

Images haven't loaded yet. Please exit printing, wait for images to load, and try to print again.

Decomposition with Interactive Code (feat Peter Mills)



Jae Duk Seo

Follow

Jul 15, 2018 · 8 min read



GIF from this website

I wanted to gain deeper understanding of singular value decomposition, how it is done and how we can implement in python.

Please note that this post is to help my understanding of linear algebra in the context of Machine Learning. Specifically, singular value decomposition.

. . .

Simple overview of linear algebra / Warning for Numpy/Scipy Users



Image from this website

If anyone wishes to brush up the knowledge regarding linear algebra please click here to read a [great blog post](#). The authors of the post have done a great job at explaining the basic mechanisms of linear algebra.

Paper from this website

And for Numpy / Scipy users please take note of the pdf that I have linked above. The authors of that paper did a good job on reasons why computing Eigenvalue using Numpy might be unstable. (And I will cover this in the later part of this post as well, but the results when using Numpy is can be unstable.)

. . .

Theoretical Review of SVD



Image from this website

For a theoretical explanation of SVD please check this [blog post](#) out, it is actually one of the best blog post for SVD. And among that blog post, I want to attempt on solving some example problems Peter have given us. (But first we need to see some of the given equation Peter expressed SVD as $A = USV.T$)

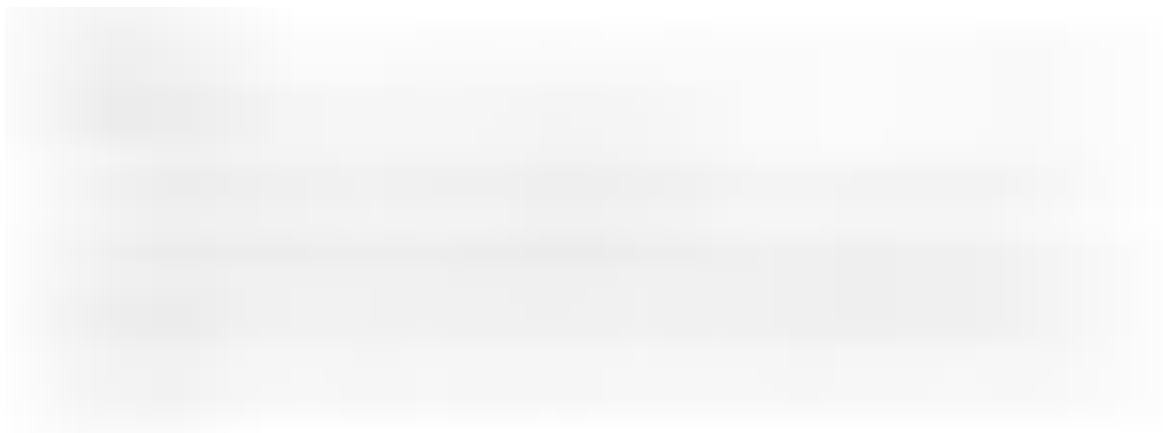
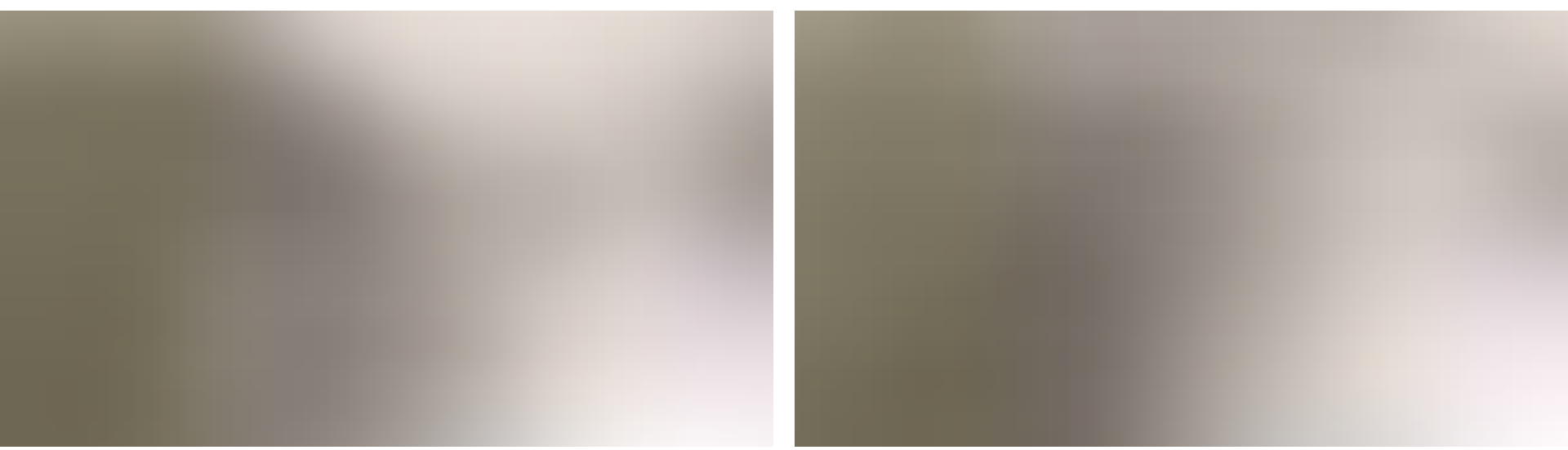
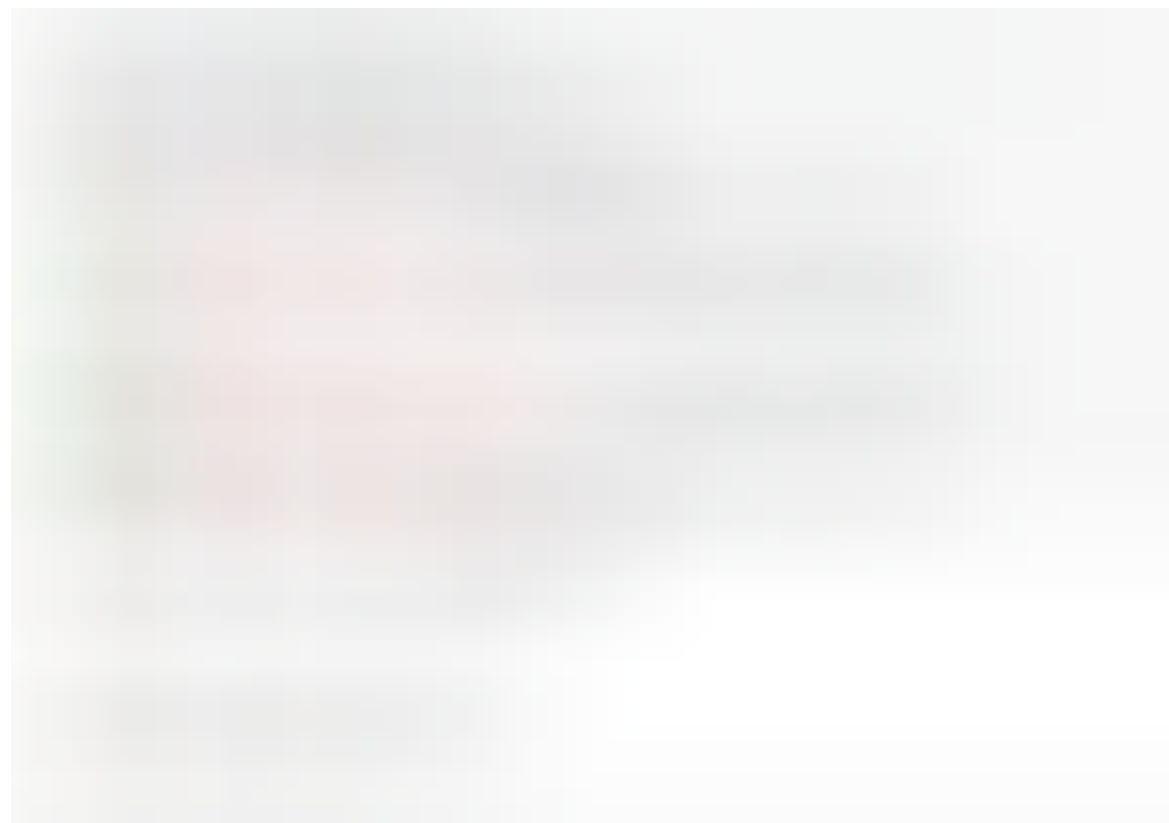


Image from this website

Although I am not 100 % sure if these are the correct answers below are my attempts to solve the given problem.



