princeton univ. F'16 cos 521: Advanced Algorithm Design

Lecture 11: Intrinsic dimensionality of data and low-rank approximations: SVD

Lecturer: *Pravesh Kothari* Scribe:*Sanjeev Arora*

Today's topic is a technique called *singular value decomposition* or SVD. We'll take two views of it, and then encounter a surprising algorithm for it, which in turn leads to a third interesting view.

0.1 View 1: Inherent dimensionality of a dataset

In many settings we have a set of m vectors $v_1, v_2, \ldots, v_m$ in $\Re^n$. Think of n, m as large. We would like to represent v_i's using fewer number of dimensions, say k. We saw one technique in an earlier lecture, namely, Johnson-Lindenstrauss dimension reduction, which achieves $k = O(\log n/\epsilon^2)$. As explored in HW 3, JL-dimension reduction is relevant if we only care about preserving all pairwise ℓ_2 distances among the vectors. Its advantage is that it works for *all* datasets. But to many practitioners, that is also a huge disadvantage: since it is oblivious to the dataset, it cannot be tweaked to leverage properties of the data at hand.

Today we are interested in datasets where the v_i's do have a special structure: they are well-approximated by some low-dimensional set of vectors. By this we mean that for some small k, there are vectors $u_1, u_2, \ldots, u_k \in \Re^n$ such that every v_i is close to the *span* of $u_1, u_2, \ldots, u_k$. In many applications k is fairly small, even 3 or 4, and JL dimension reduction is of no use.

Let's attempt to formalize the problem at hand. We are looking for k-dimensional vectors $u_1, u_2, \ldots, u_k$ and mk coefficients $\alpha_{i1}, \ldots, \alpha_{ik} \in \Re$ such that $\left|v_i - \sum_j \alpha_{ij} u_j\right|_2^2 \approx$ small. But of course any real-life data set has *outliers,* for which this may not hold. But if most vectors fit the conjectured structure, then we expect

$$\sum_i \left|v_i - \sum_j \alpha_{ij} u_j\right|_2^2 \approx \text{small} \tag{1}$$

This problem is nonlinear and nonconvex as stated. Today we will try to understand it more and learn how to solve it. We will find that it is actually easy (which I find one of the miracles of math: one of few natural nonlinear problems that are solvable in polynomial time).

But first some examples of why this problem arises in practice.

Example 1 (Understanding shopping data) Suppose a marketer is trying to assess shopping habits. He observes the shopping behaviour of m shoppers with respect to n goods: how much of each good did they buy? This gives m vectors in $\Re^n$.

The simplest model for this would be: every shopper starts with a budget, and allocates it equally among all m items. Then if B_i is the budget of shopper i and p_j is the price for item j, the ith vector is $\frac{1}{n}(\frac{B_i}{p_1}, \frac{B_i}{p_2}, \ldots, \frac{B_i}{p_n})$. Denoting by $\vec{u}$ the vector of price inverses, namely, $(1/p_1, 1/p_2, \ldots, 1/p_n)$ this is just $\frac{B_i}{n}\vec{u}$. We conclude that the data is 1-dimensional: just scalar multiples of $\vec{u}$.

But maybe the above model is too unrealistic and doesn't fit the data well. Then one could try another model. We assume that the goods partition into k categories these could correspond to supermarket sections like produce, deli items, canned goods, etc., but it could also be some unknown category (e.g., all the items purchased by families with babies). Let $S_1, S_2, \ldots, S_k$ denote the k subsets of items corresponding to categories; these are unknown to us. Assume furthermore that the ith shopper designates a budget B_{it} for the tth category, and then divides this budget equally among goods in that category. Let $u_t \in \Re^n$ denotes the vector inf $\Re^n$ whose coordinate is 0 for goods not in S_t and the inverse price for goods in S_t. Then the quantities of each good purchased by shopper i are given by the vector $\sum_{t=1}^{k} \frac{B_{it}}{|S_t|} u_t$. In other words, this model predicts that the dataset is k-dimensional.

Of course, no model is exact so the data set will only be approximately k-dimensional, and thus the problem in (1) is a possible formulation.

