**Singular Value Decomposition**

* **Unsupervised Learning + Dimensionality Reduction**

SVD is a data summary method similar to PCA. It extracts important features from data. But there is one more advantage of SVD: reconstructing original dataset into small dataset. So it has wide applications such as image compression. For example, if you have a 32\*32 = 1,024 pixel image, SVD can summary it into 66 pixels. The 66 pixels can retrieve 32\*32 pixel image without miss any important information.

SVD has been instrumental in linear algebra, but it seems “not nearly as famous as it should be” as stated in the classic textbook “Linear Algebra and Its Applications” by Gilbert Strang. To introduce SVD properly, it is essential to start with matrix operation. If A is a symmetric real n × n matrix, there exists an orthogonal matrix V and a diagonal D such that

https://miro.medium.com/max/124/1*rdEtBbHkOk4XF9dPoGQX3w.png

Columns V are eigenvectors for A and the diagonal entries of D are the eigenvalues of A. This process is called the Eigenvalue Decomposition, or EVD, for matrix A. It tells us how to choose orthonormal bases so that the transformation is represented by a matrix with the simplest possible form, that is, diagonal. (For readers who would like to go over the steps to diagonalize a matrix, [here](https://yutsumura.com/how-to-diagonalize-a-matrix-step-by-step-explanation/) is a good example.)

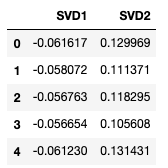
Extending the symmetric matrix, the SVD works with any real m × n matrix A. Given a real m× n matrix A, there exists an orthogonal m × m matrix U, an orthogonal matrix m × m V, and a diagonal m × n matrix Σ such that

https://miro.medium.com/max/113/1*eLi5Mt2ewpvLxEcC3FzXlQ.png

Note that an orthogonal matrix is a square matrix such that the produce of itself and its inverse matrix is an identity matrix. A diagonal matrix is a matrix in which the entries other than the diagonal are all zero.

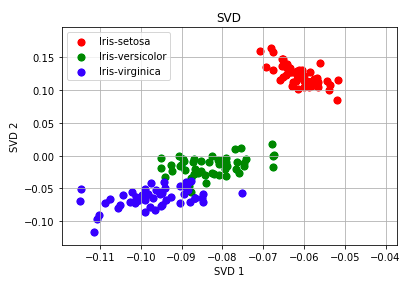
Below I will again use the iris dataset to show you how to apply SVD.

*from numpy import \*  
import operator  
import matplotlib.pyplot as plt  
import pandas as pd  
from numpy.linalg import \*url = “*[*https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data*](https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data)*"  
# load dataset into Pandas DataFrame  
df = pd.read\_csv(url, names=[‘sepal length’,’sepal width’,’petal length’,’petal width’,’target’])# Only the X variables  
data = df[[‘sepal length’,’sepal width’,’petal length’,’petal width’]]#calculate SVD  
n = 2 # We will take two Singular Values  
U, s, V = linalg.svd( data )# eye() creates a matrix with ones on the diagonal and zeros elsewhere  
Sig = mat(eye(n)\*s[:n])  
newdata = U[:,:n]  
newdata = pd.DataFrame(newdata)  
newdata.columns=[‘SVD1’,’SVD2']  
newdata.head()*



You can compare the result of SVD to that of PCA. Both achieve similar outcomes.

*# Add the actual target to the data in order to plot it  
newdata[‘target’]=df[‘target’]fig = plt.figure()  
ax = fig.add\_subplot(1,1,1)   
ax.set\_xlabel(‘SVD 1’)   
ax.set\_ylabel(‘SVD 2’)   
ax.set\_title(‘SVD’)   
targets = [‘Iris-setosa’, ‘Iris-versicolor’, ‘Iris-virginica’]  
colors = [‘r’, ‘g’, ‘b’]  
for target, color in zip(targets,colors):  
 indicesToKeep = newdata[‘target’] == target  
 ax.scatter(newdata.loc[indicesToKeep, ‘SVD1’]  
 , newdata.loc[indicesToKeep, ‘SVD2’]  
 , c = color  
 , s = 50)  
ax.legend(targets)  
ax.grid()*



**Figure: SVD**