



# Information Retrieval Using The Singular Value Decomposition

Greg Brown and Liya Zhu



# Summary of Objectives

- Examine problems with classic information retrieval
- Search for new methods of information retrieval with linear algebra
- Create a vector space model of data
- Change the basis of the vector space to create more precise results
- Use the SVD to create an approximation of the original data in the new space
- Examine a real world application



# Classic Indexing

A librarian in an old library might be faced with indexing a few thousand or so documents. When a patron comes to find a document, they look through a small card catalog to find what they need. This method has worked for hundreds of years, why would we want to change things?



Now imagine a librarian attempting to index the Internet. Faced with billions of documents the most intrepid indexer will quake in fear. Even if a librarian was able to index this massive amount of data, how could a user possibly search with any efficiency and hope to find a relevant document? It is our goal to aid the librarians and their patrons by finding a way to make their lives easier.



# Vector Space Representation

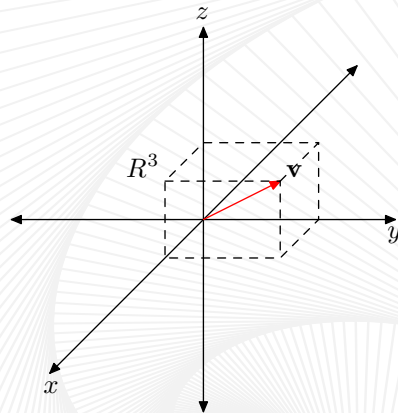
In order to represent documents mathematically we will use vectors. Each element in the vector will represent the frequency of an arbitrary term.

For example, the title “The Art of Computer Programming” will be our document. The terms “mathematics,” “computer,” and “art” will be used to index the document. Each element in the vector will represent the frequency with which each term appears in the document. The term-document vector becomes

$$v = [0 \ 1 \ 1]^T.$$



This vector happens to exist in  $\mathbf{R}^3$ . A visual representation looks like



where  $v$  is our document vector. This geometrical interpretation is called the vector space model. This geometry will remain as we advance to higher dimensions



# Expanding to Matrices

We can now represent one vector as a document. The next step is to represent a large group of documents. A matrix handles this perfectly. We call this a term by document matrix and it is represented by

$$A = \begin{pmatrix} T_1D_1 & T_1D_2 & T_1D_3 & T_1D_4 & T_1D_5 \\ T_2D_1 & T_2D_2 & T_2D_3 & T_2D_4 & T_2D_5 \\ T_3D_1 & T_3D_2 & T_3D_3 & T_3D_4 & T_3D_5 \\ T_4D_1 & T_4D_2 & T_4D_3 & T_4D_4 & T_4D_5 \\ T_5D_1 & T_5D_2 & T_5D_3 & T_5D_4 & T_5D_5 \\ T_6D_1 & T_6D_2 & T_6D_3 & T_6D_4 & T_6D_5 \end{pmatrix} \quad (1)$$

Each column represents a different document. Notice we are using six terms rather than the three in the previous example. Each of these document vectors exist in  $\mathbf{R}^6$ .



# An Example Using Matrices

First we choose some terms to index the document:

- T1: bak(e,ing)
- T2: recipes
- T3: bread
- T4: cake
- T5: pastr(y,ies)
- T6: pie



Next some documents:

- D1: How to Bake Bread Without Recipes
- D2: The Classic Art of Viennese Pastry
- D3: Numerical Recipes: The Art of Scientific Computing
- D4: Breads, Pastries, Pies and Cakes: Quantity Baking Recipes
- D5: Pastry: A Book of Best French Recipes



The matrix becomes

$$\tilde{A} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad (2)$$

Each  $j$ th column represents each document. Each  $i$ th row represents each term. Each element  $a_{ij}$  represents the frequency of the  $i$ th term in the  $j$ th document.



In order to see the relative importance of each term in a certain document, we normalize the matrix:

$$A = \begin{pmatrix} .5774 & 0 & 0 & .4082 & 0 \\ .5774 & 0 & 1 & .4082 & .7071 \\ .5774 & 0 & 0 & .4082 & 0 \\ 0 & 0 & 0 & .4082 & 0 \\ 0 & 1 & 0 & .4082 & .7071 \\ 0 & 0 & 0 & .4082 & 0 \end{pmatrix} \quad (3)$$

This representation is useful only if we have a way to compare a user's query to a document in the matrix. This will be accomplished geometrically by comparing the cosine of the angle between each document vector and a query vector of the same dimension:

$$\cos \theta_j = \frac{a_j^T q}{\|a_j\| \|q\|}.$$



Let us say a user in our previous example asks for documents about “baking.” This is represented by the query vector

$$q = (1 \ 0 \ 0 \ 0 \ 0 \ 0)$$

Using our formula, the cosines between each document and the query vector are .5774, 0, 0, .4082, and 0. If we use a cutoff value of .5, only the first document in the matrix will be returned. However, recall

Document 1: How to Bake Bread Without Recipes

Document 4: Breads, Pastries, Pies and Cakes: Quantity Baking Recipes.