One can consider alternative probabilistic models of data generation where the shopper picks items randomly from each category. You'll analyse that in the next homework.

Example 2 (Understanding microarray data in biology) The number of genes in your cell is rather large, and their activity levels —which depend both upon your genetic code and environmental factors— determine your body's functioning. *Microarrays* are tiny "chips"of chemicals sites that can screen the activity levels —also called *gene expression* levels—of a large number of genes in one go, say $n = 10,000$ genes. Typically these genes would have been chosen because they are suspected to be related to the phenomenon being studied, say a particular disease, immune reaction etc. After testing m individuals, one obtains m vectors in $\Re^n$.

In practice it is found that this gene expression data is low-dimensional in the sense of (1). This means that there are, say, 4 directions u_1, u_2, u_3, u_4 such that most of the vectors are close to their span. These new axis directions usually have biological meaning; eg they help identify genes whose expression (up or down) is controlled by common regulatory mechanisms.

0.2 View 2: Low rank matrix approximations

We have an $m \times n$ matrix M. We suspect it is actually a noisy version of a rank-k matrix, say $\tilde{M}$. We would like to find out $\tilde{M}$. One natural idea is to solve the following optimization problem

$$\min \sum_{ij} \left| M_{ij} - \tilde{M}_{ij} \right|^2 \qquad \text{s.t.} \quad \tilde{M} \text{ is a rank-}k \text{ matrix} \tag{2}$$

Again, seems like a hopeless nonlinear optimization problem. Peer a little harder and you realize that, first, a rank-k matrix is just one whose rows are linear combinations of k independent vectors, and second, if you let M_i denote the ith column of M then you are trying to solve nothing but problem (1)! (Recall that for a vector x, the value $|x|_2^2$ is just the sum of the squares of its coordinates.)

Example 3 (Planted bisection/Stochastic Block Model) Graph bisection is the problem where we are given a graph $G = (V, E)$ and wish to partition V into two equal sets $S, \overline{S}$ such that we minimize the number of edges between $S, \overline{S}$. It is NP-complete. Let's consider the following average case version.

Nature creates a random graph on n nodes as follows. It partitions nodes into S_1, S_2. Within S_1, S_2 it puts each edge with prob. p, and between S_1, S_2 put each edge with prob. q where $q < p$. Now this graph is given to the algorithm. Note that the algorithm doesn't know S_1, S_2. It has to find the optimum bisection. (A real-life explanation may be that the graph is a social network of people who have friended each other. Then S_1 could be "Princeton Alums"and S_2 could be "Yale Alums."Clearly, the probability of friending is higher within S_1, S_2 than between.)

It is possible to show using Chernoff bounds —as we will verify later—that if $q = \Omega(\frac{\log n}{n})$ then with high probability the optimum bisection in the graph is the planted one, namely, S_1, S_2. How can the algorithm recover this partition?

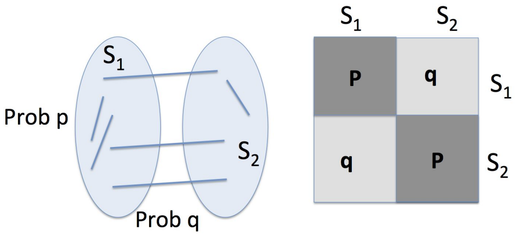

Figure 1: Planted Bisection problem: Edge probability is p within S_1, S_2 and q between S_1, S_2 where $q < p$. On the right hand side is the adjacency matrix. If we somehow knew S_1, S_2 and grouped the corresponding rows and columns together, and squint at the matrix from afar, we'd see more density of edges within S_1, S_2 and less density between S_1, S_2. Thus from a distance the adjacency matrix looks like a rank 2 matrix.

The observation in Figure 1 suggests that the adjacency matrix is close to a rank 2 matrix shown there: the block within S_1, S_2 have value p in each entry; the blocks between S_1, S_2 have q in each entry. (It is rank 2 since it has exactly two types of columns.)

Maybe if we can solve (2) with $k = 2$ we are done? This turns out to be correct as we will see in next lecture.