For the reason why S is a positive number, is because it is squared. And there is one area that I wish to touch on before moving on. Which is this question “*Show that if $m < n$ there will be at most m non-zero singular values.*”. In practice I found that this is not always the case, since in python implementation I was able to get non-zero eigenvalues more than the number of smaller dimension in the matrix A . (An example can be seen below.)



As an example above, for a given data, when I computed the number of non zero elements for each matrix ($\text{dot}(A.T, A)$ and $\text{dot}(A, A.T)$) I was able to observe that in some cases the number of non zero eigen values did not match exactly. However, the values that are very small does match up to be the same number. I suspect that this is due to how numpy calculates the eigen values. Currently, I understand SVD as representing

the original data A in a coordinate system, where U Matrix measures how each row's are correlated with other rows, V Matrix measures how each column's are correlated with other columns, and S measures the strength of that correlation. Finally, I want to mention one additional material that really helped me.

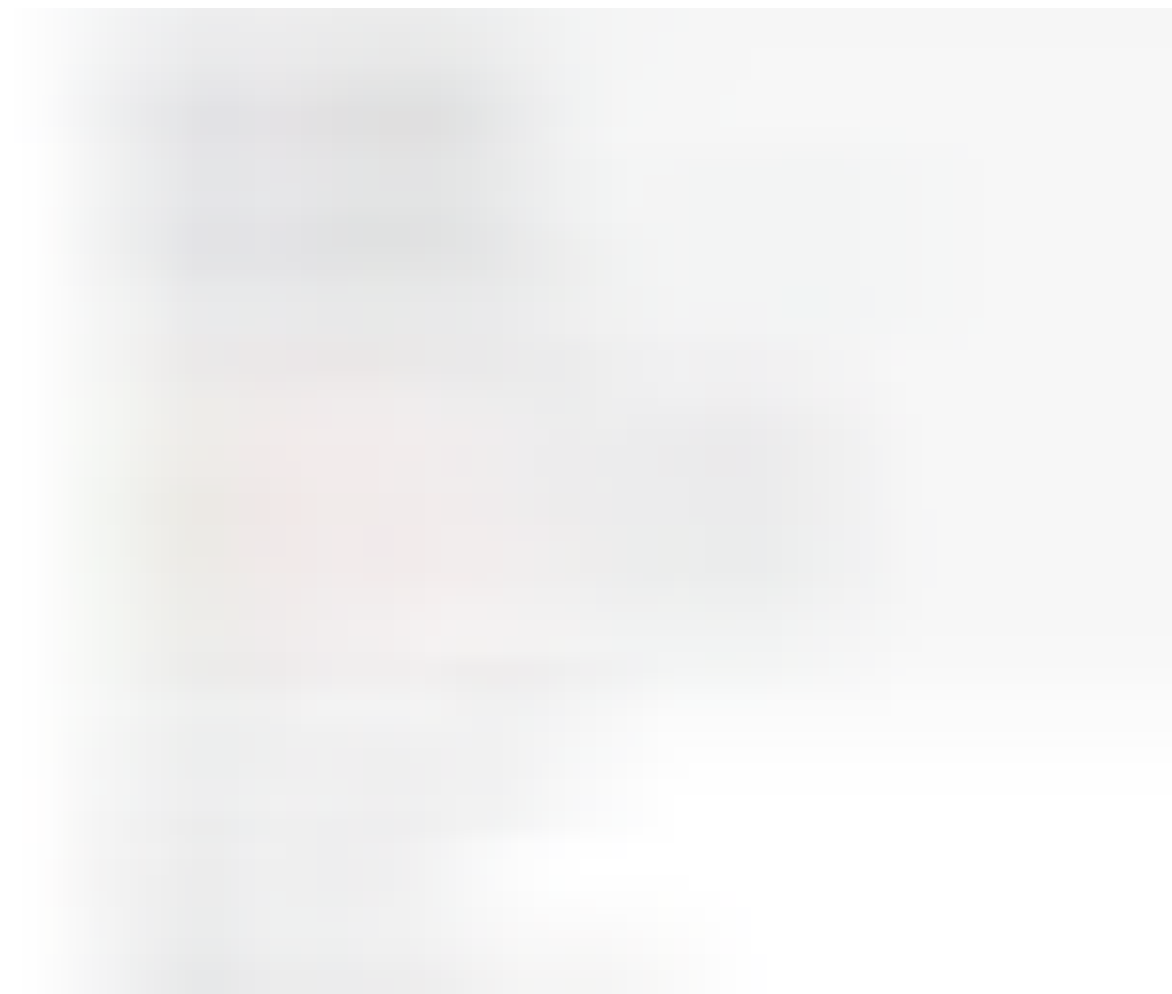
Paper from this website

Please note that I will be following the method presented on the above paper as well as from Peter's blog post.

• • •

SVD on Small Matrix

To start off simple, I have created a (2,3) matrix as seen above, and using the numpy linear algebra library, we are easily able to compute the eigen vectors for both matrix $\text{dot}(A, A.T)$ and $\text{dot}(A.T, A)$.



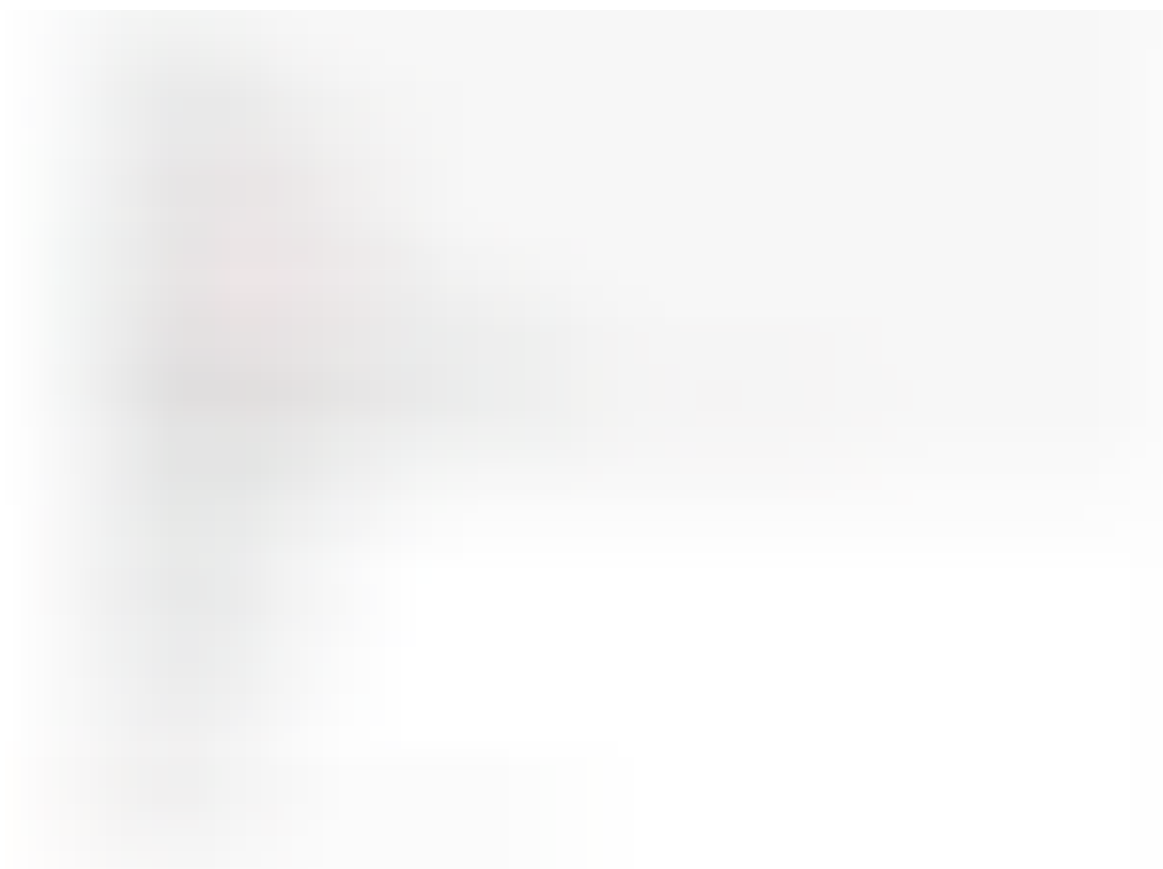
Next step is the sort the eigen value in a decreasing order and change the eigen vector matrix accordingly. And again, we can notice that the num-

