# Dimensionality reduction techniques: Part 1

Sanzhar Askaruly (San)

Ulsan National Institute of Science and Technology
Ph.D. Candidate in Biomedical Engineering

CodeSeoul MLA
December 10, 2022

# Overview

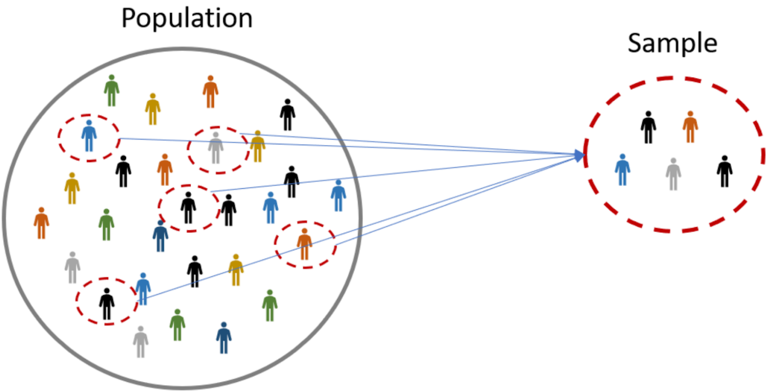
# Motivation

**Why dimensionality reduction?**

Important "toolkit" in machine learning:

- Preprocessing step for other algorithms (reduce size)
- Visualization
- Compression
- Interpolation

# Population vs sample



# Population vs sample

**Population mean**

$$\mu = \frac{\sum_{i=1}^{N} x_i}{N} \qquad (1)$$

$N$ is number of items in the population

**Sample mean**

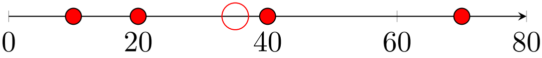
$$\bar{X} = \frac{\sum_{i=1}^{n} x_i}{n} \qquad (2)$$

$n$ is number of items in the sample

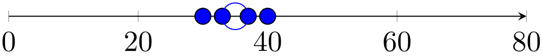
# Standard deviation

Let's take a look on two samples:

$$A = [10\ 20\ 40\ 70]$$




$$B = [30\ 33\ 37\ 40]$$




Here, $\bar{A} = \bar{B} = 35$. Unfortunately, mean doesn't tell us a lot except for a middle point.

# Standard deviation

For our two sets, $A = [10\,20\,40\,70]$ and $B = [30\,33\,37\,40]$, we would be more interested in the *spread* of the data. So, how do we calculate it?

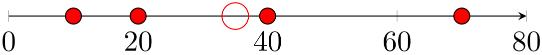
## Standard deviation

$$s = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{(n-1)}} \tag{3}$$

In plain English, it is the "average distance from the mean of the data set to a point."

# Standard deviation

Set 1: $A = [10\ 20\ 40\ 70]$, and $\bar{A} = 35$

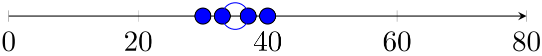



Let's calculate standard deviation:

| $A$ | $(A - \bar{A})$ | $(A - \bar{A})^2$ |
|---|---:|---:|
| 10 | -25 | 625 |
| 20 | -15 | 225 |
| 40 | 5 | 25 |
| 70 | 35 | 1,225 |
| Total | | 2,100 |
| Divided by (n-1) | | 700 |
| Square root | | **26.4575** |

# Standard deviation

Set 2: $B = [30\ 33\ 37\ 40]$, and $\bar{B} = 35$




Let's calculate standard deviation:

| $B$ | $(B - \bar{B})$ | $(B - \bar{B})^2$ |
|---|---|---|
| 30 | -5 | 25 |
| 33 | -2 | 4 |
| 37 | 2 | 4 |
| 40 | 5 | 25 |
| Total | | 58 |
| Divided by (n-1) | | 19.333 |
| Square root | | **4.397** |

# Variance

Similar to standard deviation So, how do we calculate it?

## Variance

$$s^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{(n-1)} \tag{4}$$

Almost identical to the standard deviation (there is no square root in the formula).

# Covariance

Previous measures are 1-dimensional. How do we check relationship between the dimensions?

