

## 주성분 분석 (Principal Component Analysis)

~~상관관계가 있는 변수들을 선형 결합하여 변수를 축약하는 기법. 요인 분석의 한 종류.~~

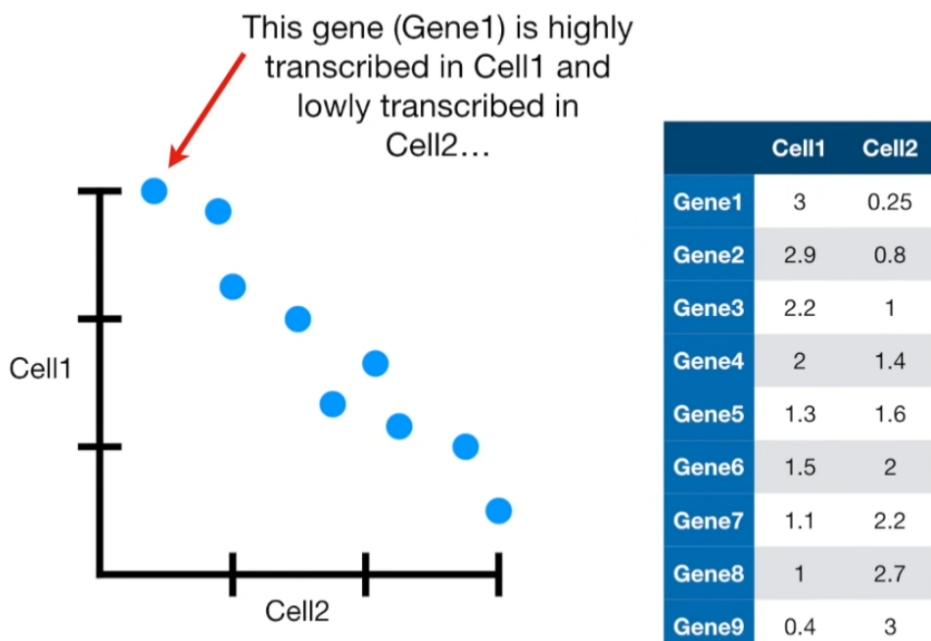
첫번째 주성분이 전체 변동을 가장 많이 설명하고, 두번째 주성분은 첫번째 주성분과 상관성이 낮아 첫 번째 주성분이 설명하지 못하는 나머지 변동을 정보의 손실없이 가장 많이 설명할 수 있도록 변수들을 조합.

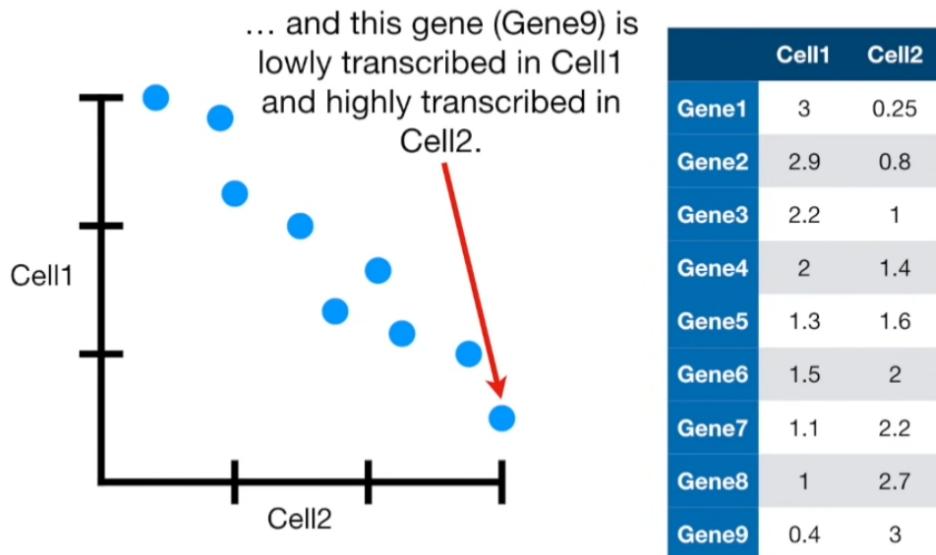
### 1. 목적

- ~~- 여러 변수들 간에 내재하는 상관관계, 연관성을 이용해 소수의 주성분으로 차원을 축소~~
- 다중공선성이 존재하는 경우, 상관성이 적은 주성분으로 변수들을 축소하여 모형 개발에 활용
  - 주성분분석을 통해 차원을 축소한 후 군집분석을 수행하면 결과와 연산속도를 개선할 수 있음
  - 다량의 센서 데이터를 주성분분석으로 차원 축소한 후 시계열로 분포나 추세의 변화를 분석하여 고장 징후를 사전에 파악

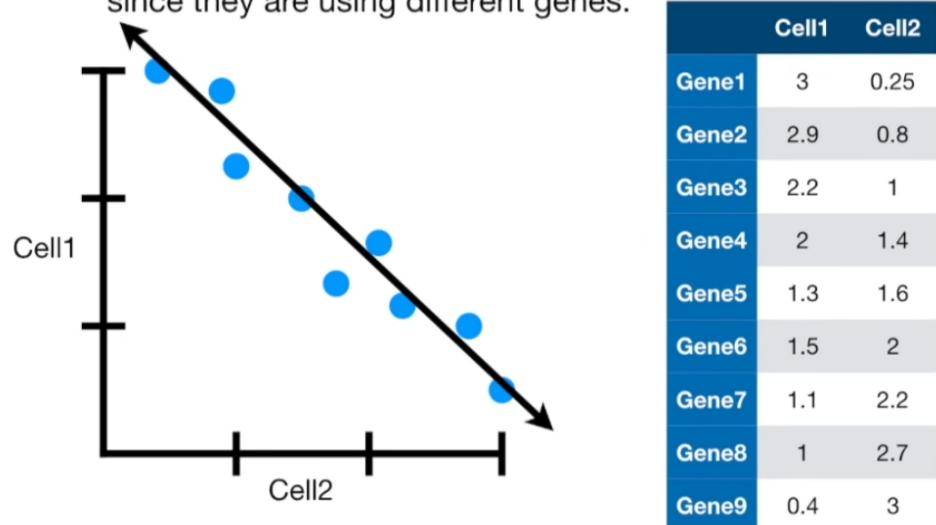
Here's the data...

	Cell1	Cell2	Cell3	Cell4	...
Gene1	3	0.25	2.8	0.1	...
Gene2	2.9	0.8	2.2	1.8	...
Gene3	2.2	1	1.5	3.2	...
Gene4	2	1.4	2	0.3	...
Gene5	1.3	1.6	1.6	0	...
Gene6	1.5	2	2.1	3	...
Gene7	1.1	2.2	1.2	2.8	...
Gene8	1	2.7	0.9	0.3	...
Gene9	0.4	3	0.6	0.1	...



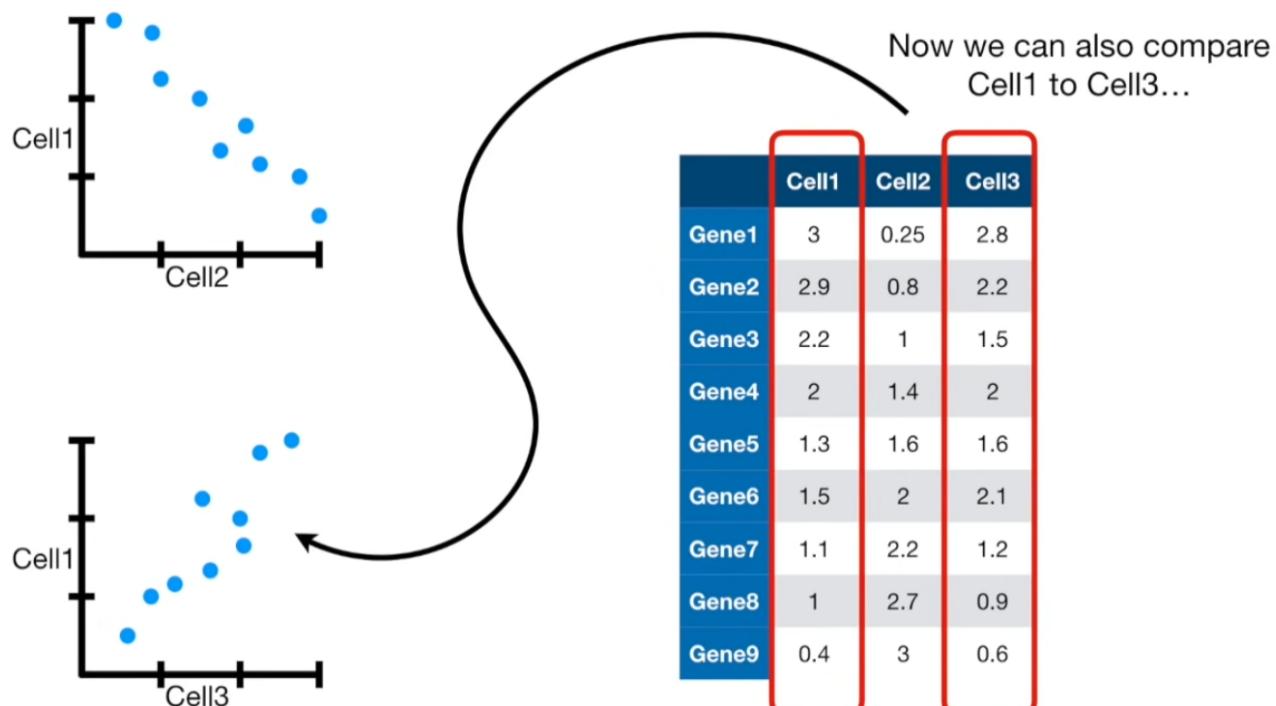


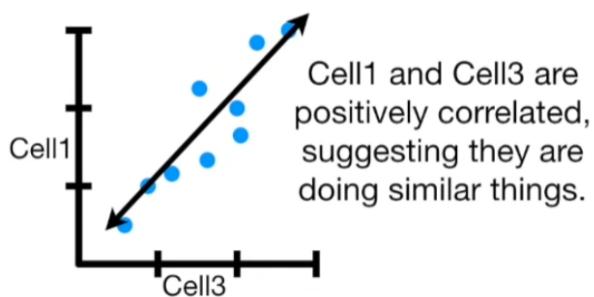
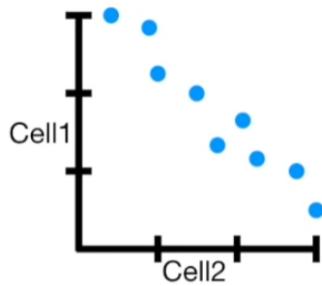
In general, Cell1 and Cell2 have an inverse correlation. This means that they are probably two different types of cells since they are using different genes.



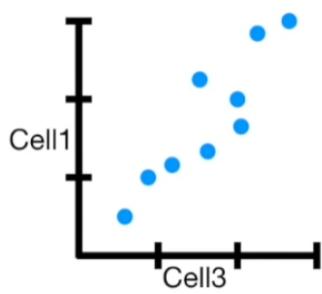
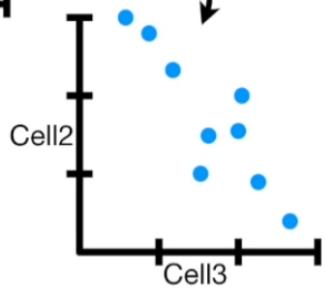
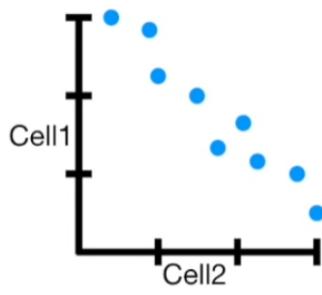
Now let's imagine there  
are 3 cells.

