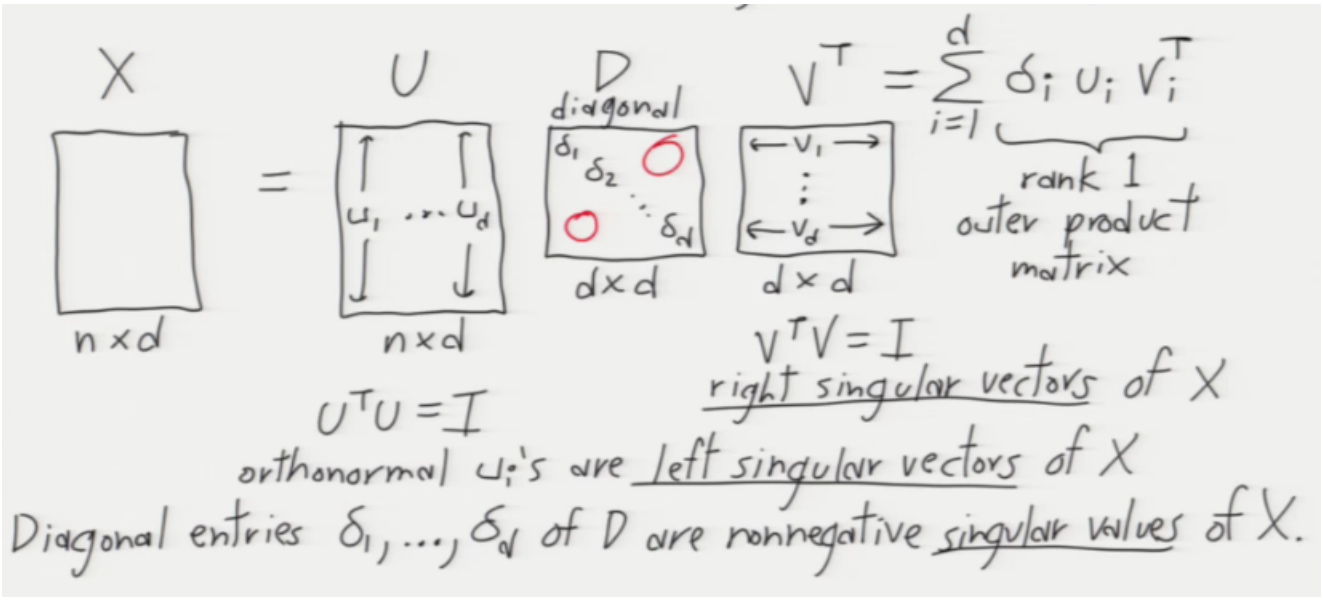
04/13/2016

Singular Value Decomposition (SVD)

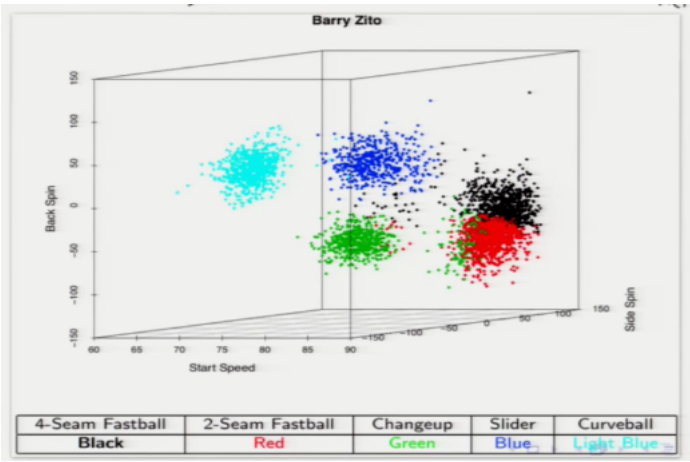
- Problems: Computing $X^T X$ takes $\mathcal{O}(nd^2)$ time.
- $X^T X$ is poorly conditioned $\rightarrow$ numerically inaccurate eigenvectors.
- Fact: If $n \geq d$, we can find a singular value decomposition



- Poorly conditioned: look at biggest and smallest eigenvalues, divide.
- Fact: v_i is an eigenvector of $X^T X$ with eigenvalue δ_i^2.
- Proof: $X^T X = VDU^T UDV^T = VD^2 V^T$ which is an eigen-decomposition of $X^T X$.
- Fact: We can find the k greatest singular values and the corresponding vectors in $\mathcal{O}(ndk)$ time.

