COMS 4995: Unsupervised Learning (Summer'18) May 22, 2018

Lecture 1 – Introduction and Clustering

Instructor: *Nakul Verma* Scribes: *Vincent Liu, Yadin Rozov, Laura Tinsi*

Today, we introduce the content of the class and begin the first topic of this class: clustering. Unsupervised Learning is based on the assumption that given a dataset $\mathcal{X}$, there exists an underlying structure in $\mathcal{X}$. Our learning task is to discover this structure given n examples from data. We aim to come up with a summary of the data using the discovered structure.

1 Introduction and Logistics

1.1 Agenda

- Partition the data into meaningful structure: Clustering (2 weeks).
- Find a low-dimensional representation that retains important information about the dataset and suppresses irrelevant/noisy information: dimensionality reduction (2 weeks).
- Understand and model how data is distributed in space: density estimation (1 week).

1.2 Logistics and Grading

- Website: http://www.cs.columbia.edu/~verma/classes/uml/index.html
- Piazza to access homeworks and ask questions, Gradescope to submit the homeworks submission
- HW0 due by 11:59 Sunday
- Grading: 45% HW, 10% scribing, 25% project, 20% participation in class

2 Clustering

- What is clustering and why do we need it?
  - Given input data, partition the data into multiple meaningful groups
- Why?
  - Approximate large/infinite/continuous set of objects with finite set of representatives.
  Example of applications: vector quantization, codebook learning, dictionary learning.
  It is used in computer vision to map from pixel space to HOG (histogram of oriented gradient) space - and edge detection - histogram profile or edge orientation gives rise to new features
  - Find a meaningful way to group data.
  Example of applications: in exploratory data analysis, gives a good understanding and summary of your input data (ex of biology).

2.1 Problem setup: Metric space

2.1.1 Definition

Definition 1. *Let $\mathcal{X}$ be a space. We define the metric space $(\mathcal{X}, \rho)$ such that $\rho : \mathcal{X} \times \mathcal{X} \longrightarrow \mathbb{R}$ is the distance function such that it verifies :*

- *Non-negativity (reflexivity):* $\forall x, y \in \mathcal{X}, \rho(x, y) \geq 0$ *with equality iff* $x = y$

- *Symmetry:* $\forall x, y \in \mathcal{X}, \rho(x, y) = \rho(y, x)$

- *Triangle Inequality:* $\forall x, y, z \in \mathcal{X}, \rho(x, y) \leq \rho(x, z) + \rho(z, y)$

Definition 2. *For a set $T \subset \mathcal{X}$, we define :*

$$\rho(s, T) := \inf_{t \in T} \rho(s, t)$$

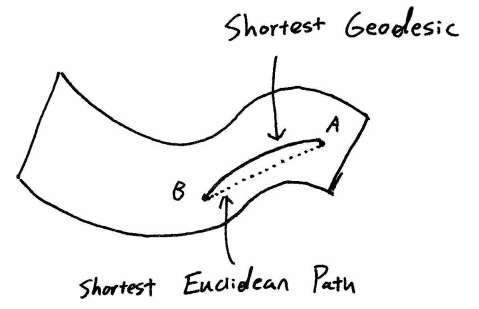
2.1.2 Examples

- ℓ_2 distance on $\mathbb{R}^d$: $\forall x, y \quad \ell_2(x, y) = \rho(x, y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$

- ℓ_1 distance on $\mathbb{R}^d$: $\forall x, y \quad \ell_1(x, y) = \rho(x, y) = \sum_{i=1}^{d} |x_i - y_i|$

- ℓ_∞ distance on $\mathbb{R}^d$: $\ell_\infty = \max_i |x_i - y_i|$

- Geodesics on manifolds: In a Non-linear space the distance between two points is defined as the shortest path between those two points. Note that the path must be included in the space. It is called the shortest geodesic.



- Shortest path on graphs (for example a social network graph): the distance is the shortest path i.e the smallest number of edges between two vertices.

- **Note**: metric space isn't always Euclidean and we cannot perform Euclidean operations (ie. taking the mean) in any metric space.

2.1.3 Covering of a metric space

Covering, ϵ-covering, covering number

Definition 3. *Let $S \subset \mathcal{X}$, $\epsilon \geq 0$. A set $T \subset \mathcal{X}$ is an ϵ-cover of S if: $\forall s \in S$, $\exists t \in T$ s.t. $\rho(s,t) \leq \epsilon$.*

In other words, $\rho(s,T) := \inf_{t \in T} \rho(s,t)$ and T ϵ-covers S if $\sup_{s \in S} \rho(s,T) \leq \epsilon$.

Questions to consider :
- Is S a cover of S?
  Yes.
- Vertices of a d-cube with distance ℓ_∞: What is a good 1-cover, $\frac{1}{2}$-cover, 0.9-cover?
  You can think of it in the 2-dimensional case. $\{-1,+1\}^d$ vertices of the cube. 1-cover is just the zero vector.

2.2 k-center problem

- Input: a set of n points $x_1, ...x_n \in \mathcal{X}$ assuming (X,ρ); a positive integer k
- Output: $T \subset \mathcal{X}$ s.t. $|T| = k$
- Goal: minimize "cost" of T where: $cost(T) := \max_{i \in \{1...n\}} \rho(x_i, T)$

How would we solve it? k-means? No, since we don't have an Euclidean space and distance here.

