CHAPTER 1 SINGULAR VALUE DECOMPOSITION (SVD)

The Singular Value Decomposition (SVD) is one of the most important concepts in applied mathematics. It is used for a number of application including dimension reduction and data analysis. Principal Components Analysis (PCA) is a special case of the SVD. Let's start with the formal definition, and then see how PCA relates to that definition.

Definition 1.0.1: Singular Value Decomposition

For any $m \times n$ matrix $\mathbf{A}$ with $rank(\mathbf{A}) = r$, there are orthogonal matrices $\mathbf{U}_{m\times m}$ and $\mathbf{V}_{n\times n}$ and a diagonal matrix $\mathbf{D}_{r\times r} = diag(\sigma_1, \sigma_2, \dots, \sigma_r)$ such that

$$\mathbf{A} = \mathbf{U} \underbrace{\begin{pmatrix} \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}}_{m\times n} \mathbf{V}^T \quad \text{with} \quad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0 \qquad (1.1)$$

The σ_i's are called the nonzero **singular values** of $\mathbf{A}$. (When $r < p = \min\{m, n\}$ (i.e. when $\mathbf{A}$ is not full-rank), $\mathbf{A}$ is said to have an additional $p - r$ zero singular values). This factorization is called a **singular value decomposition** of $\mathbf{A}$, and the columns of $\mathbf{U}$ and $\mathbf{V}$ are called the left- and right-hand **singular vectors** for $\mathbf{A}$, respectively.

Properties of the SVD

- The left-hand singular vectors are a set of orthonormal eigenvectors for $\mathbf{A}\mathbf{A}^T$.
- The right-hand singular vectors are a set of orthonormal eigenvectors for $\mathbf{A}^T\mathbf{A}$.

- The singular values are the square roots of the eigenvalues for $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}\mathbf{A}^T$, as these matrices have the same eigenvalues.

When we studied PCA, one of the goals was to find the new coordinates, or *scores*, of the data in the principal components basis. If our original (centered or standardized) data was contained in the matrix $\mathbf{X}$ and the eigenvectors of the covariance/correlation matrix ($\mathbf{X}^T\mathbf{X}$) were columns of a matrix $\mathbf{V}$, then to find the scores (call these $\mathbf{S}$) of the observations on the eigenvectors we used the following equation:

$$\mathbf{X} = \mathbf{S}\mathbf{V}^T.$$

This equation mimics Equation **??** because the matrix $\mathbf{V}^T$ in Equation **??** is also a matrix of eigenvectors for $\mathbf{A}^T\mathbf{A}$. This means that the principal component scores $\mathbf{S}$ are a set of unit eigenvectors for $\mathbf{A}\mathbf{A}^T$ scaled by the singular values in $\mathbf{D}$:

$$\mathbf{S} = \mathbf{U}\begin{pmatrix} \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}.$$

1.1 Resolving a Matrix into Components

One of the primary goals of the singular value decomposition is to resolve the data in $\mathbf{A}$ into r mutually orthogonal components by writing the matrix factorization as a sum of outer products using the corresponding columns of $\mathbf{U}$ and rows of $\mathbf{V}^T$:

$$\mathbf{A} = \mathbf{U}\begin{pmatrix} \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}\mathbf{V}^T = \begin{pmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_m \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 & \dots & 0 & 0 \\ 0 & \ddots & 0 & \vdots & 0 \\ \vdots & 0 & \sigma_r & 0 & \vdots \\ 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_n^T \end{pmatrix}$$

$$= \sigma_1\mathbf{u}_1\mathbf{v}_1^T + \sigma_2\mathbf{u}_2\mathbf{v}_2^T + \dots + \sigma_r\mathbf{u}_r\mathbf{v}_r^T.$$

$$\sigma_1 \geq \sigma_2 \geq \dots \sigma_r$$

For simplicity, let $\mathbf{Z}_i = \mathbf{u}_i\mathbf{v}_i^T$ act as basis matrices for this expansion, so we have

$$\mathbf{A} = \sum_{i=1}^{r} \sigma_i \mathbf{Z}_i. \tag{1.2}$$

This representation can be regarded as a Fourier expansion. The coefficient (singular value) σ_i can be interpreted as the proportion of $\mathbf{A}$ lying in the "direction" of $\mathbf{Z}_i$. When σ_i is small, omitting that term from the expansion will cause only a small amount of the information in $\mathbf{A}$ to be lost. This fact has important consequences for compression and noise reduction.

