CS 450 Final Project: Computing Singular Value Decomposition (SVD) Via Matrix Diagonalization

Enyi Jiang (enyij2@illinois.edu)

May 10, 2022

1 Introduction

Singular Value Decomposition (SVD), as introduced in class, is a matrix factorization method that generalizes the eigen-decomposition of a square matrix $n \times n$ to any matrix $n \times m$. Following shows the general formula of SVD (Figure 1),

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathbf{T}$$

where $\mathbf{U}$ is an $m \times m$ orthogonal matrix; $\mathbf{\Sigma}$ is an $m \times n$ diagonal matrix with $\sigma_\mathbf{i} > 0$ on the diagonal; $\mathbf{V}$ is an $n \times n$ orthogonal matrix. In the class, we learned how to use SVD method to solve the Linear Least Squares Problem. However, in my final project, I combined the ideas from chapter 5 of iteration methods to find matrices $\mathbf{\Sigma}$, $\mathbf{U}$, and $\mathbf{V}$ iteratively with/without using the covariance matrix of the original data. The basic idea is to construct a square matrix and find its eigenvalues and eigenvectors, which are equal to or have a mathematical relationship with the original matrix's SVD results.

2 Method

2.1 Interpretation of SVD in Terms of Covariance Matrices

If we have a matrix $\mathbf{X}$ with a shape $n \times p$, where n is the number of samples and p is number of the dimensions we want to reduce from. We first need to **normalize X** to make the following math easier. Then the $p \times p$ covariance matrix $\mathbf{C}$ is given by $\mathbf{C} = \mathbf{X}^\top\mathbf{X}/(n-1)$. It is a **symmetric** matrix and so it can be **diagonalized**:

$$\mathbf{C} = \mathbf{V}\mathbf{L}\mathbf{V}^\mathbf{T}$$

where $\mathbf{V}$ is a matrix of eigenvectors (each **column** is an eigenvector) and $\mathbf{L}$ is a diagonal matrix with decreasing eigenvalues λ_i on the diagonal.

On the other hand, we do SVD on the original $\mathbf{X}$ and can get a decomposition as follows:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathbf{T}$$

From here one can easily see that

$$\mathbf{C} = \mathbf{V}\mathbf{\Sigma}\mathbf{U}^\top\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top/(n-1) = \mathbf{V}\frac{\mathbf{\Sigma}^2}{n-1}\mathbf{V}^\top$$

Thus, in this case, rows of the eigenvectors $\mathbf{V}^\mathbf{T}$ of the covariance matrix $\mathbf{C}$ are the same as $\mathbf{V}^\mathbf{T}$ of SVD results. Also, singular values are related to the eigenvalues of covariance matrix $\mathbf{C}$ via $\lambda_i = \sigma_i^2/(n-1)$.

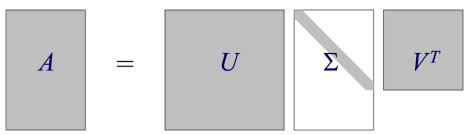


Figure 1: Visualization of SVD method.

2.2 Power Iteration Methods

(1) Use Normalized Power Iteration to find one pair of (eigenvalue, eigenvector) for the covariance matrix $\mathbf{C}$, which corresponding to the **major** component of SVD. Using the mathematical equation obtained above, I could compute the actual singular value from this eigenvalue. Here, the eigenvector is the same as the first row of $\mathbf{V^T}$ in SVD.

(2) Use Normalized Power Iteration to find one pair of (eigenvalue, eigenvector) for the square matrix $\mathbf{X^T X}$, which corresponding to the **major** component of SVD. Eigenvector is the same as the first row of $\mathbf{V^T}$ in SVD; singular value is the square root of the eigenvalue ($\sigma_i = \sqrt{\lambda_i}$).

(3) Use Normalized Power Iteration to find k pairs of (eigenvalue, eigenvector) for the square matrix $\mathbf{A} = \mathbf{X^T X}$, which corresponding to part of/all the components of SVD. I compute $\mathbf{v}_1$ from applying (2). After that, I use $\mathbf{v}_1$ to compute $\mathbf{u}_1 = \mathbf{A}\mathbf{v}_1/\sigma_1$. Then I simply subtract the rank 1 component of $\mathbf{A}$ corresponding to $\mathbf{v}_1$, i.e., set

$$\mathbf{A}' = \mathbf{A} - \sigma_1 \mathbf{u}_1 \mathbf{v_1^T}$$

Then it's easy to see that $\sigma_1(\mathbf{A}') = \sigma_2(\mathbf{A})$ and basically all the singular vectors shift indices by 1 when going from $\mathbf{A}$ to $\mathbf{A}'$. Then I repeat. In the end, I can get $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{\Sigma}$ using normalized power iteration.

2.3 Orthogonal and Lanczos Iteration Methods

(1) Use Orthogonal Iteration to find k pairs of (eigenvalue, eigenvector) for the square matrix $\mathbf{A} = \mathbf{X^T X}$, which corresponding to part of/all the components of SVD. For Orthogonal iteration, the code performs QR factorization many times until the threshold is met. At that point, the **square roots** of the **diagonal** values of matrix $\mathbf{R}$ are the singular values (we can construct matrix $\mathbf{\Sigma}$ from that) and $\mathbf{V^T} = \mathbf{Q^T}$. In addition, $\mathbf{U} = \mathbf{AQ\Sigma}$.