One can study planted versions of many other NP-hard problems as well.

Many practical problems involve graph partitioning. For instance, image recognition involves first partitioning the image into its component pieces (sky, ground, tree, etc.); a process called *image segmentation* in computer vision. This is done by graph partitioning on a graph defined on pixels where edges denote *pixel-pixel similarity*. Perhaps graphs with planted partitions are a better model for such real-life settings than worst-case graphs.

Example 4 (Semantic Word Embeddings) Low rank approximation of a matrix is also used in Natural Language Processing (NLP) to capture the "meaning"of words via a vector.

The idea is to take a large text corpus (eg Wikipedia) and calculate the statistics of word coccurences. Let N be the number of distinct words (in practice, this is around 10^5). Construct an $N \times N$ matrix M where M_{ij} is the number of times that word i and word j cooccur within a window of size 3 (say) in the corpus.

In practice it is found that M is well-approximated by a matrix of rank around 300. This allows one to compute a 300-dimensional vector u_i for the ith word such that $\sum_{ij} |M_{ij} - u_i \cdot u_j|^2$ is small. These u_i's are called *word embeddings*. They capture the meaning of the word, in the sense that similarity of words (as judged by humans) tends to correlate with the inner product between their vector representations. (See the paper on *Latent Semantic Analysis*.)

It has been discovered that the quality of the word embeddings improves if one takes the *square root* of each matrix entry, or the *logarithm*. Other more mysterious reweightings are also popular. A recent paper by Arora et al. tries to explain this phenomenon.

0.3 Singular Value Decomposition

Now we describe the tool that lets us solve the above problems.

For simplicity let's start with a symmetric matrix M. Suppose its eigenvalues are $\lambda_1, \ldots, \lambda_n$ in decreasing order by absolute value, and the corresponding eigenvectors (scaled to be unit vectors) are $e_1, e_2, \ldots, e_n$. (These are column vectors.) Then M has the following alternative representation.

THEOREM 1 (SPECTRAL DECOMPOSITION)
$M = \sum_i \lambda_i e_i e_i^T$.

PROOF: At first sight, the equality does not even seem to pass a "typecheck"; a matrix on the left and vectors on the right. But then we realize that $e_i e_i^T$ is actually an $n \times n$ matrix (it has rank 1 since every column is a multiple of e_i). So the right hand side is indeed a matrix. Let us call it B.

Any matrix can be specified completely by describing how it acts on an orthonormal basis. By definition, M is the matrix that acts as follows on the orthonormal set $\{e_1, e_2, \ldots, e_n\}$: $Me_j = \lambda_j e_j$. How does B act on this orthonormal set? We have

$$\begin{aligned} Be_j &= (\sum_i \lambda_i e_i e_i^T) e_j \\ &= \sum_i \lambda_i e_i (e_i^T e_j) \qquad \text{(distributivity and associativity of matrix multiplication)} \\ &= \lambda_j e_j \end{aligned}$$

since $e_i^T e_j =< e_i, e_j >$ is 1 if $i = j$ and 0 else. We conclude that $B = M$. $\Box$

THEOREM 2 (BEST RANK k APPROXIMATION)
The solution $\tilde{M}$ to (2) is simply the sum of the first k terms in the previous Theorem.

The proof of this theorem uses the following.

Theorem 3 (Courant-Fisher)
If $e_1, e_2, \ldots, e_n$ are the eigenvectors as above then:

1. *e_1 is the unit vector that maximizes $|Mx|_2^2$.*
2. *e_{i+1} is the unit vector that is orthogonal to $e_1, e_2, \ldots, e_i$ and maximizes $|Mx|_2^2$.*

Proof:(Sketch) Let's prove the first statement. A general unit vector x can be expressed as a combination of eigenvectors as $\sum_i \alpha_i e_i$ where the coefficients satisfy $\sum_i \alpha_i^2 = 1$ since x is a unit vector. Then $Mx = \sum_i \lambda_i \alpha_i e_i$ by definition of eigenvalues. Thus $|Mx|_2^2 = \sum_i \lambda_i^2 \alpha_i^2$, which is maximised if $\alpha_1 = 1$ and the other $\alpha_i = 0$.

