CSE 4502/5717: Big Data Analytics
April 23, 2018; Lecture 24: Data Reduction

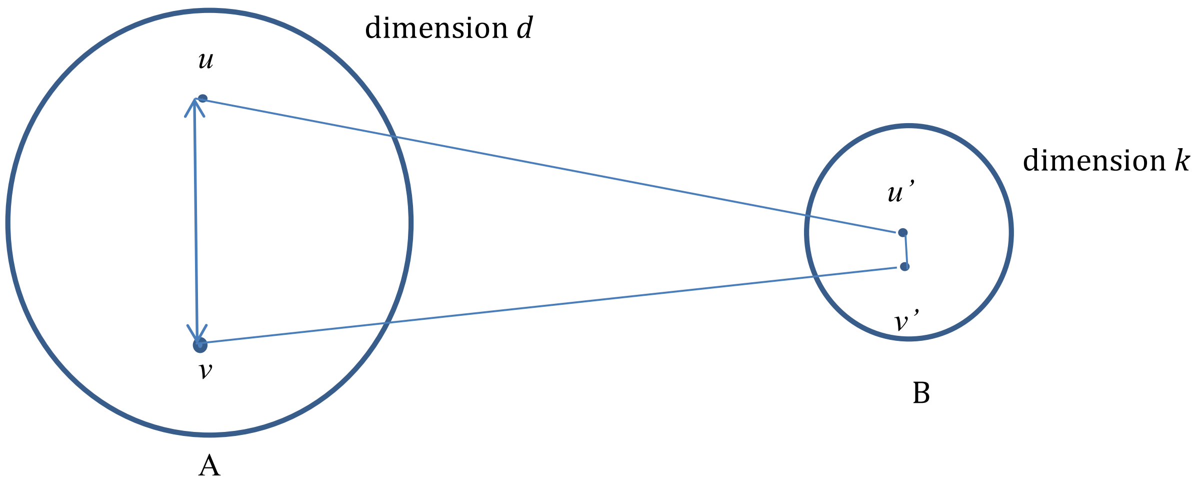
Data Reduction

When we have big data, relevant data reduction techniques are called for to reduce the data size. Two forms of data reduction can be useful.

The first kind is to reduce the dimension. There are many algorithms whose run times depend exponentially on the dimension. For example, the randomized algorithm of Rabin for finding the closest pair of points from a given set of points has an expected run time that is linear in the number of input points and exponential in the dimension. As another example, the best-known algorithms for finding the convex hull of a given set of points have run times that are exponential in the dimension. If the dimension is large, these algorithms may not perform well in practice. This problem is referred to in the literature as the "curse of dimensionality". Algorithms that reduce the dimension while preserving the information closely could be quite useful while dealing with big data. One such technique is to employ random projections.

Random Projections:

Random Projection was first proposed by Johnson & Lindenstrauss (1984).



The idea here is to project from a high dimensional space to a lower dimensional space.
The distance between any two points should be closely preserved. (In the above figure, points u and v are from a high dimensional space A with dimension d. Their distance should be preserved when projected to a lower dimensional space B with dimension k).

Application: Analyzing genome data, medical image analysis, etc.,

Theorem:
Let S be any set of n points from d-dimensional space.
Let $k \geq 4\left(\frac{\varepsilon^2}{2} - \frac{\varepsilon^3}{3}\right)^{-1} \log_e n$
where ε is a constant, $1 > \varepsilon > 0$.
Then $\exists$ a projection f of points in S into a k-dimensional space such that distance is closely preserved for any pair of points u and v from S. Specifically, the following inequalities hold:

$$(1-\varepsilon)\ \|u-v\|^2 \leq \|f(u)-f(v)\|^2 \leq (1+\varepsilon)\|u-v\|^2$$

(Please note || is the L2 norm).
We can find such a projection in randomized polynomial time.

Another kind data reduction can be achieved by reducing the number of input points. This can be done with the employment of clustering algorithms.

Clustering Algorithms
We can think of these as data-reduction (this would be point reduction by replacing each cluster by a single point (centroid of the cluster, for example)).

Let $\boldsymbol{X}$ be a given point set. Then, clustering is defined as a partitioning of $\boldsymbol{X}$ into k groups:
$\boldsymbol{X} \longmapsto \boldsymbol{X_1}, \boldsymbol{X_2}, \ldots \boldsymbol{X_k}$ *s.t.* points in each cluster are similar to each other.

Let $d(i,j)$ be a distance measure between $p_i, p_j \in \boldsymbol{X}$. We say p_i and p_j are similar if $d(i,j) \leq \epsilon$ (for some small value).