1.1.1 Data Compression

We've already seen how PCA can be used to reduce the dimensions of our data while keeping the most amount of variance. The way this is done is by simply ignoring those components for which the proportion of variance is small. Supposing we keep k principal components, this amounts to truncating the sum in Equation **??** after k terms:

$$\mathbf{A} \approx \sum_{i=1}^{k} \sigma_i \mathbf{Z}_i. \tag{1.3}$$

As it turns out, this truncation has important consequences in many applications. One example is that of image compression. An image is simply an array of pixels. Supposing the image size is m pixels tall by n pixels wide, we can capture this information in an $m \times n$ matrix if the image is in grayscale, or an $m \times 3n$ matrix for a [r,g,b] color image (we'd need 3 values for each pixel to recreate the pixel's color). These matrices can get very large (a 6 megapixel photo is 6 million pixels).

Rather than store the entire matrix, we can store an approximation to the matrix using only a few (well, more than a *few*) singular values and singular vectors.

This is the basis of image compression. An approximated photo will not be as crisp as the original - some information will be lost - but most of the time we can store much less than the original matrix and still get a good depiction of the image.

1.1.2 Noise Reduction

Many applications arise where the relevant information contained in a matrix is contaminated by a certain level of noise. This is particularly common with video and audio signals, but also arises in text data and other types of (usually high dimensional) data. The **truncated SVD** (Equation **??**) can actually reduce the amount of noise in data and increase the overall **signal-to-noise** ratio under certain conditions.

Let's suppose, for instance, that our matrix $\mathbf{A}_{m\times n}$ contains data which is contaminated by noise. If that noise is assumed to be random (or nondirectional) in the sense that the noise is distributed more or less uniformly across the components $\mathbf{Z}_i$, then there is just as much noise "in the direction" of one $\mathbf{Z}_i$ as there is in the other. If the amount of noise along each direction is approximately the same, and the σ_i's tell us how much (relevant) information in $\mathbf{A}$

is directed along each component $\mathbf{Z}_i$, then it must be that the ratio of "signal" (relevant information) to noise is decreasing across the ordered components, since

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$$

implies that the signal is greater in earlier components. So letting $SNR(\sigma_i \mathbf{Z}_i)$ denote the signal-to-noise ratio of each component, we have

$$SNR(\sigma_1 \mathbf{Z}_1) \geq SNR(\sigma_2 \mathbf{Z}_2) \geq \cdots \geq SNR(\sigma_r \mathbf{Z}_r)$$

This explains why the **truncated SVD**,

$$\mathbf{A} \approx \sum_{i=1}^{k} \sigma_i \mathbf{Z}_i \quad \text{where} \quad k < r$$

can, in many scenarios, filter out some of the noise without losing much of the significant information in $\mathbf{A}$.

1.1.3 Latent Semantic Indexing

Text mining is another area where the SVD is used heavily. In text mining, our data structure is generally known as a **Term-Document Matrix**. The *documents* are any individual pieces of text that we wish to analyze, cluster, summarize or discover topics from. They could be sentences, abstracts, webpages, or social media updates. The *terms* are the words contained in these documents. The term-document matrix represents what's called the "bag-of-words" approach - the order of the words is removed and the data becomes unstructured in the sense that each document is represented by the words it contains, not the order or context in which they appear. The (i, j) entry in this matrix is the number of times term j appears in document i.