Clustering

- Partition data into clusters so points within 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.
  - Graph partitioning: Image segmentation; find groups in social network.



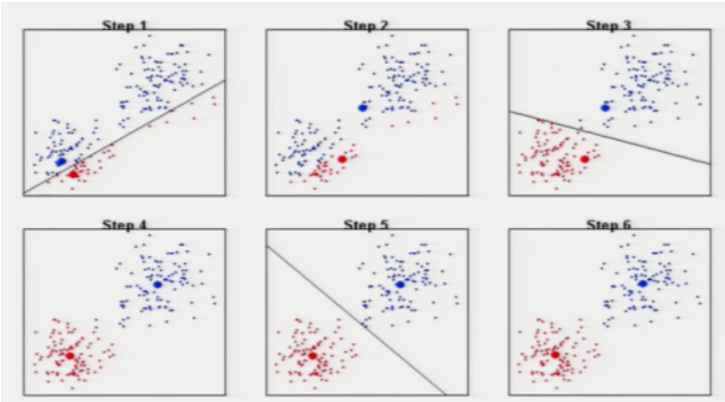
k-Means Clustering aka Lloyd's Algoritm

- Goal: Partition n points into k disjoint clusters. Assign each input point X_i a cluster label $y_i \in [1, k]$.
- Cluster i's <u>mean</u> is $\mu_i = \frac{1}{n_i} \sum_{y_j=i} X_j$, where $n_i = \#$ of points in i.

$$\boxed{\text{Find } y \text{ that minimizes } \sum_{i=1}^{k} \sum_{y_j=i} |X_j - \mu_i|^2}$$

- NP-Hard problem. Solvable in $\mathcal{O}(nk^n)$ time.
- k-means heuristic: Alternate between:
  1. y_j's are fixed; update μ_i's.
     - One can show (w/calculus) the optimal μ_i is the mean of the points in cluster i.
  2. μ_i's are fixed; update y_j's.
     - The optimal y assigns each point X_j to closest center μ_i.

  Halt when step 2. changes no assignments.



- Both steps decrease objective function unless they change nothing. We never revisit the same set of assignments.
- Hence algorithm must terminate.
- Usually very fast in practice.
- Finds a local minimum, often not global.

- Getting started:
  - Forgy method: Choose k random sample points to be initial μ_i's; go to step 2.
  - Random partition: randomly assign each sample point to a cluster; go to step 1.
- For best results, run k-means multiple times with random starts.
- Equivalent objective function: the within-cluster variation

$$\boxed{\text{Find } y \text{ that minimizes } \sum_{i=1}^{k} \frac{1}{n_i} \sum_{y_j=i} \sum_{y_m=i} |X_j - X_m|^2}$$

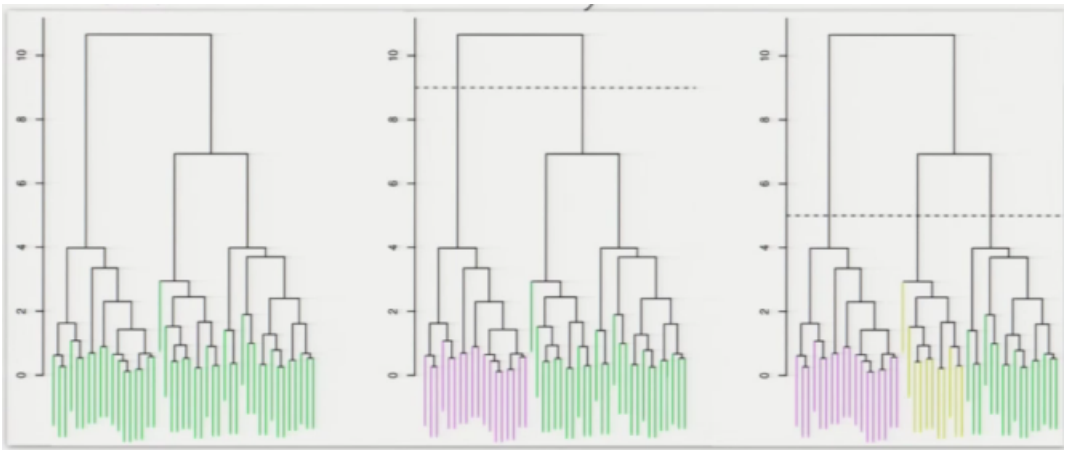
- Normalize the data? Same advice as for PCA.

