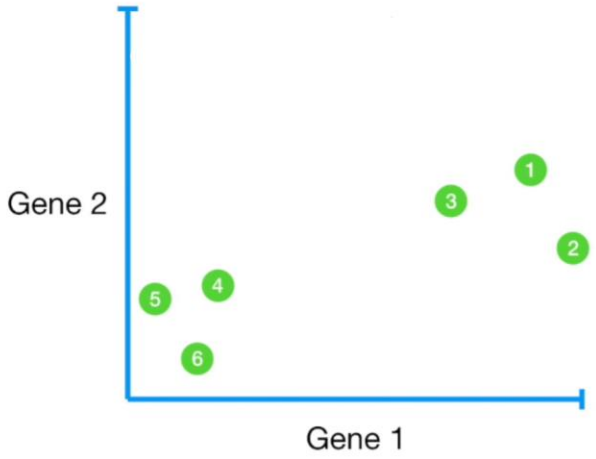
COS30018 – Intelligent Systems

Principal Component Analysis (PCA) with sklearn

In short, PCA is a ML method to reduce the dimensions of data. In machine learning and statistics, dimension means the number of features in a dataset.

For example, with this two-features dataset (Gene 1 and Gene 2 are the 2 features), we can plot it on a 2D graph like this:

	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1



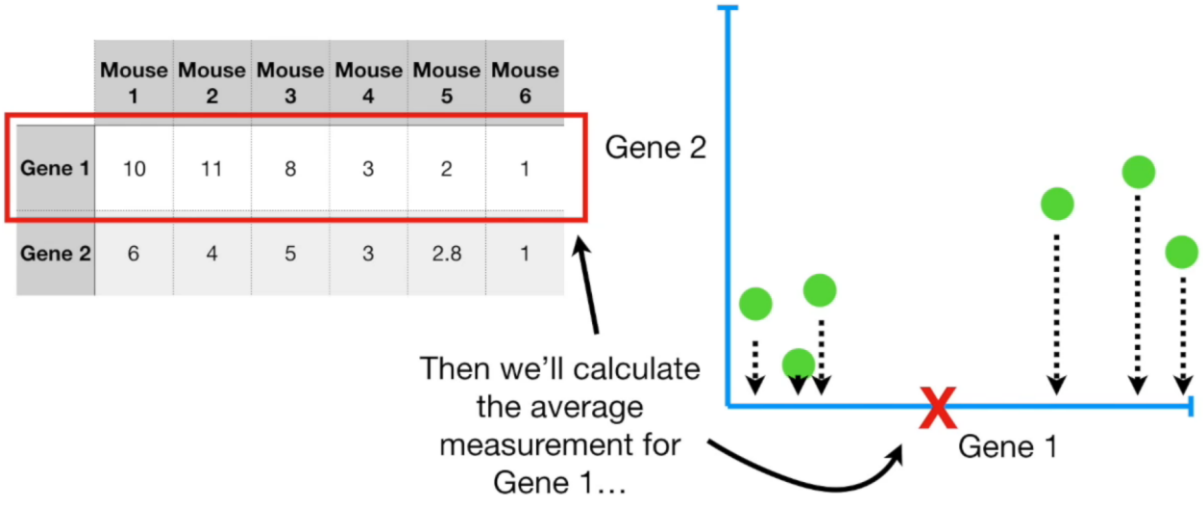
If we have 3 features, we can make a graph in 3D to visualise it. What if the data has more than 3 features? There is no way to visualise it in our 3D world => We need some solutions to reduce the dimension of the data (without losing the relative relationship between the data) so that we can visualise them in 2D/3D graphs.

In this tutorial, we will go through **PCA (using Singular Value Decomposition (SVD))** to reduce the dimensions of data.

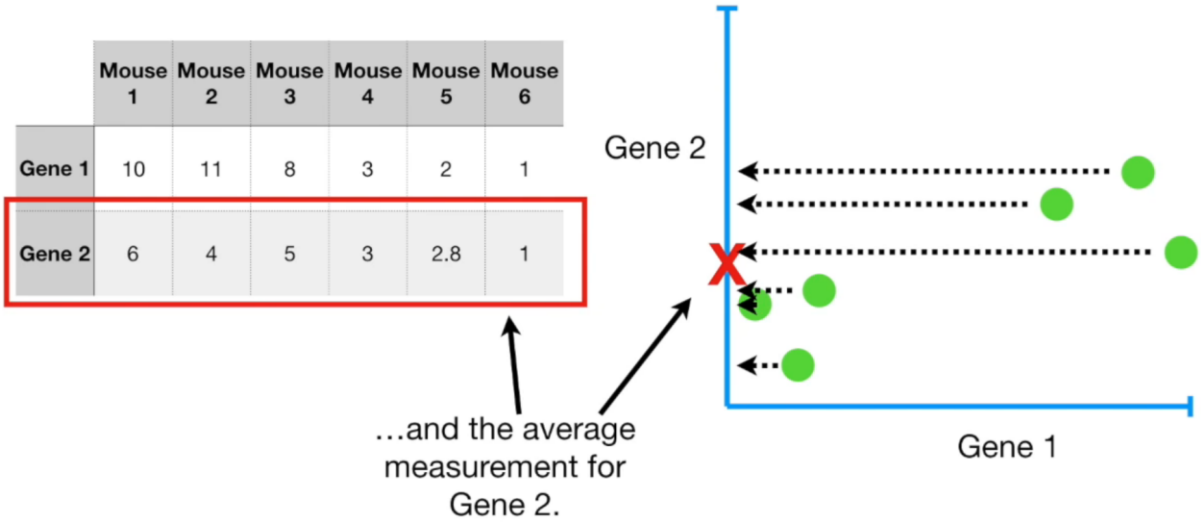
To know how PCA can be implemented, let's start with the 2-features dataset and use PCA with 2 components.

First, consider the data below and calculate the average of gene 1 and gene 2:

