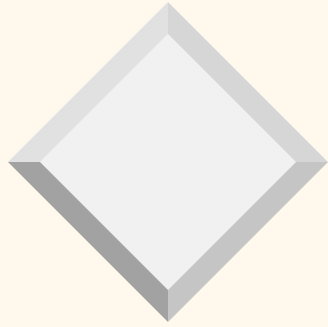
Into 3 groups

- ❖ Note that every object is assigned to exactly 1 cluster: no outliers, no overlapping clusters



Hierarchical Clustering Methods

- ❖ Instead of forcing m , the number of clusters, to be pre-determined, **hierarchical clustering** methods find the best $d-1, d-2, \dots, 1$ clusters
- ❖ Start with every object in its own cluster, and continue to find the best two clusters to merge
- ❖ All clusters are naturally represented as trees
- ❖ Pros: for some applications, it is hard to determine m
- ❖ Cons: “complete” clustering, expensive for large datasets



Tree for Simple 2D Example



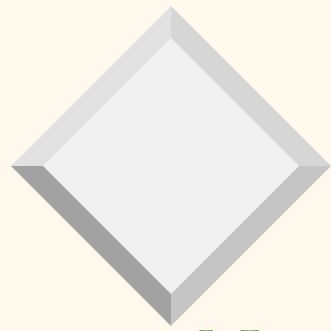
E.g., Food Group Clustering

- ❖ How are “similar” and “dissimilar” really defined?
- ❖ How do we handle different attributes/ dimensions that have drastically different scales?



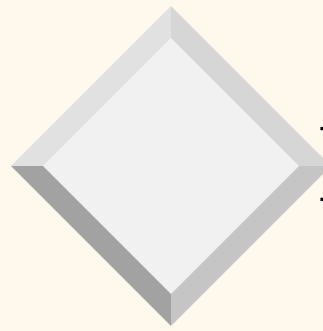
Dissimilarity or Distance Functions

- ❖ Intuition: clustering aims to put “similar” objects into one cluster, but “dissimilar” objects in different clusters
- ❖ We need to define how much every pair of objects are similar or dissimilar to each other
- ❖ Distance function for d data objects can be captured in a $d \times d$ matrix of non-negative values



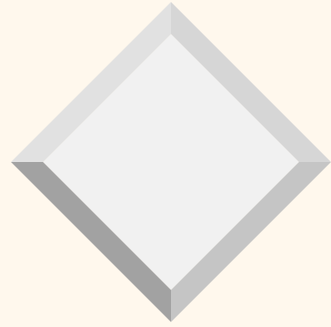
Lp-norm Distances

- ❖ Notation: each data object x_i be of k dimensions or attributes, denoted as $x_{i,1}, \dots, x_{i,k}$
- ❖ Euclidean distance btw x_i and x_j :
$$df(x_i, x_j) = [\sum_{w=1}^k (x_{i,w} - x_{j,w})^2]^{1/2}$$
- ❖ Manhattan distance: $\sum_{w=1}^k |x_{i,w} - x_{j,w}|$
- ❖ General Lp-norm distance:
$$[\sum_{w=1}^k |x_{i,w} - x_{j,w}|^p]^{1/p}$$
- ❖ What is L^∞ - norm?



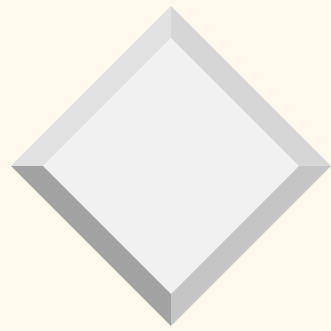
Metric Distances

- ❖ What properties are generally desirable for a distance function?
 1. $df(x_i, x_i) = 0$
 2. [Symmetry] $df(x_i, x_j) = df(x_j, x_i)$
 3. [Triangle Inequality] for any x_i, x_j, x_p
$$df(x_i, x_p) \leq df(x_i, x_j) + df(x_j, x_p)$$
- ❖ Distance functions that satisfy all the 3 conditions are called **metric** distances



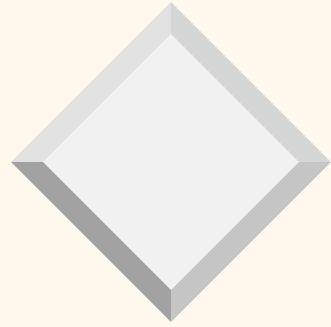
Metric Distances Example

- ❖ Exercise: show that Euclidean distance is metric
- ❖ In a proof, need to show
 - $df(x_i, x_i) = 0$ (typically the easiest)
 - Symmetry (typically not hard)
 - Triangle inequality (typically the hardest to show)





Clicker Question

- ❖ Is L_p -norm distance metric?
 - a) No, it is not symmetric, even though it satisfies the triangle inequality.
 - b) No, it does not satisfy the triangle inequality, even though it is symmetric.
 - c) No, it is not symmetric and does not satisfy the triangle inequality.
 - d) Yes.



Metric Distances Examples (2)

- ❖ Does non-symmetry distance happen in real-life?
- ❖ Does triangle inequality violation happen in real-life?
- ❖ Exercise: is relative entropy (KL divergence) metric: 
$$df(x_i, x_j) = \sum_{w=1}^k (x_{i,w} \log(x_{i,w} / x_{j,w}))$$
 ?
- ❖ What about Jensen-Shannon divergence, Bhattacharyya coefficient, or Hellinger distance?



