Note: These lecture notes are closely based on lecture notes by Sanjeev Arora [1].

1 View 1: "Best projection for a dataset"

Assume we have a set of m vectors, $v_1, v_2, \ldots v_m$ in $\mathbb{R}^n$, where m and n are both large. As before, we want to represent these vectors using fewer dimensions, say k.

We've seen that Johnson-Lindenstrauss can be very useful if we care about preserving pairwise ℓ_2 distances among the vectors. JL works for any data set - in fact, it doesn't even pay attention to the data set when determining the subspace that we project onto. But this is potentially a disadvantage, since there may be many data sets where there is a natural low dimensional subspace inherent in the dat.

Our goal in this lecture is to find a low-dimensional subspace that is "inherent" to the data set. More formally, this means that we want to find a small set of vectors $u_1, u_2, \ldots u_k \in \mathbb{R}^n$ such that every v_i is close to the *span* of $u_1, u_2, \ldots u_k$. Recall that a vector v is in the span of $u_1, u_2, \ldots u_k$ if $v = \sum_{i=1}^{k} \alpha_i u_i$ for some set of real numbers $\alpha_1, \ldots \alpha_k$. For many data sets, we can find very small k since the data set has intrinsically small dimensionality.

Let's formalize the problem a bit more. Mathematically, we want to find a set of vectors $u_1, u_2, \ldots u_k$ (the basis of the low dimensional space), such that every vector in our data set is "close" to the span of these k vectors. It makes sense for "close" to be defined in terms of the Euclidean distance (i.e. 2-norm) between each vector v_i and the span of $u_1, u_2, \ldots u_k$. In particular, we want to minimize $|v_i - \sum_j \alpha_{i,j} u_j|^2$, where the $\alpha_{i,j}$ are any real numbers.

Then, our optimization problem over all vectors is to find $u_1, u_2, \ldots u_k \in \mathbb{R}n$ and $\alpha_{i,j}$ for $1 \leq i \leq m$, $1 \leq j \leq k$ that minimizes:

$$\sum_{i=1}^{n} \left| v_i - \sum_{j=1}^{k} \alpha_{i,j} u_j \right|^2 \tag{1}$$

This optimization problem is non-linear and nonconvex. It's amazing that it can actually be solved. But it can, using the tool of *singular value decomposition*. The term singular value decomposition (also principle component analysis, spectral decomposition, etc.) is just a fancy word for computing all the eigenvectors and eigenvalues of the matrix. Before we talk more about how this works mathematically, some example applications.

1.1 Big Data

Suppose a marketer wants to understand behavior of m shoppers with respect to n goods. A very simple model of shopping behavior is that each shopper has a budget B_i and they allocate it equally over the m items. Then if p_j is the price of the j-th item, the vector associated with shopper i is:

$$\left(\frac{B_i}{p_1}, \frac{B_i}{p_2}, \ldots \frac{B_i}{p_m} \right)$$

So what dimensionality is the data in this model? The data in this model has dimension 1.

But maybe the above model is too simplistic. A more realistic model might be the following. We assume the goods are partitioned into k (unknown) categories like: produce, deli, items purchased by families with babies, etc. Let $S_1, S_2, \ldots S_k$ denote the k subsets of items in these categories. They are unknown to us and may overlap. Now assume that the i-th shopper designates a budget B_{ij} for the j-th category and then divides that budget equally among all items in that category.

How does this model work? For $1 \leq j \leq k$, let $u_j \in \mathbb{R}^n$ be the vector, where for $i \neq S_t$ $u_j[i] = 0$ and for $i \in S_t$, $u_j[i] = 1/p_j$. Then the vector for each shopper i that gives the quantity of each good purchased by that shopper is:

$$\sum_{j=1}^{k} \frac{B_i j}{|S_j|} u_j$$

What dimensionality is this model? The dimensionality of this model is k. Note that no model will exactly match the data so the data set will only be "approximately" low-dimensional.

Consider 3 items and let $v_1 = (1/3, 1/3, 1/3)$ and $v_2 = (1/2, 0, 1/2)$. Every customer is now a combination of these 2 vectors, so every customer is defined by exactly 2 coordinates so $span(v_1, v_2)$ is a 2-D space.

1.2 Microarray Data in Biology

The human body has a huge number of genes, and the activity level of these genes depends both on your genetic code and on environmental factors. *Microarrays* are small "chips" of chemical sites that can measure the gene expression level of a large number of genes (say $n = 10,000$) in one experiment. After testing m individuals with a microarray, we get a m by n matrix of real values.

