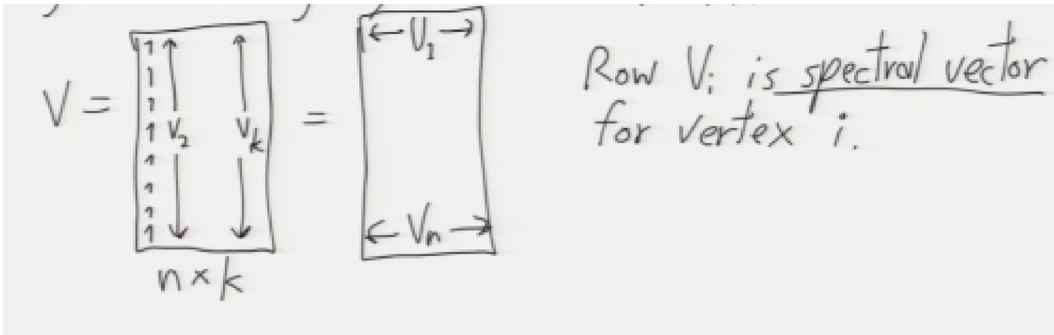
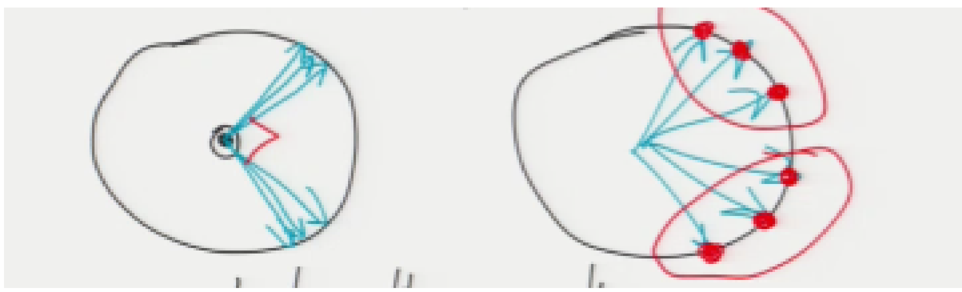
04/20/2016

Clustering w/Multiple Eigenvectors

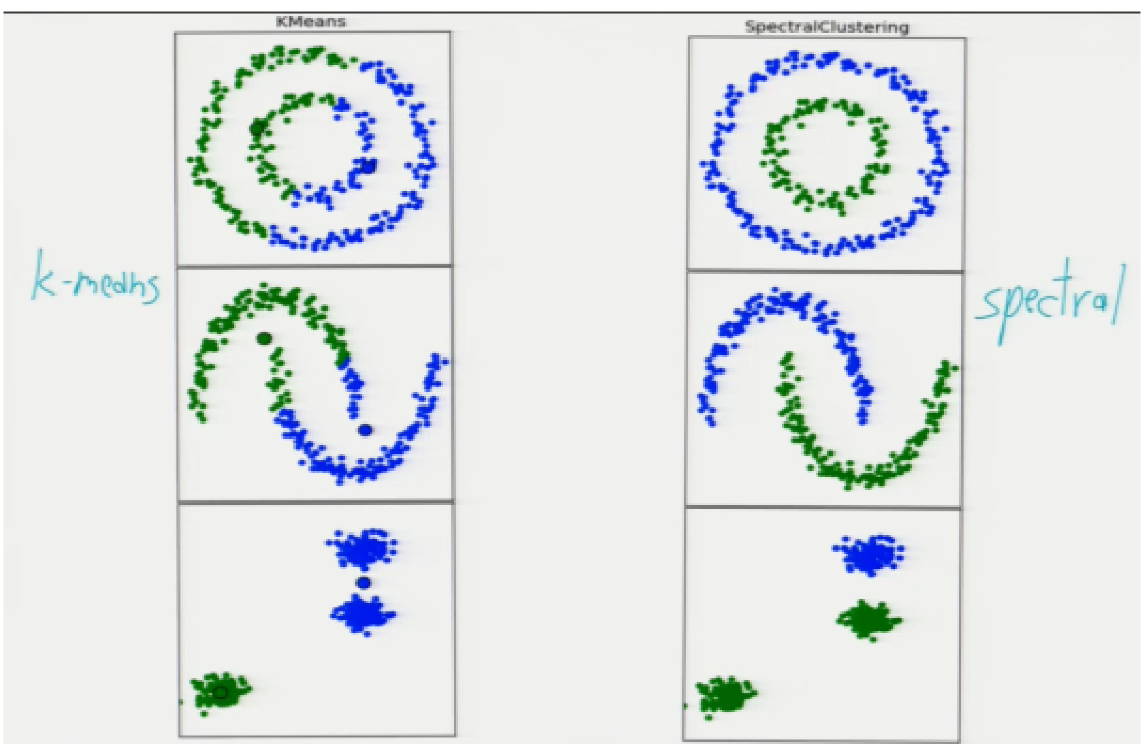
- For k clusters, compute first k eigenvectors $v_1 = \mathbf{1}, v_2, \ldots, v_k$ of generalized eigensystem $Lv = \lambda Mv$.



- Normalize each row V_i to unit length.

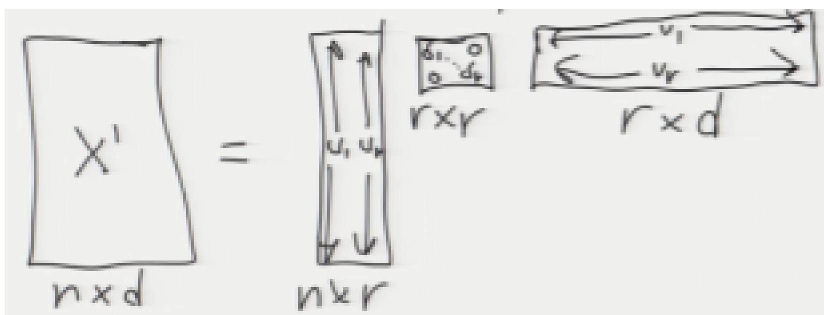


- k-means cluster these vectors.