	Cell1	Cell2	Cell3
Gene1	3	0.25	2.8
Gene2	2.9	0.8	2.2
Gene3	2.2	1	1.5
Gene4	2	1.4	2
Gene5	1.3	1.6	1.6
Gene6	1.5	2	2.1
Gene7	1.1	2.2	1.2
Gene8	1	2.7	0.9
Gene9	0.4	3	0.6



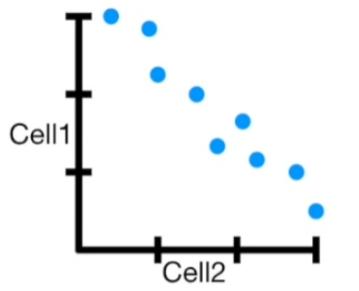


	Cell1	Cell2	Cell3
Gene1	3	0.25	2.8
Gene2	2.9	0.8	2.2
Gene3	2.2	1	1.5
Gene4	2	1.4	2
Gene5	1.3	1.6	1.6
Gene6	1.5	2	2.1
Gene7	1.1	2.2	1.2
Gene8	1	2.7	0.9
Gene9	0.4	3	0.6

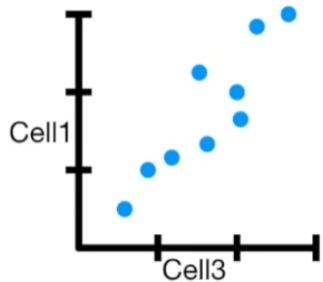
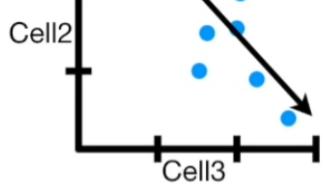


Lastly, we can also compare Cell2 to Cell3...

	Cell1	Cell2	Cell3
Gene1	3	0.25	2.8
Gene2	2.9	0.8	2.2
Gene3	2.2	1	1.5
Gene4	2	1.4	2
Gene5	1.3	1.6	1.6
Gene6	1.5	2	2.1
Gene7	1.1	2.2	1.2
Gene8	1	2.7	0.9
Gene9	0.4	3	0.6



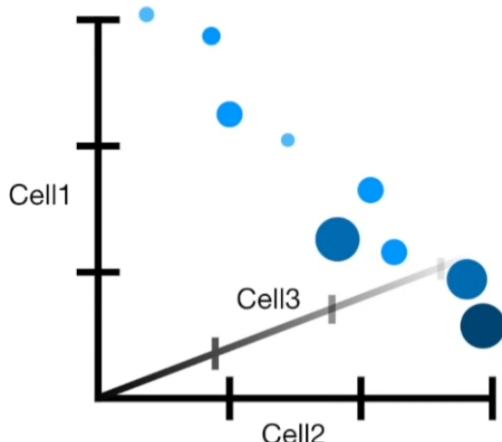
The negative correlation suggests that Cell2 is doing something different from Cell3...



	Cell1	Cell2	Cell3
Gene1	3	0.25	2.8
Gene2	2.9	0.8	2.2
Gene3	2.2	1	1.5
Gene4	2	1.4	2
Gene5	1.3	1.6	1.6
Gene6	1.5	2	2.1
Gene7	1.1	2.2	1.2
Gene8	1	2.7	0.9
Gene9	0.4	3	0.6

Alternatively, we could try to plot all three cells at once on a 3-dimensional graph.

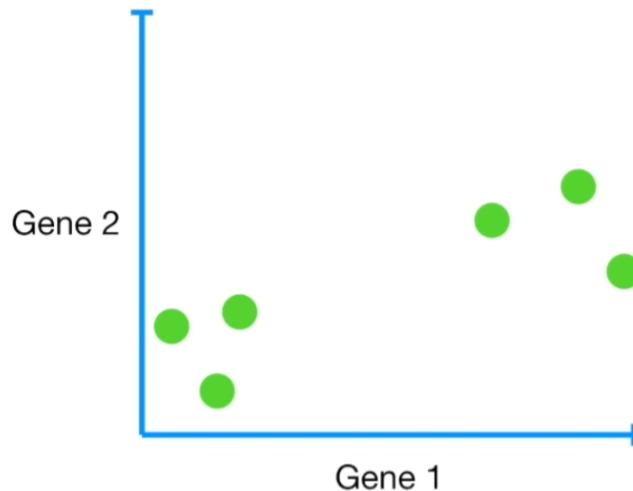
이(이)드는 3차원 공간에서 선형을 띠고 있는 때로면,  
세 변수를 한 개의 변수로 측정할 수 있음.



	Cell1	Cell2	Cell3
Gene1	3	0.25	2.8
Gene2	2.9	0.8	2.2
Gene3	2.2	1	1.5
Gene4	2	1.4	2
Gene5	1.3	1.6	1.6
Gene6	1.5	2	2.1
Gene7	1.1	2.2	1.2
Gene8	1	2.7	0.9
Gene9	0.4	3	0.6

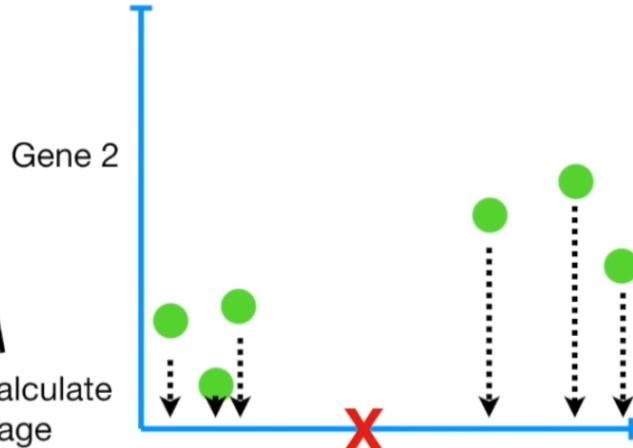
	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1

We'll start by plotting the data...



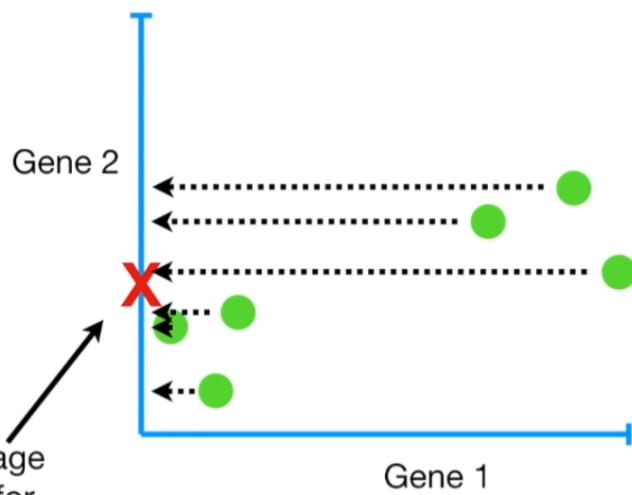
	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1

Then we'll calculate  
the average  
measurement for  
Gene 1...

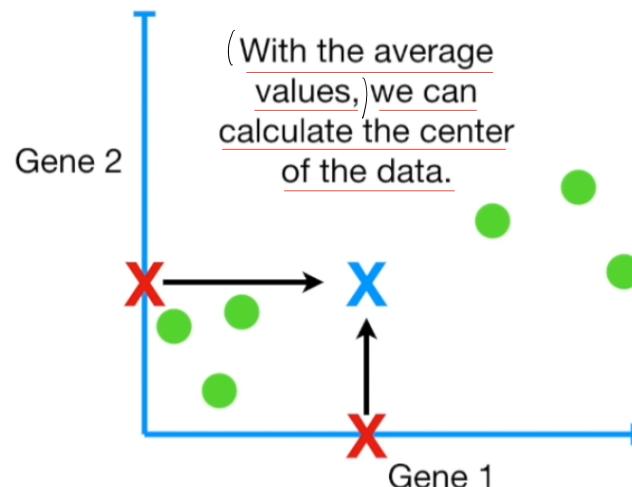


	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1

...and the average measurement for Gene 2.