(2) Use Lanczos Iteration to find k pairs of (eigenvalue, eigenvector) for the square matrix $\mathbf{A} = \mathbf{X^T X}$, which corresponding to part of/all the components of SVD. Using the homework code for Lanczos Iteration, I got $\mathbf{V^T} = \mathbf{Q^T}$ (same as orthogonal iteration) and $\mathbf{T}$ which consists of **square roots** of the singular values on the diagonal. Same as (1), $\mathbf{U} = \mathbf{AQ\Sigma}$.

3 Results

Code, figures, and data can be found at https://github.com/enyijiang/cs450_final_project, where a jupyter notebook named **Final Project.ipynb** consists of all the following results and visualizations.

3.1 Dataset

I conducted experiments on Beer Consumption dataset from Kaggle. After some simple data cleaning procedure, the resulting data $\mathbf{X}$ has a shape of 365×6 and I did SVD on these 6 features. Figure 2 exhibits a two-dimensional scatter plot of two features: Maximal temperature and Consumption.

3.2 Power Iteration Methods

(1) Normalized Power Iteration on the covariance matrix $\mathbf{C}$: In this setting, I used only two features: Maximal temperature and Consumption and plotted the vectors $\mathbf{v_1}$ of gold results using SVD API in NumPy and my results of normalized power iteration. In Figure 3, we can see left and right absolute eigenvectors are identical. For the singular value, I got $\lambda_1 = 31.21 = \frac{106.58^2}{364} = \sigma_1^2/(n-1)$ which confirms the math relationship.

(2) Normalized Power Iteration on $\mathbf{A} = \mathbf{X^T X}$: this function returns the same singular value and $\mathbf{v_1}$ as the gold approach. This part of the code already took the square root of the eigenvalue before returning it.

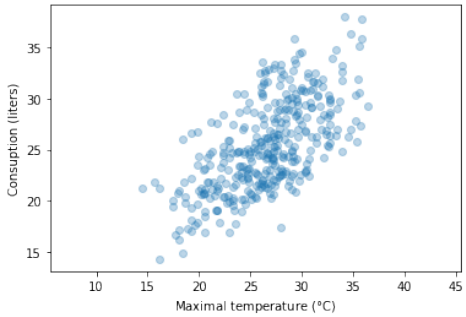
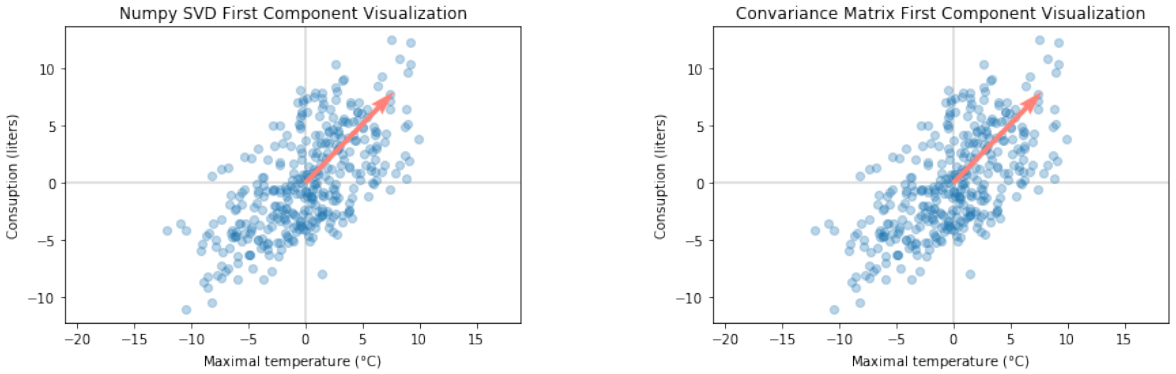


Figure 2: Scatter plot of two features in Beer dataset.



(a) SVD in NumPy

(b) Normalized Power Iteration on $\mathbf{C}$

Figure 3: Visualization of Gold and Power Iteration Results

(3) I ran Normalized Power Iteration for $k = 3$ times and compared its results with the gold results. Confirmed their closeness after calling *np.allclose*() function. More results are shown in the jupyter notebook.

3.3 Orthogonal and Lanczos Iteration Methods

I ran both methods for $k = 3$ times and compared its results with the gold results. Confirmed their closeness after calling *np.allclose*() function. More results are shown in the jupyter notebook.

4 Conclusion

In this project, I successfully used iteration methods introduced in class to compute matrices $\mathbf{\Sigma}$, $\mathbf{U}$, and $\mathbf{V}$ iteratively with sufficient accuracy. Throughout this process, I got a deeper understanding of SVD and how its singular values are correlated to eigenvalues when doing matrix diagonalization. Besides, I got more familiar with linear algebra at the same time. In short, it is a very exciting and fulfilling learning experience for me.

5 References

1. Lecture slides.
2. https://towardsdatascience.com/simple-svd-algorithms-13291ad2eef2
3. https://en.wikipedia.org/wiki/Singular_value_decomposition
4. https://github.com/RRisto/learning/blob/master/linear_algebra_learn/PCA_SVD/power_method.ipynb
5. https://stats.stackexchange.com/questions/134282/relationship-between-svd-and-pca-how-to-use-svd-t

6. https://jeremykun.com/2016/05/16/singular-value-decomposition-part-2-theorem-proof-algorithm
7. https://www.kaggle.com/datasets/dongeorge/beer-consumption-sao-paulo