Solution:
- Exhaustive search (takes a long time, exponential complexity)
- Farthest First Traversal Algorithm (polynomial complexity)

Algorithm 1 Farthest First Traversal Algorithm

```
Pick randomly z ∈ X and set T = {z}
while |T| < k do
  z = arg max_{x∈X} ρ(x, T)
  T = T ∪ {z}
end while
```

Important: This is not an optimal solution to the k-center problem. A counter example we can think of is the 2-center problem ($|T| = 2$) on the space $\mathcal{X} = \{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}$, with the usual metric ρ. The farthest fast traversal algorithm might yield to $\{0,1\}$ while the optimal solution is $\{\frac{1}{4}, \frac{3}{4}\}$.

Theorem 4. *Let T^* be the optimal solution of the k-center problem and $cost(T^*)$ be its optimal cost. Let T be a solution given by the Farthest First Traversal Algorithm. Then we have that :*

$$cost(T^*) \leq cost(T) \leq 2 * cost(T^*)$$

In other words, Farthest First Traversal Algorithm is 2-optimal for the k-center problem.

Proof. Let $cost(T) = r = \max_{s \in \mathcal{X}} \rho(s, T)$. There exists $x_0 \in \mathcal{X}$ such that $x_0 = \arg\max_{x \in \mathcal{X}} \rho(x, T)$. We set $T' = T \cup \{x_0\}$, and observe that $\forall t_i, t_j \in T', t_i \neq t_j : \rho(t_i, t_j) \geq r$, because if $x_0 \notin T$, then during any iteration of the farthest first traversal algorithm, for the new center t_i added to T it must be true that $\rho(t_i, T) \geq r$, otherwise x_0 would be already added to T. We note that $|T'| = k + 1$ and $|T^*| = k$. Because there are k elements in T^* covering $k + 1$ elements in T', by the pigeonhole principle, there exists some $t^* \in T^*$ that covers at least two element $t_1, t_2 \in T'$. As $\rho(t_1, t_2) \geq r$, by the triangle inequality, at least one of $\rho(t_1, t^*)$ or $\rho(t_2, t^*)$ is at least $\frac{r}{2}$. This implies: $\text{cost}(T^*) \geq \frac{r}{2}$. □

Remark: The algorithm that gives the an optimal solution to this problem isn't a polynomial time algorithm but is in fact NP-hard. Even a $(2-\epsilon)$-approximation is NP-hard for general metric spaces.

Some related open problems: Hardness in Euclidean spaces. Is k-center problem still hard in these spaces? Can we get better than 2-optimality? Is this the better algorithm to solve the problem?

2.3 k-means problem

- Input: A set of n points $x_1, ...x_n \in \mathbb{R}$ and a positive integer $k < n$.

- Output: $T \subset \mathbb{R}$ s.t. $|T| = k$.

- Goal: minimize "cost" of T where: $cost(T) := \sum_{i=1}^{n} \min_{\mu_j \in T} \|x_i - \mu_j\|^2$.

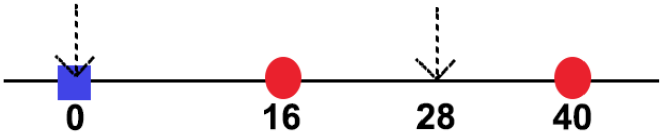
Solution:
- Exhaustive search (takes a long time, exponential complexity)
- Farthest First Traversal Algorithm?
- Lloyd's method for the k-means (Here, T is the set of the centroids $\mu_j, j \in \{1, ..., k\}$).

Algorithm 2 Lloyd k-means algorithm

Pick randomly $x_1, ..., x_k$ from $\mathcal{X}$
while stopping criteria not satisfied **do**
  Assign points of the dataset to the closest center
  Get the clusters $C_1, ..., C_k$
  Compute the new centrods $\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$, $\forall j \in \{1, ..., k\}$
end while

Important: This is not an optimal solution of the k-means problem and we do not have any approximation guarantee on the solution: it is an unbounded approximation solution problem. This is explained by the fact that you can find a local optimum as far as possible of the global optimum. This illustrates the **hardness of k-means theorem**: k-means is a NP-hard optimization problem (k-means ++).

Example of k-means non-optimality: Consider the following setting and dataset where $k = 2$ and $d = 1$:



The dashed lines point to the current cluster centers, with 0 for the blue cluster and 28 for the red cluster. On this setting, Lloyd's algorithm makes no further improvement (it terminates after one iteration), since the middle point at 16 is closer to the red cluster center 28 than to the blue cluster center 0. And the total cost is $0^2 + 12^2 + 12^2 = 288$. However, the optimal cluster center assignment should be 8 for the blue cluster and 40 for the red cluster, which gives a cost of $8^2 + 8^2 + 0^2 = 128$. Thus, Lloyd's algorithm does not give the optimal solution.

References

[1] Gonzalez, F. "Clustering to minimize the maximum intercluster distance." *Theoretical Computer Science 38* (1985): 293-306.

[2] Hartigan, John A. "Clustering Algorithms" *John wiley & sons* (1977).

[3] Sanjoy Dasgupta. "The hardness of k-means clustering" *Department of Computer Science and Engineering University of California, San Diego* (2008): Technical Report CS2008-0916.