Applications

- Compression, matrix completion
- Recommender systems
- Sports team rankings

Topics

- Low-rank approximations
- Matrix completion problem
- Frobenius norm
- SVD for optimal low-rank approximations

These lecture notes are mostly based off lecture 9 from Stanford CS168:

https://web.stanford.edu/class/cs168/l/l9.pdf

What are the Missing Entries?

Suppose that I run a web streaming service for movies for three of my friends, Amy, Bob, and Carol. It's a very specialized movie service, with only five movie options: The Matrix, Inception, Star Wars: Episode I, Moana, and Inside Out. After 1 month, we ask our friends Amy, Bob, and Carol to rate the movies they've watched from one to five. We collect their ratings into a table below (we mark unrated movies with ?):

	The Matrix	Inception	Star Wars: Ep I	Moana	Inside Out
Amy	2	?	?	?	5
Bob	?	3	4	?	2
Carol	?	?	2	1	?

and are asked to provide recommendations to Amy, Bob, and Carol as to which movie they should watch next. Said another way, we are asked to fill in the unknown ? entries in the table above.

This seems a bit unfair! Each of the unknown entries could be any value in 1-5 after all! But what if I told you an additional hint: Amy, Bob, and Carol have the same relative preferences for each movie: for example, Amy likes Inside Out $\frac{5}{2}$ more than Bob likes Inside Out, and this ratio is the same across all movies. Mathematically, we are making the assumption that all columns of the table above are multiples of each other.

Thus we can conclude that Bob likes the Matrix $\frac{2}{5} \cdot (\text{Amy's rating}) = 4/5$. Similarly, Carol's rating of Inception is $\frac{1}{2}(\text{Bob's rating}) = 1.5$, Carol's rating of Inside Out is $\frac{1}{2} \times (\text{Bob's rating}) = 1$, and so on. Here's the completed matrix:

$$M = \begin{bmatrix} 2 & 7.5 & 10 & 5 & 5 \\ 0.8 & 3 & 4 & 2 & 2 \\ 0.4 & 1.5 & 2 & 1 & 1 \end{bmatrix}$$

The point of this example is that when you know something about the structure of a partially known matrix, then sometimes it is possible to intelligently fill in missing entries. In this previous example, the assumption that every column is a multiple of each other means that rank $M = 1$ (since $\dim \mathrm{Col}(M) = 1$), which is pretty extreme! One natural and useful definition is that assuming a matrix M has low-rank. What rank counts as "low" is application dependent, but it typically means that for a matrix $M \in \mathbb{R}^{m \times n}$, that rank $M = r \ll \min\{m, n\}$.

This lecture will explore how we can use this idea of structure to solve the matrix completion problem by finding the best low-rank approximation to a partially known matrix. The SVD will of course be our main tool.

Low-Rank Matrix Approximations: Motivation

Before diving into the math, let's highlight some applications of low-rank matrix approximations:

1. Compression: we saw this idea last class, but its worth revisiting through the lens of low-rank approximations. If the original matrix $M \in \mathbb{R}^{m \times n}$ is described by mn numbers, then a rank k approximation only requires $k(m+n)$ numbers. To see this, recall that if $\hat{M}$ has rank k, then we can write its SVD as

$$\hat{M} = \begin{bmatrix} \\ U \\ \\ \end{bmatrix}_{m \times k} [\Sigma]_{k \times k} \begin{bmatrix} & V^T & \end{bmatrix}_{k \times n}$$

$$\left(\Sigma^{1/2} = \operatorname{diag}(\lambda_1^{1/2}, \ldots, \lambda_k^{1/2})\right)$$

$$= \begin{bmatrix} \underbrace{U\Sigma^{1/2}}_{=Y} \\ \\ \end{bmatrix}_{m \times k} \begin{bmatrix} & \underbrace{\Sigma^{1/2} V^T}_{=Z^T} & \end{bmatrix}_{k \times n}$$

a product $\hat{M} = YZ^T$, where $Y \in \mathbb{R}^{m \times k}$ and $Z \in \mathbb{R}^{n \times k}$. For example, if M represents a gray scale image (with entries = pixel intensities), m and n are typically in the 100s (or 1000s for HD images), and a modest value of k ($\sim$100-150) is usually enough to give a good approximation of the original image.