Document 4 is certainly a relevant document to the users query even though it is not returned. This partial failure, which is one of many problems inherent in the vector space model, provides inspiration to craft a superior method of information retrieval.



# Rank Reduction

One powerful method for dealing with problems inherent in the vector space model is to examine the rank of the term-by-document matrix. Usually large term-by-document matrices have a large amount of redundant data. Removing this information allows a more precise and efficient search.

We will accomplish this removal of data by approximating the original term by document matrix  $A$  with a new matrix  $A_k$  where

$$\text{rank}(A) = r > \text{rank}(A_k) = k$$

In order to perform rank reduction we must find a method that will allow us to easily see the rank of matrix  $A$ .



# The Singular Value Decomposition

One method used for solving the rank reduction problem, as well as other inherent problems with the vector space model is the Singular Value Decomposition

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

If the term-by-document matrix is  $t \times d$  then  $U$  is a  $t \times t$  orthogonal matrix,  $V$  is a  $d \times d$  orthogonal matrix and  $\Sigma$  is a  $t \times d$  diagonal matrix. The values on the diagonal of  $\Sigma$  are called the singular values where

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

and  $r$  is the rank of the term-by-document matrix.



It can be shown that the first  $r = \text{rank}(A)$  columns of  $U$  are a basis for the column space of  $A$ . Also, the first  $r$  rows of  $V^T$  are a basis for the row space of  $A$ . The help with rank reduction comes from  $\Sigma$ .

$$\Sigma = \begin{pmatrix} \sigma_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_k & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_r & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (4)$$

By changing all but the top  $k$  rows of  $\Sigma$  to zero rows, we can create a lower rank approximation to  $A$  called  $A_k$  where

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

The rank of  $A$  has been lowered from  $r$  to  $k$ . This low rank approximation has removed redundancy from our original data.



Now a method of comparison is required for  $A_k$

$$\cos \theta_j = \frac{(A_k e_j)^T q}{\|A_k e_j\| \|q\|} \quad (5)$$

The vector  $e_j$  is the  $j$ th vector of the  $d \times d$  identity matrix. It is used here to extract the  $j$ th column of  $A_k$ . Then the equation can be rewritten as

$$\cos \theta_j = \frac{(\hat{a}_j)^T q}{\|\hat{a}_j\| \|q\|}, \quad (6)$$

where  $\hat{a}_j = A_k e_j$ . In theory this new equation should return more relevant results.



## Back to our Example

Using the SVD on our original matrix  $A$  returns

$$\Sigma = \begin{pmatrix} 1.69 & 0 & 0 & 0 & 0 \\ 0 & 1.12 & 0 & 0 & 0 \\ 0 & 0 & 0.84 & 0 & 0 \\ 0 & 0 & 0 & 0.42 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (7)$$

along with the matrices  $U$  and  $V^T$ . If we replace the value .42 with 0, we have a rank 3 approximation to the original matrix.



Recalculating  $A_k$  with the new  $\Sigma_k$  gives

$$A_k = \begin{pmatrix} 0.50 & -0.03 & 0.02 & 0.49 & -0.01 \\ 0.60 & 0.01 & 0.99 & 0.39 & 0.71 \\ 0.50 & -0.03 & 0.02 & 0.49 & -0.01 \\ 0.18 & 0.07 & -0.05 & 0.23 & 0.02 \\ -0.03 & 0.99 & 0.01 & 0.44 & 0.70 \\ 0.18 & 0.07 & -0.05 & 0.23 & 0.02 \end{pmatrix} \quad (8)$$

This certainly looks nothing like the original  $A$ ! However, recall we are using a geometrical model, the appearance of the vectors has nothing to do with the similarity.



Recall the old query vector

$$q = (1 \ 0 \ 0 \ 0 \ 0 \ 0)^T.$$

Using the new cosine equation returns values of .5181, -.0332, .0233, .5064, and -.0069. Compare these values with the original values .5774, 0, 0, .4082, and 0. Using our cutoff of .5 on the new cosine values returns documents 1 and 4. Unlike the original vector space model, the new matrix returns both of the relevant documents, rendering our search superior.

The SVD has lowered the rank of the original matrix, thereby reducing redundancy in our original data. Then we allowed a user to query this data. All of this has occurred without human intervention. We have succeeded in our original task, creating a method for indexing extremely large amounts of data and giving the user the ability to query this index without introducing human error or work.



# Conclusion

Here is a summary of what was covered in our speech:

- From a database of documents we created a term-by-document matrix where each element represented the frequency of a certain term in a certain document.
- Rank reducing a matrix removes redundancy in a database.
- The SVD provides an efficient method for rank reduction
- We created a method for querying this new data, essentially creating a basic search engine.

