Unsupervised Machine Learning

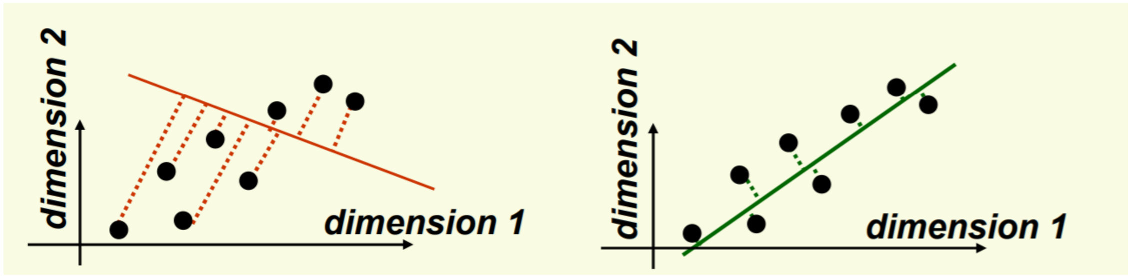
Unsupervised Learning

Principal Component Analysis Concepts

Principal Component Analysis

1. Main idea: seek most accurate data representation in a lower dimensional space

2. Example in 2-D, project data to 1-D subspace (a line) with minimal projection error



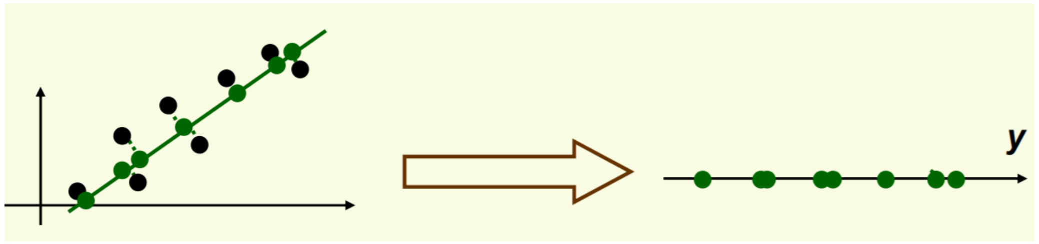
3. In both the pictures above, the data points (black dots) are projected to one line but the second line is closer to the actual points (less projection errors) than first one

4. Notice that the good line to use for projection lies in the direction of largest variance

Ref: http://www.cs.haifa.ac.il/~rita/uml_course/add_mat/PCA.pdf

Principal Component Analysis

5. After the data is projected on the best line, need to transform the coordinate system to get 1D representation for vector y



6. Note that new data y has the same variance as old data x in the direction of the green line

7. PCA preserves largest variances in the data

Ref: http://www.cs.haifa.ac.il/~rita/uml_course/add_mat/PCA.pdf

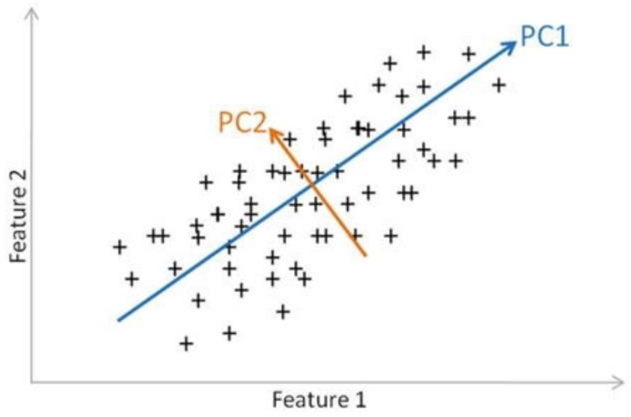
Unsupervised Machine Learning

greatlearning
Learning for Life

Principal Component Analysis

8. In general PCA on n dimensions will result in another set of new n dimensions. The one which captures maximum variance in the underlying data is the principal component 1, principal component 2 is orthogonal to it

