Clustering

Muchang Bahng

Spring 2025

Contents

Clustering and dimensionality reduction is used for many purposes, such as preprocessing data, visualizing it, or encoding it in a sparser, more efficient way.

1 K Means Clustering

The simplest type of unsupervised learning is **clustering**. In the clustering problem, we are given a training set of *unlabeled* data

$$\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(n)}\} \tag{1}$$

and want to group the data into a few cohesive "clusters."

1. Determine the number of clusters that we want to find and set it as k (this can be a disadvantage if we do not know how many clusters we are looking for beforehand).
2. We initialize the **cluster centroids** $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_k \in \mathbb{R}^d$ randomly or by some other method.
3. The next part takes the centroids and moves them to the center of each cluster accordingly. The following two steps are repeated until convergence (and convergence is guaranteed):
   (a) We assign each training sample $\mathbf{x}^{(i)}$ to the closest cluster centroid $\boldsymbol{\mu}_j$. That is, for every $i = 1, \dots, n$, set
   $$c^{(i)} \equiv \arg\min_j ||\mathbf{x}^{(i)} - \boldsymbol{\mu}_j||^2 \tag{2}$$
   where this argmin function returns the input to a function that yields the minimum (in this case, the number j that yields the minimum value of $||\mathbf{x}^{(i)} - \boldsymbol{\mu}_j||^2$ for each i).
   (b) We move each training cluster centroid $\boldsymbol{\mu}_j$ to the mean of the points assigned to it. That is, for each $j = 1, \dots, k$, set
   $$\boldsymbol{\mu}_j \equiv \frac{\sum_{i=1}^n 1\{c^{(i)} = j\}\, \mathbf{x}^{(i)}}{\sum_{i=1}^n 1\{c^{(i)} = j\}} \tag{3}$$

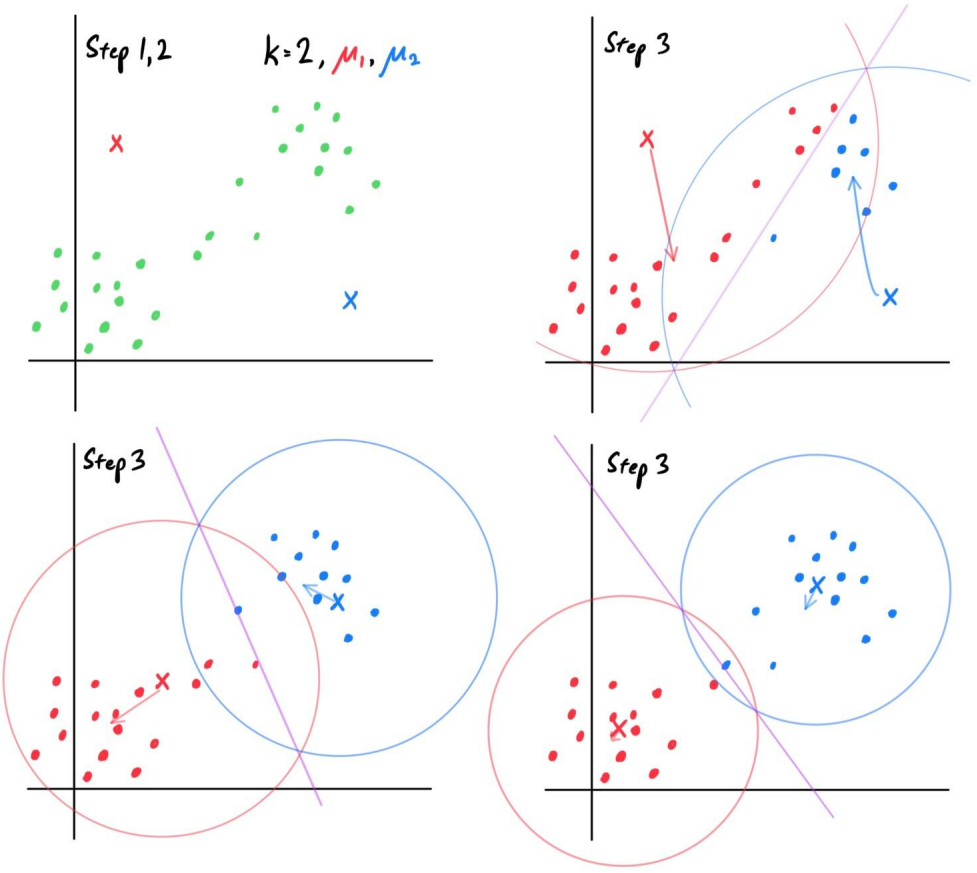

Figure 1: The steps can be visualized for a set of unlabeled data (green points) in $\mathbb{R}^2$ clustered into $k = 2$ groups (red and blue). The crosses represent the cluster centroids.

We can interpret this algorithm in another equivalent way. k-means is precisely coordinate descent on the cost function called the **distortion function**:

$$L(\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k) \equiv \sum_{i=1}^{n} \min_k ||\mathbf{x}^{(i)} - \boldsymbol{\mu}_k||^2 \tag{4}$$

but since L is not necessarily convex, it might be susceptible to local extrema.