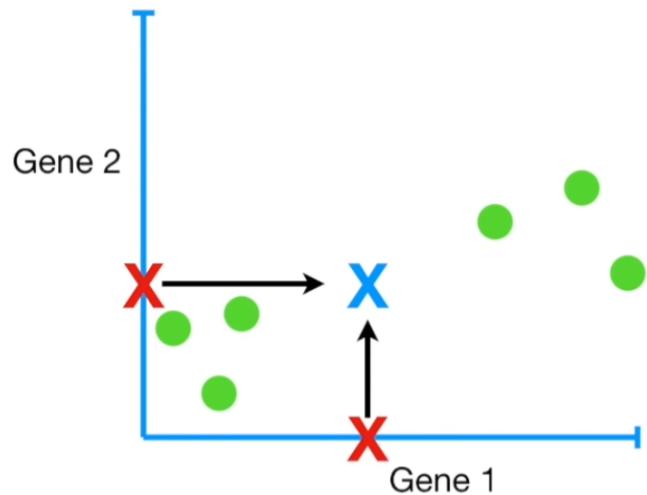


	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1

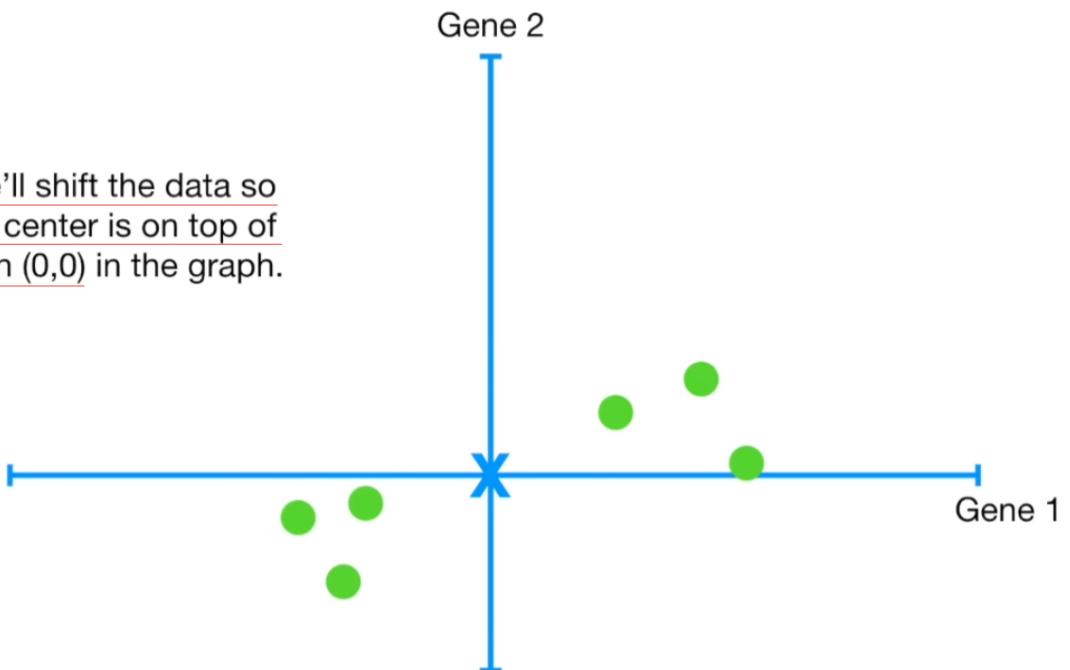


	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1

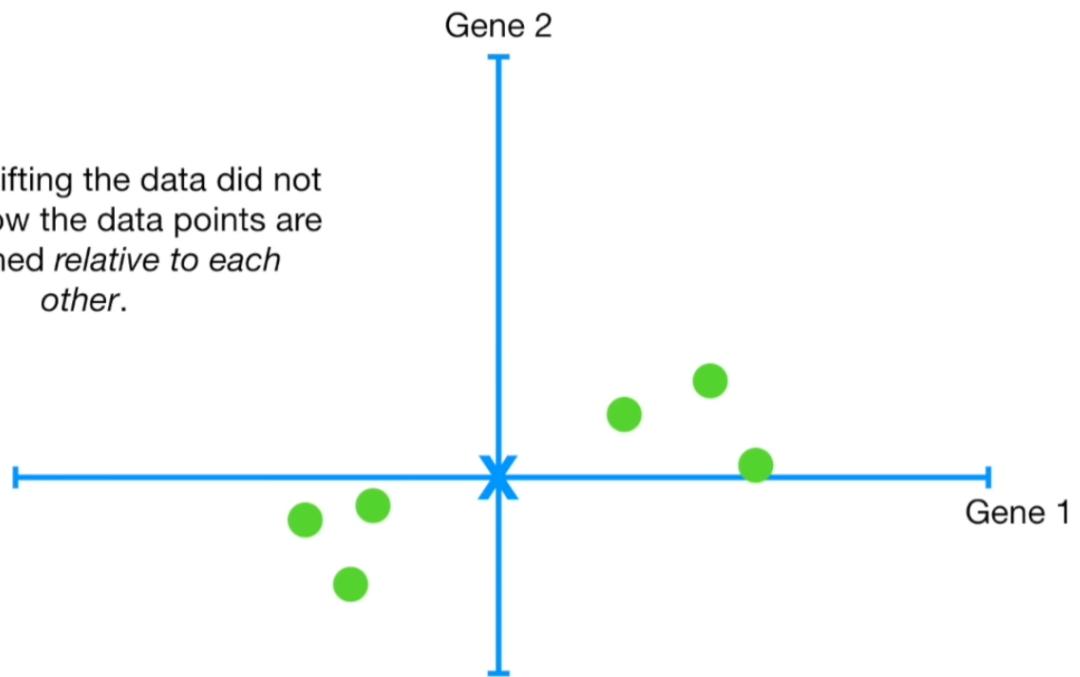
From this point on, we'll focus on what happens in the graph; we no longer need the original data...



Now we'll shift the data so that the center is on top of the origin (0,0) in the graph.

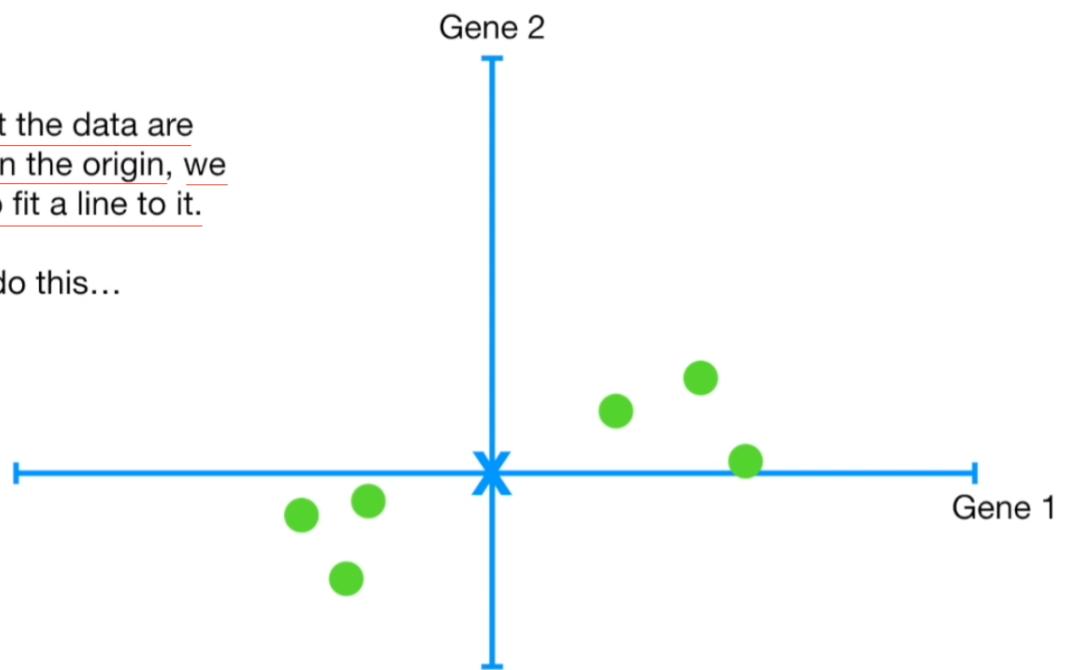


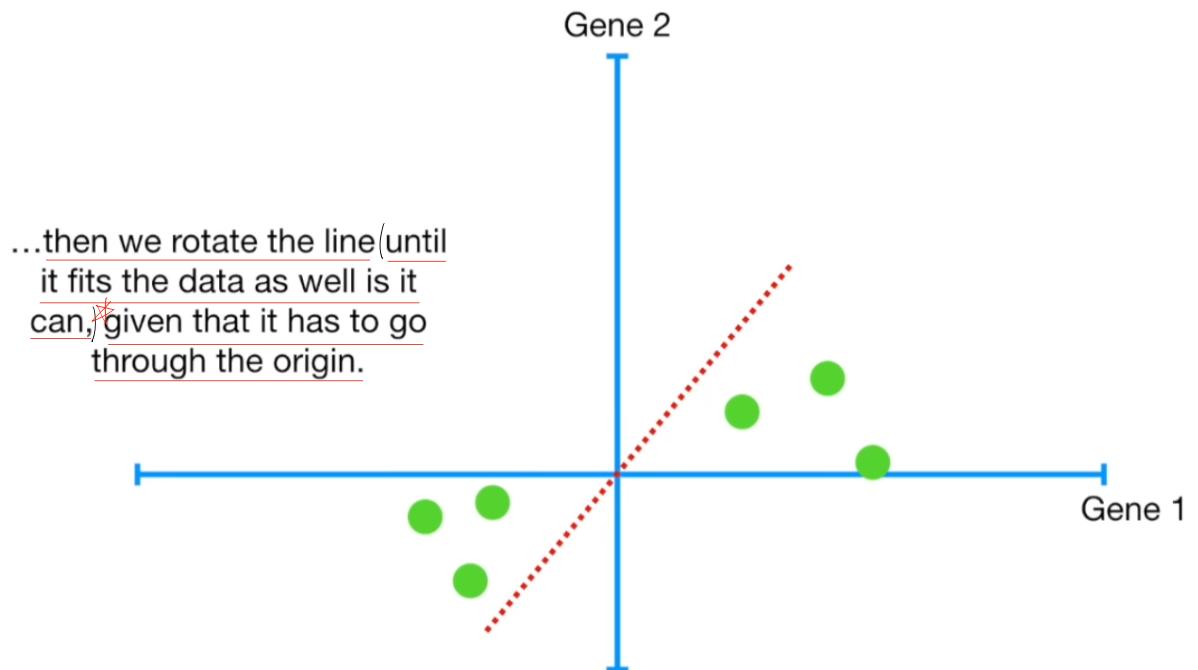
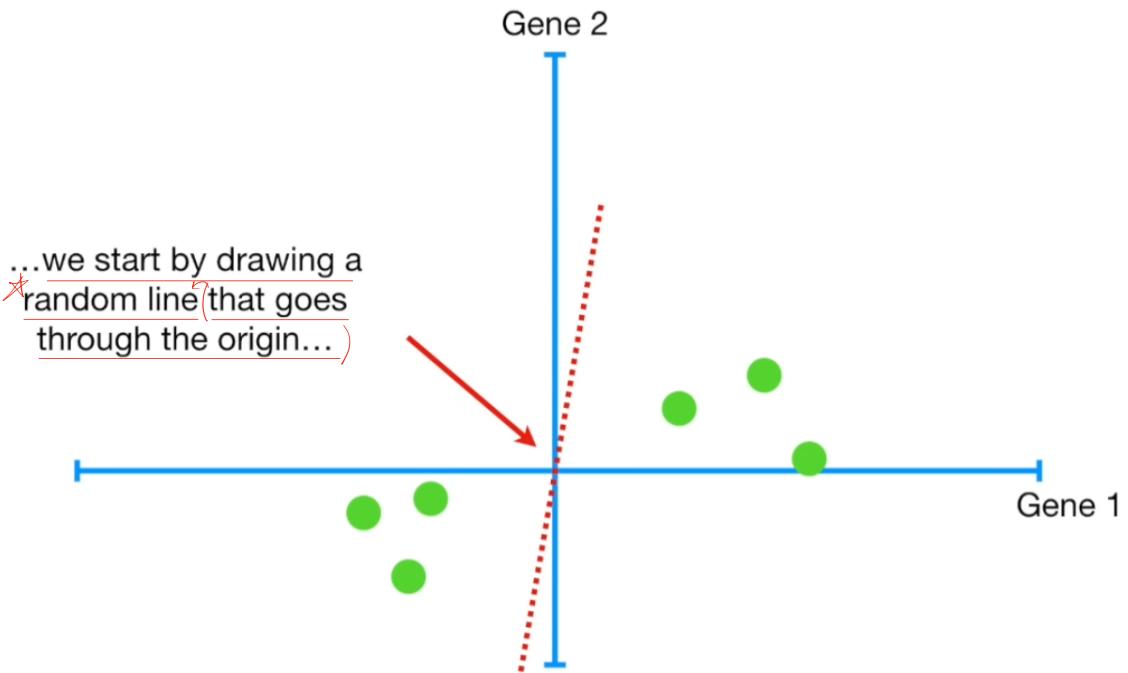
**NOTE:** Shifting the data did not change how the data points are positioned *relative to each other*.