9. Example in 2-D, project data to 1-D subspace (a line) with minimal projection error



Ref: http://www.cs.haifa.ac.il/~rita/uml_course/add_mat/PCA.pdf

Mechanics of Principal Component Analysis

http://setosa.io/ev/principal-component-analysis/

Principal Component Analysis steps

1. Begins by standardizing the data. Data on all the dimensions are subtracted from their means to shift the data points to the origin. i.e. the data is centered on the origins

2. Generate the covariance matrix / correlation matrix for all the dimensions

3. Perform eigen decomposition, that is, compute eigen vectors which are the principal components and the corresponding eigen values which are the magnitudes of variance captured

4. Sort the eigen pairs in descending order of eigen values and select he one with the largest value. This is the first principal component that covers the maximum information from the original data

Ref: http://www.cs.haifa.ac.il/~rita/uml_course/add_mat/PCA.pdf

Principal Component Analysis (Performance issues)

1. PCA effectiveness depends upon the scales of the attributes. If attributes have different scales, PCA will pick variable with highest variance rather than picking up attributes based on correlation

2. Changing scales of the variables can change the PCA

3. Interpreting PCA can become challenging due to presence of discrete data

4. Presence of skew in data with long thick tail can impact the effectiveness of the PCA (related to point 1)

5. PCA assumes linear relationship between attributes. It is ineffective when relationships are non linear

Principal Component Analysis steps

Lab-3 Principal Component Analysis on iris data set

Description – Explore the iris data set and perform PCA

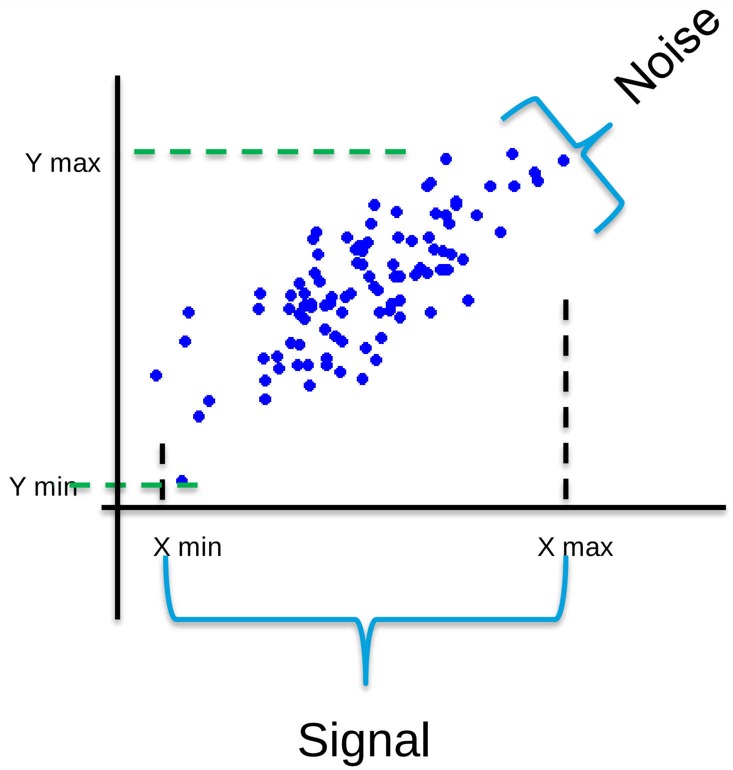
The data set is winequality-red.csv

Sol: PCA-iris.ipynb

Unsupervised Machine Learning

greatlearning
Learning for Life

Principal Component Analysis (Signal to noise ratio)



Signal – all valid values for a variable (show between max and min values for x axis and y axis). Represents a valid data

Noise – The spread of data points across the best fit line. For a given value of x, there are multiple values of y (some on line and some around the line). This spread is due to random factors