## Covariance

**Variance**

$$var(X) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(X_i - \bar{X})}{(n-1)} \tag{5}$$

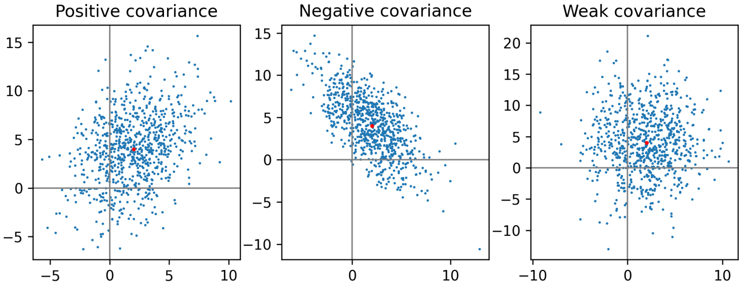
**Covariance**

$$covar(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)} \tag{6}$$

How much dimensions vary from the mean *with respect to each other*.

- Covariance between one dimension and itself gives variance.

# Covariance


Positive covariance
Negative covariance
Weak covariance
15
10
5
0
−5
−10
20
0
5
10

# Covariance

Exercise: Find covariance between 2-dimensional dataset

| Item number: | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| x | 10 | 39 | 19 | 23 | 28 |
| y | 43 | 13 | 32 | 21 | 20 |

**Ans:** -120.55

# Covariance matrix

When we have more than two dimensions, covariances between individual dimensions could be described in covariance matrix

## Covariance matrix

$$C^{n \times n} = (c_{i,j} : c_{i,j} = cov(Dim_i, Dim_j)) \tag{7}$$

where $C^{m \times n}$ is a matrix with $n$ rows and $m$ columns. Example for three dimensions:

$$c = \begin{pmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{pmatrix} \tag{8}$$

Since $cov(a,b) = cov(b,a)$, the matrix is symmetrical about the main diagonal.

# Covariance matrix

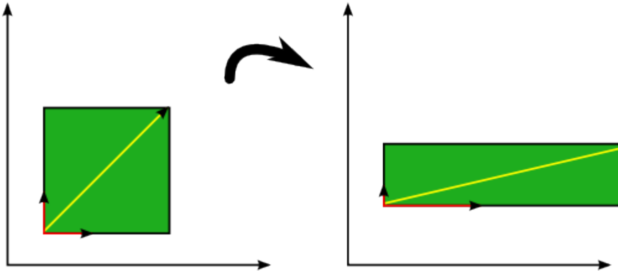
Exercise: Find covariance matrix for the 3-dimensional dataset

| Item number: | 1 | 2 | 3 |
|---|---|---|---|
| x | 1 | -1 | 4 |
| y | 2 | 1 | 3 |
| z | 1 | 3 | -1 |

**Ans:** $$\begin{pmatrix} 4.222 & 1.667 & -3.333 \\ 1.667 & 0.667 & -1.333 \\ -3.333 & -1.333 & 2.667 \end{pmatrix}$$

More examples and linear algebra basics recommended here, [1].

# Eigenvectors



- Eigenvectors (red) do not change direction when a linear transformation (e.g. scaling) is applied to them. Other vectors (yellow) do.

# Eigenvectors

An example of non-eigenvector:

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \boxed{\begin{pmatrix} 1 \\ 3 \end{pmatrix}} = \begin{pmatrix} 11 \\ 5 \end{pmatrix} \tag{9}$$

An example of eigenvector:

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \boxed{\begin{pmatrix} 3 \\ 2 \end{pmatrix}} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} \tag{10}$$

# Eigenvalues

The amount by which the original vector was scaled after multiplication by the square matrix

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = \boxed{4} \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} \tag{11}$$

# PCA

**What is Principal Component Analysis?**
A way of identifying patterns in data: by highlighting their similarities and differences [2]:

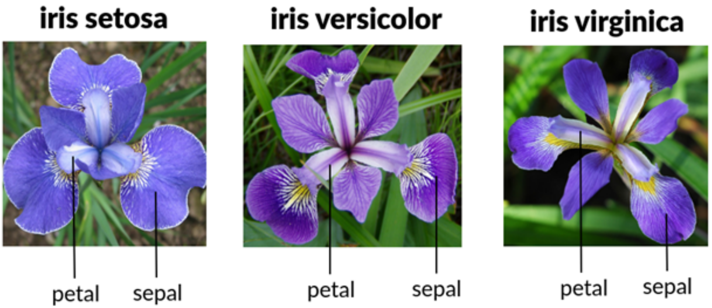
**Pros**

- Once patterns in the data, and you compress the data, ie. by reducing the number of dimensions, without much loss of information.

