Lecture 20

SVD & Least-Squares
Power Method & PageRank

Owen L. Lewis

Department of Mathematics and Statistics
University of New Mexico

Nov. 5, 2024

SVD: Singular Value Decomposition

SVD takes an $m \times n$ matrix A and factors it:

$$A = USV^T$$

where U ($m \times m$) and V ($n \times n$) are orthogonal. S ($m \times n$) is diagonal, with the diagonal entries made up of “singular values”:

$$\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_r \geqslant \sigma_{r+1} = \cdots = \sigma_p = 0$$

Here, $r = rank(A)$ and $p = min(m, n)$.

Recall

We want

$$A = USV^T$$

multiply by A^T from the left gives

$$A^TA = VS^TSV^T = VS^2V^T.$$

So V and S^2 contain the eigenvectors & eigenvalues of A^TA. Similarly,

$$AA^T = US^2U^T.$$

So U is the matrix of eigenvectors of AA^T (S^2 is also the eigenvalues of AA^T).

In the end...

We get

$$A = \begin{bmatrix} \vdots & \vdots & \vdots \\ u_1 & \dots & u_m \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_r & & \\ & & & \ddots & \\ & & & & 0 \end{bmatrix} \begin{bmatrix} \dots & v_1^T & \dots \\ \dots & \vdots & \dots \\ \dots & v_n^T & \dots \end{bmatrix}$$

Example

Decompose

$$A = \begin{bmatrix} 2 & -2 \\ 1 & 1 \end{bmatrix}$$

First construct $A^T A$:

$$A^T A = \begin{bmatrix} 2 & 1 \\ -2 & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 5 & -3 \\ -3 & 5 \end{bmatrix}$$

Eigenvalues: $\lambda_1 = 8$ and $\lambda_2 = 2$. So

$$S^2 = \begin{bmatrix} 8 & 0 \\ 0 & 2 \end{bmatrix} \quad \Rightarrow \quad S = \begin{bmatrix} 2\sqrt{2} & 0 \\ 0 & \sqrt{2} \end{bmatrix}$$

Example

Now find V^T and U. The columns of V^T are the eigenvectors of A^TA.

- $\lambda_1 = 8$: $(A^TA - \lambda_1 I)v_1 = 0$

$$\Rightarrow \begin{bmatrix} -3 & -3 \\ -3 & -3 \end{bmatrix} v_1 = 0 \quad \Rightarrow \quad \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} v_1 = 0 \quad \Rightarrow \quad v_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -\sqrt{2}/2 \\ \sqrt{2}/2 \end{bmatrix}$$

- $\lambda_2 = 2$: $(A^TA - \lambda_2 I)v_2 = 0$

$$\Rightarrow \begin{bmatrix} 3 & -3 \\ -3 & 3 \end{bmatrix} v_2 = 0 \quad \Rightarrow \quad \begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix} v_2 = 0 \quad \Rightarrow \quad v_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} \sqrt{2}/2 \\ \sqrt{2}/2 \end{bmatrix}$$

- Finally:

$$V = \begin{bmatrix} -\sqrt{2}/2 & \sqrt{2}/2 \\ \sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix}$$

Example

Now find U. The columns of U are the eigenvectors of AA^T.

- $\lambda_1 = 8$: $(AA^T - \lambda_1 I)u_1 = 0$

$$\Rightarrow \begin{bmatrix} 0 & 0 \\ 0 & -6 \end{bmatrix} u_1 = 0 \quad \Rightarrow \quad \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} u_1 = 0 \quad \Rightarrow \quad u_1 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

- $\lambda_2 = 2$: $(AA^T - \lambda_2 I)u_2 = 0$

$$\Rightarrow \begin{bmatrix} 6 & 0 \\ 0 & 0 \end{bmatrix} u_2 = 0 \quad \Rightarrow \quad \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} u_2 = 0 \quad \Rightarrow \quad u_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

- Finally:

$$U = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

- Together:

$$A = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2\sqrt{2} & 0 \\ 0 & \sqrt{2} \end{bmatrix} \begin{bmatrix} -\sqrt{2}/2 & \sqrt{2}/2 \\ \sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix}$$

Eigenvalues seem important!

The SVD is just one situation where we might want to calculate the eigenvalues/eigenvectors of some matrix.

This is not hard for 2×2 or 3×3 matrices…but what if we have a large matrix?

Eigenvalues, Eigenvectors

- For a matrix A, the scalar-vector pairs (λ, v) such that

$$Av = \lambda v, \qquad v \neq 0$$

  are eigenvalue-eigenvectors.