Signal to Noise Ratio – Variance of signal / variance in noise. $\frac{\sigma^2_{signal}}{\sigma^2_{noise}}$

Greater the SNR the better the model will be

```
X_std_df = pd.DataFrame(X_std)
axes = pd.plotting.scatter_matrix(X_std_df)
plt.tight_layout()
```

Principal Component Covariance Matrix

1. Variance is measured within the dimensions and co-variance is among the dimensions

$$\mathrm{var}(X) = \frac{\sum_{i=1}^{n}\left(X_i - \overline{X}\right)\left(X_i - \overline{X}\right)}{(n-1)}$$

$$\mathrm{cov}(X,Y) = \frac{\sum_{i=1}^{n}\left(X_i - \overline{X}\right)\left(Y_i - \overline{Y}\right)}{(n-1)}$$

2. Express total variance (variance and cross variance between dimensions as a matrix (variance matrix)

3. Covariance matrix is a mathematical representation of the total variance of individual dimension and across dimensions .

$$C = \begin{bmatrix} \mathrm{cov}(X,X) & \mathrm{cov}(X,Y) & \mathrm{cov}(X,Z) \\ \mathrm{cov}(Y,X) & \mathrm{cov}(Y,Y) & \mathrm{cov}(Y,Z) \\ \mathrm{cov}(Z,X) & \mathrm{cov}(Z,Y) & \mathrm{cov}(Z,Z) \end{bmatrix}$$

Covariance matrix for three dimensions x,y and z

```
eig_vals, eig_vecs = np.linalg.eig(cov_matrix)
```

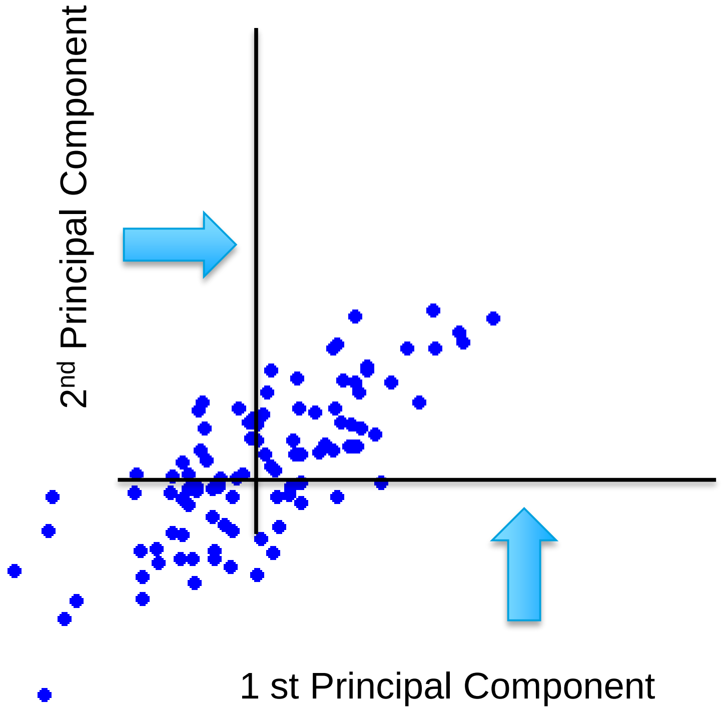
Unsupervised Machine Learning

Improving SNR through PCA (Scaling the dimensions)