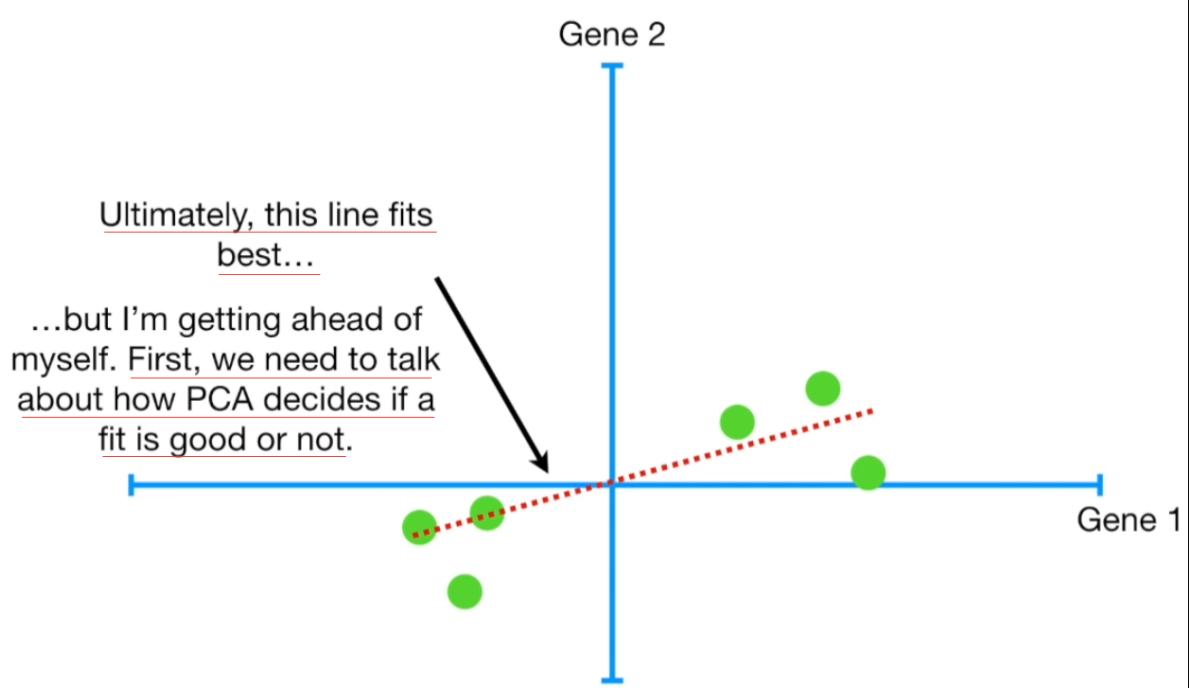
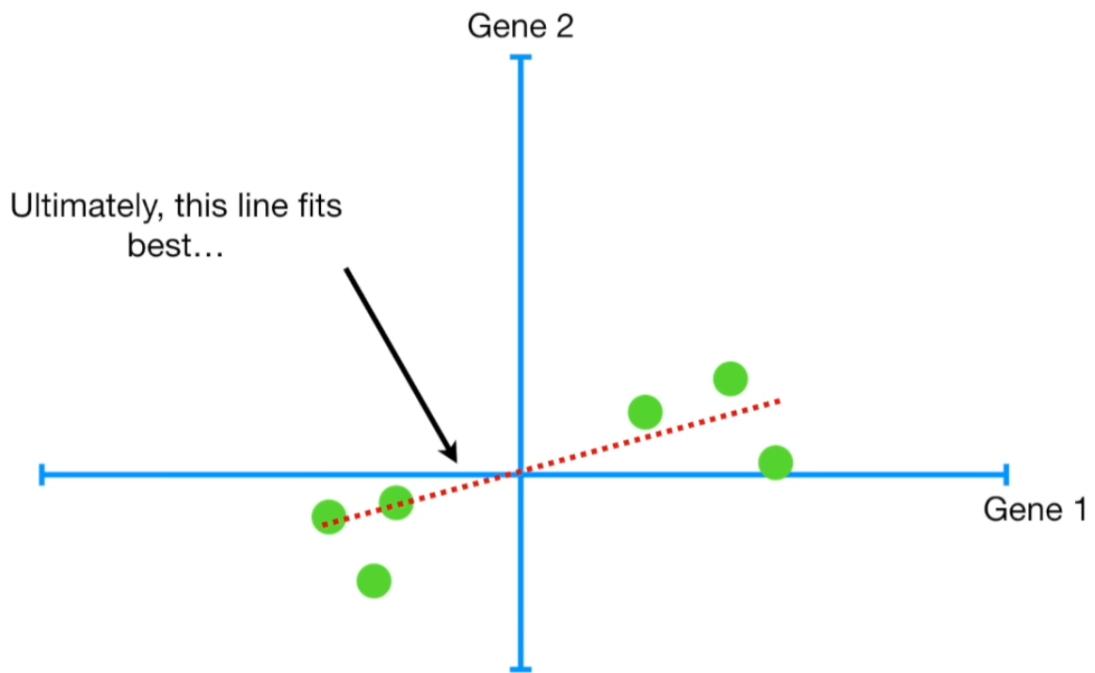


Now that the data are centered on the origin, we can try to fit a line to it.

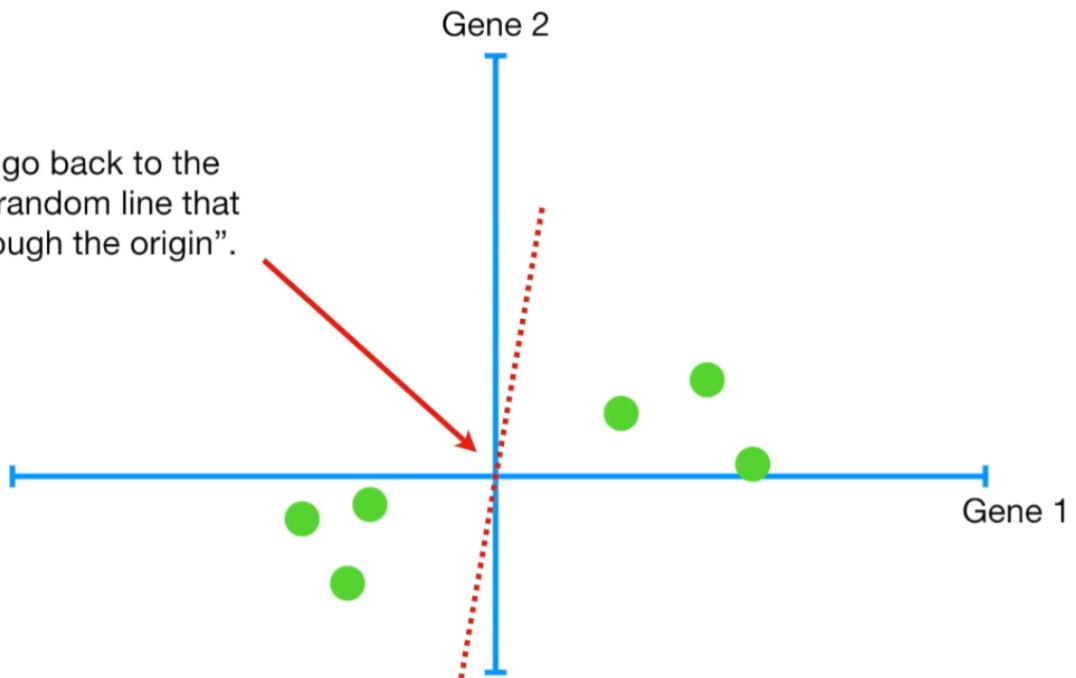
To do this...



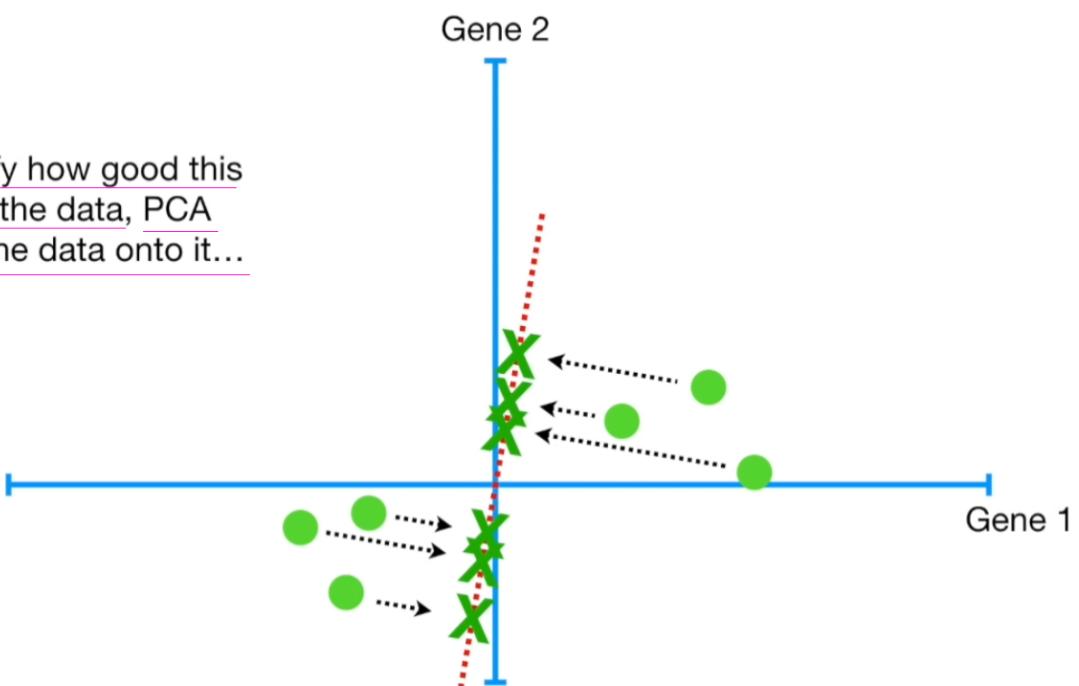


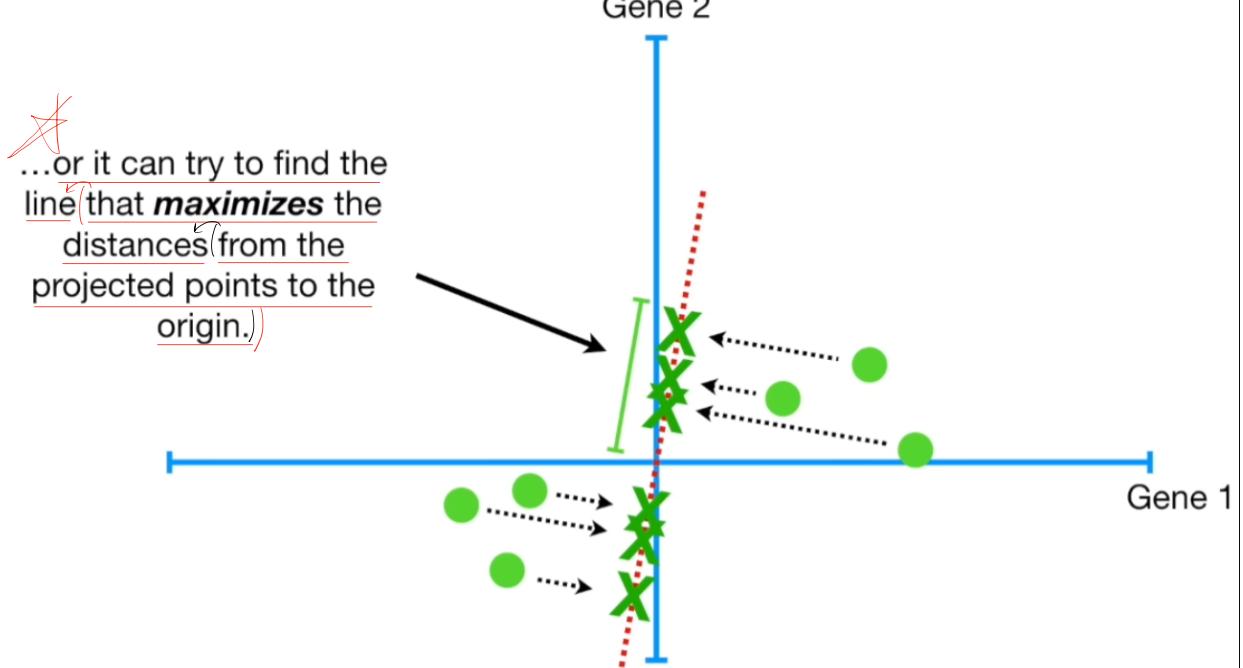
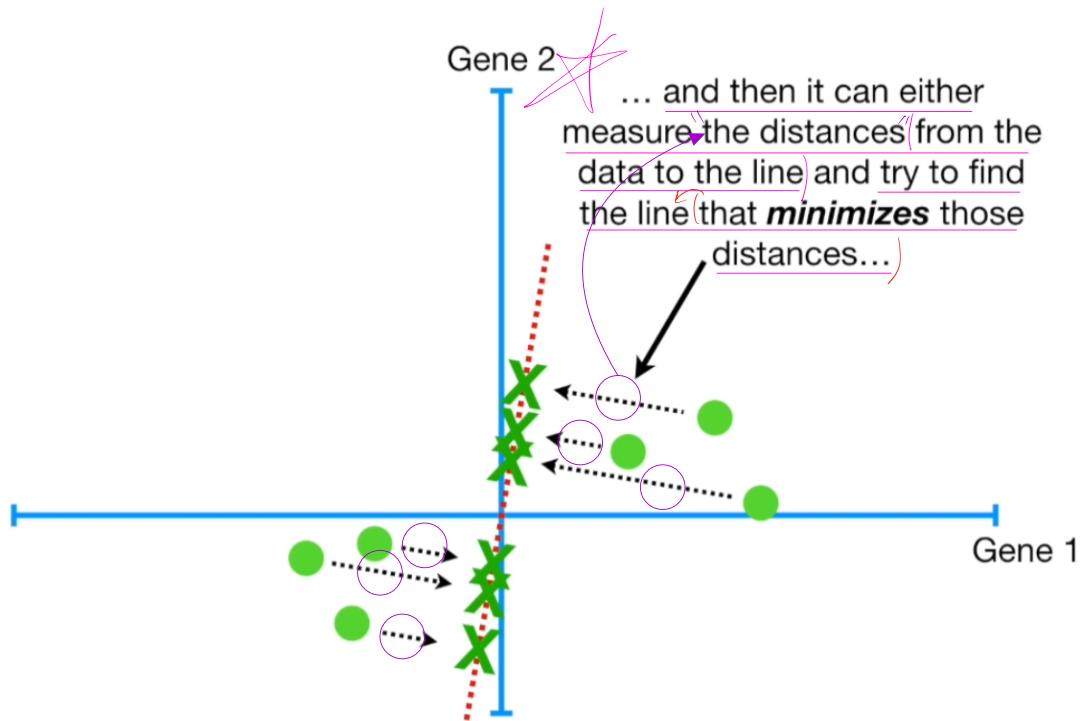


So let's go back to the original "random line that goes through the origin".