In practice, these matrices turn out to be "low-dimensional" in the sense of equation 1. For example, there may tend to be three common "directions" such that most of the columns are close to the span of these three directions. These new axis directions usually have some biological meaning: for example sets of genes whose is expression is controlled by the same regulatory mechanism (i.e. the same "regulatory molecule").

2 View 2: Low-rank Approximation

We have a m by n matrix M and suspect it is a noisy version of a rank k matrix, say $\tilde{M}$. Given M, we want to find $\tilde{M}$. A natural optimization problem then is to find a rank k matrix $\tilde{M}$ that minimizes:

$$\sum_{1 \leq i \leq m, 1 \leq j \leq n} \left(M[i,j] - \tilde{M}[i,j] \right)^2 \tag{2}$$

Again this seems like a hopelessly complicated optimization problem. But taking a closer look, you can tell that a rank k matrix is just one whose columns are just linear combinations of k independent vectors. Then if you let v_i be the i-th column of M, this problem is equivalent to solving Equation 1. (Recall that for vector x, $|x|^2$ is just the sum of the squares of its coordinates.

Now some applications of this viewpoint.

2.1 Data Compression

Note that storing the matrix M requires storing mn real values. In contrast, storing the matrix $\tilde{M}$ requires first storing km real values for the k new basis vectors. Then we need to k real values for each vector of M, for a total of an additional kn real values. Thus, the total storage cost is $kn + km$ which may be *much* smaller than mn. For example, if m and n are in the millions and $k = 10$.

Thus, singular value decomposition gives a handy way to compress structured data with minimum loss.

2.2 Semantic Word Embeddings

Low rank approximations of data can also be used in natural language processing (NLP) to capture the semantic meaning of words via a vector.

For example, in a large test corpus (say Wikipedia) let N be the number of distinct words (usually on the order of 10^5) and construct a N by N matrix M where $M_{i,j}$ is the number of times that word i and word j cooccur within a window of size 3 (say).

In practice, M can be very well-approximated by a matrix of rank 300. This means there is a 300 dimensional space, such that, for every word i, we can compute a vector u_i in this space that is pretty close to the true cooccurrence vector for the i-th word. These u_i vectors are called *word embeddings*, and they capture the semantic meaning of words. Also the linguistic similarity between two words tends to correlate with the inner product of these u_i vectors.

3 Eigenvectors to the Rescue

Now we describe the incredible tool that lets us solve all of the above problems. For simplicity, let's start with a symmetric n by n matrix M. Suppose its eigenvalues are $\lambda_1, \lambda_2, \ldots \lambda_n$ in decreasing order of absolute value, and the corresponding eigenvectors (scaled to be unit vectors) are $e_1, e_2, \ldots, e_n$. Recall that e_i is an eigenvector, with eigenvalue λ_i for matrix M iff $Me_i = \lambda_i e_i$.

Then, M has the following alternative representation.

Theorem 1. *(Spectral Decomposition) Consider any symmetric matrix M with eigenvalues $\lambda_1, \ldots \lambda_n$ and eigenvectors $e_1, \ldots e_n$. Then the following holds:*

$$M = \sum_{i=1}^{n} \lambda_i e_i e_i^T$$

Proof: Note that $e_i e_i^T$ is actually a n by n, rank 1 matrix (it's rank 1 since each column is a multiple of e_i). Let $B = \sum_{i=1}^{n} \lambda_i e_i e_i^T$. Note that any matrix is completely specified by how it acts on an orthonormal basis. This is because $M(x+y) = Mx + My$ for any vectors x and y.

By definition, M is the matrix that acts as follows on the set $e_1, e_2, \ldots e_n$: $Me_j = \lambda_j e_j$.

How does B act on this orthonormal set? For any $1 \leq j \leq n$, we have:

$$\begin{aligned} Be_j &= \left(\sum_{i=1}^{n} \lambda_i e_i e_i^T \right) e_j \\ &= (\lambda_j e_j (e_j^T e_j)) \\ &= \lambda_j e_j \end{aligned}$$

This proves that B behaves the same way as M on an orthonormal basis. Hence $B = M$. □

3.1 Best Rank k approximation

Theorem 2. *(Courant-Fisher) If $e_1, \ldots, e_n$ are the eigenvectors of M, then*

1. *e_1 is the unit vector that maximizes $|Mx|^2$ among all unit vectors x*
2. *e_{i+1} is the unit vector that is orthogonal to $e_1, \ldots e_i$ that maximizes $|Mx|^2$, among all unit vectors x orthogonal to $e_1, \ldots e_i$.*

Proof: We will prove the first statement. Note that any arbitrary unit vector x can be represented as a combination of eigenvectors: $x = \sum_{i=1}^{n} \alpha_i e_i$, where $\sum_{i=1}^{n} \alpha_i^2 = 1$.

