CS 683: Advanced Algorithms	**SVD applications to clustering and**
information retrieval	
May 1st, 2001	**Scribe: Many**

1 Clustering problems and Singular Value Decomposition

In this lecture, we consider the following three clustering problems. Let $d(\cdot, \cdot)$ be the Euclidean distance function.

k-median: Given $a_1, \ldots, a_n \in \mathbb{R}^d$, find $c_1, \ldots, c_k \in \mathbb{R}^d$ to minimize

$$\sum_{1 \leq i \leq n} d(a_i, \{c_j\}).$$

k-means: Given $a_1, \ldots, a_n \in \mathbb{R}^d$, find $c_1, \ldots, c_k \in \mathbb{R}^d$ to minimize

$$\sqrt{\sum_{1 \leq i \leq n} d(a_i, \{c_j\})^2}.$$

k-dimensional subspace clustering: Given $a_1, \ldots, a_n \in \mathbb{R}^d$, find the k-dimensional subspace S to minimize

$$\sqrt{\sum_{1 \leq i \leq n} d(a_i, S)^2}.$$

1.1 Simple Clustering Algorithm

For simplicity, assume that $k = 2$ (Both k-median and k-means problems are still NP-hard for variable d). Next, we analyze an algorithm that outputs a solution within a factor of 2 for either problem. The algorithm is simple and it outputs two centers c_1, c_2 from the given point set: Try all pairs $\{a_i, a_j\}$ as centers and output the pair that achieves the smallest objective function value.

Let c_1 and c_2 be two points in $\mathbb{R}^d$ that achieves the optimal objective function value for the set $\{a_i\}$. Then, without loss of generality, assume that a_1 and a_2 are the closest points to c_1 and c_2, respectively, in the given point set. Suppose we are trying to solve the 2-median problem; a similar argument follows for 2-means. For a fixed a_i, assume, without loss of generality, that c_1 is closer to a_i than c_2. So, we have picked a_1, a_2 as our centers, a_i would have contributed to the objective function value

$$d(a_i, a_1) \leq d(a_i, c_1) + d(c_1, a_1) \leq 2d(a_i, c_1).$$

The first inequality follows from the triangle inequality, and the second one follows from the choice of a_1. Therefore, we get

$$\sum_i d(a_i, \{a_1, a_2\}) \leq 2 \sum_i d(a_i, \{c_1, c_2\}).$$

Since we choose the best $\{a_i, a_j\}$ pair as our centers, we get a solution which is no worse that what $\{a_1, a_2\}$ achieves. Therefore, the algorithm is a 2-approximation algorithm.

For the k-means problem, a similar argument gives

$$d(a_i, a_1)^2 \leq (d(a_i, c_1) + d(c_1, a_1))^2 \leq 4d(a_i, c_1)^2$$

and the square root in the objective function, yields a 2-approximation algorithm.

Note that the arguments above holds for any k, and we get a 2-approximation algorithm. The running time of the algorithm is $O(\binom{n}{k})$.

1.2 k-means clustering via SVD

Next we will get a $\sqrt{2}$-approximation algorithm for the k-means problem by reducing it to the k-dimensional subspace clustering problem and using Singular Value Decomposition to solve the latter. For simplicity assume $k = 2$ (The argument will generalize to $k = O(1)$ and give a polynomial time $\sqrt{2}$-approximation algorithm).

Find a k-dimensional subspace S that minimizes $\sum_i d(a_i, S)^2$ via SVD. Let a'_i be orthogonal projection of each point a_i to the subspace S. We have reduced the problem from d to k dimensions, where we can solve the problem optimally. Solve k-means by brute-force for $\{a'_i\}$. For example, for $k = 2$, we have a 2-dimensional subspace and want to find two centers q_1, q_2. For any position of the centers, the sets of points closer to q_1 than q_2 and vice versa are separated by a line. We thus try all possible all divisions of $\{a'_i\}$ by a line thorough two of the a_i's. In each part, the optimum solution is just 1-mean or centroid. Use the centroids ($\sum \frac{a_i}{|subset|}$) of these two subsets as a a candidate solution. We pick the best of all $\binom{n}{2}$ candidate solutions, say $\{q_1, q_2\}$. Note that the brute force algorithm can be generalized for any $k = O(1)$ with running time $O(n^{f(k)})$ for some function f.

Let c_1, c_2 be the centers in an optimal solution and S^* be the 2-dimensional (in general k-dimensional) subspace containing c_1 and c_2. Then,

$$\begin{aligned}
\sum_i d(a_i, a'_i)^2 &= \sum_i d(a_i, S)^2 \quad \text{by the definition of } a'_i \\
&\leq \sum_i d(a_i, S^*)^2 \quad \text{by the optimality of } S \\
&\leq \sum_i d(a_i, \{c_1, c_2\})^2 = \text{ optimal value OPT}
\end{aligned}$$

Let c'_1, c'_2 be the projections of c_1, c_2 onto S, respectively. We also know the following facts:

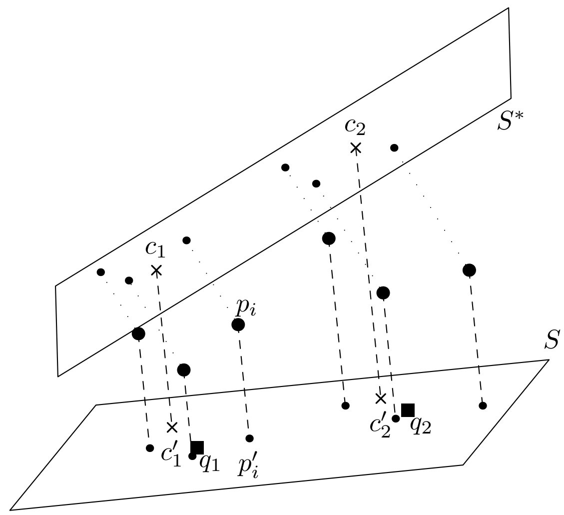

Figure 1: The optimal subspace S and subspace S^* containing c_1, c_2.