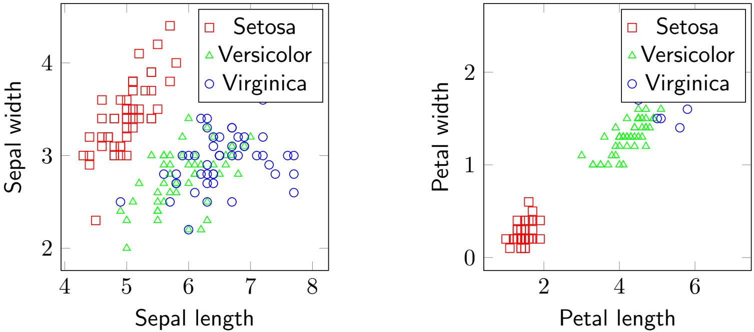
**Cons**

- Linear $\therefore$ difficult to unfold single ambiguous object really belongs in several disparate locations in the low-dimensional space.

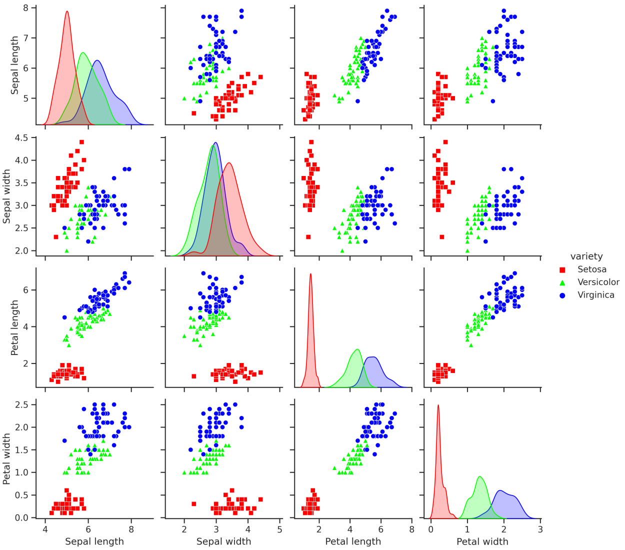
# Iris flower dataset



# Iris flower dataset



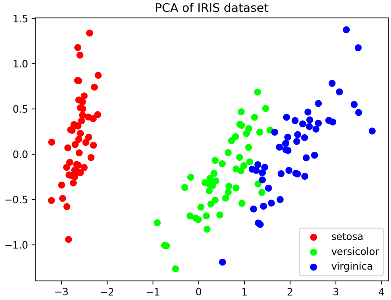
# Iris flower dataset



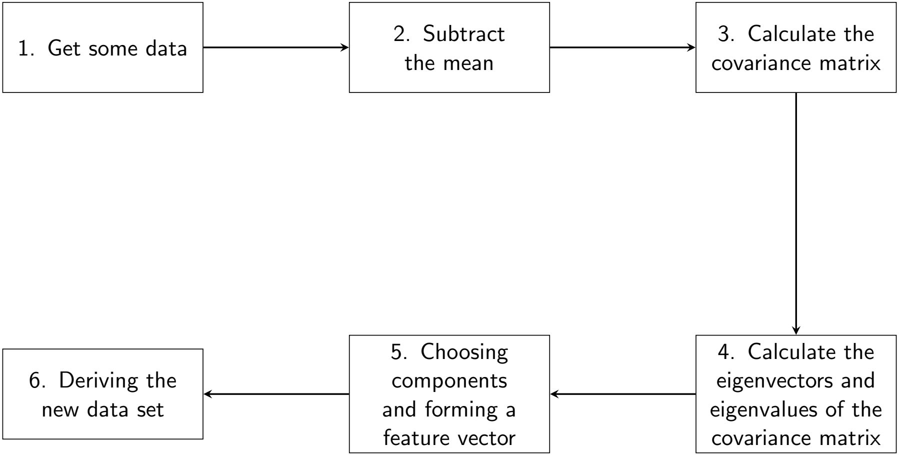
# Applying PCA to Iris flower dataset

PCA automatically finds principal components (eg. *axes*) in feature space. Here, we defined two, so we can visualize in 2D easily.

```
from sklearn import datasets
from sklearn.decomposition import PCA

iris = datasets.load_iris()

X = iris.data
y = iris.target
target_names = iris.target_names

pca = PCA(n_components=2)
X_r = pca.fit(X).transform(X)
```



