Latent Semantic Analysis (Tutorial)

Alex Thomo

1 Eigenvalues and Eigenvectors

Let A be an $n \times n$ matrix with elements being real numbers. If $\mathbf{x}$ is an n-dimensional vector, then the matrix-vector product $A\mathbf{x}$ is well-defined, and the result is again an n-dimensional vector. In general, multiplication by a matrix changes the direction of a *non-zero* vector $\mathbf{x}$, unless the vector is special and we have that

$$A\mathbf{x} = \lambda\mathbf{x},$$

for some scalar λ. In such a case, the multiplication by matrix A only stretches or contracts or reverses vector $\mathbf{x}$, but it does not change its direction. These special vectors and their corresponding λ's are called *eigenvectors* and *eigenvalues* of A. For diagonal matrices it is easy to spot the eigenvalues and eigenvectors. For example matrix

$$A = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

has eigenvalues and eigenvectors

$$\lambda_1 = 4 \text{ with } \mathbf{x}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \lambda_2 = 3 \text{ with } \mathbf{x}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \text{ and } \lambda_3 = 2 \text{ with } \mathbf{x}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

We will see that the number of eigenvalues is n for an $n \times n$ matrix. Regarding eigenvectors, if $\mathbf{x}$ is an eigenvector then so is $a\mathbf{x}$ for any scalar a. However, if we consider only one eigenvector for each $a\mathbf{x}$ family, then there is a 1-1 correspondence of such eigenvectors to eigenvalues. Typically, we consider eigenvectors of unit length.

Diagonal matrices are simple, the eigenvalues are the entries on the diagonal, and the eigenvectors are their columns. For other matrices we find the eigenvalues first by reasoning as follows. If $A\mathbf{x} = \lambda\mathbf{x}$ then $(A-\lambda I)\mathbf{x} = 0$, where I is the identity matrix. Since $\mathbf{x}$ is non-zero, matrix $A-\lambda I$ has dependent columns and thus its determinant $|A - \lambda I|$ must be zero. This gives us the equation $|A - \lambda I| = 0$ whose solutions are the eigenvalues of A.

As an example let

$$A = \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix} \text{ and } A - \lambda I = \begin{bmatrix} 3-\lambda & 2 \\ 2 & 3-\lambda \end{bmatrix}$$

Then the equation $|A - \lambda I| = 0$ becomes $(3-\lambda)^2 - 4 = 0$ which has $\lambda_1 = 1$ and $\lambda_2 = 5$ as solutions.

For each of these eigenvalues the equation $(A - \lambda I)\mathbf{x} = 0$ can be used to find the corresponding eigenvectors, e.g.

$$A - \lambda_1 I = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} \text{ yields } \mathbf{x}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \text{ and } A - \lambda_2 I = \begin{bmatrix} -2 & 2 \\ 2 & -2 \end{bmatrix} \text{ yields } \mathbf{x}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

In general, for an $n \times n$ matrix A, the determinant $|A - \lambda I|$ will give a polynomial of degree n which has n roots. In other words, the equation $|A - \lambda I| = 0$ will give n eigenvalues.

Let us create a matrix S with columns the n eigenvectors of A. We have that

$$\begin{aligned} AS &= A[\mathbf{x}_1, \ldots, \mathbf{x}_n] \\ &= A\mathbf{x}_1 + \ldots + A\mathbf{x}_n \\ &= \lambda_1\mathbf{x}_1 + \ldots + \lambda_n\mathbf{x}_n \\ &= [\mathbf{x}_1, \ldots, \mathbf{x}_n] \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}. \\ &= S\Lambda, \end{aligned}$$

where Λ is the above diagonal matrix with the eigenvalues of A along its diagonal. Now suppose that the above n eigenvectors are linearly independent. This is true when the matrix has n distinct eigenvalues. Then matrix S is invertible and by mutiplying both sides of $AS = S\Lambda$ we have

$$A = S\Lambda S^{-1}.$$

So, we were able to "diagonalize" matrix A in terms of the diagonal matrix Λ spelling the eigenvalues of A along its diagonal. This was possible because matrix S was invertible. When there are fewer than n eigenvalues then it might happen that the diagonalization is not possible. In such a case the matrix is "defective" having too few eigenvectors.