2 Kernel Density Estimation

3 Multi-Dimensional Scaling

Again, we want to reduce our dimension, but the goal is slightly different from PCA.

Definition 3.1 (Multi-Dimensional Scaling)

Given our data $X \in \mathbb{R}^d$, we want to construct a linear map $T : \mathbb{R}^d \to \mathbb{R}^k$ such that it preserves the pairwise differences between the data points. That is, we want to minimize the following loss function

$$\min_T \sum_{i \neq j} \left(d_{\mathbb{R}^k}(T(x_i), T(x_j)) - d_{\mathbb{R}^d}(x_i, x_j) \right) \tag{5}$$

where d_V is a distance metric in the space V.

Note that we can easily modify this formulation to preserve other structures, such as dot products, weights between distances, or different types of metrics in each space. It turns out that when the distance metric is the Euclidean L2 distance, then the solution to this linear map turns out to be PCA. This may be a more intuitive way to think about PCA, since we're trying to preserve the pairwise distances between the data points.

Theorem 3.1 (Equivalence of Classical MDS and PCA)

If the distance metric is the Euclidean L2 distance, then the solution to the MDS problem is equivalent to PCA. That is,

$$T_k = \underset{T}{\text{argmin}} \sum_{i \neq j} \left(||T(x_i) - T(x_j)||^2 - ||x_i - x_j||^2 \right) \tag{6}$$

Generally, if you don't use classical MDS, then you will get a different answer than PCA and there doesn't exist a closed form solution, so you'll have to minimize this numerically.

Example 3.1 (Non Classical MDS)

The loss

$$\sum_{i \neq j} \left(||T(x_i) - T(x_j)|| - ||x_i - x_j|| \right)^2 \tag{7}$$

does not give the same solution as PCA.

4 Isomap

Isomap is a bit different in the way that it tries to capture more of the global structure of the data, which brings advantages and disadvantages. It is simply a modification of MDS but with geodesic distances.

Definition 4.1 (Isomap)

You start off with the point cloud, but with every point, X_i, you find the local neighborhood N_i and you make a weighted graph over the whole dataset in the high dimensional space. Then, the distance between any two arbitrary points is the weighted sum of the path between them, calculated by Dijkstra's algorithm. Intuitively, this is an approximation of the geodesic distance between these two points on a manifold. Call this distance d_G. Then, we simply do MDS by minimizing

$$\min_{T} \sum_{i \neq j} \left(d_{\mathbb{R}^k}(T(x_i), T(x_j)) - d_G(x_i, x_j) \right) \tag{8}$$

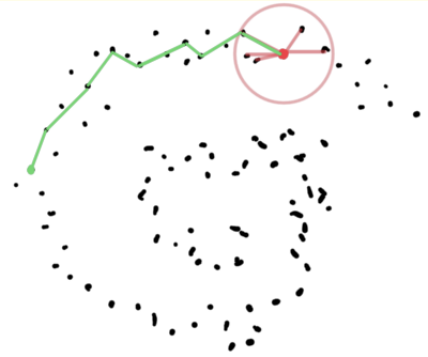

Figure 2: The classical example is the spiral manifold. The data lies in this manifold, and the geodesic distance helps us gain an accurate distance metric within this data.

The problem with this is that it is very sensitive to noise. For example, if we had a few points lying between the spirals, then the geodesic distance between the two spirals would be very small, and so the MDS would try to bring them closer together.

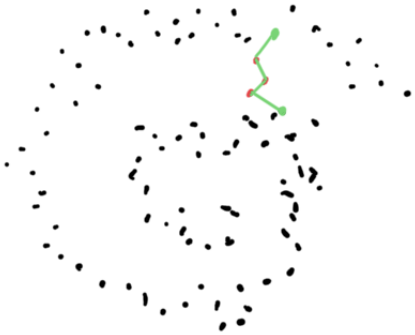

Figure 3: With extra noisy points (red), the geodesic distance can get corrupted.

To fix this, we use *diffusion maps*, which looks at all possible paths between two points and looks at some average of them, which increases robustness.

5 Local Linear Embedding

PCA and MDS are linear embedding methods. Let's move onto nonlinear ones. The first nonlinear models that we work with again use the idea of locality (remember kernel regression). You have data that is globally nonlinear, but if you look at a point and its local neighborhood around it, then it is approximately linear since we assume that it lives in some smooth manifold.

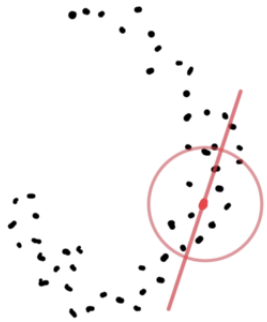

Figure 4: Local linear embedding assumes that the data is locally linear.

The concept of neighborhood can be defined in two ways. You can either just fix an ϵ and take the ϵ-ball around each point x_i. Or you can fix a k and take the k nearest neighbors of each point. The general idea of using kernel PCA is to take a local neighborhood of the data and construct some linear approximation of it.

6 UMAP

7 t-SNE