2. Updating Huge AI Models: A modern application of low-rank matrix approximations is for "fine-tuning" huge AI models. In the setting of Large Language Models (LLM), like ChatGPT we are typically given some huge off-the-shelf model with billions (or more) parameters. Given this large model that has been trained on an enormous but generic corpus of text from the web, one often performs "fine-tuning." This fine-tuning is a second round of training, typically using a much smaller domain specific dataset (for example, the lecture notes for this class could be used to fine-tune a "LinearAlgebraGPT"). The challenge of fine-tuning is that because these models are so big, making these updates is extremely challenging. The 2021 paper "LoRA: Low-Rank Adaptation of Large Language Models" argued that fine-tuning updates are generally approx. low-rank and that one can learn these updates in their factorized YZ^T forms, allowing model fine-tuning with 1000x - 10000x fewer parameters!

3. Denoising: If M is a noisy version of some "true" matrix that is approx. low-rank, then finding a low-rank approximation to M will typically remove a lot of noise (and maybe some signal), resulting in a matrix that is actually more informative than the original.

4. Matrix Completion: Low-rank approximations offer a way of solving the matrix completion problem we introduced above. Given a matrix M with missing entries, the first step is to obtain a full matrix $\hat{M}$ by filling in the missing entries with "default" values:

what these default values should be often requires trial and error, but natural things to try include 0, the average of known entries in the same column, row, or the entire matrix. The second step is then to find a rank k approximation to $\hat{M}$. This approach works well when the unknown matrix is close to a rank k matrix and there aren't too many missing entries

With this motivation in mind, let's see how the SVD can help us in finding a good rank r approximation of a matrix M. Once we've described our procedure, and seen some examples of it in action, we'll make precise how our method is actually producing the "best" rank r approximation possible.

<u>Low-Rank Approximations from the SVD</u>

Given an $m \times n$ matrix $M \in \mathbb{R}^{m \times n}$, which we'll assume has rank r. Then the SVD of M is given by

$$M = U \Sigma V^T = \sum_{i=1}^{r} \sigma_i \underline{u}_i \underline{v}_i^T, \qquad \text{(SVD)}$$

for $U = [\underline{u}_1 \cdots \underline{u}_r] \in \mathbb{R}^{m \times r}$, $V = [\underline{v}_1 \cdots \underline{v}] \in \mathbb{R}^{n \times r}$, and $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$ the matrices of left singular vectors, right singular vectors, and singular values, respectively.

The right-most expression of (SVD) is a particularly convenient expression for our purposes, which expresses M as a sum of rank 1 matrices $\sigma_i \underline{u}_i \underline{v}_i^T$ with mutually orthogonal column and row spaces.

This sum expression suggests a very natural way of forming a rank k approximation to M: simply truncate the sum to the top k terms, as measured by the singular values σ_i:

$$\hat{M}_k = \sum_{i=2}^{k} \sigma_i \underline{u}_i \underline{v}_i^T = U_k \Sigma_k V_k^T, \quad \text{(SVD-k)}$$

where the right most expression is defined in terms of the truncated matrices:

$$U_k = [\underline{u}_1 \cdots \underline{u}_k] \in \mathbb{R}^{m \times k}, \quad V_k = [\underline{v}_1 \cdots \underline{v}_k] \in \mathbb{R}^{n \times k}, \quad \Sigma_k = \mathrm{diag}(\sigma_1, \ldots, \sigma_k) \in \mathbb{R}^{k \times k}$$

Before analyzing the properties of $\hat{M}_k = U_k \Sigma_k V_k^T$, let's examine if $\hat{M}_k$ could plausibly address our motivating applications. Storing the matrices U_k, V_k, and Σ_k requires storing $km + kn + k^2 \approx k(m+n)$ numbers if $k << \min\{m,n\}$, which is much less than the mn numbers needed to store $M \in \mathbb{R}^{m \times n}$ when m and n are relatively large.

It is also natural to interpret (SVD-k) as approximating the raw data M in terms of k "concepts" (e.g., "sci-fi", "romcom", "drama", "classic"), where the singular values $\sigma_1, \ldots, \sigma_k$ express the "prominence" of the concepts, the rows of V^T and columns of U express the "typical row/column" associated with each concept (e.g., a viewer likes only sci-fi movies, or a movie liked only by romcom viewers), and the rows of U (or columns of V^T) approximately express each row (or column) of M as a linear combination (scaled by $G_1, \ldots, G_k$) of the "typical rows" (or "typical columns"),

This method of producing a low-rank approximation is beautiful: we interpret the SVD of a matrix M as a list of "ingredients" ordered by "importance", and we retain only the k most important ingredients. But is this elegant procedure any "good"?