	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1

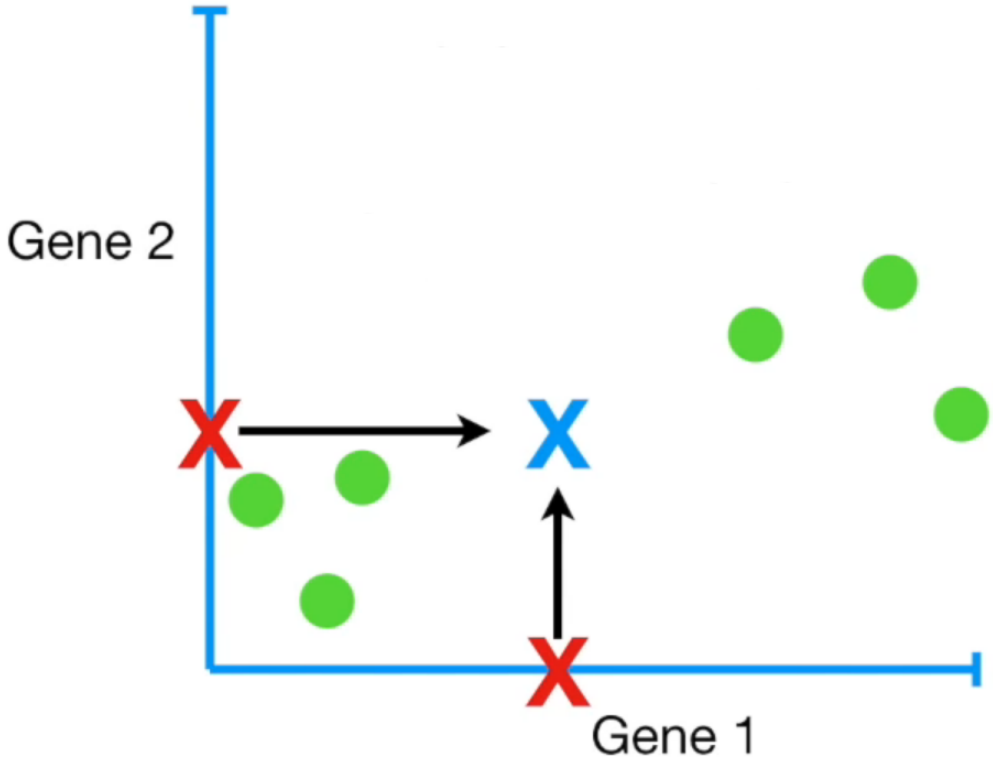


	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1



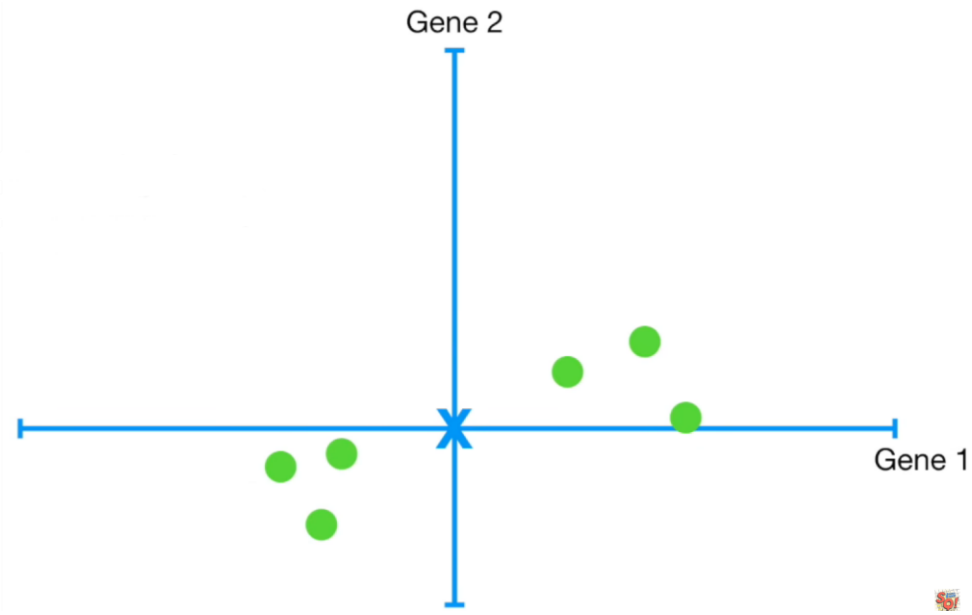
Note: the red X is the average value of each feature.

With the average values, we can calculate the centre of the data:



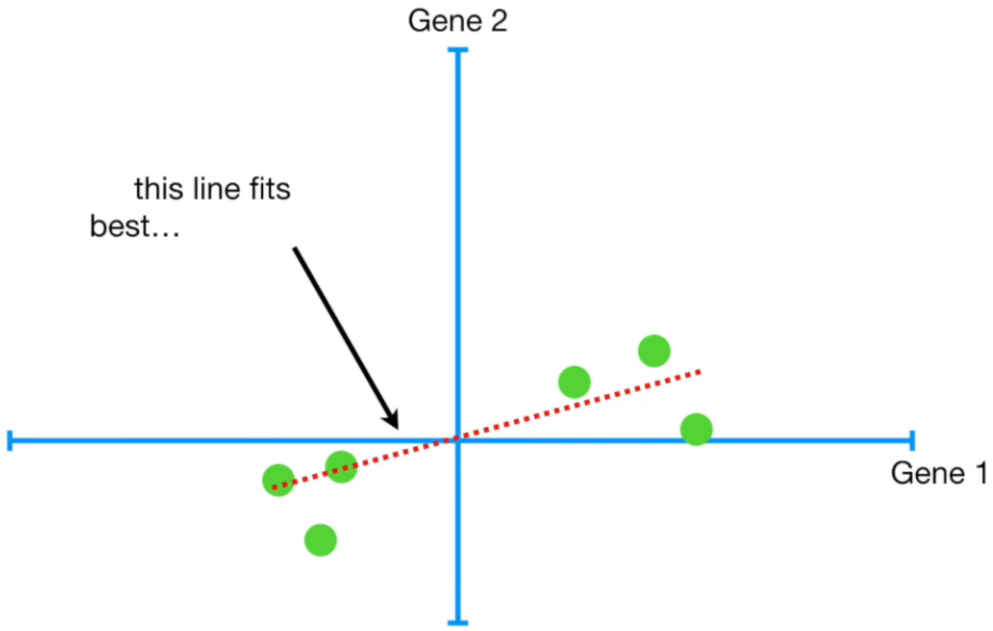
Note: the blue X is the centre of the data

Next, we will shift the data so that the centre point is at the origin (0, 0) in the graph

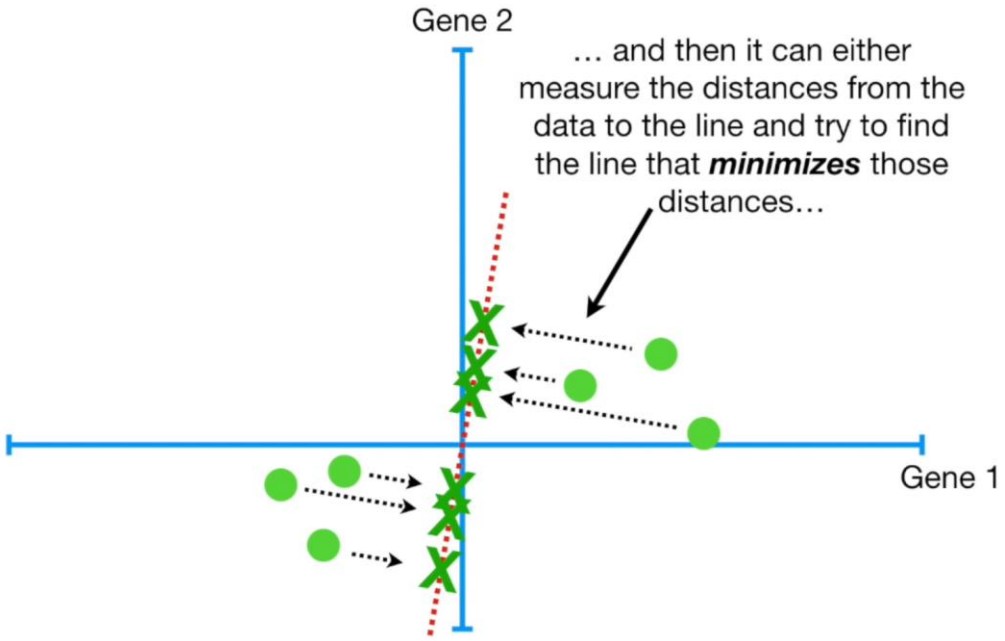


Note: shifting data doesn't change how data points are positioned relative to each other.