ber of non-zero eigen values does not match up from one another. However, $-2e-15$ is significantly a small number and we can discard it.

Finally, using the created matrix U S and V , we can reconstruct the original data via dot product. We can just scale the reconstructed matrix (by -1) to match the original data and when we create a correlation matrix we can see something like below.



And by looking at the first two rows, we can observe that we have successfully reconstructed the original matrix.



Additionally, we can take the approach presented in Peter’s blog post and get the same results.



...

SVD on Large Matrix



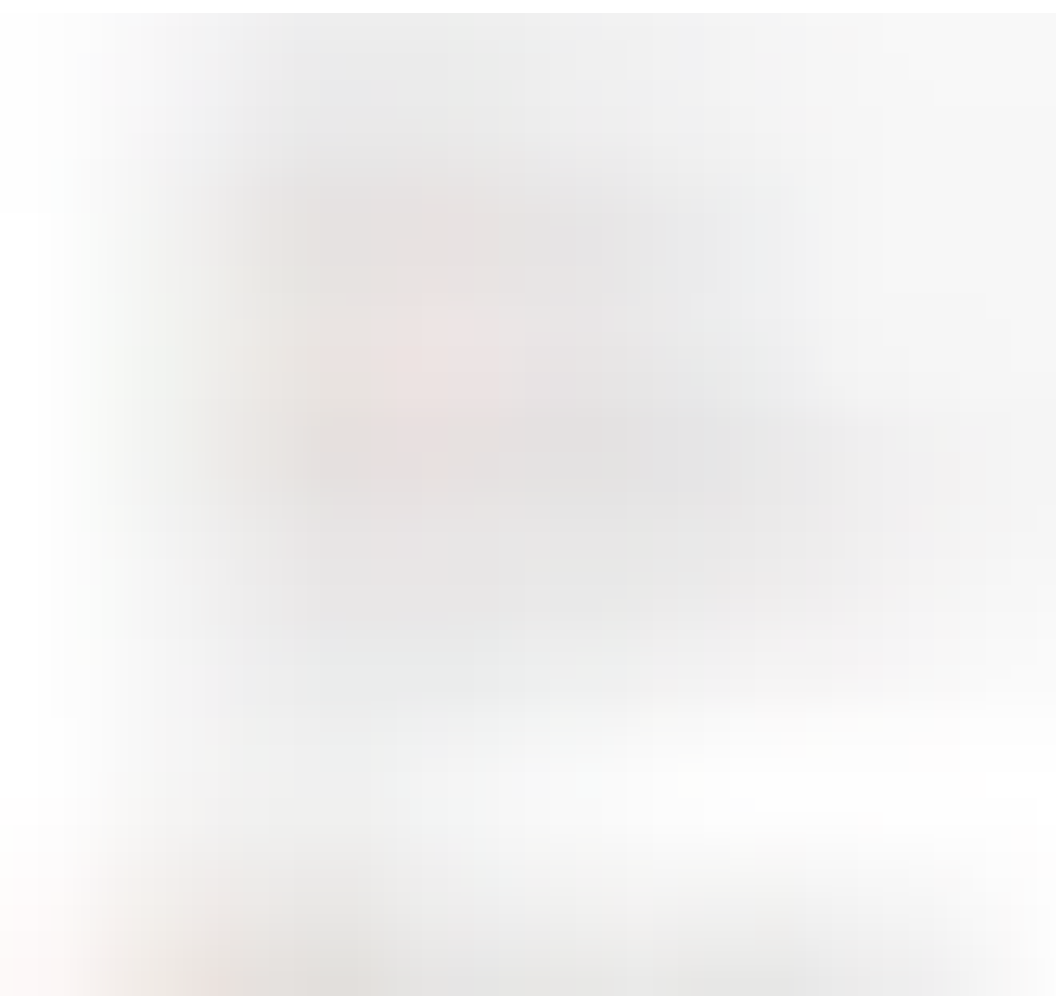
Next lets apply SVD on a larger matrix, specifically the data from load - boston. Just from the first sight we can see that we have 506 examples with 14 columns each.