In this tutorial, for reasons to be clear soon we will be interested in symmetric matrices ($A = A^T$). For $n \times n$ symmetric matrices it has been shown that that they always have real eigenvalues and their eigenvectors are perpendicular. As such we have that

$$S^T S = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix} [\mathbf{x}_1, \ldots, x_n] = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} = I.$$

In other words, for symmetric matrices, S^{-1} is S^T (which can be easily obtained) and we have

$$A = S\Lambda S^T.$$

2 Singular Value Decomposition

Now let A be an $m \times n$ matrix with entries being real numbers and $m > n$. Let us consider the $n \times n$ square matrix $B = A^T A$. It is easy to verify that B is symmetric; namely $B^T = (A^T A)^T = A^T (A^T)^T = A^T A = B$). It has been shown that the eigenvalues of such matrices ($A^T A$) are real non-negative numbers. Since they are non-negative we can write them in decreasing order as squares of non-negative real numbers: $\sigma_1^2 \geq \ldots \geq \sigma_n^2$. For some index r (possibly n) the first r numbers $\sigma_1, \ldots, \sigma_r$ are positive whereas the rest are zero. For the above eigenvalues, we know that the corresponding eigenvectors $\mathbf{x}_1, \ldots, \mathbf{x}_r$ are perpendicular. Furthemore, we normalize them to have length 1. Let

$$S_1 = [\mathbf{x}_1, \ldots, \mathbf{x}_r].$$

We create now the vectors $\mathbf{y}_1 = \frac{1}{\sigma_1} A\mathbf{x}_1$, ..., $\mathbf{y}_r = \frac{1}{\sigma_r} A\mathbf{x}_r$. These are perpendicular m-dimensional vectors of length 1 (orthonormal vectors) because

$$\begin{aligned}
\mathbf{y}_i^T \mathbf{y}_j &= \left(\frac{1}{\sigma_1} A\mathbf{x}_i\right)^T \frac{1}{\sigma_j} A\mathbf{x}_j \\
&= \frac{1}{\sigma_i \sigma_j} \mathbf{x}_i^T A^T A \mathbf{x}_j \\
&= \frac{1}{\sigma_i \sigma_j} \mathbf{x}_i^T B \mathbf{x}_j \\
&= \frac{1}{\sigma_i \sigma_j} \mathbf{x}_i^T \sigma_j^2 \mathbf{x}_j \\
&= \frac{\sigma_j}{\sigma_i} \mathbf{x}_i^T \mathbf{x}_j
\end{aligned}$$

is 0 for $i \neq j$ and 1 for $i = j$ (since $\mathbf{x}_i^T \mathbf{x}_j = 0$ for $i \neq j$ and $\mathbf{x}_i^T \mathbf{x}_i = 1$). Let

$$S_2 = [\mathbf{y}_1, \ldots, \mathbf{y}_r].$$

We have

$$\mathbf{y}_j^T A \mathbf{x}_i = \mathbf{y}_j^T (\sigma_i \mathbf{y}_i) = \sigma_i \mathbf{y}_j^T \mathbf{y}_i,$$

which is 0 if $i \neq j$, and σ_i if $i = j$. From this we have

$$S_2^T A S_1 = \Sigma,$$

where Σ is the diagonal $r \times r$ matrix with σ_1, ..., σ_r along the diagonal. Observe that S_2^T is $r \times m$, A is $m \times n$, and S_1 is $n \times r$, and thus the above matrix multiplication is well defined.