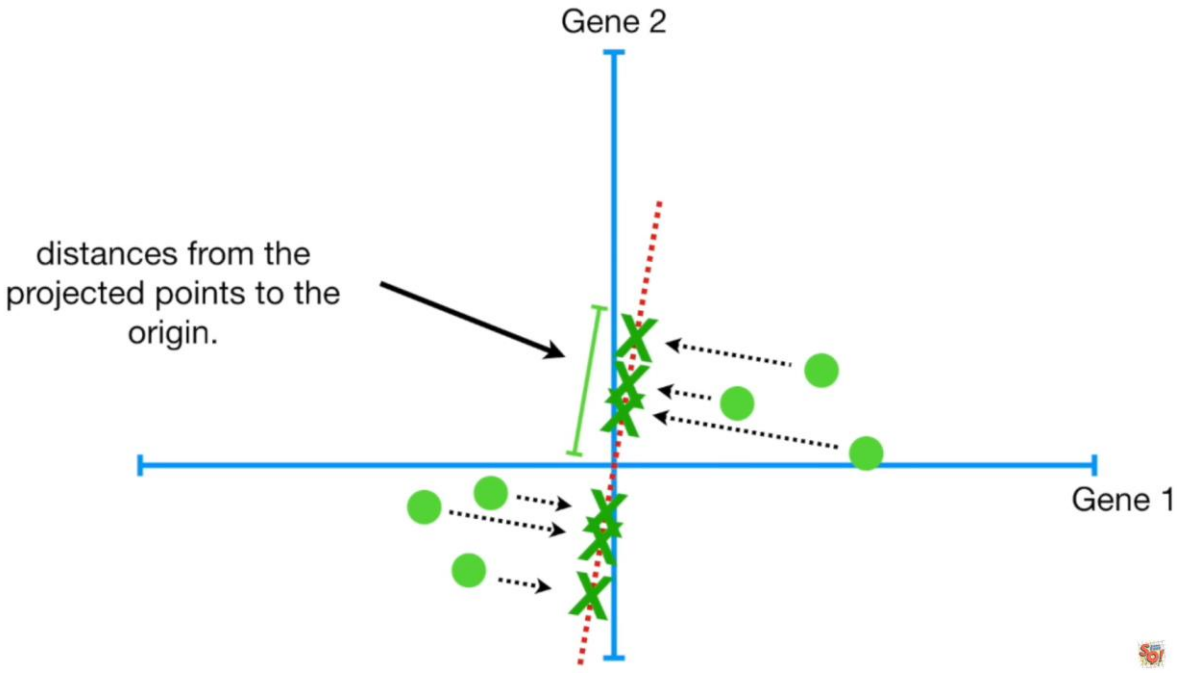
Then, we will find a line that fits best the data:



A line fits best the data is the line that minimises the projected errors between the data points and the line:

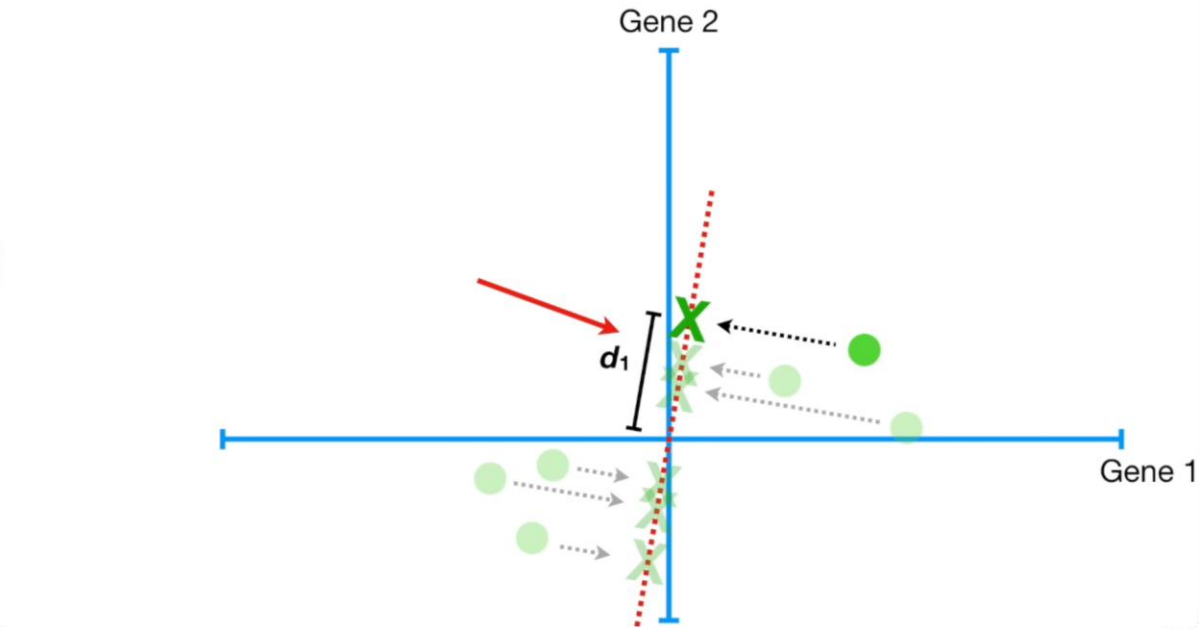


Or equivalently, the line that maximises the distances from the projected points to the origin:



In practice, people use the second way => PCA finds the best fitting line by **maximising** the **sum of squared distances** from the projected points to the origin.

For example, the distance from the 1st point to the origin is called d_1 as below:

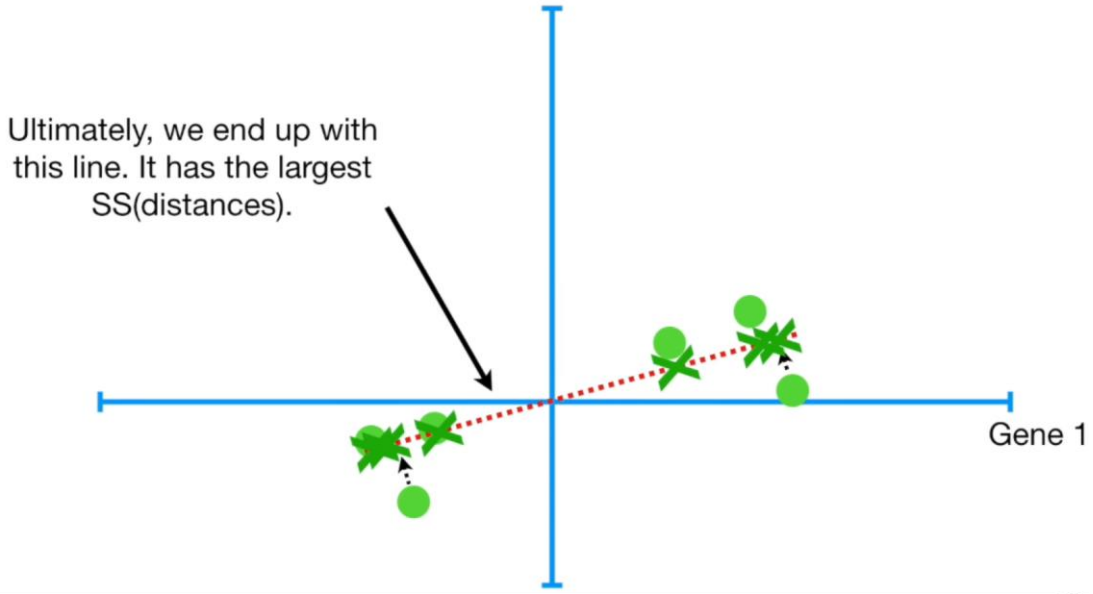


Similarly, we calculate d_2, d_3, d_4, d_5, d_6 for the other points, then the sum of squared distances of this line is:

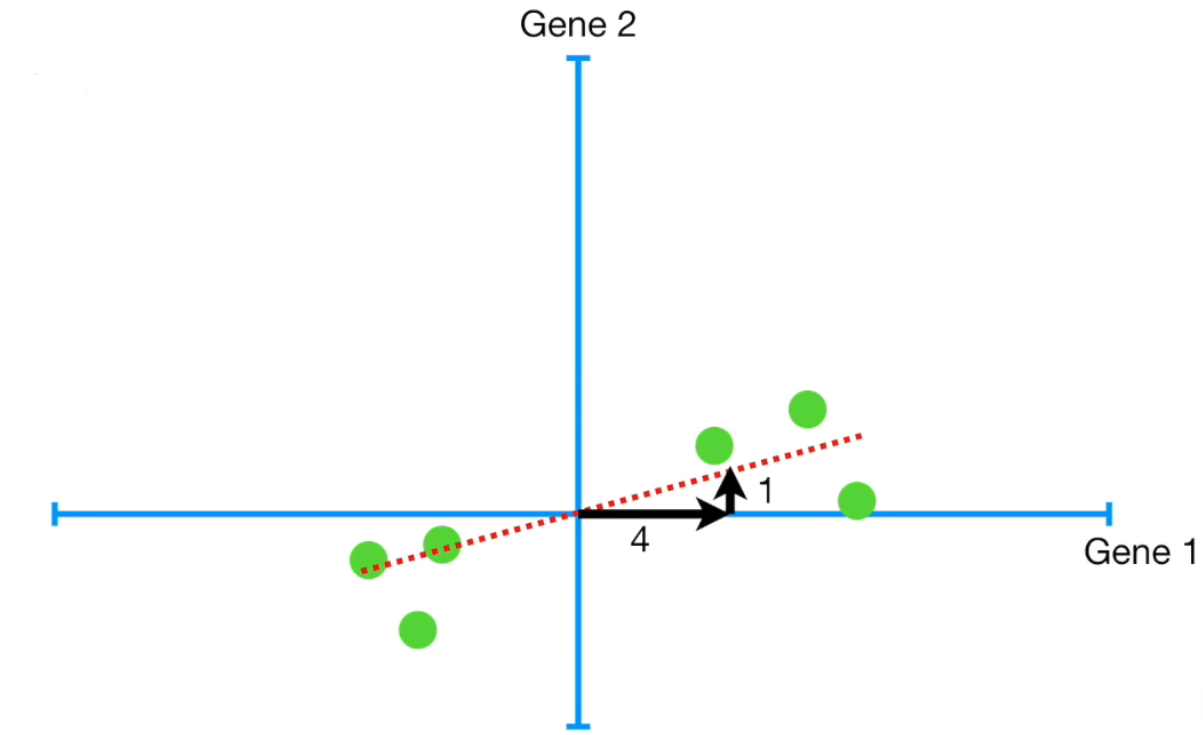
$$SS\_distances = (d_1)^2 + (d_2)^2 + (d_3)^2 + (d_4)^2 + (d_5)^2 + (d_6)^2$$

The distances are squared so that negative values don't cancel out positive values.

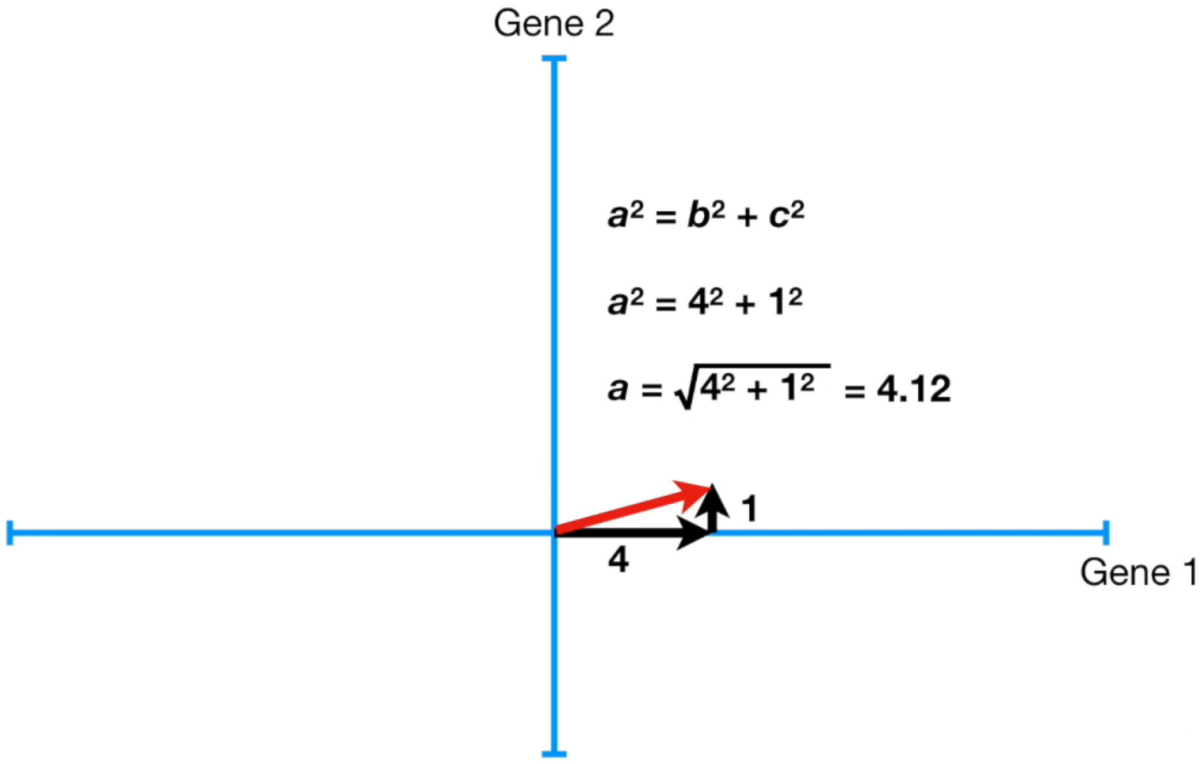
The line with the largest SS_distances is the best fitting line, which is something like this:

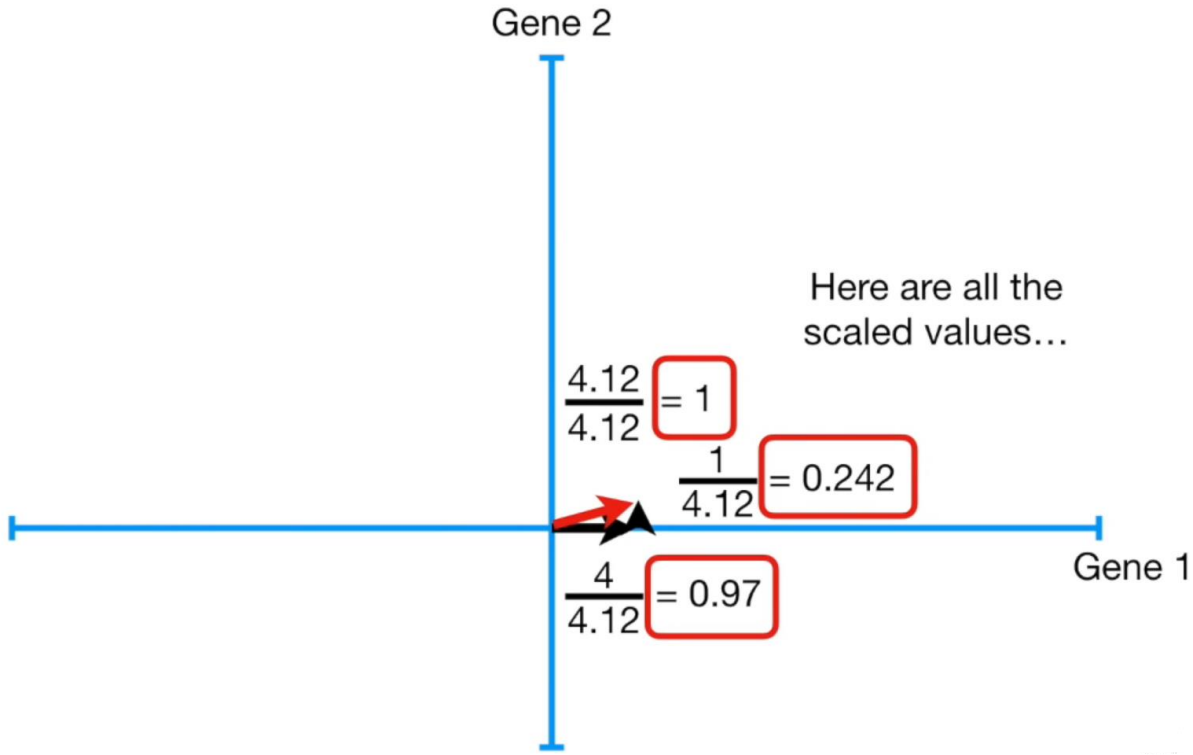


This line is called **Principal Component 1 (PC1)**. This PC1 has the slope = $\frac{1}{4}$, which means for every 4 units that we go along the Gene 1 axis, we will go up by 1 unit along the Gene 2 axis:

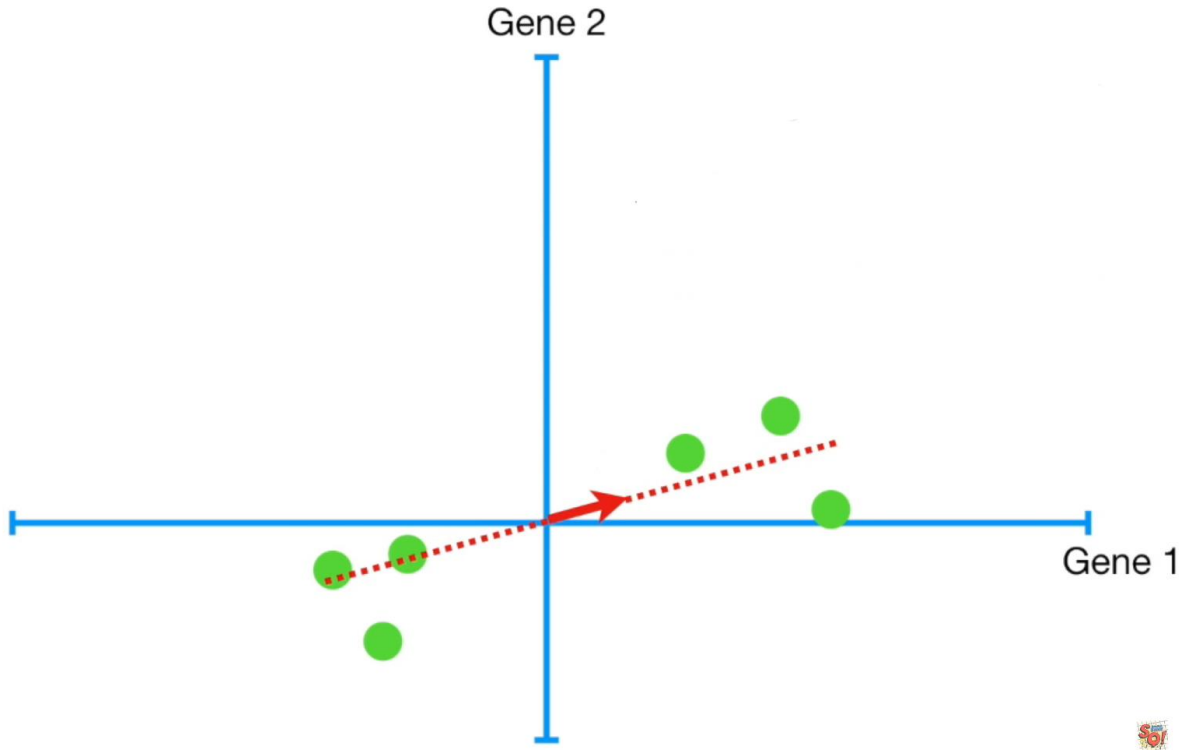


The **Eigenvector** of PC1 is the vector that has 1-unit length (unit vector) along the PC1 axis:





This 1-unit long vector, consisting of 0.97 units on Gene 1 and 0.242 units on Gene 2 is called **Eigenvector** or **Singular Vector** of PC1. The Eigenvector helps you to find **Loading Score,** which is introduced later.



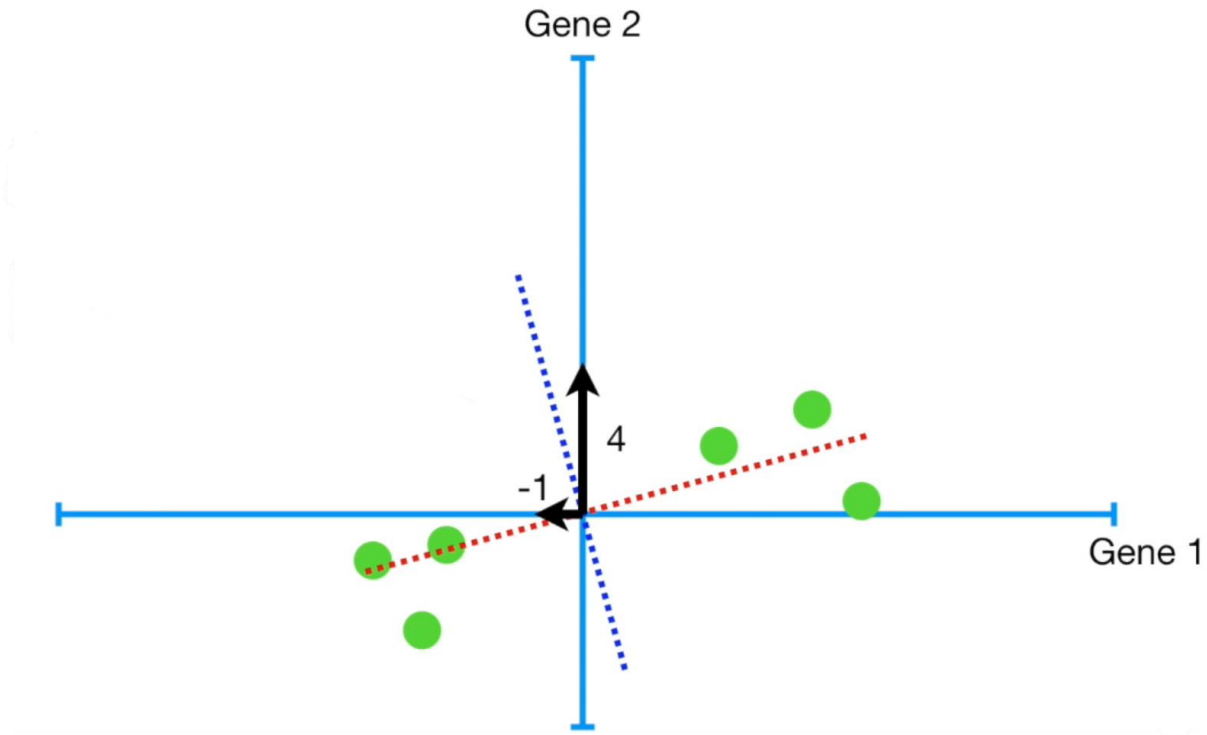
And the $\boldsymbol{SS\_distances}$ along the PC1 line is called **Eigenvalue** of PC1. And the square root of Eigenvalue of PC1 is called **Singular Value** of PC1. The Eigenvalue helps you to find **Variation**, which is introduced later.