A Matrix Norm

For an $m \times n$ matrix $M \in \mathbb{R}^{m \times n}$, let $\hat{M}$ be a low-rank approximation of M, and define the approximation error as $E = M - \hat{M}$. Intuitively, a "good" approximation will lead to "small" error E. But we need to quantify the "size" of $E \in \mathbb{R}^{m \times n}$. We saw that for vectors $x \in \mathbb{R}^n$, the right way to quantify the size of x was through its norm $\|x\|$, where $\|\cdot\|$ is a function that needs to satisfy the axioms of a norm

1. $\|ax\| = |a|\,\|x\|$ for all $x \in \mathbb{R}^n$, $a \in \mathbb{R}$
2. $\|x\| \geq 0$ for all $x \in \mathbb{R}^n$, and $\|x\| = 0$ if and only if $x = 0$
3. $\|x + y\| \leq \|x\| + \|y\|$ for all $x, y \in \mathbb{R}^n$.

It turns out we can define functions on the vector space of $m \times n$ matrices that satisfy these same properties: these are called matrix norms. We'll introduce one of them here that is particularly relevant to low-rank matrix approximations, but be aware that just as for vectors, there are many kinds of matrix norms.

The Frobenius Norm

The Frobenius norm of a $m \times n$ matrix $M \in \mathbb{R}^{m \times n}$ simply computes the Euclidean norm of M as if it were a mn vector:

$$\|M\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} m_{ij}^2} \,. \qquad (F)$$

Example $M = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ has $\|M\|_F^2 = 1^2 + 2^2 + 3^2 + 4^2$
$= 30.$

which is the same as $\|\mathrm{vec}(M)\|_2^2 = \left\| \begin{bmatrix} 1 \\ 3 \\ 2 \\ 4 \end{bmatrix} \right\|^2$.

We need a couple of properties of the Frobenius norm before we can connect the SVD to low-rank matrix approximation.

Property 1: For $A \in \mathbb{R}^{n \times n}$ a square matrix, $\|A\|_F = \|A^T\|_F$.

This isn't too hard to check from the definition of (F): taking the transpose just swaps the role (i,j) in the sum, but you still end up adding together the square of all entries in A, which are the same as the square of all of the entries in A^T.

Property 2: If $Q \in \mathbb{R}^{n \times n}$ is an orthogonal matrix and $A \in \mathbb{R}^{n \times n}$ is a square matrix, then $\|QA\|_F = \|AQ\|_F = \|A\|_F$, i.e., the Frobenius norm of a matrix A is unchanged by left or right multiplication by an orthogonal matrix.

To see why this is true, recall that if $A = [a_1 \cdots a_n]$ are the columns of A, then $QA = [Qa_1 \cdots Qa_n]$. Then, since we can write the Frobenius norm squared of a matrix as the sum of the Euclidean norm squared of its columns, we have:

$$\|QA\|_F^2 = \|Qa_1\|^2 + \cdots + \|Qa_n\|^2$$
$$= \|a_1\|^2 + \cdots + \|a_n\|^2 = \|A\|_F^2.$$

Here, the second equality holds because multiplying a vector by an orthogonal matrix does not change its Euclidean norm. Finally we use this and Property 1 to conclude

$$\|AQ\|_F = \underbrace{\|Q^T A^T\|_F = \|A^T\|_F}_{\text{since } Q^T \text{ is also orthogonal}} \underbrace{= \|A\|_F}_{\text{property 1}}.$$

We will measure the quality of our rank k approximation (SVD-k) $\hat{M}$ to M in terms of the Frobenius norm of their difference.

The following theorem tells us that the SVD-based approximation (SVD-k) is optimal with respect to the Frobenius norm of the approximation error!

Theorem: For every $m \times n$ matrix $M \in \mathbb{R}^{m \times n}$, every rank target $k \geq 1$, and every rank k $m \times n$ matrix $B \in \mathbb{R}^{m \times n}$,

$$\|M - \hat{M}_k\|_F \leq \|M - B\|_F$$

where $\hat{M}_k$ is the rank k approximation derived from the SVD $M = U \Sigma V^T$ as in (SVD-k).

We won't formally prove this theorem, but let's get some intuition as to why this is true. To keep things simple, we'll assume M is square and full rank, i.e., $M \in \mathbb{R}^{n \times n}$ with rank $M = n$. Nearly the exact same argument works for general M, but we have to use the non-compact SVD of M (which keeps zero singular values around).