Were attributes “created equally”?

- ❖ What do we assume so far in distance functions?
- ❖ Different attributes/dimensions could have drastically different scales
 - E.g., in our Nutrients dataset, energy in kcal, fats in grams, minerals in milligrams
 - One solution is **standardization** per attribute by replacing $x_{i,w}$ with $(x_{i,w} - \mu_w) / \sigma_w$, where μ_w and σ_w are the mean and standard deviation of the w -th attribute



Quadratic Form Distances

- ❖ What about the scenario that attribute B is more similar to attribute C than to attribute E?
- ❖ Now talking about *similarity of attributes*, adding on top of *similarity of objects*
- ❖ Rewrite Euclidean distance as follows in vector form: $df(x_i, x_j) = [(x_i - x_j)^T A (x_i - x_j)]^{1/2}$
 - A is an identity matrix of $k \times k$;
 - x_i is a vector $[x_{i,1}, \dots, x_{i,k}]^T$



Quadratic Form Distances (2)

- ❖ If A is not an identity matrix, we can use A to capture the similarity of attributes:
 - A_{ij} denotes the similarity between the i -th and the j -th attributes
 - A needs to be positive semi-definite, so that distances are non-negative
 - A usually symmetric
- ❖ E.g., x 's are images and the A captures the distances of colors



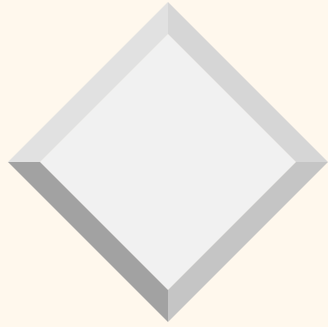
Quadratic Form Distances (3)

- ❖ One important special case is Mahalanobis distance, where A is the inverse of the covariance matrix
- ❖ Exercise: show that Mahalanobis distance is metric
- ❖ Mini-summary: clustering your data always produces something; whether clustering is effective depends *critically* on the choice of the distance function



K-Means Clustering Algorithm

1. Start with a random set of m clusters
2. Compute for each cluster, a **representative** which is the mean of all the members in the cluster
3. Re-assign each object to the closest cluster as represented by the distance btw the object and the m representatives
4. Stop if no object changes clusters; otherwise repeat steps 2 and 3



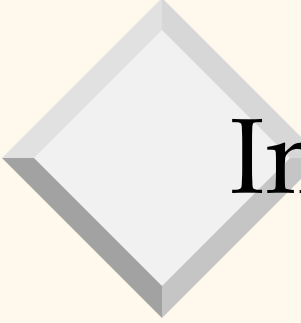
Simple 2D Example

- ❖ Note that efficiency depends on how good the initial random clusters are



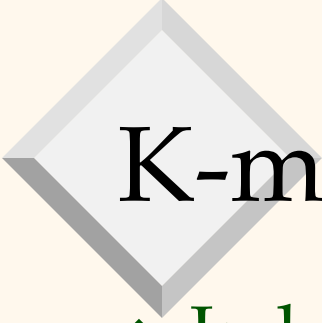
Quality of K-means Clusters

- ❖ A global optimum exists as defined by the smallest total distance btw each data object and its cluster representative
- ❖ Depending on the initial clusters, the final k-means clusters may not be globally optimal – but only locally optimal
- ❖ Local optima can vary greatly in quality
- ❖ Thus, it is often recommended that multiple runs of k-means are used to pick the best set of clusters



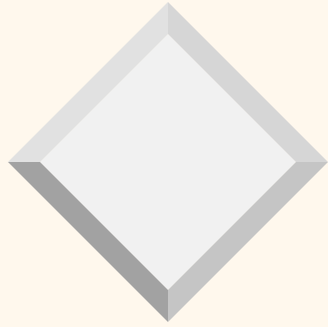
Impact of Outliers

- ❖ What is the difference between the mean and the median?
- ❖ K-means can be severely impacted by the presence of outliers
- ❖ The K-medoids algorithm can be seen as the robust version of k-means, i.e., much less affected by outliers

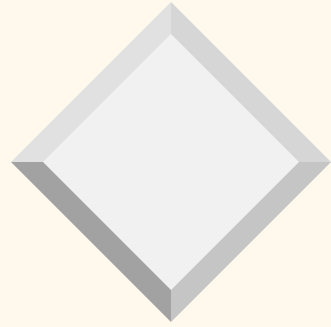


K-medoids Algorithm

- ❖ In k-means, the representative of a cluster is the mean of the members; it is not necessarily a true data object
- ❖ In k-medoids, the cluster representative is the most centrally located member of the cluster, i.e., minimum total pairwise distance from every other member in the same cluster
- ❖ Algorithmic structure basically the same as k-means but there is no new distance calculations as a medoid is a true data object



Simple 2D Example



Clicker Question

- ❖ Does K-medoids algorithm guarantee optimality?
 - a) Yes, it guarantees to find the global optimum.
 - b) Like K-means, it only guarantees a local optimum.
 - c) No, it is worse than K-means, as it does not even guarantee a local optimum.



Concluding Remarks

- ❖ Big Data requires much more scalable clustering methods than K-means/medoids
- ❖ They do exist after 20+ years of R&D
- ❖ Clustering is quite parallelizable and thus amendable to cloud computing
- ❖ Algorithms and hardware aside, a critical question is always whether an appropriate distance function is chosen