

Clustering ctd. & Feature selection

Lecture 9

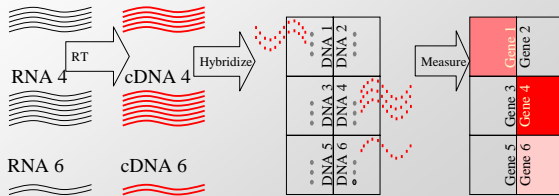
October 6, 2005

Plan

- Clustering – running times
- Feature selection, using Singular Value Decomposition

DNA MicroArrays

- To measure levels of messages in a cell
 - Construct an array with DNA sequences for multiple genes
 - Hybridize each RNA in your sample to a sequence in your array (All sequences from the same gene hybridize to the same spot)
 - Measure the number of hybridizations for each spot



Gene expression data

- For each gene j we have a vector

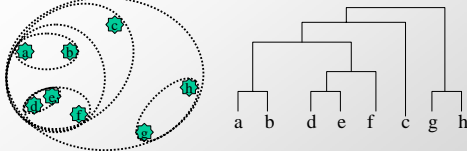
$$t_j = (t_{1j}, t_{2j}, \dots, t_{nj})$$

- What can we do with this data ?
- Cluster !

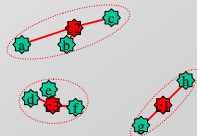
Images removed due to copyright restrictions.

2. Clustering Algorithms

- Hierarchical: Merge data successively to construct tree

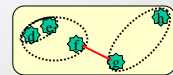


- Non-Hierarchical: place k -means to best explain data
 - Clusters: via closest center
 - Centers: cluster centroids

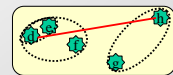


Distance between clusters

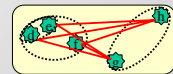
- $CD(X, Y) = \min_{x \in X, y \in Y} D(x, y)$
Single-link method



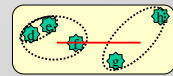
- $CD(X, Y) = \max_{x \in X, y \in Y} D(x, y)$
Complete-link method



- $CD(X, Y) = \text{avg}_{x \in X, y \in Y} D(x, y)$
Average-link method

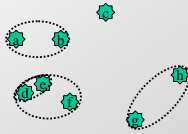


- $CD(X, Y) = D(\text{avg}(X), \text{avg}(Y))$
Centroid method



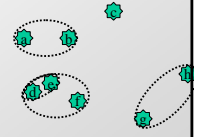
Running time: Hierarchical methods

- Repeat:
 - Choose the pair of **closest clusters**
 - Merge
- What is the running time ?
- Number of iterations:
 - Exactly $n-1$
- Iteration cost:
 - At most n^2 computations of $CD(,)$
 - How many point-point distance computations ?
 - At most n^2 as well !
- Total running time: $O(n^3)$



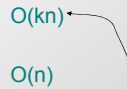
Improvements

- Single-link: $O(n^2 \log n)$ time
 - Sort all n^2 links by length in ascending order
 - For each link (a,b)
 - If a and b are in different clusters, merge them
- Analysis:
 - Sorting: $O(n^2 \log n)$
 - Maintaining the clusters: $O(\log n)$ per pair (a,b) ("Union-find" data structure)
- Even faster algorithm
 - Minimum Spanning Tree
 - Time: $O(n^2 + n \log n)$



Running time: k-means

- For each iteration:
 - For each point, find its closest center
 - For each cluster, compute its centroid
- Total time $O(kn)$
 - $\#iterations * O(kn)$
- Can improve by using fast nearest neighbor methods
 - Kd-tree (in low dimensions)
 - Locality-Sensitive Hashing/Random Projection (as in Lecture 4)
- Overall much faster than hierarchical methods (but suffers from initialization problems)

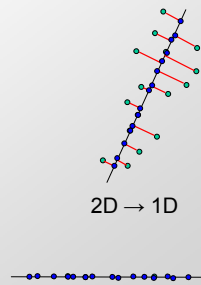


Visualizing clustering output

Images removed due to copyright restrictions.

Dimensionality reduction

- A.k.a.:
 - Hyper-plane fitting
 - SVD/PCA (techniques)
 - Feature selection
 - Latent Semantic Analysis (in text retrieval)
 - ...
- What it is:
 - Suppose you have a set of points in high (say, d) dimensions
 - You want to map this set to low (say, k) dimensions, in a way that preserves **structure** of the data set



Why would we do that ?