Since S_2 and S_1 have orthonormal columns, $S_2 S_2^T = I_{m \times m}$ and $S_1 S_1^T = I_{n \times n}$, where $I_{m \times m}$ and $I_{n \times n}$ are the $m \times m$ and $n \times n$ identity matrices. Thus, by multiplying the above equality by S_2 on the left and S_1 on the right, we have

$$A = S_2 \Sigma S_1^T.$$

Reiterating, matrix Σ is diagonal and the values along the diagonal are σ_1, ..., σ_r, which are called *singular values*. They are the square roots of the eigenvalues of $A^T A$ and thus completely determined by A. The above decomposition of A into $S_2^T A S_1$ is called *singular value decomposition*.

For the ease of notation, let us denote S_2 by S and S_1 by U (getting thus rid of the subscripts). Then

$$A = S \Sigma U^T.$$

3 Latent Semantic Indexing

Latent Semantic Indexing (LSI) is a method for discovering hidden *concepts* in document data. Each document and term (word) is then expressed as a vector with elements corresponding to these concepts. Each element in a vector gives the degree of participation of the document or term in the corresponding concept. The goal is not to describe the concepts verbally, but to be able to represent the documents and terms in a unified way for exposing document-document, document-term, and term-term similarities or semantic relationship which are otherwise hidden.

3.1 An Example

Suppose we have the following set of five documents

d_1 : *Romeo and Juliet.*

d_2 : *Juliet: O happy dagger!*

d_3 : *Romeo died by dagger.*

d_4 : *"Live free or die", that's the New-Hampshire's motto.*

d_5 : *Did you know, New-Hampshire is in New-England.*

and a search query: *dies, dagger.*

Clearly, d_3 should be ranked top of the list since it contains both *dies, dagger*. Then, d_2 and d_4 should follow, each containing a word of the query. However, what about d_1 and d_5? Should they be returned as possibly interesting results to this query? As humans we know that d_1 is quite related to the query. On the other hand, d_5 is not so much related to the query. Thus, we would like d_1 but not d_5, or differently said, we want d_1 to be ranked higher than d_5.

The question is: Can the machine deduce this? The answer is yes, LSI does exactly that. In this example, LSI will be able to see that term *dagger* is related to d_1 because it occurs together with the d_1's terms *Romeo* and *Juliet*, in d_2 and d_3, respectively. Also, term *dies* is related to d_1 and d_5 because it occurs together with the d_1's term *Romeo* and d_5's term *New-Hampshire* in d_3 and d_4, respectively. LSI will also weigh properly the discovered connections; d_1 more is related to the query than d_5 since d_1 is "doubly" connected to *dagger* through *Romeo* and *Juliet*, and also connected to *die* through *Romeo*, whereas d_5 has only a single connection to the query through *New-Hampshire*.

3.2 SVD for LSI

Formally let A be the $m \times n$ term-document matrix of a collection of documents. Each column of A corresponds to a document. If term i occurs a times in document j then $A[i, j] = a$. The dimensions of A, m and n, correspond to the number of words and documents, respectively, in the collection. For our example, matrix A is:

	d_1	d_2	d_3	d_4	d_5
romeo	1	0	1	0	0
juliet	1	1	0	0	0
happy	0	1	0	0	0
dagger	0	1	1	0	0
live	0	0	0	1	0
die	0	0	1	1	0
free	0	0	0	1	0
new-hampshire	0	0	0	1	1

Observe that $B = A^T A$ is the document-document matrix. If documents i and j have b words in common then $B[i, j] = b$. On the other hand, $C = AA^T$ is the term-term matrix. If terms i and j occur together in c documents then $C[i, j] = c$. Clearly, both B and C are square and symmetric; B is an $m \times m$ matrix, whereas C is an $n \times n$ matrix.

Now, we perform an SVD on A using matrices B and C as described in the previous section

$$A = S\Sigma U^T,$$

where S is the matrix of the eigenvectors of B, U is the matrix of the eigenvectors of C, and Σ is the diagonal matrix of the singular values obtained as square roots of the eigenvalues of B.

