Image Compression and Recommendation Systems

Tool: Singular Value Decomposition.

The Analytics Edge: Images are an important type of unstructured data on the internet. Companies that share user photos—such as Instagram, Facebook, Flickr, and Twitter—need efficient ways to store them. Data compression is thus an important tool that companies use to tradeoff quality and size of photos. Tools for matrix factorization, such as Singular Value Decomposition (SVD), are particularly useful in this domain. These tools are also adopted in recommendation systems to handle matrices with missing entries.

Overview

Photo and video-sharing social networking services

There are various social networking services that enable users to take and share photos. Instagram is arguably one of the most popular. It was created in 2010 as a free mobile app and acquired by Facebook in 2012. According to Bloomberg, the company is now worth more than $ 100 billion (McCormick, 2018). Since its launch, more than 40 billion photos have been shared. Another example of such social networking service is Google Photos, a photo and video-sharing platform launched by Google in an attempt to increase the popularity of Google +. So, how many photos are uploaded to the Internet? It is estimated that in 2014 people uploaded an average of 1.8 billion digital images every single day (Eveleth, 2015). Assuming that an image is on average 100 kb, this means that for just one day the amount of bytes needed for storing the images is 1.8 x 10^{14} bytes, that is, 180 terabytes.

Image representation

A digital (raster) image contains a finite set of *pixels*, which are arranged in a matrix with a given number of rows and columns.

Greyscale images can be seen as matrices in which the value of each pixel (cell) is a single number representing an amount of light—the intensity information. Such images have intensity going from black (0) to white (1), with the different shades of grey taking values in $[0, 1]$.

Color images are built as a set of stacked color channels, each of them representing value levels of the given channel. The most common color model is arguably the RGB one, which is based on the three primary colors, red, green, and blue. Hence, an RGB image contains the red, green, and blue color in three separate matrices (generally represented with 8-32 bit).

Summary

Data: Greyscale and color images are stored as matrices containing the intensity information or the information on the red, green, and blue color.

Model: Singular Value Decomposition can be used to extract key information from these matrices.

Value and Decision: an effective image compression system helps store a large amount of images without compromising their quality, and, therefore, the user experience.

Singular Value Decomposition

Introduction

Given a rectangular matrix X of dimension m x n, the Singular Value Decomposition (SVD) of X takes the following form:

$$\underbrace{X}_{m \text{ x } n} = \underbrace{U}_{m \text{ x } n} \underbrace{S}_{n \text{ x } n} \underbrace{V^T}_{n \text{ x } n},$$

where:

- U is an m x n unitary matrix, that is, $U^TU = UU^T = I$ (where I is the identity matrix);
- V is an n x n unitary matrix, that is, $V^TV = VV^T = I$;
- S is an n x n diagonal matrix with non-negative real numbers on the diagonal. Let's denote these non-negative entries with $\sigma_1, \dots, \sigma_n \geq 0$, which are typically referred to as *singular values*.

The SVD applies to any m x n (rectangular) matrix, while the eigenvalue decomposition applies only to certain classes of square matrices.

Relation to eigenvalue decomposition

To understand the meaning of U, let's consider the following expression:

$$\begin{aligned}
X &= USV^T. \\
XX^T &= USV^TVS^TU^T \\
&= USIS^TU^T \text{ (since } V^TV = I) \\
&= U(SS)U^T \text{ (since } S^T = S \text{ is diagonal)} \\
&= US^2U^T.
\end{aligned}$$

Thus, the column vectors in U are the eigenvectors of XX^T, and the non-zero elements $\sigma_1, \dots, \sigma_r$ are the square root of the non-zero eigenvalues of XX^T. Similarly, we can write:

$$\begin{aligned}
X^TX &= VS^TU^TUSV^T \\
&= V(S^TS)V^T \\
&= VS^2V^T.
\end{aligned}$$

This means that the column vectors in V are the eigenvectors of X^TX.

Low-rank approximation

The matrix X can also be expressed as:

$$X = \sum_{j=1}^{n} \underbrace{\sigma_j U_j V_j^T}_{\substack{\text{Sum of rank} \\ \text{one matrices}}},$$

where $\sigma_j \geq 0$, U_j is the j-th column of U and V_j is the j-th column of V.

Without loss of generality, say $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$. The main idea of low-rank approximation is that we develop a simpler version of X by dropping the smallest singular values. A rank-k approximation to X is thus given by:

$$\hat{X} = \sum_{j=1}^{k} \sigma_j U_j V_j^T,$$

where $k \leq n$. A graphical representation is given in Figure 0.1.