# PCA under the hood



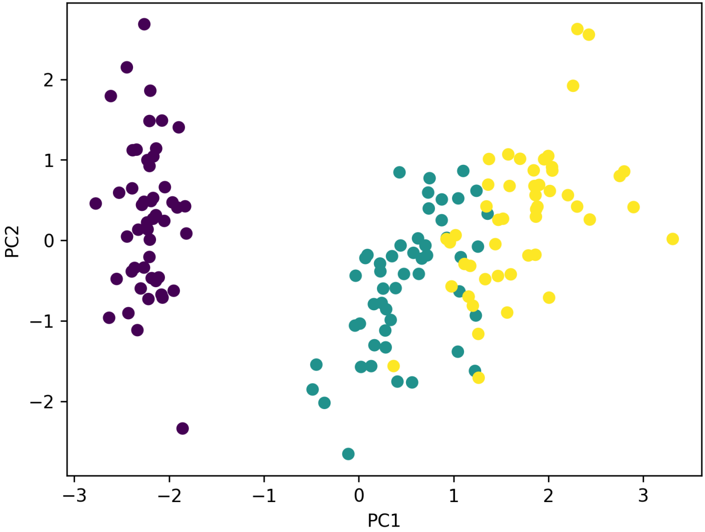
# PCA under the hood

```
import numpy as np
from numpy.linalg import eig

def PCA_numpy(X,y):
    def mean(x): # np.mean(X, axis = 0)
        return sum(x)/len(x)

    def std(x): # np.std(X, axis = 0)
        return (sum((i - mean(x))**2 for i in x)/len(x))**0.5

    def Standardize_data(X):
        return (X - mean(X))/std(X)

    def covariance(x):
        return (x.T @ x)/(x.shape[0]-1)

    # Step 1: Standardize the data
    X_std = Standardize_data(X)
    # Step 2: Find the covariance matrix
    cov_mat = covariance(X_std) # np.cov(X_std.T)

    # Step 3: Find the eigenvectors and eigenvalues of the covariance matrix
    eig_vals, eig_vecs = eig(cov_mat)
```

# PCA under the hood

```
def continued ...
    max_abs_idx = np.argmax(np.abs(eig_vecs), axis=0)
    signs = np.sign(eig_vecs[max_abs_idx, range(eig_vecs.shape[0])])
    eig_vecs = eig_vecs*signs[np.newaxis,:]
    eig_vecs = eig_vecs.T

    # Step 4: Rearrange the eigenvectors and eigenvalues
    eig_pairs = [(np.abs(eig_vals[i]), eig_vecs[i,:]) for i in range(len(eig_vals))

    # Then, we sort the tuples from the highest to the lowest based on eigenvalues m
    eig_pairs.sort(key=lambda x: x[0], reverse=True)
    eig_vals_sorted = np.array([x[0] for x in eig_pairs])
    eig_vecs_sorted = np.array([x[1] for x in eig_pairs])

    # Step 5: Choose principal components
    k = 2
    W = eig_vecs_sorted[:k, :] # Projection matrix

    # Step 6: Project the data
    X_proj = X_std.dot(W.T)

    return X_proj
```

# PCA under the hood



# t-SNE (Stochastic neighbor embedding)

To be continued next time,
Dimensionality reduction techniques: Part 2

# Thank you for your attention!

- Slides:
  https://github.com/CodeSeoul/machine-learning/blob/master/221210-pca/lecture.pdf
- Recorded video:
  https://www.youtube.com/watch?v=UfwQyWDOKRM
- Solutions for exercises:
  https://github.com/CodeSeoul/machine-learning/blob/master/221210-pca/excel_ex.xlsx
- PCA from scratch:
  https://github.com/CodeSeoul/machine-learning/blob/master/221210-pca/practicum.ipynb
- Follow-up QA?
  http://discord.com/users/tuttelikz

# References


Howard Anton and Chris Rorres. *Elementary linear algebra: applications version*. John Wiley & Sons, 2013.

Lindsay I Smith. "A tutorial on principal components analysis". In: (2002).