- For example, $A = \begin{bmatrix} 3 & 2 \\ 7 & -2 \end{bmatrix}$,
  - eigenvalues: $\lambda_1 = 5, \lambda_2 = -4$
  - eigenvectors:

$$v_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \qquad \begin{bmatrix} 2 \\ -7 \end{bmatrix}$$

  - Verify that $Av_1 = \lambda_1 v_1$ and $Av_2 = \lambda_2 v_2$

Computing Eigenvalues and Eigenvectors

- If either is given, then computing the other is straightforward.
- For example, if an eigenvalue λ is given, then compute eigenvector by solving:

$$(A - \lambda I)v = 0$$

- If eigenvector v given?

$$\lambda = (Av)_i / v_i$$

Computing Eigenvalues and Eigenvectors

- Characteristic polynomial
$$\det(A - \lambda I)$$
- Roots of the characteristic polynomial are the eigenvalues
- Example: $A = \begin{bmatrix} 3 & 2 \\ 7 & -2 \end{bmatrix}$
- $\det(A - \lambda I) = (\lambda - 5)(\lambda + 4)$
- Note: eigenvalues can be complex/imaginary.
- Numerically, the characteristic polynomial is not evaluated directly
- We look at an approach to find eigenvectors one at a time...

Power Method

Suppose that A is $n \times n$ and that the eigenvalues are ordered:

$$|\lambda_1| > |\lambda_2| \geqslant |\lambda_3| \geqslant \cdots \geqslant |\lambda_n|$$

We have a linearly independent set of v_i such that $Av_i = \lambda_i v_i$.

Goal

Computing the value of the largest (in magnitude) eigenvalue, λ_1.

Notes

Power Method

Take a guess at the eigenvector, $x^{(0)}$ associated with λ_1. We know

$$x^{(0)} = c_1 v_1 + \cdots + c_n v_n$$

For simplicity, say all $c_j = 1$:

$$x^{(0)} = v_1 + \cdots + v_n$$

Then compute

$$\begin{aligned}
x^{(1)} &= Ax^{(0)} \\
x^{(2)} &= Ax^{(1)} \\
x^{(3)} &= Ax^{(2)} \\
&\vdots \\
x^{(k+1)} &= Ax^{(k)}
\end{aligned}$$

Power Method

Or $x^{(k)} = A^k x^{(0)}$. Or

$$
\begin{aligned}
x^{(k)} &= A^k x^{(0)} \\
&= A^k v_1 + \cdots + A^k v_n \\
&= \lambda_1^k v_1 + \ldots \lambda_n^k v_n
\end{aligned}
$$

And this can be written as

$$x^{(k)} = \lambda_1^k \left(v_1 + \left(\frac{\lambda_2}{\lambda_1}\right)^k v_2 + \cdots + \left(\frac{\lambda_n}{\lambda_1}\right)^k v_n \right)$$

So as $k \to \infty$, we are left with

$$x^{(k)} \to \lambda^k v_1$$

The Power Method (with normalization)

```
for k = 1 to kmax
    y⃗ = A x⃗
    r = φ(y⃗)/φ(x⃗)
    x⃗ = y⃗/‖y⃗‖
```

- Why normalization? (overflow, underflow)
- ϕ just needs to be some “measurable” quantity (scalar).
- Algorithm converges to $r \approx \lambda_1$ and $x \approx v_1$

- If $\phi(\vec{z}) = \vec{x} \cdot \vec{z}$, then
$$r = \frac{\vec{x} \cdot A\vec{x}}{\vec{x} \cdot \vec{x}},$$
is called the Rayleigh quotient
- Often $\phi(\vec{z}) = z_1$ (i.e., the first nonzero component of z) is sufficient

Finding next eigenvalue

Suppose we know λ_1 and $\vec{v}_1$, now what?

Construct a new matrix B such that A and B have the same eigenvectors, but B has eigenvalues $0, \lambda_2, \lambda_3, \ldots$.

$$\vec{x} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_n\vec{v}_n.$$

$$A\vec{x} = 0\vec{v}_1 + c_2\lambda_2\vec{v}_2 + \ldots + c_n\,\lambda_n\vec{v}_n.$$

Power method *can't* pick out λ_1 and $\vec{v}_1$.
It will find λ_2 and $\vec{v}_2$.