Our goal is to find a rank k matrix $\hat{M}$ which minimizes $\|\hat{M}-M\|_F^2$. Let $M = U\Sigma V^T$ be the SVD of M, where $U, \Sigma, V \in \mathbb{R}^{n\times n}$ since rank $M = n$. By Property 2 of the Frobenius norm, we then have the following sequence of equalities:

$$\begin{aligned}
\|\hat{M}-M\|_F^2 &= \|\hat{M}-U\Sigma V^T\|_F^2 \\
&= \|U^T(\hat{M}-U\Sigma V^T)\|_F^2 \qquad (\|A\|_F = \|QA\|_F) \\
&= \|U^T\hat{M} - \Sigma V^T\|_F^2 \qquad (U^TU = I) \\
&= \|(U^T\hat{M} - \Sigma V^T)V\|_F^2 \qquad (\|A\|_F = \|AQ\|_F) \\
&= \|U^T\hat{M}V - \Sigma\|_F^2 \qquad (V^TV = I)
\end{aligned}$$

Now notice that since Σ is a diagonal matrix, any non-diagonal entry in $U^T\hat{M}V$ adds to our approx error, so $U^T\hat{M}V$ should be diagonal. Let's posit $\hat{M} = UDV^T$ for some diagonal matrix D. Then

$$\|\hat{M}-M\|_F^2 = \|U^T(UDV^T)V - \Sigma\|_F^2 = \|D-\Sigma\|_F^2 = \sum_{i=1}^{n}(d_{ii}-\sigma_i)^2. \qquad (*)$$

Therefore, we want to pick the diagonal entries d_{ii} of D to minimize the right-most expression in $(*)$. If there were no rank restriction on $\hat{M}$, we simply would set $d_{ii} = \sigma_i$. However, notice $\hat{M} = UDV^T$ is an SVD of $\hat{M}$! Therefore, for $\hat{M}$ to be rank k, only k of the d_{ii} can be nonzero: if we can only knock off k of the $(d_{ii}-\sigma_i)^2$ terms in $(*)$, we should pick the top k, i.e., $d_{ii} = \sigma_i$ for $i = 1, \dots, k$, and $d_{ii} = 0$ for $i = k+1, \dots, n$.

Then, $$\hat{M} = [u_1 \cdots u_k \; u_{k+1} \cdots u_n]\begin{bmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_k & & \\ & & & 0 & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix}\begin{bmatrix} v_1^T \\ \vdots \\ v_k^T \\ v_{k+1}^T \\ \vdots \\ v_n^T \end{bmatrix} = \sum_{i=1}^{k} \sigma_i u_i v_i^T = U_k \Sigma_k V_k^T,$$

is exactly the expression in (SVD-k), and the square approximation error it incurs is

$$\|\hat{E}\|_F^2 = \|\hat{M}-M\|_F^2 = \sum_{i=k+1}^{n} \sigma_i^2,$$

i.e. the sum of the squares of the "tail" singular values of M.

<u>Example:</u> Recall the matrix $A = \begin{bmatrix} 4 & 11 & 14 \\ 8 & 7 & -2 \end{bmatrix}$ from Lecture 18; we computed its SVD as:

$$A = \begin{bmatrix} 3/\sqrt{10} & 1/\sqrt{10} \\ 1/\sqrt{10} & -3/\sqrt{10} \end{bmatrix}\begin{bmatrix} 6\sqrt{10} & 0 \\ 0 & 3\sqrt{10} \end{bmatrix}\begin{bmatrix} 1/3 & 2/3 & 2/3 \\ -2/3 & -1/3 & 2/3 \end{bmatrix} = U\Sigma V^T.$$

A is rank 2, and its rank 1 approximation is, according to (SVD-k), given by

$$\hat{A}_1 = \begin{bmatrix} 3/\sqrt{10} \\ 1/\sqrt{10} \end{bmatrix} 6\sqrt{10} \begin{bmatrix} 1/3 & 2/3 & 2/3 \end{bmatrix} = \begin{bmatrix} 6 & 12 & 12 \\ 2 & 4 & 4 \end{bmatrix}$$

If we compute $\|\hat{A}_1 - A\|_F^2$ we get:

$$\left\| \begin{bmatrix} 2 & -1 & 2 \\ -6 & -3 & 6 \end{bmatrix} \right\|_F^2 = 2^2 + (-1)^2 + 2^2 + (-6)^2 + (-3)^2 + 6^2$$

$$= 90$$

which is exactly $\sigma_2^2 = (3\sqrt{10})^2 = 90$.

Finally, we address an obvious question when applying these ideas in practice: how should we pick the rank k of our approximation?

In a perfect world, the singular values of the original data matrix will give strong guidance: if the top few singular values are much larger than the rest, then the obvious solution is to take k = # of big values. This was the case in the Landsat example last class: the 1st singular value was significantly larger than others, suggesting a rank 1 approximation would be a good choice (this was the image (d)).

In less clear settings, the rule of thumb is to take k as small as possible while still providing a "useful" approximation of the original data. For example, it is common to choose k so that the sum of the top k singular values is at least c times larger than the sum of the other singular values. The ratio c is typically a domain-dependent constant picked based on the application.