- For each $i \in \{1, \ldots, n\}, j \in \{1, 2\}$, $d(a_i, c_j)^2 \geq d(a'_i, c'_j)^2$.
- For each $i \in \{1, \ldots, n\}, j \in \{1, 2\}$, $d(a_i, c'_j)^2 = d(a_i, a'_i)^2 + d(a'_i, c'_j)^2$ by the Pythagoras' theorem applied to the projections on S.

By combining the facts from above, we get

$$\begin{aligned} 2OPT &\geq \sum_i d(a_i, a'_i)^2 + \sum_i d(a'_i, \{c'_1, c'_2\})^2 \\ &= \sum_i d(a_i, a'_i)^2 + d(a'_i, \{c'_1, c'_2\})^2 \\ &= \sum_i d(a_i, \{c'_1, c'_2\})^2 \\ &\geq \sum_i d(a_i, \{q_1, q_2\})^2 = \text{ our solution} \end{aligned}$$

Since the summations are within a factor of 2, the square root enclosing the objective function will yield a $\sqrt{2}$-approximation factor. [1]

[1]Note that in the algorithm we assumed that both S and S^* are subspaces, i.e. they contain the origin. If we allow them to be arbitrary hyperplanes, then we need S to be only $k-1$ dimensional hyperplane, as there is a $k-1$ dimensional hyperplane S^* containing all k centers of the optimal solution. However, we need a little trick in order to use SVD to find the optimal hyperplane S instead of just an optimal subspace.

2 Connections to Information Retrieval

2.1 Motivation

Recall the singular value decomposition:

$$A = U\Sigma V^T$$

In information retrieval, A is a term-document matrix. Rows represent terms and columns represent documents, with $A[i,j] \neq 0 \iff$ doc j contains term i. Which non-zero value to use is based on heuristics.

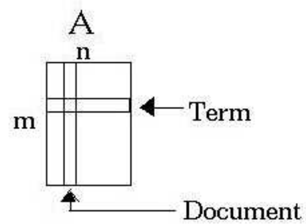

Figure 2: term-document matrix A

Queries are mapped to vectors in term space, $\mathbb{R}^m$, just like documents. Given a query, the goal is to find documents (columns of A) that are close to the query.

If we use this approach in the original term space, we will not necessarily get the results we want. In particular, there are two big problems:

synonomy: many different words can have the same meaning (e.g. car/auto)

polysemy: one word can have many meanings (e.g. bank, apple)

Ideally, we want a query about cars to be able to return a document about automobiles, and a query about bank robberies NOT to return a document about erosion of river banks. And although big companies like Microsoft have ongoing efforts to hand-code this kind of information, we would really like this to be handled automatically by our clustering algorithm.

Namely, we want terms to be considered similar because they show up in similar documents, and documents considered similar because they use similar terms. How to solve this "chicken and egg" problem? This is where the SVD comes in.

2.2 Application of SVD

The key idea is to "compress out" redundant term usage by working with an optimal rank-k matrix A_k instead of matrix A. $A_k = U_k \Sigma_k V_k^T$

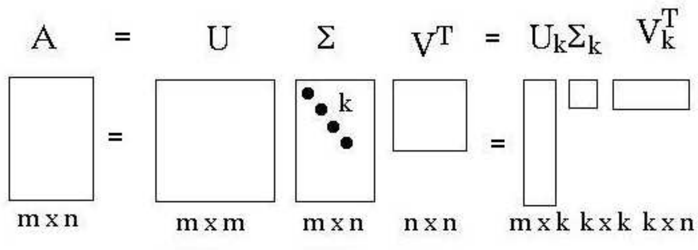

Figure 3: "Compression" of A

This gives us a "lossy compression" of matrix A, which may be useful for reducing the problems of synonomy and polysemy.

How to think of this matrix A_k?

The j^{th} column of A_k ("compressed" document j), is formed by multiplying the j^{th} column of $\Sigma_k V_k^T$ (which is in $\mathbb{R}^k$) by U_k. Hence the j^{th} column of $\Sigma_k V_k^T$ represents the k coordinates of document j according to the k axes given by the columns of U_k.

Each axis (a column of U_k, in $\mathbb{R}^m$) has a coordinate for each term $\Longrightarrow$ it is a "mixture" of terms.

Each axis aggregates terms the co-occur frequently ("clusters").

2.3 History

These SVD techniques have been used in statistics since the 1930's, and in information retrieval since the late 1980's.

Deerwester et al, in the late 1980's, first used this approach to "Latent Semantic Indexing". It is typically used to reduce 10K-100K dimensional problems to 100-200 dimensions. The "right" choice of low dimension to use is still a subject of research.