Now, what is Mx???

$$\begin{aligned} Mx &= M\left(\sum_{i=1}^{n} \alpha_i e_i\right) \\ &= \sum_{i=1}^{n} M\alpha_i e_i \\ &= \sum_{i=1}^{n} \lambda_i \alpha_i e_i \end{aligned}$$

Thus we have,

$$\begin{aligned} |Mx|^2 &= \left|\sum_{i=1}^{n} \lambda_i \alpha_i e_i\right|^2 \\ &= \sum_{i=1}^{n} \lambda_i^2 \alpha_i^2 \end{aligned}$$

Note that to maximize this last sum subject to the constraint that $\sum_i \alpha_i^2 = 1$ requires setting $\alpha_1 = 1$. This is true since $\lambda_1 \geq \lambda_i$ for all $i > 1$. □

Given a symmetric matrix M, with eigenvalues $\lambda_1, \ldots \lambda_n$ in decreasing order of absolute value, and eigenvectors $e_1, \ldots e_n$, for any $k, 1 \leq k \leq n$, let

$$M_k = \sum_{i=1}^{k} \lambda_i e_i e_i^T$$

3.2 Digression

Note that one can have a rank 1 matrix where all entries are non-zero. For example if $e_1 = (1/2, 1/2, 1/2, 1/2)$. Assume that $\lambda_1 = 1$. Then the matrix $M_1 = \lambda_1 e_1 e_1^T$ equals

$$\begin{pmatrix} 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \end{pmatrix}$$

Now consider the case where we have a matrix like the following:

$$\begin{pmatrix} 1/4+\epsilon & 1/4-\epsilon & -1/4+\epsilon & -1/4-\epsilon \\ 1/4-\epsilon & 1/4+\epsilon & -1/4-\epsilon & -1/4+\epsilon \\ -1/4+\epsilon & -1/4-\epsilon & 1/4+\epsilon & 1/4-\epsilon \\ -1/4-\epsilon & -1/4+\epsilon & 1/4-\epsilon & 1/4+\epsilon \end{pmatrix}$$

Note that this matrix is "well-represented" by $M_1 = e_1 e_1^T$, where $e_1 = (1/2, 1/2, -1/2, -1/2)$ even though it is not a rank 1 matrix. But to capture it exactly, we'd need another matrix $M_2 = \lambda_2 e_2 e_2^T$, where $e_2 = (1/2, -1/2, 1/2, -1/2)$. This matrix M_2 would be:

$$\begin{pmatrix} +\epsilon & -\epsilon & +\epsilon & -\epsilon \\ -\epsilon & +\epsilon & -\epsilon & +\epsilon \\ +\epsilon & -\epsilon & +\epsilon & -\epsilon \\ -\epsilon & +\epsilon & -\epsilon & +\epsilon \end{pmatrix}$$

3.3 Back to Proof

Theorem 3. *Among all rank k matrices, $\tilde{M} = M_k$ is the matrix that minimizes:*

$$\sum_{1 \le i \le m, 1 \le j \le n} \left| M[i,j] - \tilde{M}[i,j] \right|^2$$

Proof: We will prove the theorem for $k = 1$, the remaining proof will follow by an induction and is left as an exercise.

We want to prove that M_1 is the best rank 1 approximation to M. A rank 1 matrix is one where each row is a multiple of some unit vector x. Denote the rows of M by $M[1], M[2], \ldots M[n]$. Then the multiple of x that is closest to row $M[i]$ is just the projection $(M[i] \cdot x)x$. Thus, the appoximation comes down to finding a unit vector x that minimizes:

$$\sum_{i=1}^{n} \left| M[i] - (M[i] \cdot x)x \right|^2 = \sum_{i=1}^{n} |M[i]|^2 - \sum_{i=1}^{n} |M[i] \cdot x|^2$$

To see the above equality, note that the square of the norm of the vector $M[i]$ after the projection onto x is subtracted out is just the squared of the norm of $M[i]$ minus the square of the norm of the projection. This holds by the Pythagorean theorem!

Can we simplify the above optimization problem in terms of x? Yes note that minimizes the above expression is the same as maximizing:

$$\sum_{i=1}^{n} |M[i] \cdot x|^2 = |Mx|^2$$

By the Courant-Fisher theorem, $|Mx|^2$ is maximized when $x = e_1$. □

4 Singular Vectors

Consider general matrices that are not necessarily symmetric. Note that these matrices may not have eigenvectors in the tradition sense that $Mx = \lambda x$ for some vector x. For example, consider the following matrix M: $\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$

Note that M is not symmetric. Moreover, there is no nonzero vector x such that $Mx = \lambda x$ for some positive value λ. (Try to find one!).