$$\text{SS(distances for PC1)} = \text{Eigenvalue for PC1}$$

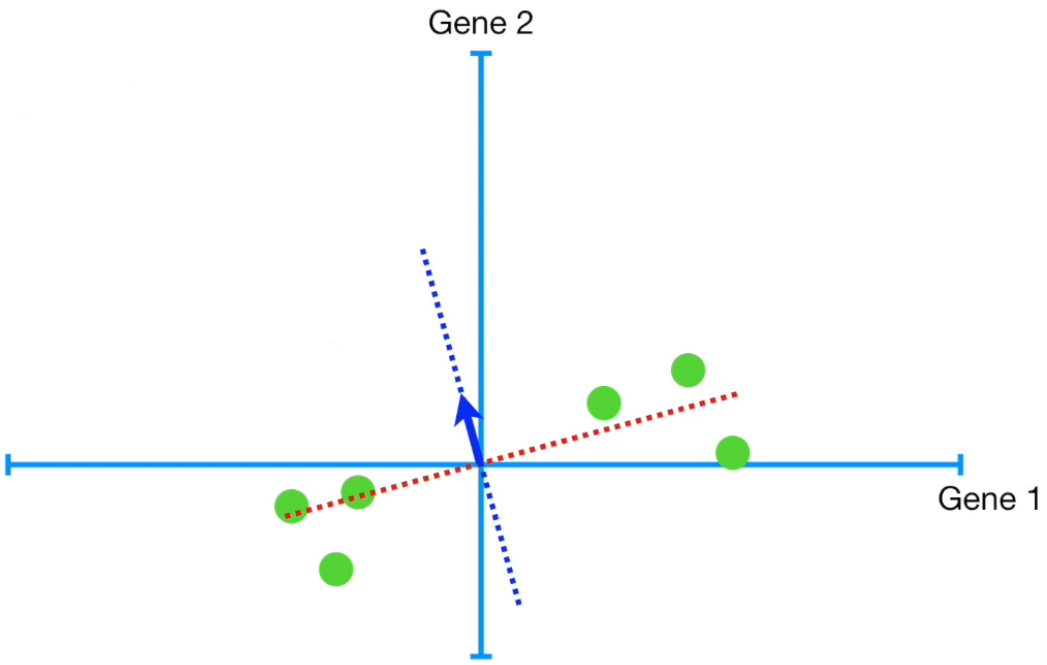
$$\sqrt{\text{Eigenvalue for PC1}} = \text{Singular Value for PC1}$$

And the proportion of each Gene $\frac{4}{1}$ is called **"Loading Scores"** of PC1. The Loading Scores shows which component has the greatest influence on PC1.

Now we find the PC2, PC2 is simply the line through the origin that is perpendicular to PC1. And the slope of this PC2 is -4, which means for every 1 unit that we go along negative Gene 1 axis, we will go up by 4 units along the Gene 2 axis:

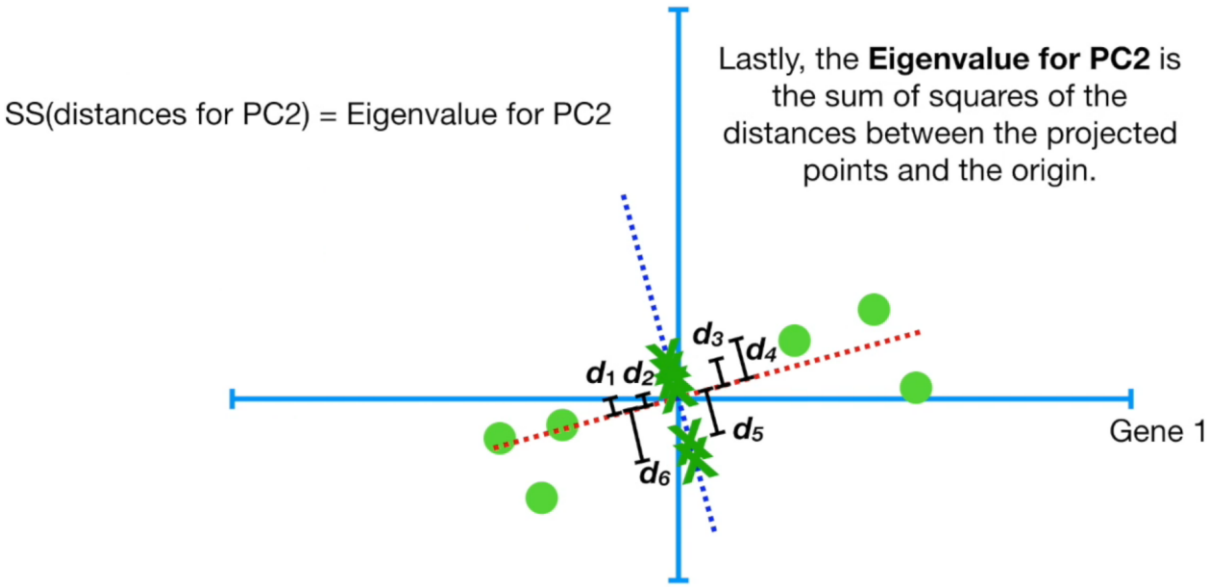


The **Eigenvector/Singular vector** of PC2 is the vector that has 1-unit length (unit vector) along the PC2 axis. If we scale everything to get that unit vector, we will have for every **-0.242** units that we go along Gene 1 axis, we will go up by **0.97** units along the Gene 2 axis. And the proportion of each Gene $\frac{-1}{4}$ is the **Loading Scores** of PC2.

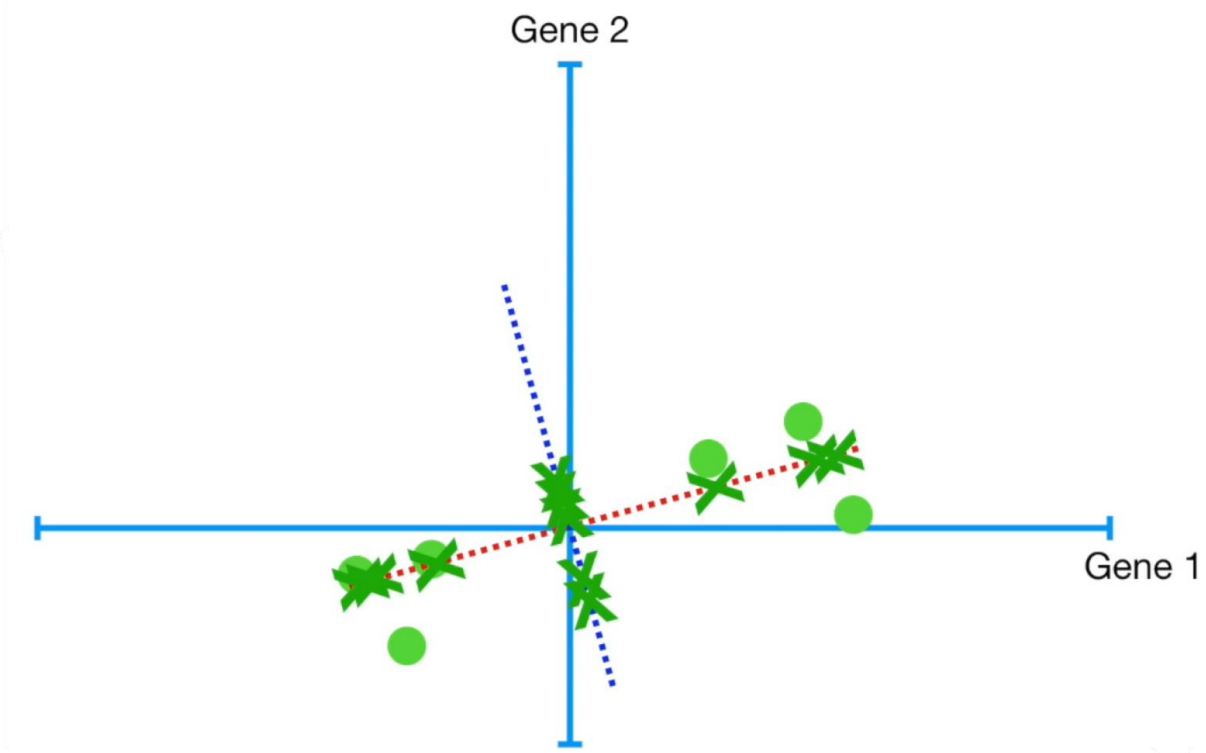


The **Loading Scores** of PC2 is $-\frac{1}{4}$.