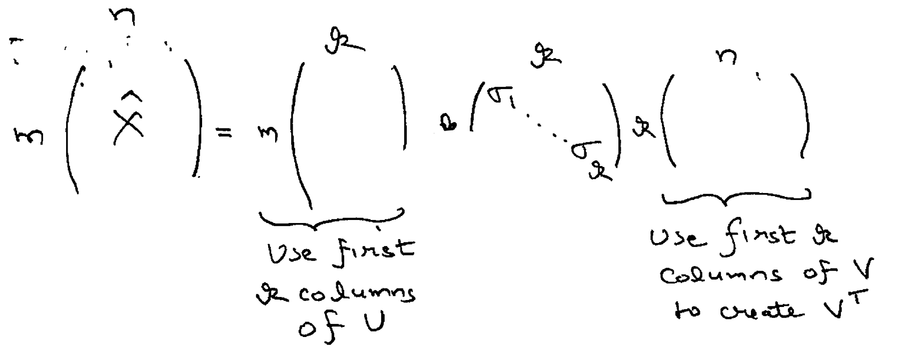

Figure 0.1: Low rank approximation.

One of the big advantages of performing a low rank approximation is that to represent an m x n matrix we only need $mk + k + nk = k(m + n + 1)$ parameters. This can help save a significant amount of memory. Consider, for example, a case in which $m = 200$ and $n = 320$: this matrix has 64,000 entries. If we use $k = 20$, we need $20(200 + 320 + 1) = 10,420$ entries to approximate the matrix.

How accurate is a low-rank approximation? Let's first introduce the Frobenius norm of a matrix:

$$||X||_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} x_{i,j}^2}.$$

Equivalently, $||X|| = \sqrt{\sigma_1^2 + \ldots + \sigma_n^2}$, where $\sigma_1, \ldots, \sigma_n$ are the singular values of X. In approximating a matrix with a low-rank matrix, the accuracy of the approximation (in terms of explained variance) is thus given by the following expression:

$$\frac{||\hat{X}||_F}{||X||_F} = \frac{\sigma_1^2 + \ldots + \sigma_k^2}{\sigma_1^2 + \ldots + \sigma_n^2}.$$

SVD in recommendation systems

Suppose a matrix R contains the ratings provided by customers (rows) to a number of movies (columns). Ideally, we could use SVD to reduce the matrix R into $\hat{R}$ and then predict a rating by simply looking up the entry for the appropriate user/movie pair in the matrix $\hat{R}$.

In most cases, R is a sparse matrix, so it is not possible to apply SVD. A possible way to solve this problem is to fill in the missing entries using a reasonable method, such as taking the mean of the columns (or the rows), and then apply SVD to the full matrix. Yet, an inaccurate filling of the matrix R could affect the results. Another way to find the matrices U and V is to find all vectors p_u and q_i such that:

- $\hat{r}_{ui} = p_u^T q_i$ or $q_i^T p_u$;
- All vectors p_u are mutually orthogonal, as well as the vectors q_i.

This idea is illustrated in Figure 0.2.

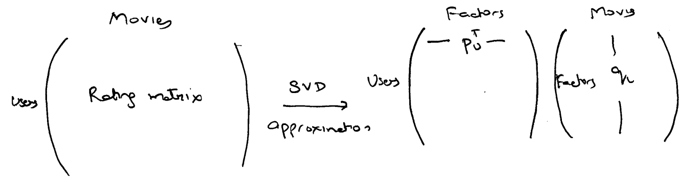

Figure 0.2: SVD in recommendation systems.

Finding the vectors p_u and q_i for all users and items can be done by solving the following optimization problem:

$$\min_{q_i, p_u, \forall (i,u)} \sum_{(u,i) \in \text{observed user-rating pairs}} (r_{ui} - q_i^T p_u)^2.$$

This was one of the approaches used in the final Netflix winning contribution. This optimization problem is not convex, so, one can use simple stochastic gradient methods to solve it. For example:

1. Start with q_i and p_u chosen arbitrarily. Then, for each user-item pair in the training set, compute the error e_{ui} defined as $r_{ui} - q_i^T p_u$.
2. Using a gradient method, update the values of q_i and p_u:

$$q_i \leftarrow q_i + \gamma \cdot p_u (r_{ui} - q_i^T p_u)$$
$$p_u \leftarrow p_u + \gamma \cdot q_i (r_{ui} - q_i^T p_u),$$

where γ is the algorithm learning rate. This can be implemented using the *funkSVD* function in the *recommenderlab* package. Once all vectors p_u and q_i have been computed, we can estimate all the ratings we want using the formula $\hat{r}_{ui} = q_i^T p_u$.

References

Eveleth, R. (2015, November 2). How many photographs of you are out there in the world? https://www.theatlantic.com/technology/archive/2015/11/how-many-photographs-of-you-are-out-there-in-the-world/413389/.

McCormick, E. (2018, June 26). Instagram is estimated to be worth more than $100 billion. https://www.bloomberg.com/news/articles/2018-06-25/value-of-facebook-s-instagram-estimated-to-top-100-billion.