UNIVERSITY OF CALIFORNIA, BERKELEY

Lecture 21: CS-189

Oscar Ortega

July 16, 2021

1 SINGULAR VALUE DECOMPOSITION

Problems:

- Computing $X^T X$ takes $O(nd^2)$ time
- $X^T X$ is poorly conditioned → numerically inaccurate eigenvectors

Compact SVD: Fact: If $n \geq d$, we can find a singular value decomposition:
$X = U\Sigma V^T$, where $X \in \mathbb{R}^{n,d}, U \in \mathbb{R}^{n,d}, \Sigma \in \mathbb{R}^{d,d} V^T \in \mathbb{R}^{d,d}$

- The orthonormal $u_i' s$ are the **left singular vectors** of X
- the orthonormal $v_i' s$ are the **right singular vectors** of X
- the diagonal matrices or $\sigma_1, ..., \sigma_d$ are non-negative **singular values** of x.

Note that $X = \sum_{i=1}^{d} \sigma_i u_i v_i^T$, in other words, what this tells us is that any matrix $X \in \mathbf{R}^{n,d}$ can be expressed as a linear combination of d outer product matrices. An outer product matrix is also known as a **dyad**.
Fact: v_i is an eigenvector of $X^T X$ w/eigenvalues σ_i^2
Proof:
$X^T X = V\Sigma U^T U\Sigma V^T = V\Sigma^2 V^T$, which is an eigen-decomposition of $X^T X$.

Fact: We can find the k greatest singular values corresponding vectors $\in O(ndk)$ time.

Important: Row i of UD gives the principle coordinates of the sample points. (i.e $X_i^T v_j \forall j$).

2 CLUSTERING

Partition data into clusters so points in a cluster are more similar than across clusters.
Why?

- discovery: Find songs similar to songs you like; determine market segments
- Hierarchy: find good taxonomy of species from genes.
- Quantization: Compress a data-set by reducing choices. Given a complex photograph with lots of different colors, find a subset of the colors that best represent the data.
- Graph Partitioning: Image segmentation; find groups in social networks.

2.1 K-MEANS CLUSTERING: AKA LLOYD'S ALGORITHM (STUART LLOYD, 1957)

Goal: Partition n points into k disjoint clusters. Assign each input point X_i a cluster label $y_i \in [1,k]$ Where each cluster's mean is the average of the points n the label that was given:

$$\mu_i = \frac{1}{n_i} \sum_{x \in \mathscr{Y}_k} X$$

Goal: Find y that minimizes $\sum_{i=1}^{k} \sum_{x \in \mathscr{Y}_k} \|X_k - \mu_i\|^2$ NP-Hard: Solvable in $O(nk^n)$ time.
K-Means heuristic: Alternate between:

1. y_j's are fixed and update μ_i's
2. μ_j's are fixed and update y_i's

Where we stop when when 2 changes no assignments.

1. We can show with calculus the optimal μ_i is the mean of the points in cluster i
2. The optimal y assigns each point X_j to the closest center μ_i

Both steps decrease the value of the objective fn unless they change nothing. It is guaranteed the algorithm will terminate. This algorithm is usually very fast in practice, and finds a local minimum, but is often the not the global minimum.

2.2 HOW TO START

- Forgy Method: choose k random sample points to be initial μ_i's to be initial, go to (2) 'better in most standard cases'.
- Random Partition: randomly assign each sample point to cluster; go to (1)
- k-means++, Like Forgy, but biased distribution, want to avoid points that are close to the first point chosen.

For best results, run k-mean multiple times with random starts.

2.3 Equivalent Objective fn: The within cluster variation

Equivalent at the minimum.

$$\min_y \sum_{i=1}^{k} \frac{1}{n_i} \sum_{y_j=i} \sum_{y_m=i} |X_j - X_m|^2$$

Sometimes we will need to normalize the data. It will depend on the problem and whether the differences in magnitudes of the scalars of the different features mean anything.

2.4 k-medoids

Generalizes k-means beyond euclidean distance: Specifically a distance function $d(x, y)$ between pts: x,y, aka **similarity**. Should ideally satisfy the triangle inequality. eg: l_∞, l_1, other distance metrics. Replace mean with a **medoid**, the sample point that minimizes total distance to other points in some cluster.

2.5 Hierarchical Clustering

Creates a tree; every subtree is a cluster

- bottom -up: aka **agglomerative clustering**
- start with each pt, a cluster: repeatedly fuse pairs
- top - down: aka **divisive clustering**
- start w/all pts in one cluster, repeatedly split it.
- We need a distance fn for clusters A,B
  - **complete linkage**: $d(A, B) = \max\{(d(w, x) : w \in A, x, \in B\}$
  - **single linkage**: $d(A, B) = \min\{(d(w, x) : w \in A, x, \in B\}$
  - **average linkage**: $d(A, B) = \{\frac{1}{|A||B|} \sum_{w \in W} \sum_{x \in X} d(w, x)\}$
  - **centroid linkage**: $d(A, B) = d(\mu_a, \mu_b)$

Greedy agglomerative alg: Repeatedly fuse the two clusters that minimize $d(A, B)$ Naively takes $O(n^3)$ time
Dendrogram: Illustration of hierarchy in which the vertical axis encodes all the linkage distances.
It usually is the case that, average linkage, and complete linkage tend to perform better than single linkage.