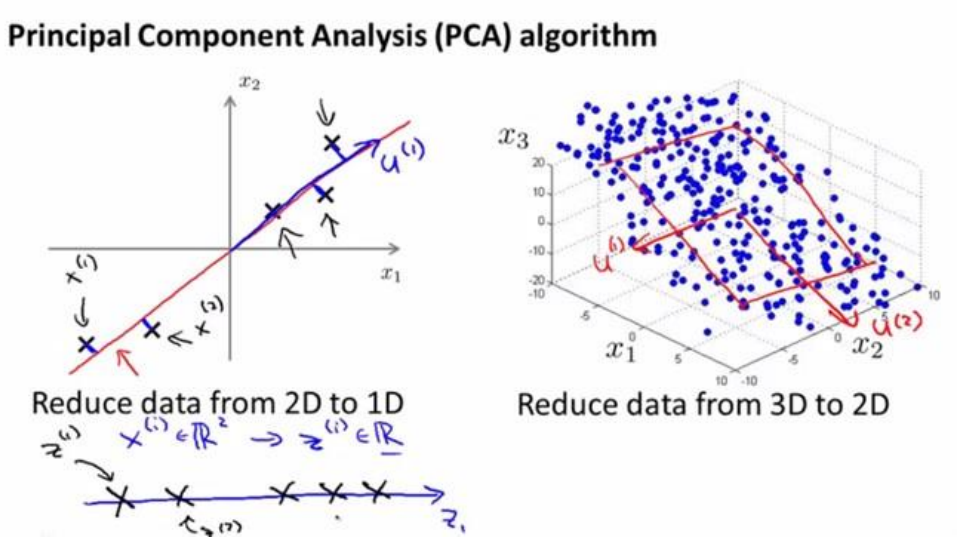
Principle Component Analysis

Principal component analysis (PCA) is a dimension reduction algorithm that extract k linearly uncorrelated features/vectors from n features/vectors (k < n) to realize minimum projection error when projecting all data onto the k features. It's widely used for image compression, data visualization and feature engineering. For example, PCA are used to reduce linearly correlated 2D data to 1D or linearly correlated 3D data to 2D as follows.



The PCA algorithm is realized in the following steps:

1, Feature Scaling and Normalization

Training set (each x is a vector):

$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots \boldsymbol{x}^{(m)}$$

Preprocessing:

calculate sample average and standard deviation of each feature j:

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$$

$$s_j = \sqrt{\frac{\sum_{i=1}^{m}(x_j^{(i)} - \mu_j)^2}{m-1}}$$

replace x as follows:

$$x_j^{(i)} = \frac{(x_j^{(i)} - \mu_j)}{s_j}$$

2, Reduce data from n-dimensions to k-dimensions

compute ‘covariance matrix’:

$$\boldsymbol{Sigma} = \frac{1}{\boldsymbol{m}} \sum_{\boldsymbol{i}=1}^{\boldsymbol{n}} (\boldsymbol{x}^{(\boldsymbol{i})})(\boldsymbol{x}^{(\boldsymbol{i})})^{\boldsymbol{T}}$$

compute ‘eigenvectors’ U of matrix sigma

$$[\boldsymbol{U}, \boldsymbol{S}, \boldsymbol{V}] = \boldsymbol{svd}(\boldsymbol{Sigma})$$

For example, eigenvectors of linearly correlated data are shown below (see Figure 1). The length of the vectors is a measure of the variance of the data when projected onto that axis.

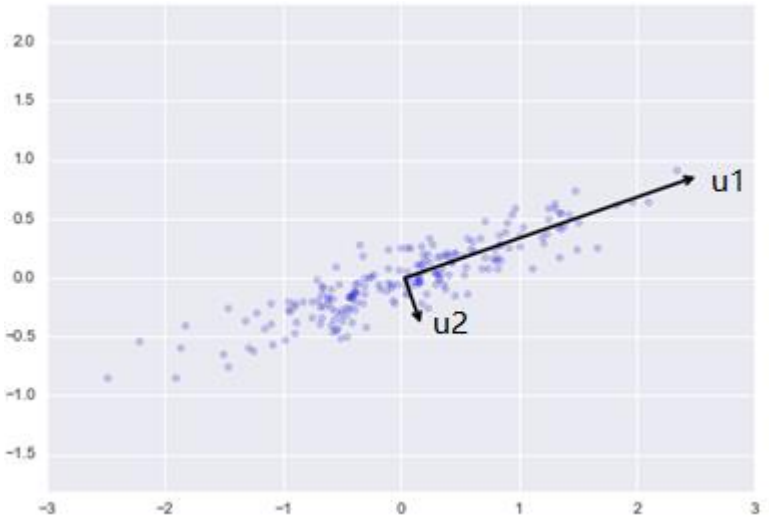


Figure 1. Eigenvectors of linearly correlated data. Ref: https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html

Here U contains principal components:

$$\boldsymbol{U} = \left[\boldsymbol{u}^{(1)}, \boldsymbol{u}^{(2)} \ldots \boldsymbol{u}^{(\boldsymbol{n})} \right] \in \boldsymbol{R}^{\boldsymbol{n} \cdot \boldsymbol{n}}$$

Reduce U to k dimension and get the k principal components:

$$\boldsymbol{U}_{\boldsymbol{reduce}} = \boldsymbol{U}(:, 1:\boldsymbol{k}) = \left[\boldsymbol{u}^{(1)}, \boldsymbol{u}^{(2)}, \ldots \boldsymbol{u}^{(\boldsymbol{k})} \right]$$

3, How to choose number k?

Typically, choose k to be the smallest value so that

$$\frac{\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}-x_{approx}^{(i)}\right\|^{2}}{\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}\right\|^{2}}\leq\alpha$$

Here the value of α is defined by the percentage of variance (1-α) we want to retain. For example, if $\alpha = 0.01$, 99% of variance is retained; if $\alpha = 0.05$, 95% of variance is retained. $x_{approx}^{(i)}$ is calculated as follows:

$$z = Ureduce * x$$

$$x_{approx}^{(i)} = Ureduce * z^{(i)}$$

Reference:
1, Machine Learning by Andrew Ng, Stanford University, Coursera.
2, https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html