The second statement follows similarly. □

Let's prove Theorem 2 for $k = 1$ by verifying that the first term of the spectral decomposition gives the best rank 1 approximation to M. A rank 1 matrix is one whose each row is a multiple of some unit vector x; in other words is on the line defined by x. Denote the rows of M as $M_1, M_2, \ldots, M_n$. Then the multiple of x that is closest to M_i is simply its projection, namely $< M_i, x > x$. Thus the matrix approximation consists of finding a unit vector x so as to minimize

$$\sum_i |M_i - < M_i, x > x|^2 = \sum_i |M_i|^2 - \sum_i |< M_i, x >|^2 .$$

This minimization is tantamount to maximising

$$\sum_i |< M_i, x >|^2 = |Mx|^2 , \tag{3}$$

which by the Courant-Fisher theorem happens for $x = e_1$. Thus the best rank 1 approximation to M is the matrix whose ith row is $< M_i, e_1 > e_1^T$, which of course is $\lambda_1 e_{1i} e_1^T$. Thus the rank 1 matrix approximation is $\lambda_1 e_1^T e_1$, which proves the theorem for $k = 1$. The proof of Theorem 2 for general k follows similarly by induction and is left as exercise.

0.3.1 General matrices: Singular values

Now we look at general matrices that are not symmetric. The notion of eigenvalues and eigenvectors have to be modified. The following theorem is proved similarly as in the symmetric case but with a bit more tedium.

Theorem 4 (Singular Value Decomposition and best rank-k-approximation)
Every $m \times n$ real matrix has $t \leq \min\{m, n\}$ nonnegative real numbers $\sigma_1, \sigma_2, \ldots, \sigma_t$ (called singular values) and two sets of unit vectors $U = \{u_1, u_2, \ldots, u_t\}$ which are in $\Re^m$ and $V = v_1, v_2, \ldots, v_t \in \Re^n$ (all vectors are column vectors) where U, V are orthogonormal sets and

$$u_i^T M = \sigma_i v_i \qquad \text{and} \qquad M v_i = \sigma_i u_i^T \tag{4}$$

Furthermore, M can be represented as

$$M = \sum_i \sigma_i u_i v_i^T . \tag{5}$$

The best rank k approximation to M consists of taking the first k terms of (5) and discarding the rest.

This solves problems (1) and (2). Next time we'll go into some detail of the algorithm for computing them. In practice you can just use matlab or another package.

0.4 View 3: Directions of Maximum Variance

The above proof of Theorem 2, especially the subcase $k = 1$ we proved, also shows yet another view of SVD that is sometimes useful in data analysis: the eigenvectors/singular vectors are directions where the data has maximum variance.

Let us phrase this formally for symmetric matrices. Suppose we shift the given points $M_1, M_2, \ldots, M_n$ so that their mean $\frac{1}{n}\sum_i M_i$ is the origin. Then the claim is that the first eigenvector corresponds to the direction x where the projections of the given data points —a sequence of n real numbers— have maximum *variance*. This variance is $\sum_i (M_i \cdot x)^2 - \frac{1}{n}(\sum_i M_i \cdot x)^2$, but the second term is 0 since the mean is 0. Thus the variance is exactly the quantity in (3). The second SVD direction corresponds to directions with maximum variance after we have removed the component along the first direction, and so on.

Thus eigenvectors can be viewed as directions in space that maximally "explain"the variations in the data.

BIBLIOGRAPHY

1. O. Alter, P. Brown, and D. Botstein. Singular value decomposition for genome-wide expression data processing and modeling. *PNAS* August 29, 2000 vol. 97 no. 18

2. S. Arora, Y. Li, Y. Liang, T. Ma, A. Risteski. RAND-WALK: A Latent Variable Model Approach to Word Embeddings. (arxiv.org, 2015)

3. T.K. Landauer, P.W. Foltz, and D. Laham. Introduction to Latent Semantic Analysis. *Discourse Processes,* 25, 259-284. (1998)

4. Relevant chapter of Hopcroft-Kannan book on data science. (link on course website)