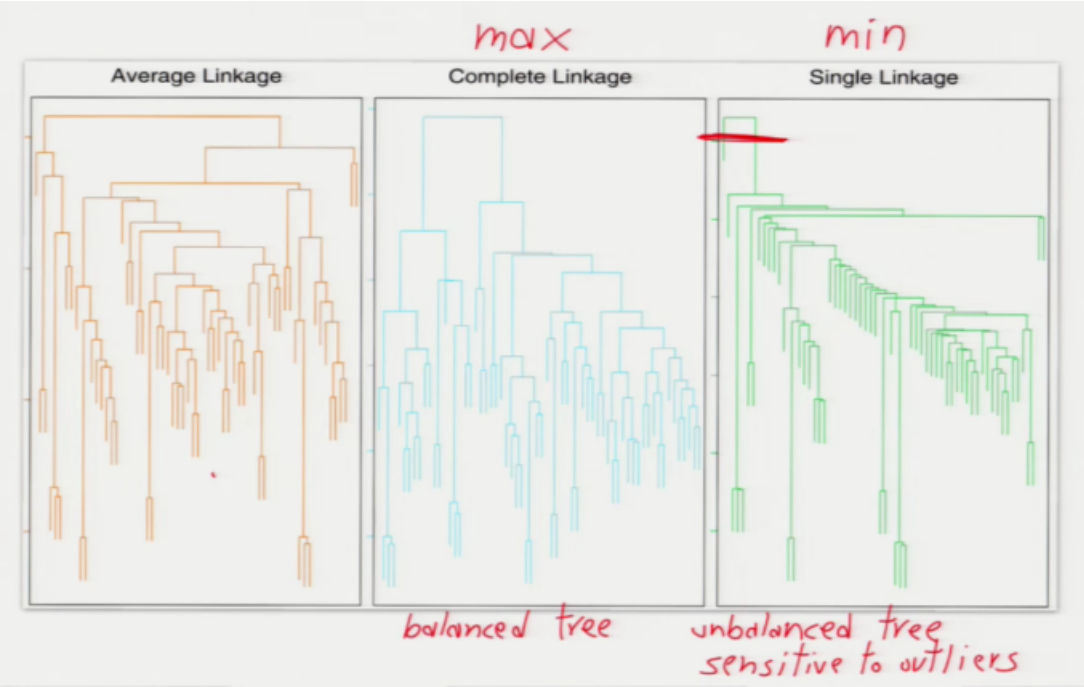
k-Medoids Clustering

- Generalizes k-means beyond Euclidean distance.
- Specify a distance function $d(x, y)$ between points x, y aka dissimilarity.
- Can be arbitrary; ideally satisfies triangle inequality, $d(x, y) \leq d(x, z) + d(z, y)$.
- Replace mean computation with medoid, the sample point that minimizes total distance to other points in same cluster.

Hierarchical Clustering

- Creates a tree; every subtree is a cluster.
- Bottom-up, aka agglomerative clustering: start with each point a cluster; repeatedly fuse pair of clusters.
- Top-down, aka divisive clustering: start with all points in one cluster; repeatedly split it. We need a distance function 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: $d(A, B) = \frac{1}{|A||B|} \sum_{w \in A} \sum_{x \in B} d(w, x)$.
  - centroid linkage: $d(A, B) = d(\mu_A, \mu_B)$.
- Greedy agglomerative algorithm:
  - Repeatedly fuse the 2 clusters that minimize $d(A, B)$. Naively takes $\mathcal{O}(n^3)$ time.
- Dendrogram: Illustration of cluster hierarchy (tree) in which the vertical axis encodes all the linkage distances.





- Warning: centroid linkage can cause inversions where a parent cluster is fused at a lower height than its children.