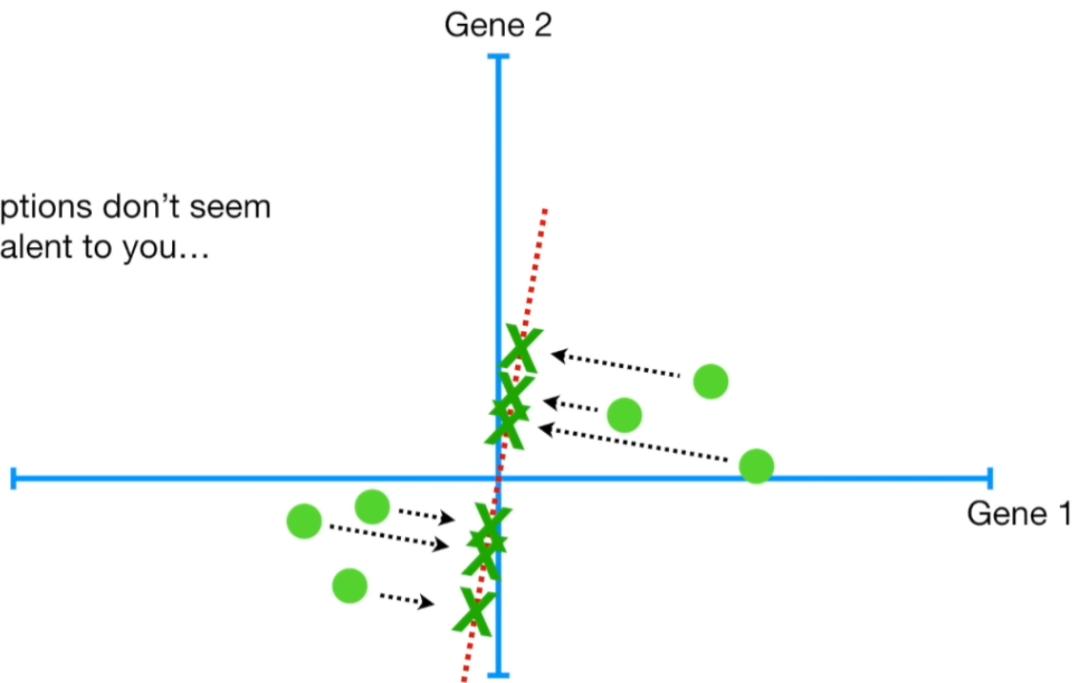


~~To quantify how good this line fits the data, PCA projects the data onto it...~~





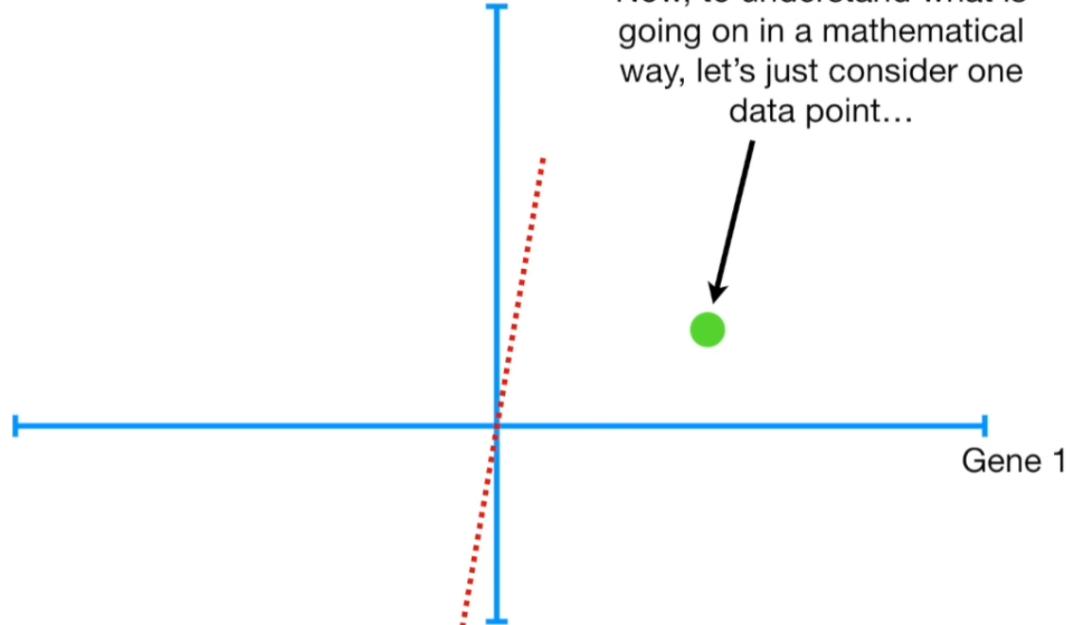
If those options don't seem equivalent to you...

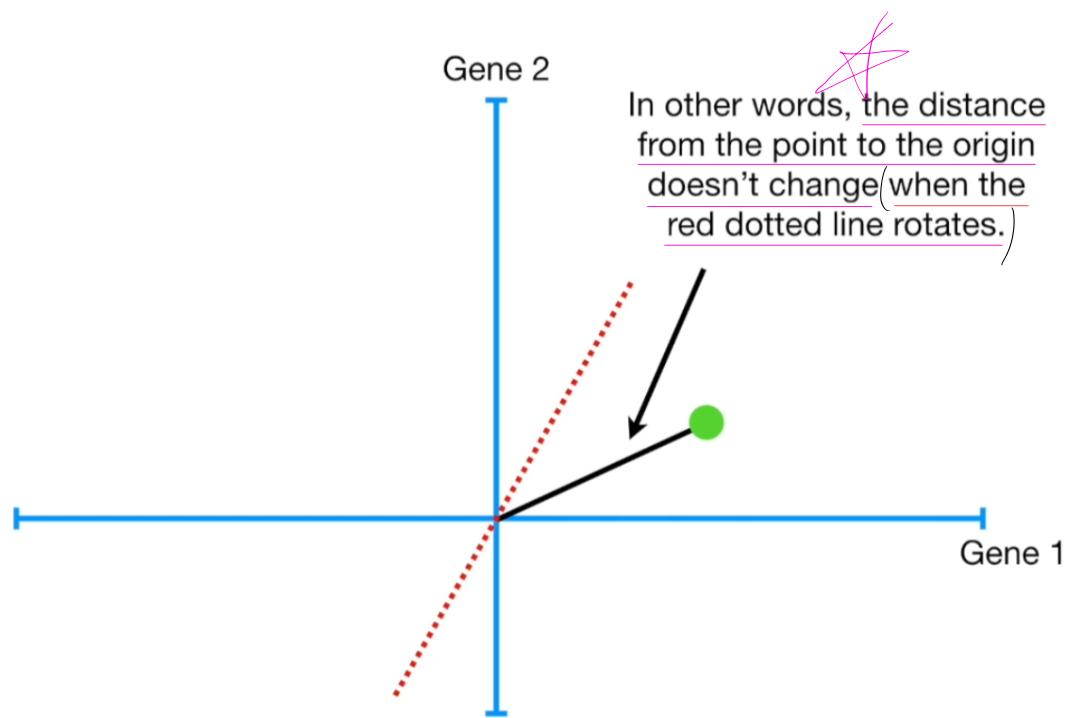
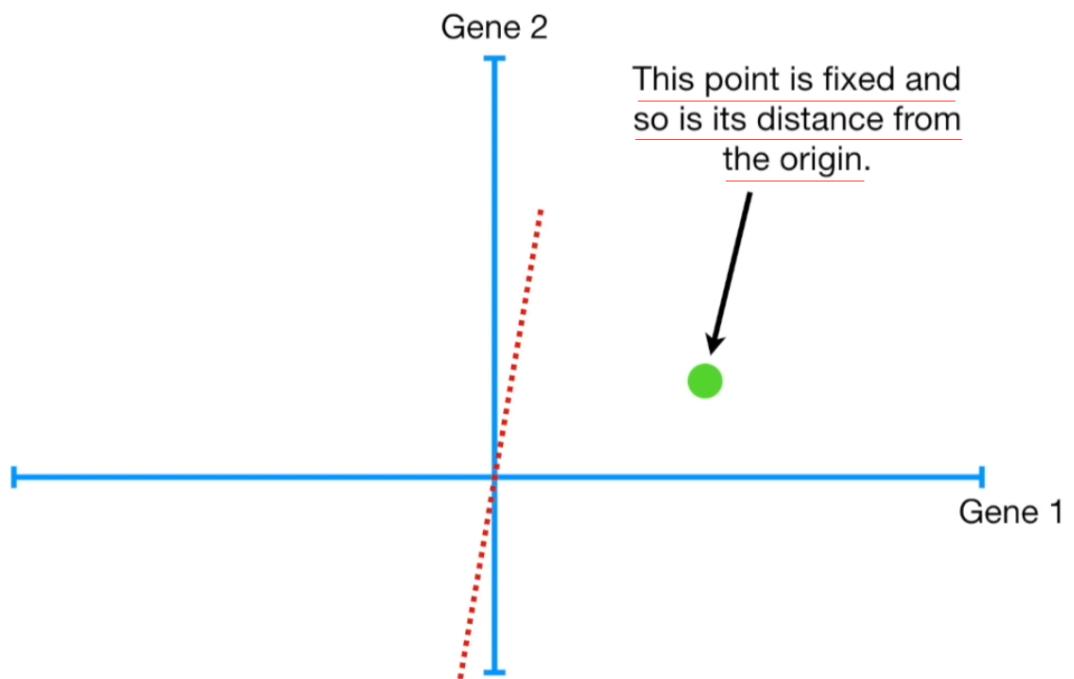