Inverse Power Method

- We now want to find the smallest eigenvalue
- $Av = \lambda v \quad \Rightarrow \quad A^{-1}v = \frac{1}{\lambda}v$
- The smallest eigenvalue of A is now the dominant eigenvalue of A^{-1}
- So "apply" power method to A^{-1} (assuming a distinct smallest eigenvalue)
- $x^{(k+1)} = A^{-1}x^{(k)}$
- Efficiently with $A = LU$

$$Lz^{(k)} = x^{(k)}, \qquad Ux^{(k+1)} = z^{(k)}$$

Uses of Power Method

Google PageRank algorithm.

Google (basics)

many slides courtesy of T. Chartier at Davidson

- A component of Google's success can be attributed to the PageRank$^{\mathrm{TM}}$ algorithm developed by Google's founders
- Algorithm determined entirely by link structure of the WWW
- Recomputed once a month (or it was in the past...)
- Involves no content of any Web page
- How are the search results ordered?

Transition Matrices for Markov Processes

If there are n possible states, then let $v = [p_1, \ldots p_n]'$ be a vector of probabilities that we are "currently" in each state ($\sum p_i = 1$).

A is the "transition matrix" for the process if $A(i, j)$ equals the probability of going from state j to state i. The column sum of A must be all ones: $\sum_i A(i, j) = 1$.

A's largest eigenvalue is 1 and the associated eigenvector v_1 is the "stationary distribution": $v_1 = [p_1, \ldots p_n]'$, where p_i is the probability that we are in state i "at the end of time".

Notes

PageRank and a random web crawler

Google's idea was to form a "random" web crawler, view it as a Markov process, and find its stationary state.

PageRank and a random web crawler

- Let W be the set of Web pages that can reached by following a chain of hyperlinks starting from a page at Google.
- Let n be the number of pages in W.
- The set W actually varies with time, and can be in the billions, trillion, ...?
- Let G be the $n \times n$ connectivity matrix of W, that is, $G_{i,j}$ is 1 if there is a hyperlink to page i from page j and 0 otherwise.

Google and Probability

- Let c_j and r_i be the column and row sums of G, respectively. That is,

$$c_j = \sum_i G_{i,j}, \qquad r_i = \sum_j G_{i,j}$$

- Then r_k and c_k are the indegree and outdegree of the k-th page.
  - In other words, r_k is the number of links into page k
  - And c_k is the number of links from page k.
- Let p be the fraction of time that the random walk follows a specific link.
- Google typically takes this to be $p = 0.85$.
- Then $1 - p$ is the fraction of time that any arbitrary page is chosen.

Google meets Markov

- Let A be an $n \times n$ matrix whose elements are

$$A_{i,j} = \begin{cases} p\,G_{i,j}/c_j + \delta, & c_j \neq 0 \\ 1/n, & c_j = 0 \end{cases}$$

  where $\delta = (1-p)/n$.
- This matrix is the transition matrix of the Markov chain of a random walk!
- Notice that A comes from scaling the connectivity matrix by its column sums.
- The j-th column is the probability of jumping from the j-th page to the other pages on the Web.

Eigenvectors and Google

Find $x = Ax$ and the elements of x are Google's PageRank.

- That is, x is the result of an infinite walk

  $$x \leftarrow A * A * \cdots * Av$$

  for any starting vector v.
- And element i of x (i.e., x_i) is the probability that you ended up on page i

For any particular search string,

- First, Google finds pages on the Web that match the search string
- Second, the pages are listed in the order of their PageRank.

To search from a homepage, you will type a statement like:

```
[U,G] = surfer('http://www.unm.edu',n).
```

This starts at the given URL and tries to surf the Web until it has visited *n* pages. That is, an *n* by *n* matrix is formed.

Note, `surfer.m` is very primitive...

- Download `surfer.m`, `pagerank.m` and `pagerankpow.m`
- `[U,G] = surfer('http://www.unm.edu',20);`

  U = a cell array of n strings, the URLs of the nodes.

  G = an n-by-n sparse matrix with $G(i,j) = 1$ if node j is linked to node i.
- Next: `spy(G)`. This shows the nonzero structure of the connectivity matrix.
- Finally: `pagerank(U,G)` and the pagerank will be computed.

Summary

1. Form transition matrix A from Google matrix G.
2. Find $x = Ax$ (with the Power method) and the elements of x are Google's PageRank.
   - That is, x_i is the probability that an infinite random walk (web surf) ended up on page i
   - Largest absolute value of an eigenvalue for a stochastic matrice like A is always 1. That is probabilities do not tend towards infinity, or zero.
3. PageRank reduces to an eigenvector problem!