Definition 1.1.1: Term-Document Matrix

Let m be the number of documents in a collection and n be the number of terms appearing in that collection, then we create our **term-document**

matrix A as follows:

$$\mathbf{A}_{m\times n} = \begin{array}{cc} & \begin{array}{ccc} \text{term } 1 & \text{term } j & \text{term } n \end{array} \\ \begin{array}{c} \text{Doc } 1 \\ \\ \\ \text{Doc } i \\ \\ \text{Doc } m \end{array} & \left(\begin{array}{ccc} & | & \\ & | & \\ & | & \\ - \quad - & f_{ij} & \\ & & \\ & & \end{array} \right) \end{array}$$

where f_{ij} is the frequency of term j in document i. A **binary** term-document matrix will simply have $\mathbf{A}_{ij} = 1$ if term j is contained in document i.

Term-document matrices tend to be large and sparse. Term-weighting schemes are often used to downplay the effect of commonly used words and bolster the effect of rare but semantically important words. The most popular weighting method is known as "Term Frequency-Inverse Document Frequency" (TF-IDF). For this method, the raw term-frequencies f_{ij} in the matrix $\mathbf{A}$ are multiplied by global weights (inverse document frequencies), w_j, for each term. These weights reflect the commonality of each term across the entire collection. The inverse document frequency of term i is:

$$w_j = \log\left(\frac{\text{total \# of documents}}{\text{\# documents containing term } j}\right)$$

To put this weight in perspective for a collection of $n = 10,000$ documents we have $0 \leq w_j \leq 9.2$, where $w_j = 0$ means the word is contained in every document (i.e. it's not important semantically) and $w_j = 9.2$ means the word is contained in only 1 document (i.e. it's quite important). The document vectors are often normalized to have unit 2-norm, since their directions (not their lengths) in the term-space is what characterizes them semantically.

The noise-reduction property of the SVD was extended to text processing in 1990 by Susan Dumais et al, who named the effect *Latent Semantic Indexing (LSI)*. LSI involves the singular value decomposition of the term-document matrix defined in Definition **??**. In other words, it is like a principal components analysis using the unscaled, uncentered inner-product matrix $\mathbf{A}^T\mathbf{A}$. If the documents are normalized to have unit length, this is a matrix of **cosine similarities** (see Chapter **??**). **In text-mining, the cosine similarity is the most common measure of similarity between documents**. If the term-document matrix is binary, this is often called the co-occurrence matrix because each entry gives the number of times two words occur in the same document.

It certainly seems logical to view text data in this context as it contains both an informative signal and semantic noise. LSI quickly grew roots in the information retrieval community, where it is often used for query processing. The idea is to remove semantic noise, due to variation and ambiguity in vocabulary and presentation style, without losing significant amounts of information. For example, a human may not differentiate between the words "car" and "automobile", but indeed the words will become two separate entities in the raw term-document matrix. The main idea in LSI is that the realignment of the data into fewer directions should force related documents (like those containing "car" and "automobile") closer together in an angular sense, thus revealing latent semantic connections.

Purveyors of LSI suggest that the use of the Singular Value Decomposition to project the documents into a lower-dimensional space results in a representation which reflects the major associative patterns of the data while ignoring less important influences. This projection is done with the simple truncation of the SVD shown in Equation **??**.

As we have seen with other types of data, the very nature of dimension reduction makes possible for two documents with similar semantic properties to be mapped closer together. Unfortunately, the mixture of signs (positive and negative) in the singular vectors (think principal components) makes the decomposition difficult to interpret. While the major claims of LSI are legitimate, this lack of interpretability is still conceptually problematic for some folks. In order to make this point as clear as possible, consider the original "term basis" representation for the data, where each document (from a collection containing m total terms in the dictionary) could be written as:

$$\mathbf{A}_j = \sum_{i=1}^{m} f_{ij}\mathbf{e}_i$$

where f_{ij} is the frequency of term i in the document, and $\mathbf{e}_i$ is the i^{th} column of the $m \times m$ identity matrix. The truncated SVD gives us a new set of coordinates (scores) and basis vectors (principal component features):

$$\mathbf{A}_j \approx \sum_{i=1}^{r} \alpha_i \mathbf{u}_i$$

but the features $\mathbf{u}_i$ live in the term space, and thus ought to be interpretable as a linear combination of the original "term basis." However the linear combination, having both positive and negative coefficients, is semantically meaningless in context - These new features cannot, generally, be thought of as meaningful *topics*.