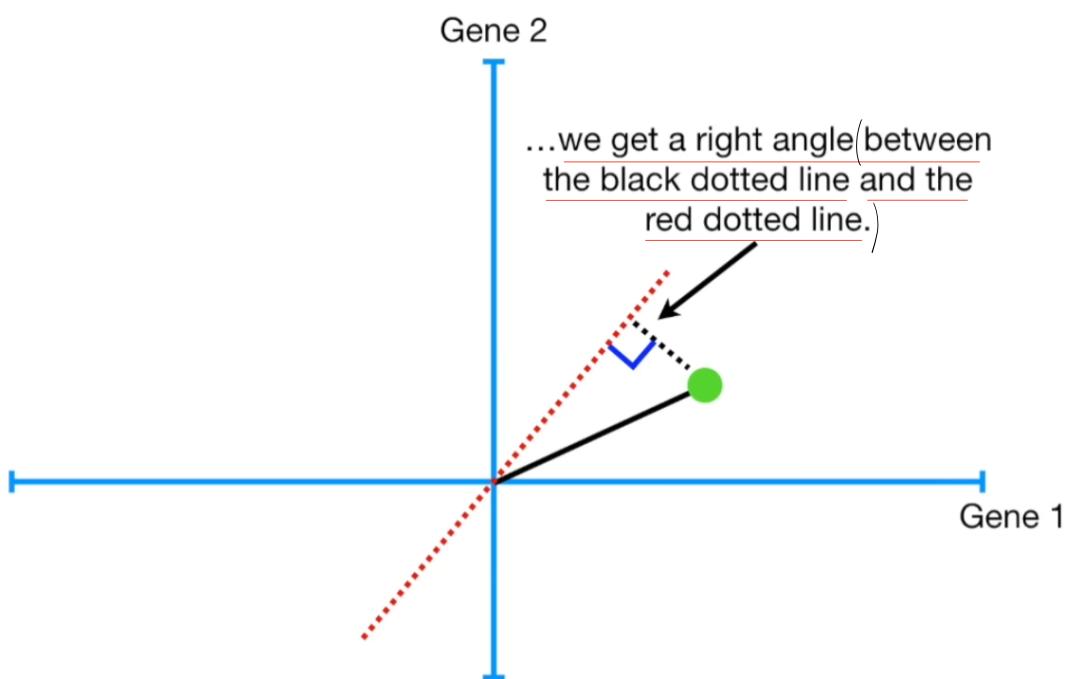
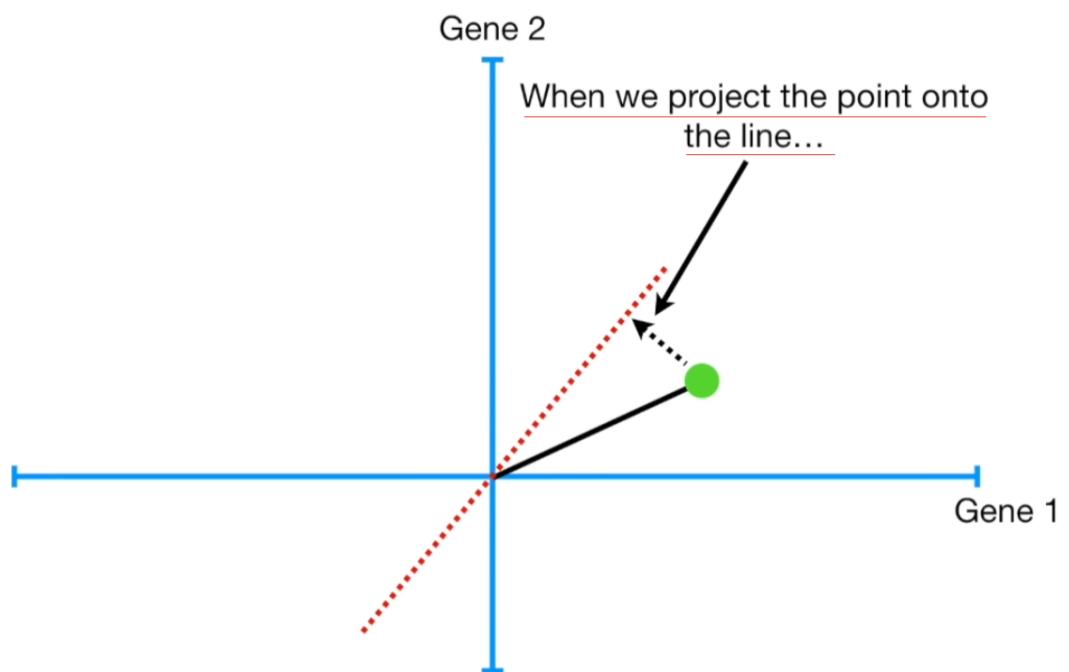
Gene 2

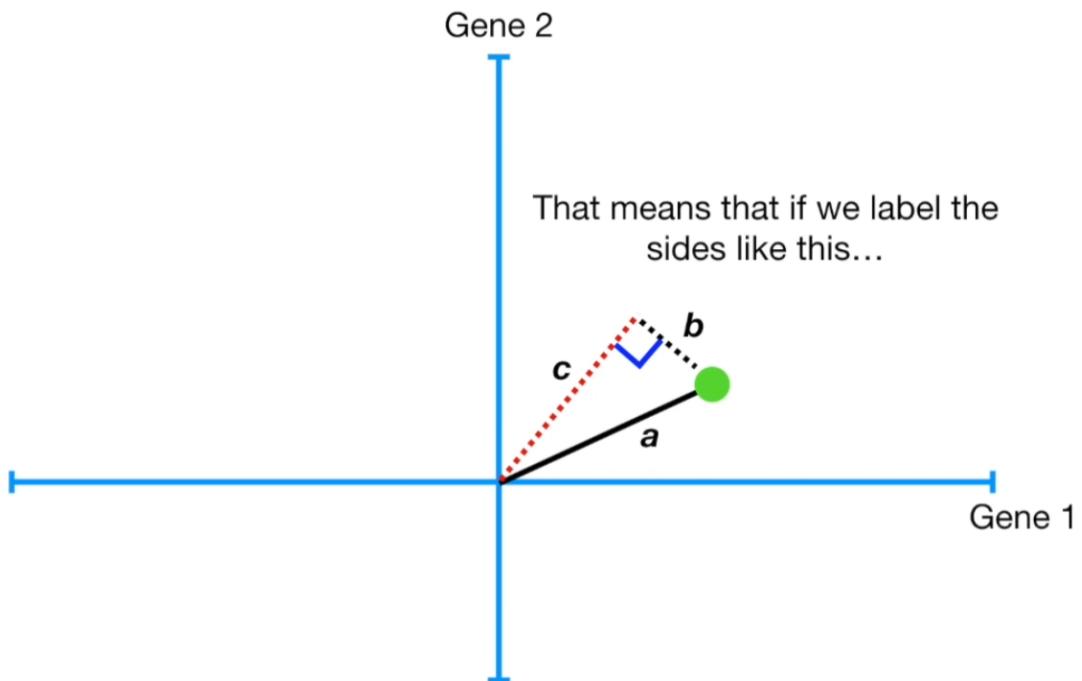
Gene 1

Now, to understand what is going on in a mathematical way, let's just consider one data point...



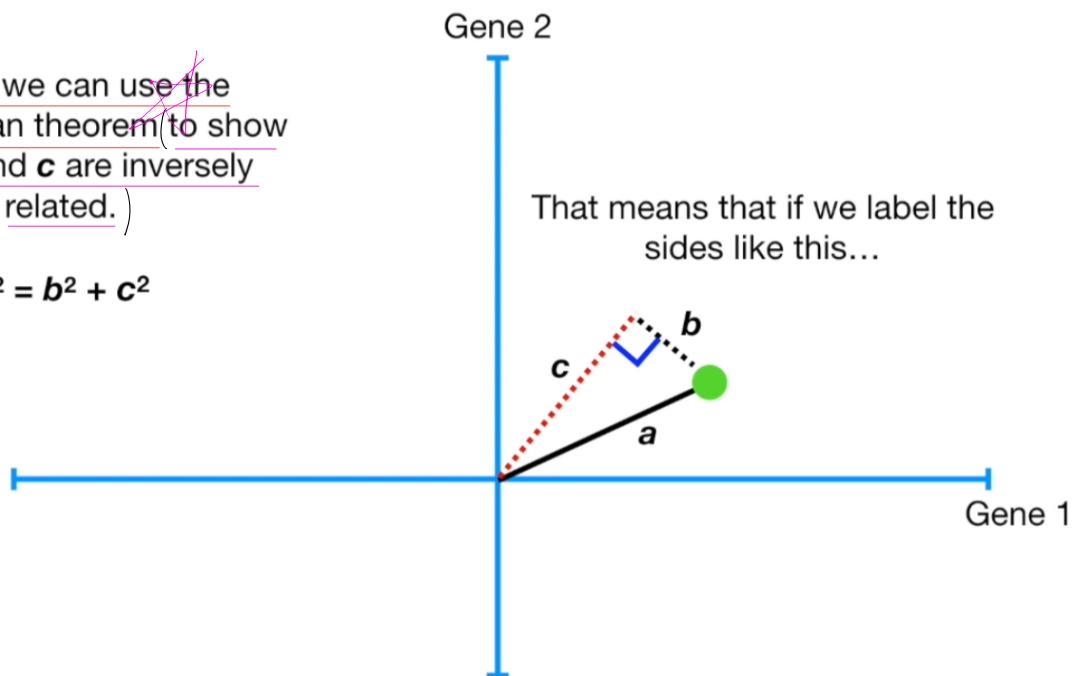




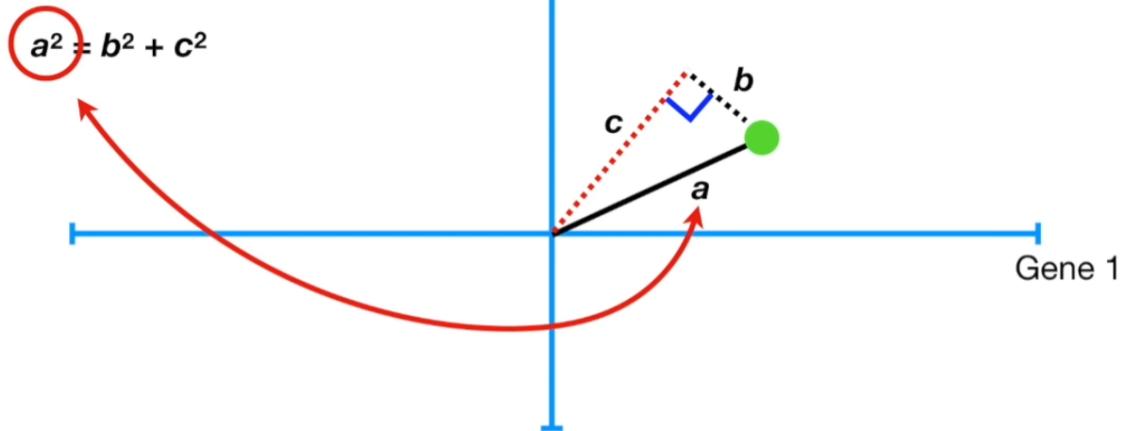


...then we can use the Pythagorean theorem (to show how  $b$  and  $c$  are inversely related.)