Matrix Σ for our example is calculated to be

$$\Sigma = \begin{bmatrix} 2.285 & 0 & 0 & 0 & 0 \\ 0 & 2.010 & 0 & 0 & 0 \\ 0 & 0 & 1.361 & 0 & 0 \\ 0 & 0 & 0 & 1.118 & 0 \\ 0 & 0 & 0 & 0 & 0.797 \end{bmatrix}$$

(For small SVD calculations, you can use the BlueBit calculator at *http://www.bluebit.gr/matrix-calculator*). As you can see, the singular values along the diagonal of Σ are listed in descending order of their magnitude.

Some of the singular values are "too small" and thus "negligible." What really constitutes "too small" is usually determined empirically. In LSI we ignore these small singular values and replace them by 0. Let us say that we only keep k singular values in Σ. Then Σ will be all zeros except the first k entries along its diagonal. As such, we can reduce matrix Σ into Σ_k which is an $k \times k$ matrix containing only the k singular values that we keep, and also reduce S and U^T, into S_k and U_k^T, to have k columns and rows, respectively. Of course, all these matrix parts that we throw out would have been zeroed anyway by the zeros in Σ. Matrix A is now approximated by

$$A_k = S_k \Sigma_k U_k^T.$$

Observe that since S_k, Σ_k, and U_k^T are $m \times k$, $k \times k$, and $k \times n$ matrices, their product, A_k is again an $m \times n$ matrix.

Intuitively, the k remaning ingredients of the eigenvectors in S and U correspond to k "hidden concepts" where the terms and documents participate. The terms and documents have now a new representation in terms of these hidden concepts. Namely, the terms are represented by the row vectors of the $m \times k$ matrix

$$S_k \Sigma_k,$$

whereas the documents by the column vectors the $k \times n$ matrix

$$\Sigma_k U_k^T.$$

Then the query is represented by the centroid of the vectors for its terms.

3.3 Completing the Example

Let us now fully develop the example we started with in this section. Suppose we set $k = 2$, i.e. we will consider only the first two singular values. Then we have

$$\Sigma_2 = \begin{bmatrix} 2.285 & 0 \\ 0 & 2.010 \end{bmatrix}$$

$$S_2 = \begin{array}{r} \textit{romeo} \\ \textit{juliet} \\ \textit{happy} \\ \textit{dagger} \\ \textit{live} \\ \textit{die} \\ \textit{free} \\ \textit{new-hampshire} \end{array} \begin{bmatrix} -0.396 & 0.280 \\ -0.314 & 0.450 \\ -0.178 & 0.269 \\ -0.438 & 0.369 \\ -0.264 & -0.346 \\ -0.524 & -0.246 \\ -0.264 & -0.346 \\ -0.326 & -0.460 \end{bmatrix}$$

$$U_2^T = \begin{bmatrix} -0.311 & -0.407 & -0.594 & -0.603 & -0.143 \\ 0.363 & 0.541 & 0.200 & -0.695 & -0.229 \end{bmatrix}$$

The terms in the concept space are represented by the row vectors of S_2 whereas the documents by the column vectors of U_2^T. In fact we scale the (two) coordinates of these vectors by multiplying with the corresponding singular values of Σ_2 and thus represent the terms by the row vectors of $S_2\Sigma_2$ and and the documents by the column vectors of $\Sigma_2 U_2^T$. Finally we have

$$\textit{romeo} = \begin{bmatrix} -0.905 \\ 0.563 \end{bmatrix},\ \textit{juliet} = \begin{bmatrix} -0.717 \\ 0.905 \end{bmatrix},\ \textit{happy} = \begin{bmatrix} -0.407 \\ 0.541 \end{bmatrix},\ \textit{dagger} = \begin{bmatrix} -1.001 \\ 0.742 \end{bmatrix},$$