1. The mean is subtracted from all the points on both dimensions i.e. (xi – xbar) and (yi – ybar)

2. The dimensions are transformed using algebra into new set of dimensions

3. The transformation is a rotation of axes in mathematical space



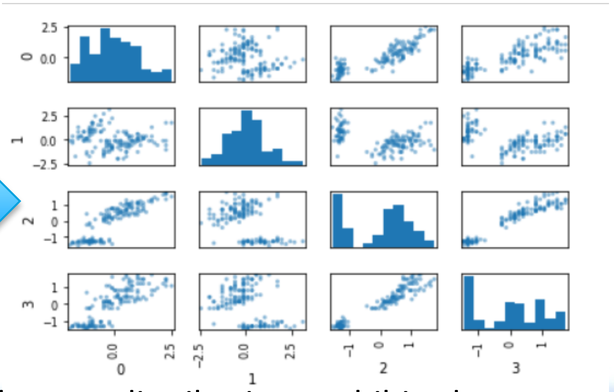
```
X_std = StandardScaler().fit_transform(X)
eig_vals, eig_vecs = np.linalg.eig(cov_matrix)
```

PCA (Calculating total variance (covariance and variance)

4. Multiplying the two matrices produces a matrix of total variance also called covariance matrix (a square and symmetric matrix).

$$\overset{A}{\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}} \times \overset{A^T}{\begin{bmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \cdots & \cdots & \cdots & \cdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix}} = C = \begin{bmatrix} \mathrm{cov}(X,X) & \mathrm{cov}(X,Y) & \mathrm{cov}(X,Z) \\ \mathrm{cov}(Y,X) & \mathrm{cov}(Y,Y) & \mathrm{cov}(Y,Z) \\ \mathrm{cov}(Z,X) & \mathrm{cov}(Z,Y) & \mathrm{cov}(Z,Z) \end{bmatrix}$$

```
Covariance Matrix
%s [[ 1.00671141 -0.11010327  0.87760486  0.82344326]
 [-0.11010327  1.00671141 -0.42333835 -0.358937  ]
 [ 0.87760486 -0.42333835  1.00671141  0.96921855]
 [ 0.82344326 -0.358937    0.96921855  1.00671141]]
```

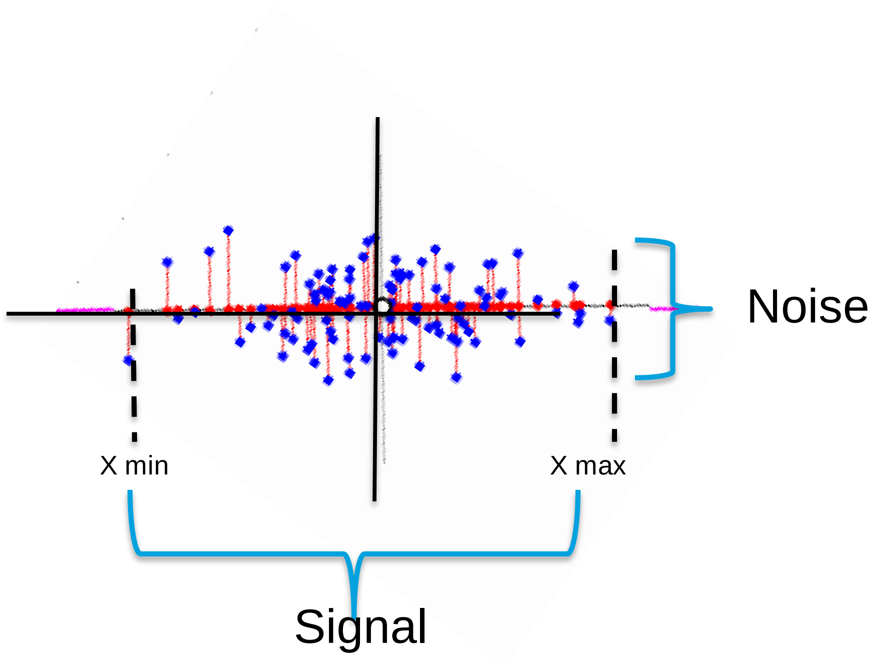


Improving SNR through PCA (Principal components)

5. The original data points are now represented by the red dots on new dimensions

6. It also introduces error of representation (vertical red lines from the blue dots to corresponding red dots on the new dimension)