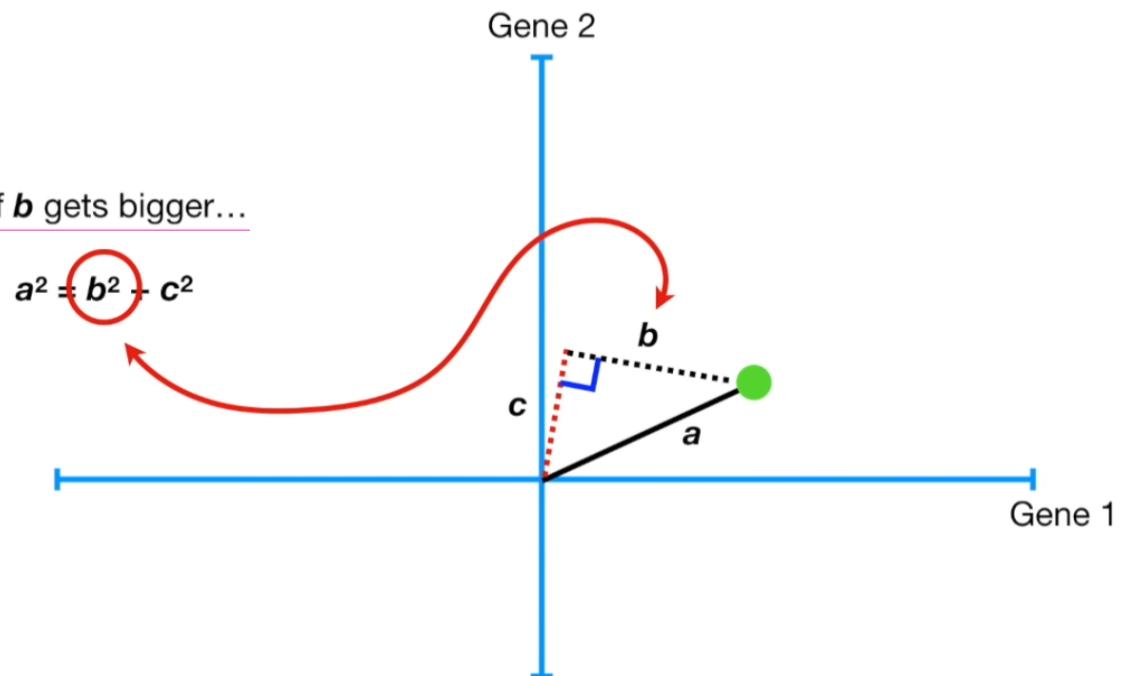
$$a^2 = b^2 + c^2$$



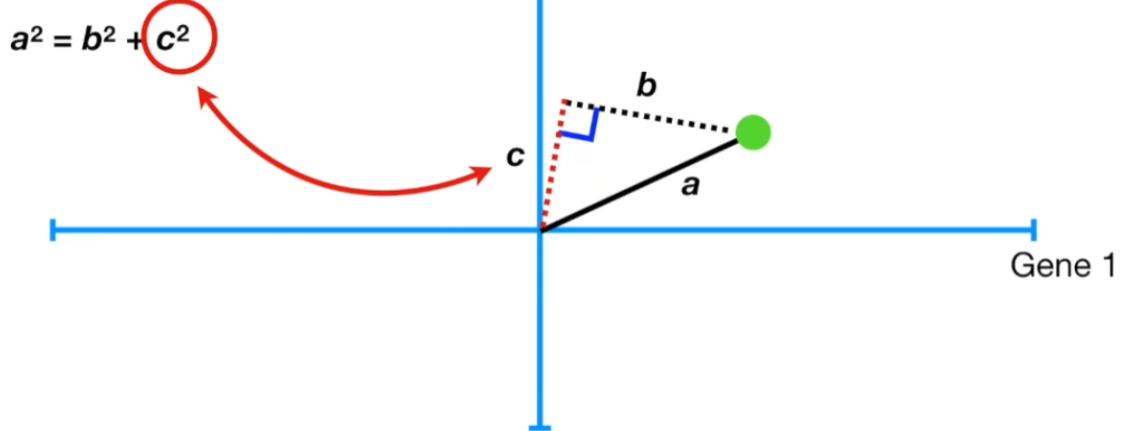
Since  $a$  (and thus  $a^2$ )  
doesn't change...



If  $b$  gets bigger...

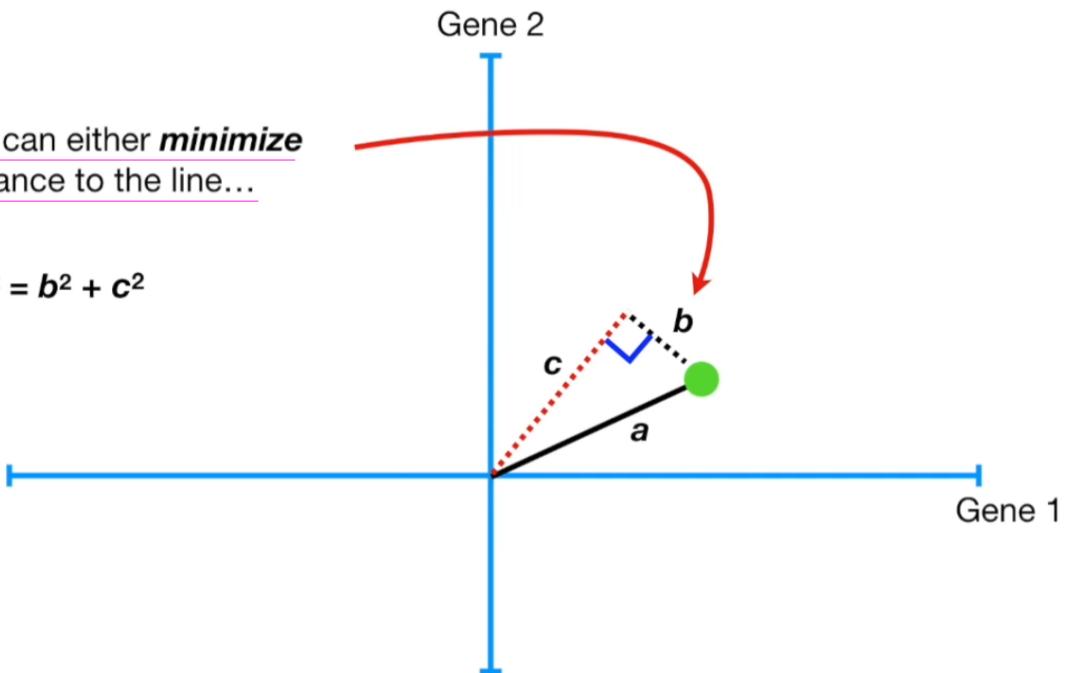


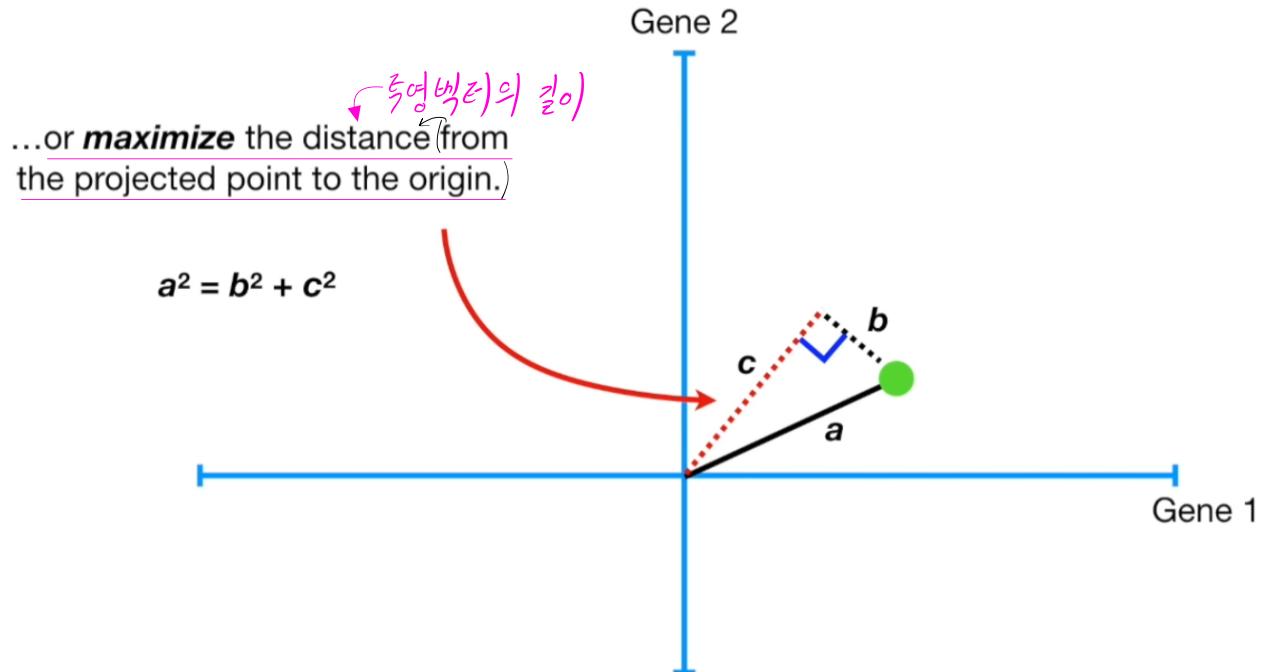
...then  $c$  must get smaller.



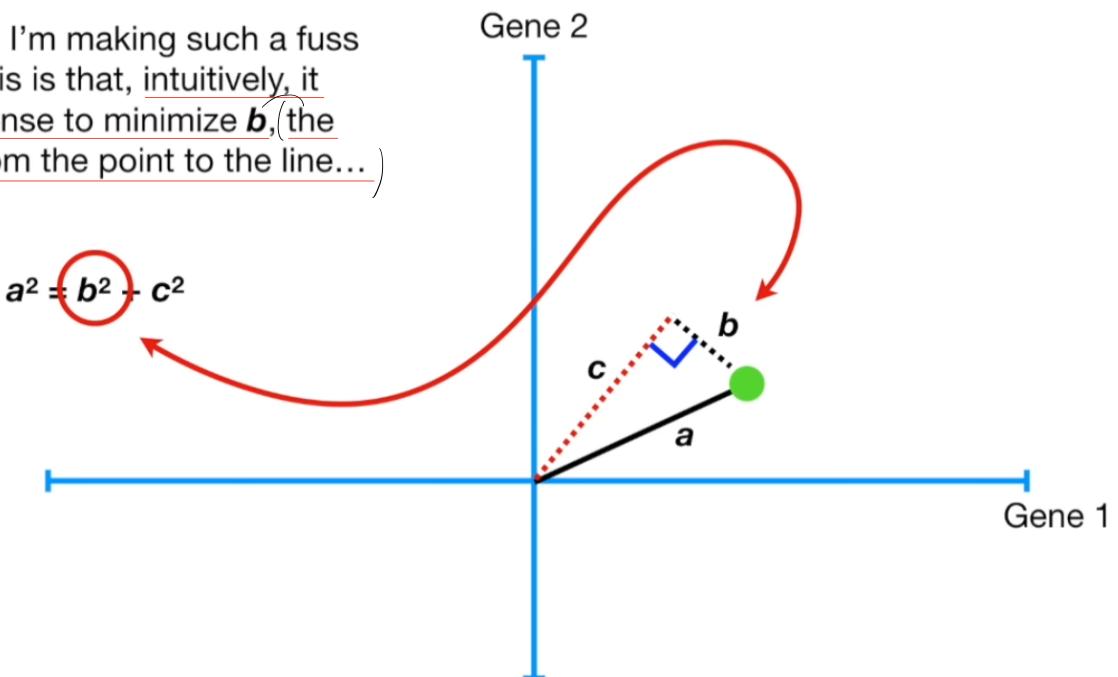
Thus, PCA can either **minimize** the distance to the line...