And above is the generated correlation matrix as well as a scatter plot of Target vs LSTAT (which was the attribute with the most highest correlation value.)

Following the same pattern we can compute the eigen vectors and values for the given matrix A.

One unfortunate fact was that the mismatch of non-zero eigen values, this seems to be a repeated problem when using numpy.



And finally, we can compute the reconstructed matrix, however there is one problem.



The reconstructed matrix loses a lot of information, as seen above the correlation matrix does not match with the original data set as well as the scatter plot.

However, when we use the method that Peter have introduced to us in his blog post, we can get something like below.



An exact same correlation matrix as well as scatter plot from the original value. So as of now, I think performing SVD via Peter's method is a better idea.

Interactive Code



For Google Colab, you would need a google account to view the codes, also you can't run read only scripts in Google Colab so make a copy on your playground. Finally, I will never ask for permission to access your files on Google Drive, just FYI. Happy Coding!

For the code for Small [Matrix please click here.](#)

For the code for Large [Matrix please click here.](#)

. . .

Final Words

Linear Algebra is, really a beautiful topic to study.

. . .

Reference

1. An Intuitive Guide to Linear Algebra—BetterExplained. (2018). Betterexplained.com. Retrieved 5 July 2018, from <https://betterexplained.com/articles/linear-algebra-guide/>
2. Singular Value Decomposition (the SVD). (2018). YouTube. Retrieved 5 July 2018, from <https://www.youtube.com/watch?>

3. Module: data—skimage v0.15.dev0 docs. (2018). Scikit-image.org. Retrieved 15 July 2018, from <http://scikit-image.org/docs/dev/api/skimage.data.html>
4. sklearn.datasets.load_boston—scikit-learn 0.19.1 documentation. (2018). Scikit-learn.org. Retrieved 15 July 2018, from http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_boston.html#sklearn.datasets.load_boston
5. Plotting Cross-Validated Predictions—scikit-learn 0.19.1 documentation. (2018). Scikit-learn.org. Retrieved 15 July 2018, from http://scikit-learn.org/stable/auto_examples/plot_cv_predict.html#sphx-glr-auto-examples-plot-cv-predict-py
6. Faces dataset decompositions—scikit-learn 0.19.1 documentation. (2018). Scikit-learn.org. Retrieved 15 July 2018, from http://scikit-learn.org/stable/auto_examples/decomposition/plot_faces_decomposition.html#sphx-glr-auto-examples-decomposition-plot-faces-decomposition-py
7. frame, N. (2018). Normalize columns of pandas data frame. Stack Overflow. Retrieved 15 July 2018, from <https://stackoverflow.com/questions/26414913/normalize-columns-of-pandas-data-frame>
8. seaborn.heatmap—seaborn 0.8.1 documentation. (2018). Seaborn.pydata.org. Retrieved 15 July 2018, from <https://seaborn.pydata.org/generated/seaborn.heatmap.html>
9. matplotlib?, H. (2018). How do you change the size of figures drawn with matplotlib?. Stack Overflow. Retrieved 15 July 2018, from <https://stackoverflow.com/questions/332289/how-do-you-change-the-size-of-figures-drawn-with-matplotlib>
10. Python, H. (2018). How to specify x and y axis for plotting dataframe in Python. Stack Overflow. Retrieved 15 July 2018, from <https://stackoverflow.com/questions/45738773/how-to-specify-x-and-y-axis-for-plotting-dataframe-in-python>
11. method, I. (2018). Inconsistency when setting figure size using pandas plot method. Stack Overflow. Retrieved 15 July 2018, from