Luckily, much of what we've said previously can still generalize to non-symmetric matrices. In particular, all matrices have what are called left and right *singular vectors* [1]. The idea is that these

[1] These are sometimes also called *left and right eigenvectors*

singular vectors occur in pairs. For example, the matrix M has a pair that is right singular vector $v_1 = (0, 1)$ and left singular vector $u_1 = (1, 0)$, along with singular value $\sigma_1 = 1$. Note that we can then say that $Mv_1 = \sigma_1 u_1$. No, since the matrix has rank!

All of the theorems in this lecture can be generalized to all matrices using singular vectors and singular values. For example, the following theorem can be proven similarly to the above proofs.

Theorem 4. *(Singular Value Decomposition). Every m by n matrix M has $t \leq \min(m, n)$ non-negative singular values $\sigma_1, \ldots, \sigma_t$, and two sets of non-zero unit vectors $U = \{u_1, \ldots u_t\}$ all in $\mathbb{R}^m$ and $V = \{v_1, \ldots v_t\}$ all in $\mathbb{R}^n$, where U, V are orthonormal sets. Moreover we have:*

$$u_i^T M = \sigma_i v_i^T \text{ and } Mv_i = \sigma_i u_i$$

Furthermore, if we let

$$M_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

then M_k is the best rank k approximation to M in the sense that it minimizes:

$$\sum_{1 \leq i \leq m, 1 \leq j \leq n} \left| M[i, j] - M_k[i, j] \right|^2$$

5 Computing the SVD

So how can we compute the singular value decomposition? Idea 1: Import your matrix, M, into Matlab and type $svd(M)$. But how does Matlab do this computation?

One of the simplest algorithms is called the power iteration method. It is based on our proof of the Courant-Fisher theorem. For a symmetric matrix, you start with some random unit vector x and just keep multiplying your matrix, M by x until the vector converges to the top eigenvector. Recall that we can write $x = \sum_{i=1}^{n} \alpha_i e_i$ where e_i is the i-th eigenvector of M. Thus, if we do k iterations of matrix multiplication by x, we get

$$\begin{aligned} M^k x &= M^k \sum_{i=1}^{n} \alpha_i e_i \\ &= \sum_{i=1}^{n} \alpha_i \lambda_i^k e_i \end{aligned}$$

Note that in this last equality, as k grows large, the value λ_1^k will grow much larger than λ_i^k for any $i > 1$. Hence, the output vector will be "mostly" in the direction of e_1. Thus, the normalized output vector will converge quickly to e_1.

In particular, the rate of convergence will be determined by the ratio λ_2/λ_1 which is know as the *spectral gap*.

Once we've found e_1 in this way, we can just project out the direction of e_1 from each column of M and repeat the process to get $e_2, e_3, \ldots e_n$.

Finding singular vectors follows a similar process. It can be done by multiplying row and column vectors back and forth across the matrix. More simply, it can be done by noting that all the right singular vectors of M are just the eigenvectors of the symmetric matrix $M^T M$. Then, we just follow the power iteration method for finding eigenvectors above.

6 View 3: Directions of Maximum Variance

Another view of SVD is that the eigen/singular vectors are directions where the data has maximum variance.

For example, consider a symmetric matrix M. Imagine shifting your points/ rows of the matrix $M[1], M[2], \ldots, M[n]$ so that their mean is the origin. In particular, $\frac{1}{n}\sum_{i=1}^{n} M[i]$ is the the origin.

Then we claim that the first eigenvector represents the direction x where the projections of the data points have maximum variance. To see this, note that the variance of the projections of the rows onto x is:

$$\sum_{i=1}^{n}\left(M[i]\cdot x - \left(\frac{1}{n}\sum_{i=1}^{n} M[i]\right)\right)^2 = \sum_{i=1}^{n}(M[i]\cdot x)^2$$

where the equality holds because we have explicitly set $\frac{1}{n}\sum_{i=1}^{n} M[i]$ to the origin.

Thus, the variance is exactly $|Mx|^2$, which we proved (Courant-Fisher) is maximized when $x = e_1$. The second eigenvector corresponds to direction with maximum variance after removing this first component and so forth.

Thus, the eigenvectors can be viewed as directions that best explain the variation in the data.

References

[1] Sanjeev Arora. Advanced Algorithm Design Class, Princeton University, 2013. https://www.cs.princeton.edu/courses/archive/fall15/cos521/.