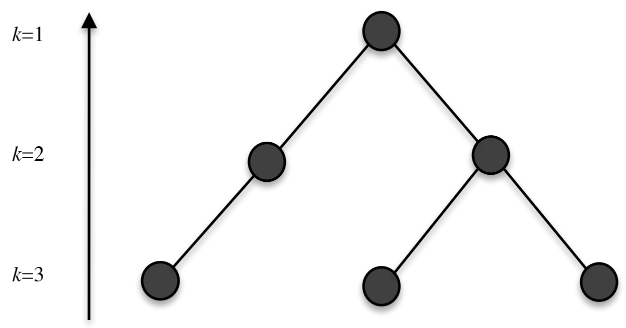
(In general, we can use some objective function to define clustering).

Hierarchical Clustering
input: $\boldsymbol{X}=\{p_1, p_2, \ldots p_n\}$, k (the number of clusters to compute)
output: k clusters

Algorithm
[0] to begin we have n clusters (each point is a cluster): $\{p_1\},\{p_2\},\ldots\{p_n\}$
[1] merge clusters (in many stages):
    at any stage we merge the pair of clusters whose distance is minimum.

We could represent the sequence of merges done using a tree called the dendrogram.

Definition: distance between two clusters c_i, c_j can be defined in several ways:
(i) **single-link** - $d(c_i, c_j) = \min\limits_{p \in c_i, q \in c_j} \{d(p,q)\}$
(ii) **complete-link** - $d(c_i, c_j) = \max\limits_{p \in c_i, q \in c_j} \{d(p,q)\}$
(iii) **average-link** - $d(c_i, c_j) = \operatorname*{mean}\limits_{p \in c_i, q \in c_j} \{d(p,q)\}$



Theorem: we can do hierarchical Clustering in $O(n^2)$ time.

Construct a pairwise distance matrix $\boldsymbol{D}[i,j] = d(p_i, p_j)$ and minimum array **min**. The **min** array stores the row minima.

D	1	…	*j*	…	*n*	**min**
1	*d(1,1)*	…	*d(1,j)*	…	*d(1,n)*	First row min
…	…		…		…	…
i	*d(i,1)*	…	*d(i,j)*	…	*d(i,n)*	…
…	…		…		…	…
n	*d(n,1)*	…	*d(n,j)*	…	*d(n,n)*	Min of row *n*

[0] to begin with using **min** find the closest pair of points. At any given time we have a single row to represent a cluster.

[1] merging: Let p_i and p_j be the two closest clusters. We merge these two rows into one. For example, the merged row could be row i. Merging of the two rows is done as follows.

$$d(i,q) = \min_{1 \le q \le n} \{d(i,q), d(j,q)\};$$

[2] do [1] n-2 times

Analysis
It takes $O(n^2)$ time to do all pairwise calculations (assuming that $d(i,j)$ is done in $O(1)$ time for a specific i and j.)
For each merge we spend $O(n)$ time for a total merging time of $O(n^2)$.
Total time then is: $O(n^2) + O(n^2) = O(n^2)$.

Note: Instead of getting the number of clusters as the input, we could also think of a variation where we get a threshold on the distance. We could then eliminate all the edges in the dendrogram whose edge weights are greater than the threshold. Each tree in the resultant forest will be an output cluster.

Parallel Hierarchical Clustering (Rajasekaran 2004)
[1] Construct a minimum spanning tree (MST).
[2] Delete all the edges whose weights are greater than the given distance threshold.
[3] Find the connected components in the resultant forest. Each component is a cluster.

claim: ∃ a Las Vegas implementation of this algorithm on the CRCW PRAM model that runs in $\tilde{O}(\log n)$ time using $\frac{n^2}{\log n}$ processors.

Theorem (Pettie & Ramachandran 2002)
We can construct a MST for any undirected graph $G(V,E)$ in $\tilde{O}(\log|V|)$ using $\frac{|E|+|V|}{\log |V|}$ CRCW PRAM processors. (use this for Rajasekaran step[1]).

Fact: We can find the connected components in a forest F in $O(\log |V|)$ using $|V|$ processors.

$\Rightarrow$ we can do $\|^l$ Hierarchical Clustering in $\tilde{O}(\log n)$ time using $\frac{n^2}{\log n}$ processors.

Application: Records Linkage
Useful in biomedical informatics where individuals may vary services, generating several partial records at disparate medical locations.

input: $D_1, D_2, \ldots D_k$ (data sets of medical records)
output: clusters where each cluster contains records belonging to the same individual exclusively.

End of lecture, continued in lecture 25.