CS725: Recap kernels

A kernel $K(x,y)$ takes x,y inputs in the original feature space and computes $\phi(x)^T\phi(y)$, which is a measure of similarity between x and y in a new (higher-dimensional) subspace defined implicitly by ϕ.

How do we determine if K is valid?

1. Can we identify a ϕ function s.t. $K(x,y) = \phi(x)^T\phi(y)$
2. Mercer's theorem
3. By applying known operations (sum, product, etc.) on known/valid kernels

Q. Is $K(x,y) = \|x+y\|^2$ a valid kernel?

No

$K(x,y) = \|x+y\|^2 = (x+y)^T(x+y)$

$K(0,0) = 0 = \phi(0)^T\phi(0) \Rightarrow \phi(0) = 0$

$K(x,0) = 0 \longrightarrow$ A $\because \phi(0) = 0$

$K(x,0) = \|x\|^2 \longrightarrow$ B $\because$ definition of $K(x,y)$

CONTRADICTION.

Examples of popular kernels

(A) Polynomial kernel: $K(x,y) = (x^T y)^d, \ d>0$

$$= (c + x^T y)^d, \ d>0$$

(B) Gaussian kernel: $K(x,y) = \exp\left(-\frac{1}{2\sigma^2}\|x-y\|^2\right)$

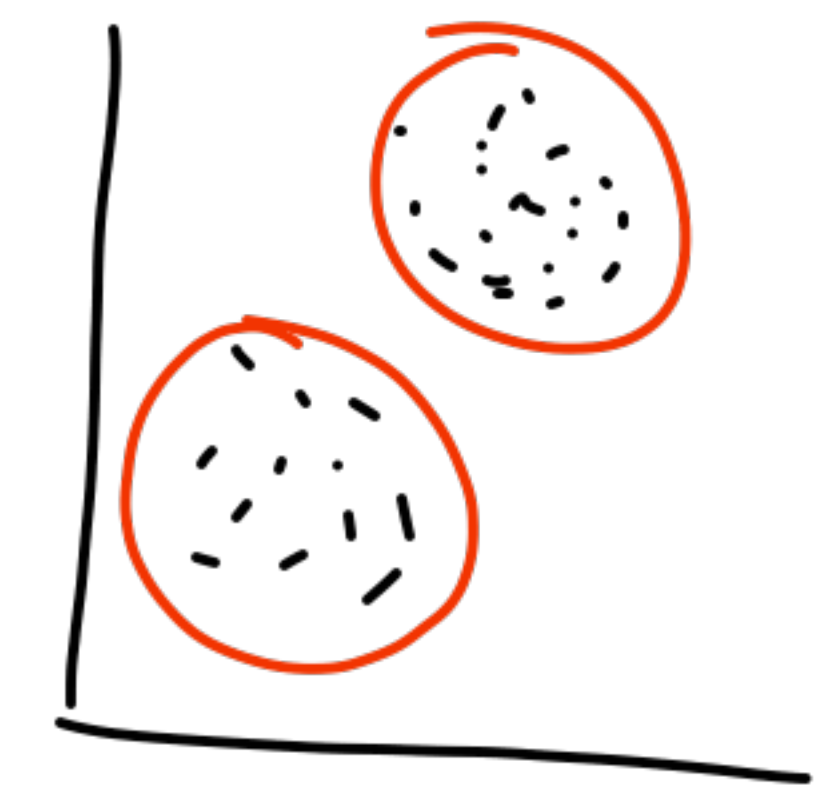
[The implicit ϕ corresponding to a Gaussian kernel is infinite-dimensional]

for v. small σ, all the pts are distinct or different (K is diagonal matrix)

for v. large σ, all the pts are v. similar (K is a uniform matrix)

CLUSTERING

Given a set of points $x_1, x_2, \ldots, x_n$, $x_i \in \mathbb{R}^d$, we aim to find K clusters of points where the points within each cluster are similar to each other [typically in terms of Euclidean distances].



Motivation: (1) first, coarse way of labeling points

(e.g. of clustering; Image segmentation, customer aggregation, etc.)

(2) Outlier detection

(3) Lossy compression of the original data

K-means clustering algorithm

Assume we have K clusters of points; each point in a cluster is closest to its centroid (more than any other $\underbrace{\text{cluster centroid}}_{\text{mean of all the datapoints in a cluster}}$)

If cluster assignment is known, it is easy to compute the centroids
If cluster centroids are known, it is easy to do cluster assignment
How do we solve this chicken-egg problem? Fix one, optimize the other!
(ALTERNATING OPTIMIZATION)

k-means objective:

Find $\mu_1, \mu_2, \ldots, \mu_k$, $\mu_i \in \mathbb{R}^d$ (cluster centroids) and $C^1, C^2, \ldots, C^n$ (cluster assignments for n training points) to minimize the sum of squared distances of each point from its cluster centroid. Here, C^i is a 1-hot vector of size k.

Objective function: $$\min_{\{\mu_k\}} \min_{\{C^i\}} \sum_{i=1}^{n} \sum_{k=1}^{K} C^i_k \underbrace{\| \mu_k - x_i \|^2}_{\text{A}}$$

where $C^i_k = \mathbb{1}[x_i \text{ is assigned to cluster index } k]$

Heuristic to solve (A) [to reach a local optimum] is to fix one set of variables and optimize the other

Step 1: fix μ and optimize the assignment

For the i^{th} datapoint, $\min\limits_{C_i} \sum_{k=1}^{K} C_k^i \, \| \mu_k - x_i \|^2$

$$C_k^i = \begin{cases} 1 & \text{if } k = \arg\min\limits_j \| \mu_j - x_i \|^2 \\ 0 & \text{otherwise} \end{cases}$$

ASSIGNMENT STEP

<u>Step 2:</u> fix the assignment and optimize for the cluster centroids

$$\frac{\partial}{\partial \mu_k} \sum_{i=1}^{n} \sum_{k=1}^{K} c_k^i \| \mu_k - x_i \|^2 \text{, set to 0 and solve for } \mu_k$$

$$\Rightarrow \quad 2 \sum_{i=1}^{n} c_k^i (\mu_k - x_i) = 0$$

UPDATE STEP

$$\Rightarrow \quad \boxed{\mu_k = \frac{\sum_i c_k^i x_i}{\sum_i c_k^i}}$$

$\longrightarrow$ cluster centroids are mean of all the assigned datapoints

k-means algorithm

Given : $x_1, x_2, \ldots, x_n$, $x_i \in \mathbb{R}^d$, number of clusters is K

Initialize $\mu_1, \mu_2, \ldots, \mu_K$

Repeat until convergence (i.e. assignments do not change)

1. Assignment : Assign each point to the closest μ_k
2. Update : Recompute the cluster centroids

k-means is guaranteed to converge (to a local optimum)

which depends on $\mu_1, \dots, \mu_K$ initialization

Because of the following two facts:

1. The cost or objective either stays the same or decreases across iterations
2. There are only a finite number of cluster assignments

k-means is very sensitive to its initialization

Common initialization strategy is to pick the centroids randomly

Another strategy is "farthest point". Pick the first centroid at random, and every succeeding centroid is the farthest point from the preceding centroids

Limitation of farthest pt $\rightarrow$ could pick outliers

Another variant is <u>k-means++</u> $\rightarrow$ instead of farthest pt, pick a pt based on a prob distribution weighted acc. to $D(x)^2$ where $D(x)$ is the distance of point x from the preceding centroids

How do we choose k in k-means clustering?

Keep varying k and compute the squared distances of all pts from its centroids