$$\textit{live} = \begin{bmatrix} -0.603 \\ -0.695 \end{bmatrix},\ \textit{die} = \begin{bmatrix} -1.197 \\ -0.494 \end{bmatrix},\ \textit{free} = \begin{bmatrix} -0.603 \\ -0.695 \end{bmatrix},\ \textit{new-hampshire} = \begin{bmatrix} -0.745 \\ -0.925 \end{bmatrix},$$

and

$$d_1 = \begin{bmatrix} -0.711 \\ 0.730 \end{bmatrix},\ d_2 = \begin{bmatrix} -0.930 \\ 1.087 \end{bmatrix},\ d_3 = \begin{bmatrix} -1.357 \\ 0.402 \end{bmatrix},\ d_4 = \begin{bmatrix} -1.378 \\ -1.397 \end{bmatrix},\ d_5 = \begin{bmatrix} -0.327 \\ -0.460 \end{bmatrix}.$$

Now the query is represented by a vector computed as the centroid of the vectors for its terms. In our example, the query is *die, dagger* and so the vector is

$$q = \frac{\begin{bmatrix} -1.197 \\ -0.494 \end{bmatrix} + \begin{bmatrix} -1.001 \\ 0.742 \end{bmatrix}}{2} = \begin{bmatrix} -1.099 \\ 0.124 \end{bmatrix}.$$

In order to rank the documents in relation to query q we will use the cosine distance. In other words, we will compute

$$\frac{d_i \cdot q}{|d_i||q|}$$

where $i \in [1, n]$ and then sort the results in descending order. For our example, let us do this geometrically. Based on Figure 1, the following interesting observations can be made.

1. Document d_1 is closer to query q than d_5. As a result d_1 is ranked higher than d_5. This conforms to our human preference, both Romeo and Juliet died by a dagger.

2. Document d_1 is slightly closer to q than d_2. Thus the system is quite intelligent to find out that d_1, containing both Romeo and Juliet, is more relevant to the query than d_2 even though d_2 contains explicitly one of the words in the query while d_1 does not. A human user would probably do the same.

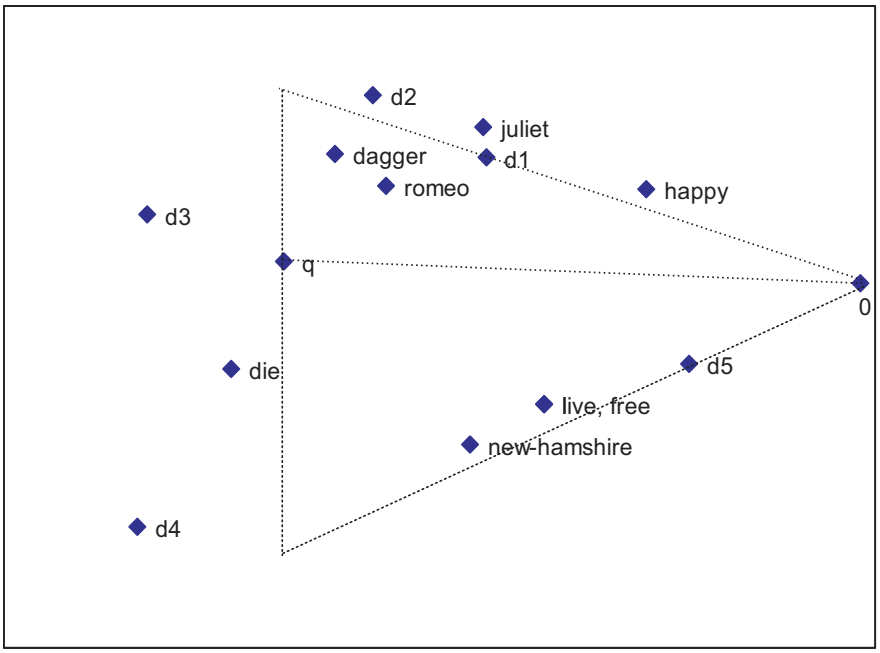

Figure 1: Geometric representation of documents $d_1, \ldots, d_5$, their terms, and query q.