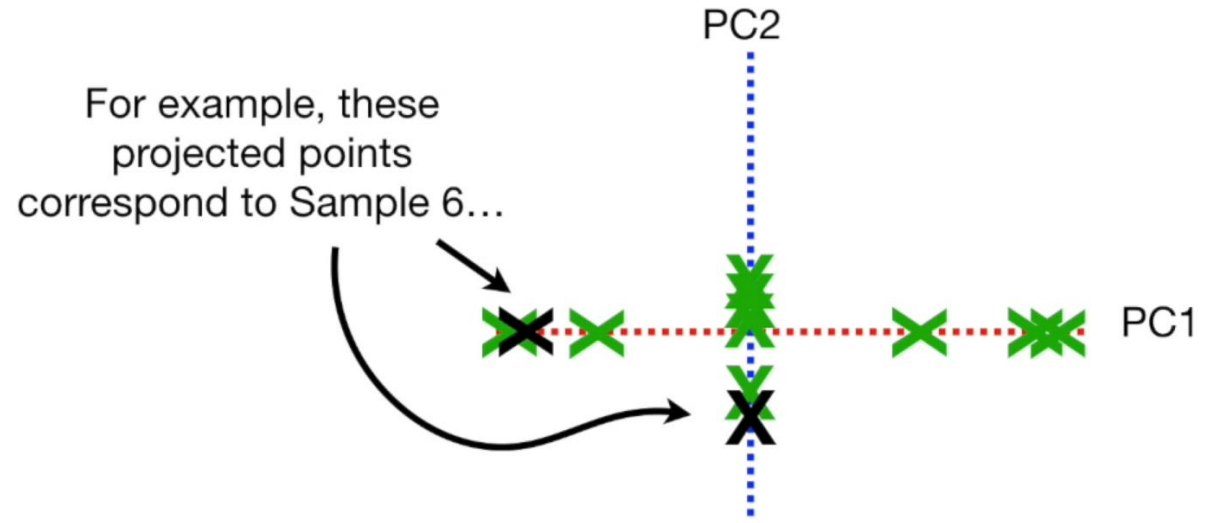
The **Eigenvalue** of PC2 is the $\boldsymbol{SS\_distances}$ between the projected points on PC2 and the origin:

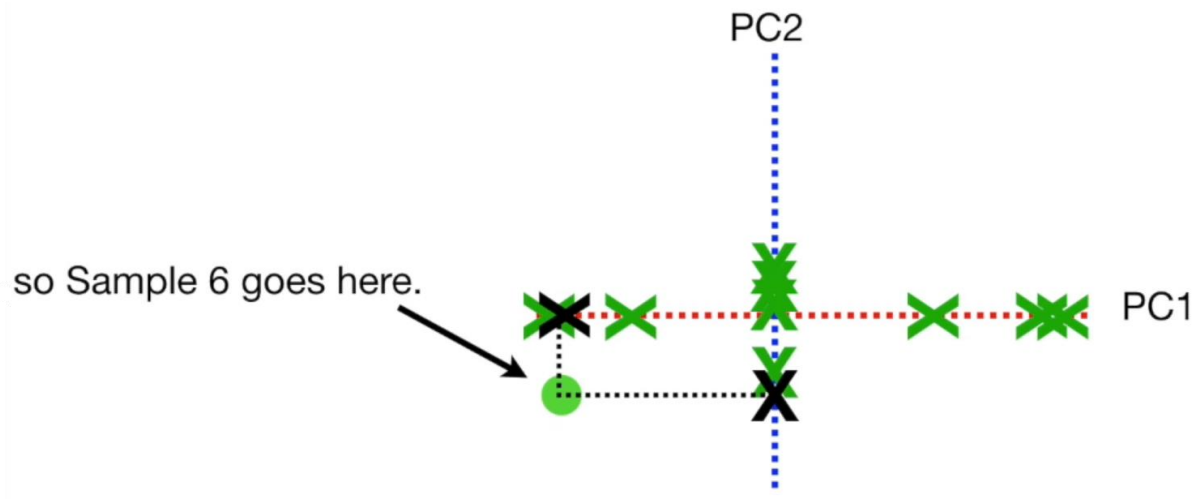


Hooray!!! Now we have found the PC1 and PC2:

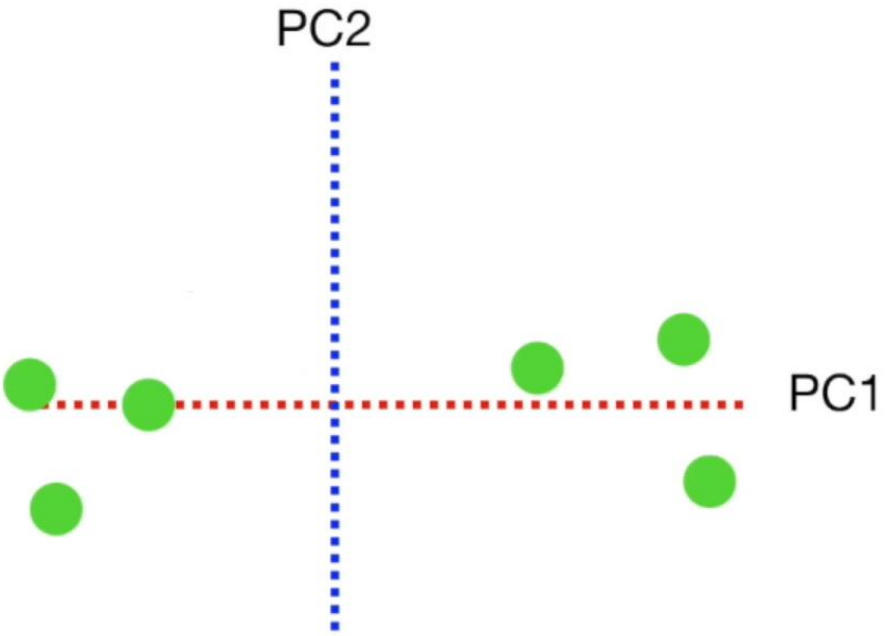


Next we rotate the plot so that PC1 is horizontal, and use the projected points to find where the samples go in the PCA plot:





Do it similarly for other samples, and we will have the final plot like this:



That's it! That's how PCA is done using Singular Value Decomposition (SVD).