7. The axis rotation is done such that the new dimension captures max variance in the data points and also reduces total error of representation



```
print('Eigen Vectors \n%s', eig_vecs)
print('\n Eigen Values \n%s', eig_vals)
```

Properties of principal components and their covariance matrix

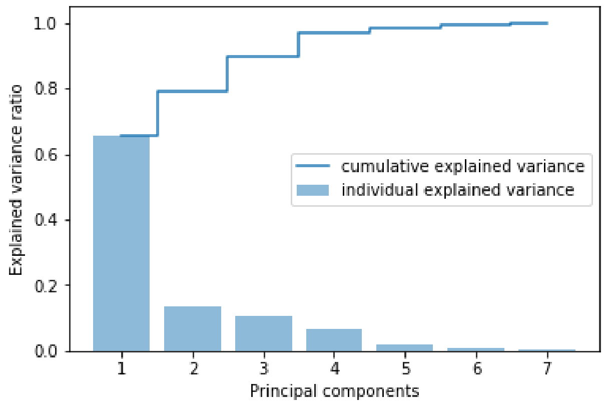
8. Thus to find principal components we need to get the diagonal matrix $\mathbf{B}\mathbf{B}^T$ from the original covariance matrix $A\,A^T$

$$\underset{A\,A^T}{\begin{bmatrix} \operatorname{cov}(X,X) & \operatorname{cov}(X,Y) & \operatorname{cov}(X,Z) \\ \operatorname{cov}(Y,X) & \operatorname{cov}(Y,Y) & \operatorname{cov}(Y,Z) \\ \operatorname{cov}(Z,X) & \operatorname{cov}(Z,Y) & \operatorname{cov}(Z,Z) \end{bmatrix}} \Longrightarrow \underset{\mathbf{B}\mathbf{B}^T}{\begin{pmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_p^2 \end{pmatrix}}$$

9. For this we have to transform the matrix **A** to a new matrix **B** such that the covariance matrix of **B** ($\mathbf{B}\mathbf{B}^T$), is a **diagonal matrix** (Ref to part 2, bullet 5)

PCA for dimensionality reduction

1. PCA can also be used to reduce dimensions
2. Arrange all eigen vectors along with corresponding eigen values in descending order of eigen values
3. Plot a cumulative eigen_value graph as shown below
4. Eigen vectors with insignificant contribution to total eigen values can be removed from analysis (for e.g. eigen vector 6 and 7 below)



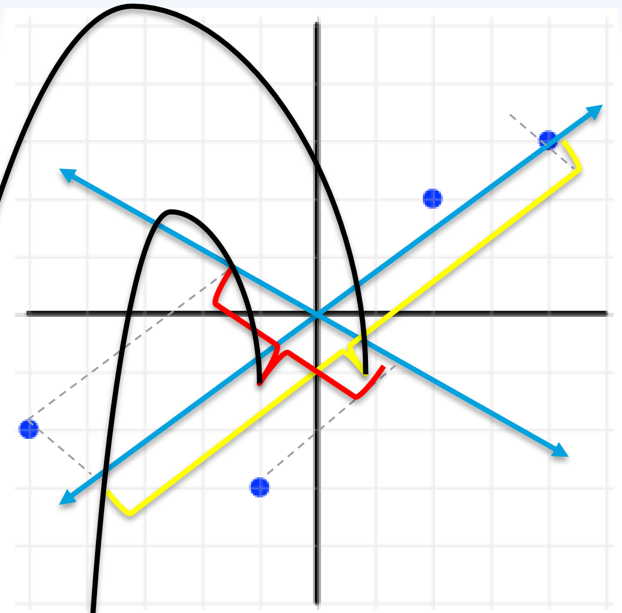
greatlearning
Learning for Life

Thanks

Unsupervised Machine Learning

greatlearning
Learning for Life

```
# Singular-value decomposition
from numpy import array
from numpy import dot
from numpy import zeros
from scipy.linalg import svd

# define a matrix
A = array([[4,3], [2,2], [-1,-3], [-5,-2]])
print(A)
# SVD
U, s, VT = svd(A)
print(U)
print(s)
print(VT)
```