- Dimensionality reduction can **improve** the quality of the data
- Consider an 8-dimensional table
 - Column 1: signal
 - Columns 2...8: noise
- Noise can "eat" the signal !
- Under natural conditions, reducing dim to 1 will recover the first column!

1.2	0.1	0.04	0.02	0.06	0.03	0.08
0.9	-0.07	-0.05	-0.07	0.04	-0.03	0.05
0.4	0.04	0.05	0.1	-0.05	0.04	0.04
0.25	-0.05	-0.1	-0.07	0.06	0.1	-0.05
0.1	0.05	-0.03	0.04	-0.1	-0.07	0.05
-0.1	-0.1	0.02	-0.05	0.03	0.04	-0.1
-0.3	0.03	0.1	0.05	0.02	-0.05	0.03
-0.4	0.02	-0.07	-0.1	0.1	-0.1	0.02
-0.8	0.003	0.04	0.03	-0.07	0.03	0.01

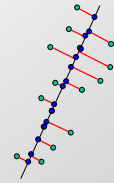
1.2
0.9
0.4
0.25
-0.1
-0.1
-0.3
-0.4
-0.8

Motivation, ctd.

- Some algorithms have time exponential in the dimension (“curse of dimensionality”)
 - Reducing dimension yields a big speed-up
- Visualization: one can see patterns in 2D
- ...

Dimensionality reduction

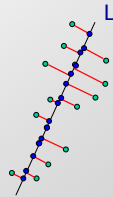
- We want to find the optimum projection
- Question:
 - What criterion we want to optimize (definition)
 - How to optimize it (algorithm)
- Ideas ?



Sum-of-squares Error

- Consider points P
- We will try to find a (hyper)-line L (of dim. k) which minimizes

$$\sum_{p \in P} \text{Dist}^2(p, L)$$
- How to solve it ?
 - k=0
 - i.e., L is a point:
 - Choose L to be the centroid of P (as per previous lecture)
 - k>0
 - Use Singular Value Decomposition

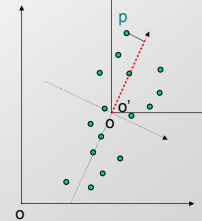


Singular Value Decomposition

- A new coordinate system
 - o' = centroid of P
 For simplicity, we assume $o = o'$
 - The 1st axis is a line L1 which maximizes

$$\sum_p p_{L1}^2$$
 Denote this max value by σ_1^2
 - The 2nd axis is a line L2, orthogonal to L1, which maximizes

$$\sum_p p_{L2}^2$$
 Denote this max value by σ_2^2
 - ...
- The values $\sigma_1, \sigma_2, \dots$ are called **singular values** of P

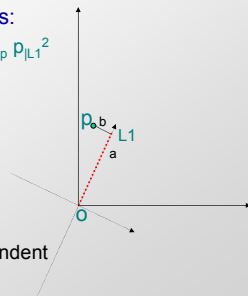


Singular Values vs. Line Fitting

- Algorithm for two dimensions:
 - Find L1 which maximizes $\sum_p p_{L1}^2$
- Why does it minimize

$$\sum_p \text{Dist}^2(p, L1) ?$$
- Argument:
 - Consider a point p
 - $\text{Dist}(p, L) = b$
 - $p_{L1} = a$
 - Also, $a^2 + b^2 = |op|^2$ is independent from L1
 - Thus

$$\sum_p \text{Dist}^2(p, L) = \sum_p |op|^2 - \sum_p p_{L1}^2$$



Computing SVD

- How to find L1 maximizing $\sum_p p_{L1}^2$
- Can be solved using an iterative algorithm
- Running time = $O(nd) * \text{\#iterations}$
- Total running time for SVD = $O(ndk) * \text{\#iterations}$

Example

- Holter NS, Mitra M, Maritan A, Cieplak M, Banavar JR, Fedoroff NV,
“Fundamental patterns underlying gene expression profiles: simplicity from complexity”,
Proc Natl Acad Sci U S A. 2000 Jul 18;97(15):8409-14.
 - Perform SVD to map gene expression data into 2D