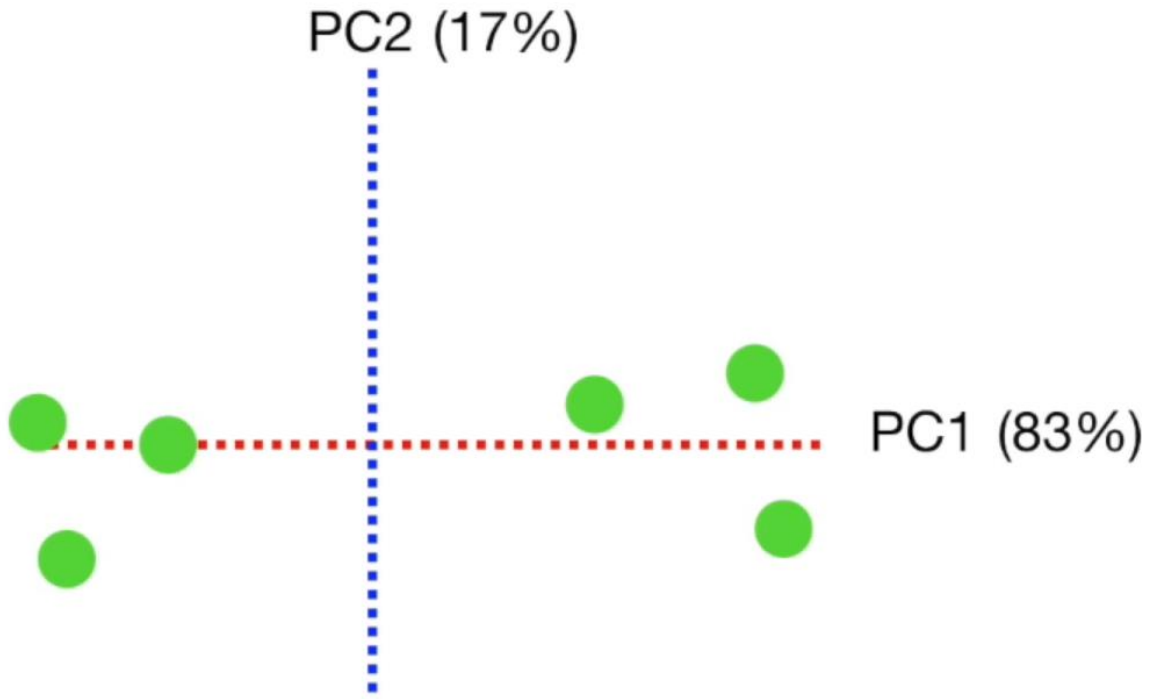
For 3-features dataset, we can repeat the same steps:

- Find the centre of the data, then shift the centre point to the origin
- Find the best fitting line that goes through the origin => this is PC1
- Find the next best fitting line given that it goes through the origin and is perpendicular to PC1 => this is PC2
- Project all samples onto PC1
- Project all samples onto PC2
- Now you have the PCA plot in 2D

Variation of each PC line and scree plot:

Finding the variation helps us know which PC line holds the most important information of the data, so that we can further reduce the dimension of data. Let's consider this example:

For the sake of example, let's say variation for PC1 = 15, and variation for PC2 = 3. That means PC1 accounts for $\frac{15}{18} = 0.83 = 83\%$ of the total variation, and PC2 accounts for $\frac{3}{18} = 17\%$.



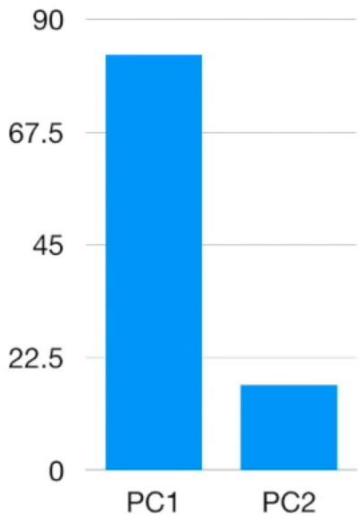
That means 83% information of the original data is reflected on PC1, and 17% is reflected on PC2. So we can tell that PC1 holds the most significant information of the original data.

The variation is calculated by dividing the Eigenvalue (or the SS_distances) by the sample size minus 1 (i.e., n-1):

$$\frac{\text{SS(distances for PC1)}}{n - 1} = \text{Variation for PC1}$$

$$\frac{\text{SS(distances for PC2)}}{n - 1} = \text{Variation for PC2}$$

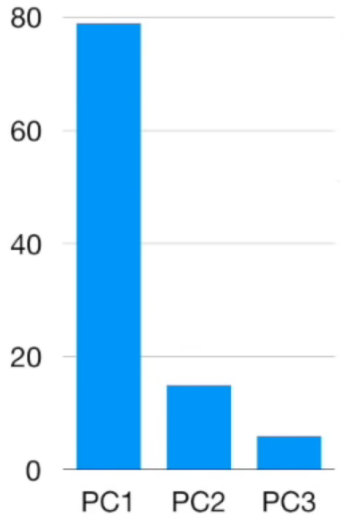
Scree plot: is the graphical representation of the percentages of variation of each PC:



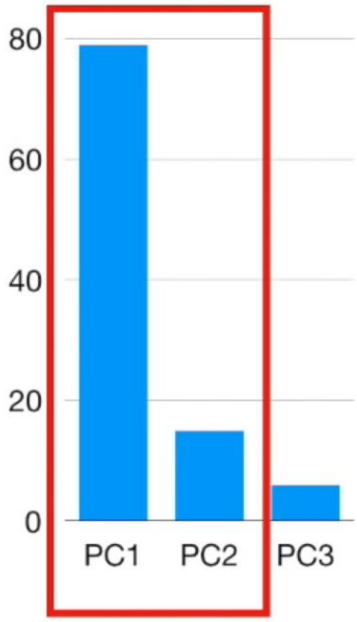
What are the variations and the scree plot for:

Let's say we are trying to reduce the 5D data into 3D data. This means we will have 3 PC lines: PC1, PC2, and PC3.

Let's say the variations of each PC is: PC1 = 79%, PC2 = 15%, PC3 = 6%. Hence, the scree plot would be:

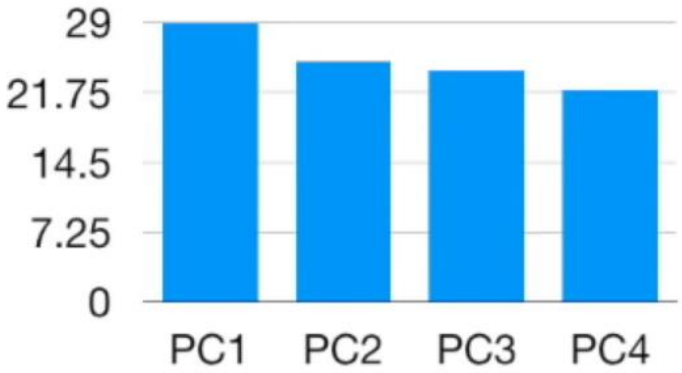


We can see PC1 and PC2 accounts for the vast majority of the variation. That means with just a 2D graph (using only PC1 and PC2), we can still represent an approximation as good as that of a 3D graph (using PC1, PC2, PC3), because PC1 & PC2 account for 94% of the variation of the data.



So we don't need to use a 3D graph for this data, 2D graph is still very informative.

Note: if you have a scree plot like this:



You can see each PC accounts for an equal amount of variation, then just using PC1 & PC2 would not create a very accurate representation of the data.

PCA implementation:

Let's see how PCA is implemented using sklearn library in the notebook named "PCA.ipynb".

References:

- StatQuest: Principal Component Analysis (PCA), Step-by-Step
  https://www.youtube.com/watch?v=FgakZw6K1QQ&t=25s