Latent Factor Analysis

- Suppose X is a term document matrix: row i represents document i; column j represents term j.
- X_{ij} = occurrences of term j in doc i? better: log(1+occurrences).
- Recall SVD $X = UDV^T = \sum_{i=1}^{d} \delta_i u_i u_i^T$, Suppose $\delta_i \leq \delta_j$ for $i \geq j$.
- For greatest δ_i,
  - each v_i lists terms in a genre/cluster of documents.
  - each u_i documents using similar/relater terms.
- e.g. u_1 might have large components for the romance novels, v_i might have large components for terms "passion," "ravish," "bodice."
- Like clustering, but clusters overlap: if u_1 picks out romances and u_2 picks out histories, they both pick out historical romances.
- Application in market research: identifying consumer types (hipsters, soccer mom) and items bought together.
- Truncated sum $X' = \sum_{i=1}^{r} \delta_i u_i v_i^T$ is low-rank approximation(rank r) of X.



$\boxed{X' \text{ is the rank-r matrix that minimizes Frobenium norm } ||X - X'||_F^2 = \sum_{i,j}(X_{ij} - X'_{ij})^2}$

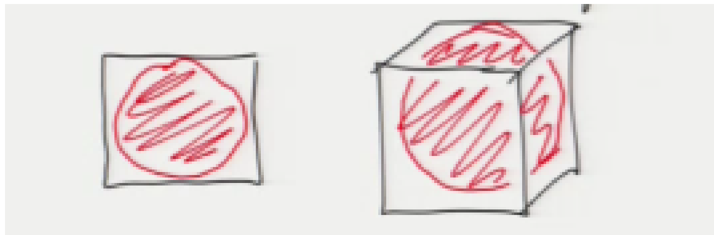
- Applications:
  - Fuzzy search.
  - Denoising.
  - Collaborative filtering: fills in unknown values, e.g. user ratings.

Nearest Neighbor Classification

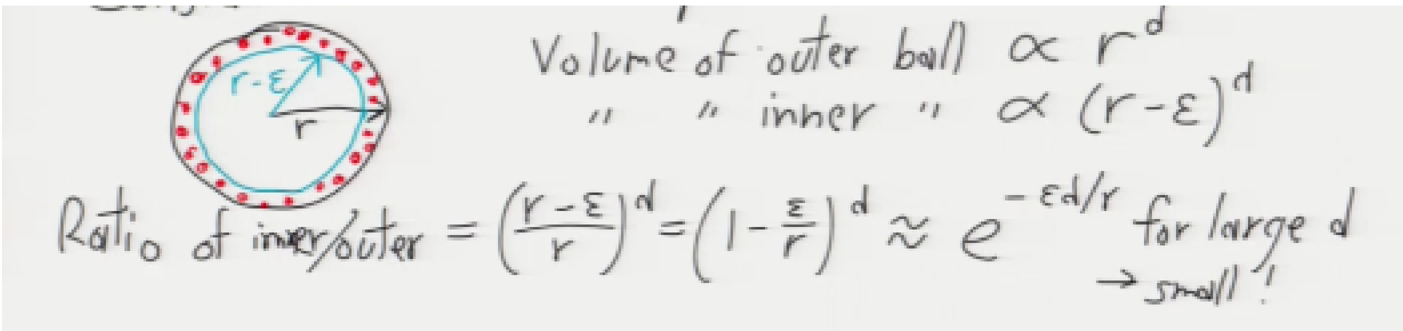
- Idea: Given query point v, find the k input points nearest v. Distance metric of your choice.
- Regression: Return average value of the k points.
- Classification: Return class with most votes from the k points or return histogram of class probabilities.
- Theorem (Cover and Hart, 1967): As $n \to \infty$, the 1-NN error rate is $< B(2 - B)$ where B=Bayes rate, if only 2 classes, $\leq 2B(1 - B)$.
- Theorem (Fix and Hodges, 1951): As $n \to \infty, k \to \infty, \frac{k}{n} \to 0$, k-NN error rate converges to B.

The Geometry of High-Dimensional Spaces

- Consider unit ball $B = \{p \in \mathbb{R}^d : |p| \le 1\}$ and hypercube $H = \{p \in \mathbb{R}^d : |p_i| \le 1\}$



- Consider a shell of the sphere



- e.g. if $\frac{\epsilon}{r} = 0.1$ and d=100, inner ball has 0.0027% of volume.
- Random points from (uniform|Gaussian) distribution in ball: nearly all are in outer shell.

Exhaustive k-NN algorithm

- Given query point v:
  - Scan through all n input points, computing (squared) distances to v.
  - Maintain max-heap with the k shortest distances seen so far.
- Time to construct the classifier: $\mathcal{O}$
- Query time: $\mathcal{O}(nd + n\log k)$ expected $\mathcal{O}(nd + k\log^2 k)$ if random point order.

Speeding up NN

- Can we preprocess the training points to obtain sub-linear query time?
- Very low dimensions: Voronoi diagrams.
- Medium dim (up to ~ 30): $k-d$ trees.
- Larger dim: locality sensitive hashing.
- Largest dim: no.
- Usually resort to <u>approximate</u> NN as d gets large.
- Can use PCA or other dimensionality reduction as preprocess.
- PCA: Row i of UD gives the coordinates of sample point X_i in principal components space (i.e. $X_i \cdot v_j$ for each j). So we don't need to project the input points onto that space; the SVD does it for us.