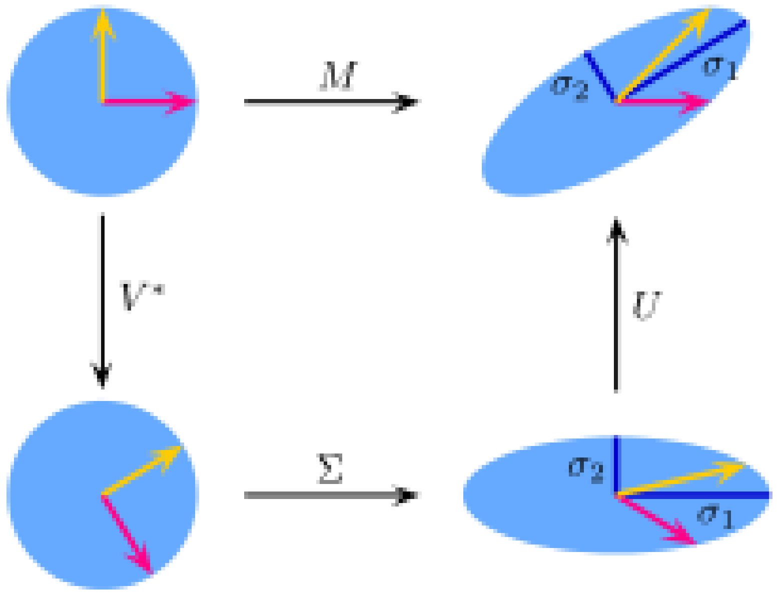


```
[[ 4  3]
 [ 2  2]
 [-1 -3]
 [-5 -2]]

[[-0.61215255 -0.05228813  0.06420835  0.78639208]
 [-0.34162337 -0.2025832   0.84891247 -0.34871356]
 [ 0.31300005  0.80704816  0.42637948  0.26249717]
 [ 0.64077586 -0.55217683  0.30565577  0.4371288 ]]

[8.16552039 2.30743942]

[[-0.81424526 -0.58052102]
 [ 0.58052102 -0.81424526]]
```

Rotation vector in higher dimensions of 4. Of these two will be nullified due to 0 values in singular matrix

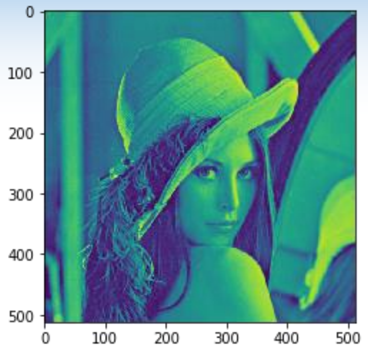
Unit orthogonal vectors in the blue directions. Pl. note, length of blue arrows is only for visibility

Unsupervised Machine Learning



$$M = U \cdot \Sigma \cdot V^*$$

Unsupervised Machine Learning



```
[[0.63529414 0.63529414 0.63529414 ... 0.6666667 0.60784316 0.5019608]
 [0.63529414 0.63529414 0.63529414 ... 0.6666667 0.60784316 0.5019608]
 [0.63529414 0.63529414 0.63529414 ... 0.6666667 0.60784316 0.5019608]
...

 [0.16862746 0.16862746 0.19607843 ... 0.40784314 0.3921568 0.38431373]
 [0.17254902 0.17254902 0.21568628 ... 0.40784314 0.4117647 0.42352942]
 [0.17254902 0.17254902 0.21568628 ... 0.40784314 0.4117647 0.42352942]]
```

512 rows

512 Columns

D:\GLI\DSE\Code\ mage_Compressic