<https://stackoverflow.com/questions/42215252/inconsistency-when-setting-figure-size-using-pandas-plot-method>

12. pandas.DataFrame.plot—pandas 0.23.1 documentation. (2018). Pandas.pydata.org. Retrieved 15 July 2018, from <http://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.plot.html>
13. numpy.linalg.eig—NumPy v1.14 Manual. (2018). Docs.scipy.org. Retrieved 15 July 2018, from <https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.eig.html>
14. numpy.sort—NumPy v1.14 Manual. (2018). Docs.scipy.org. Retrieved 15 July 2018, from <https://docs.scipy.org/doc/numpy/reference/generated/numpy.sort.html>
15. Python, n. (2018). Python, numpy sort array. Stack Overflow. Retrieved 15 July 2018, from <https://stackoverflow.com/questions/14875248/python-numpy-sort-array/14875366>
16. order?, E. (2018). Efficiently sorting a numpy array in descending order?. Stack Overflow. Retrieved 15 July 2018, from <https://stackoverflow.com/questions/26984414/efficiently-sorting-a-numpy-array-in-descending-order>
17. VanderPlas, J. (2018). Sorting Arrays | Python Data Science Handbook. Jakevdp.github.io. Retrieved 15 July 2018, from <https://jakevdp.github.io/PythonDataScienceHandbook/02.08-sorting.html>
18. numpy.diag—NumPy v1.14 Manual. (2018). Docs.scipy.org. Retrieved 15 July 2018, from <https://docs.scipy.org/doc/numpy/reference/generated/numpy.diag.html>
19. complex?, H. (2018). How to make the eigenvalues and eigenvectors stay real instead of complex?. Stack Overflow. Retrieved 15 July 2018, from <https://stackoverflow.com/questions/48695430/how-to-make-the-eigenvalues-and-eigenvectors-stay-real-instead-of-complex>

20. Matrix Eigenvectors Calculator—Symbolab. (2018). Symbolab.-com. Retrieved 15 July 2018, from <https://www.symbolab.com/solver/matrix-eigenvectors-calculator/eigenvectors%20%5Cbegin%7Bpmatrix%7D10%260%262%5C%5C%200%2610%264%5C%5C%20%264%262%5Cend%7Bpmatrix%7D>
21. [Only Numpy] Having Fun with Eigen Value s/ Vectors with Interactive Code in Numpy. (2018). Towards Data Science. Retrieved 15 July 2018, from <https://towardsdatascience.com/only-numpy-having-fun-with-eigen-value-s-vectors-with-interactive-code-in-numpy-80d3443dfd22>
22. Independent Component Analysis via Gradient Ascent in Numpy and Tensorflow with Interactive Code. (2018). Towards Data Science. Retrieved 15 July 2018, from <https://towardsdatascience.com/independent-component-analysis-via-gradient-ascent-in-numpy-and-tensorflow-with-interactive-code-98b9a73e5d19>
23. Google Colaboratory. (2018). Colab.research.google.com. Retrieved 15 July 2018, from <https://colab.research.google.com/drive/1vDozsX3CkCGMyw-MutMiUTR4PhfjKAZz2#scrollTo=W5nfPzyB6jfS>
24. Anon, (2018). [online] Available at: https://www.researchgate.net/publication/281455336_Numpy_SciPy_Recipes_for_Data_Science_EigenvaluesEigenvectors_of_Covariance_Matrices [Accessed 15 Jul. 2018].
25. Singular Value Decomposition (SVD) tutorial. (2018). Web.mit.edu. Retrieved 15 July 2018, from http://web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm
26. sklearn.datasets.load_boston—scikit-learn 0.19.1 documentation. (2018). Scikit-learn.org. Retrieved 15 July 2018, from http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_boston.html
27. (2018). Datajobs.com. Retrieved 15 July 2018, from [https://datajobs.com/data-science-repo/SVD-Tutorial-\[Kirk-Baker\].pdf](https://datajobs.com/data-science-repo/SVD-Tutorial-[Kirk-Baker].pdf)