$$a^2 = b^2 + c^2$$

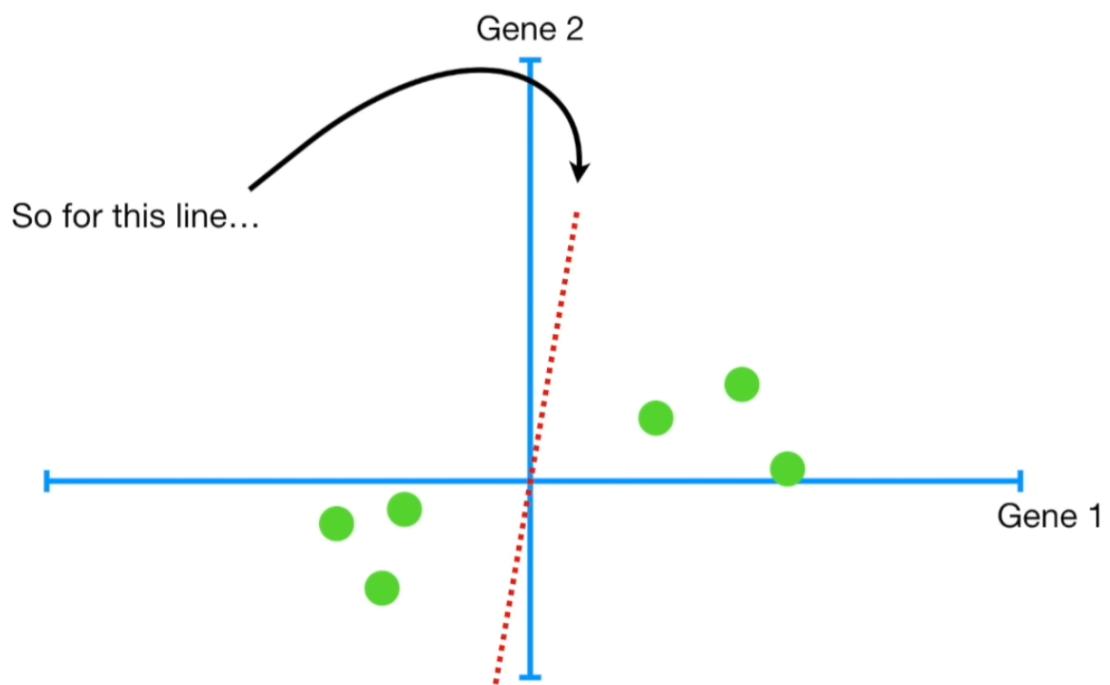
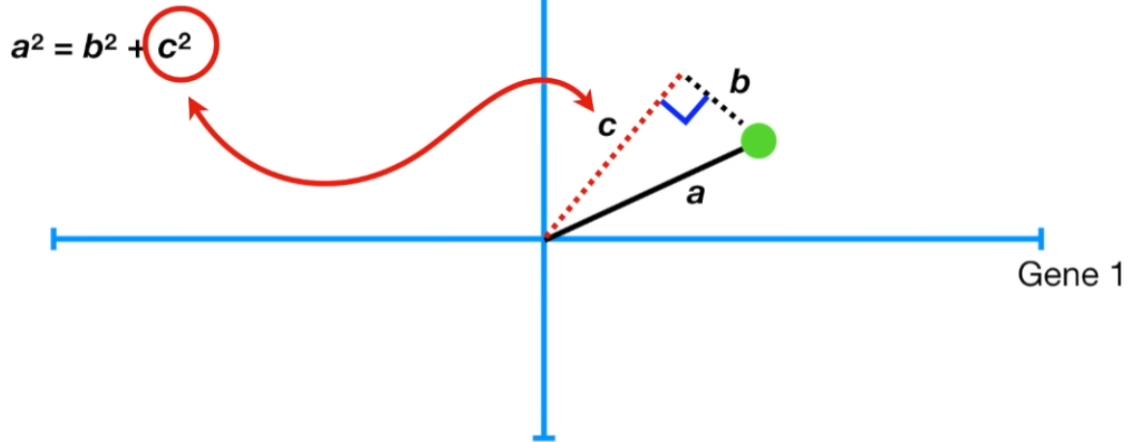


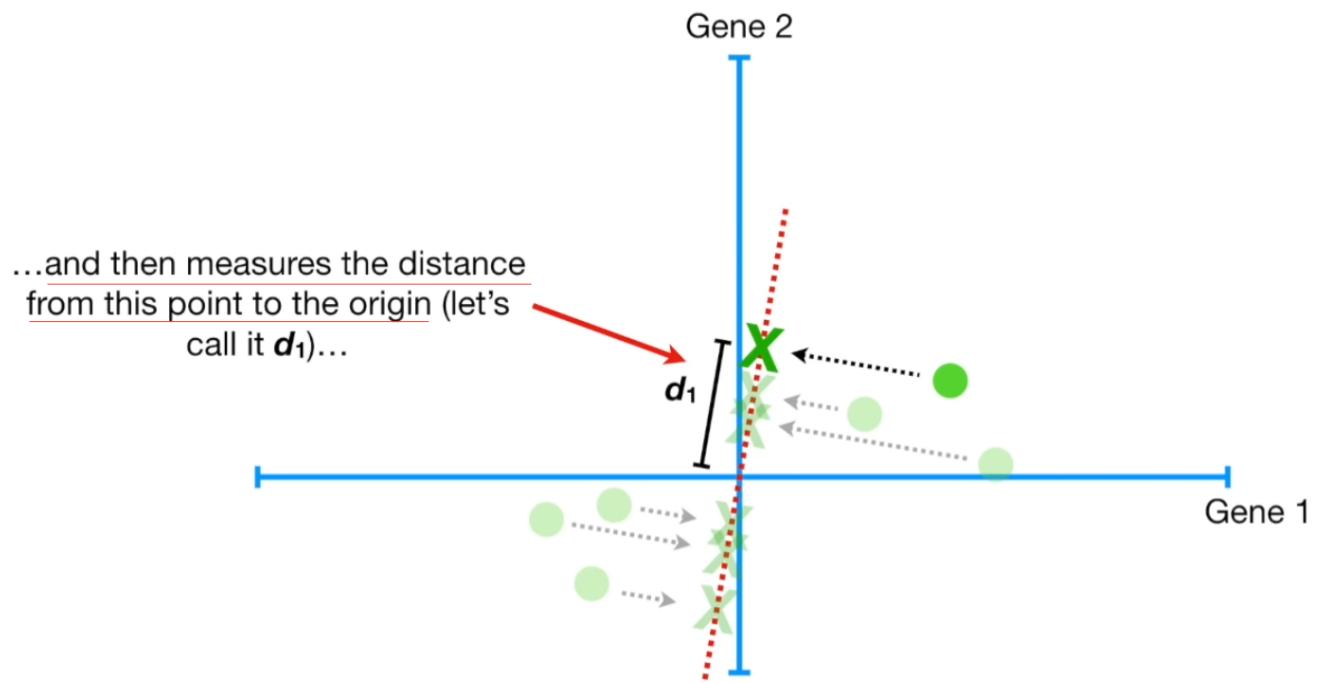
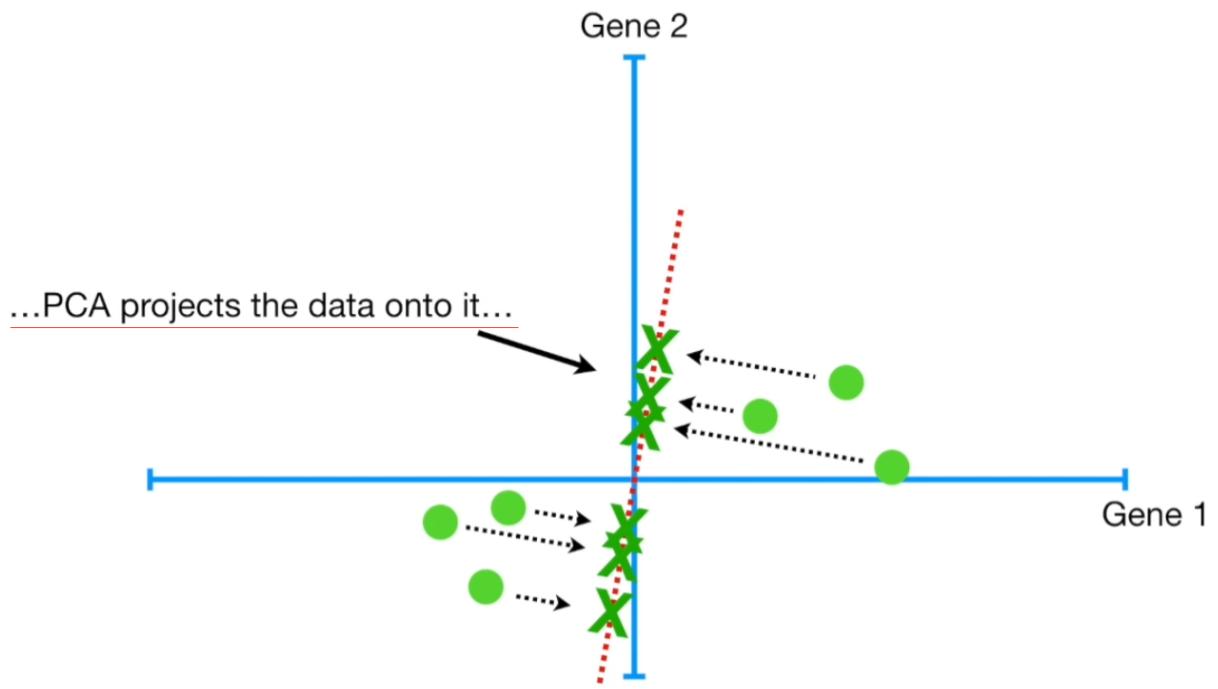


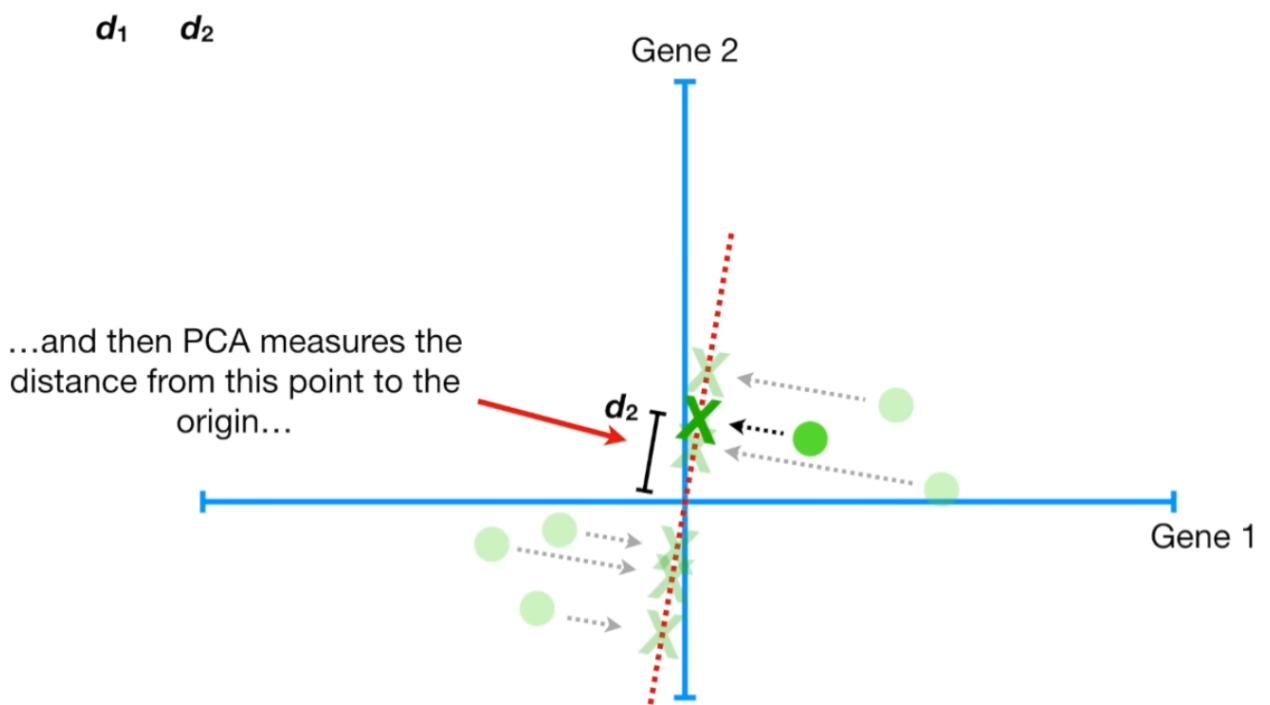
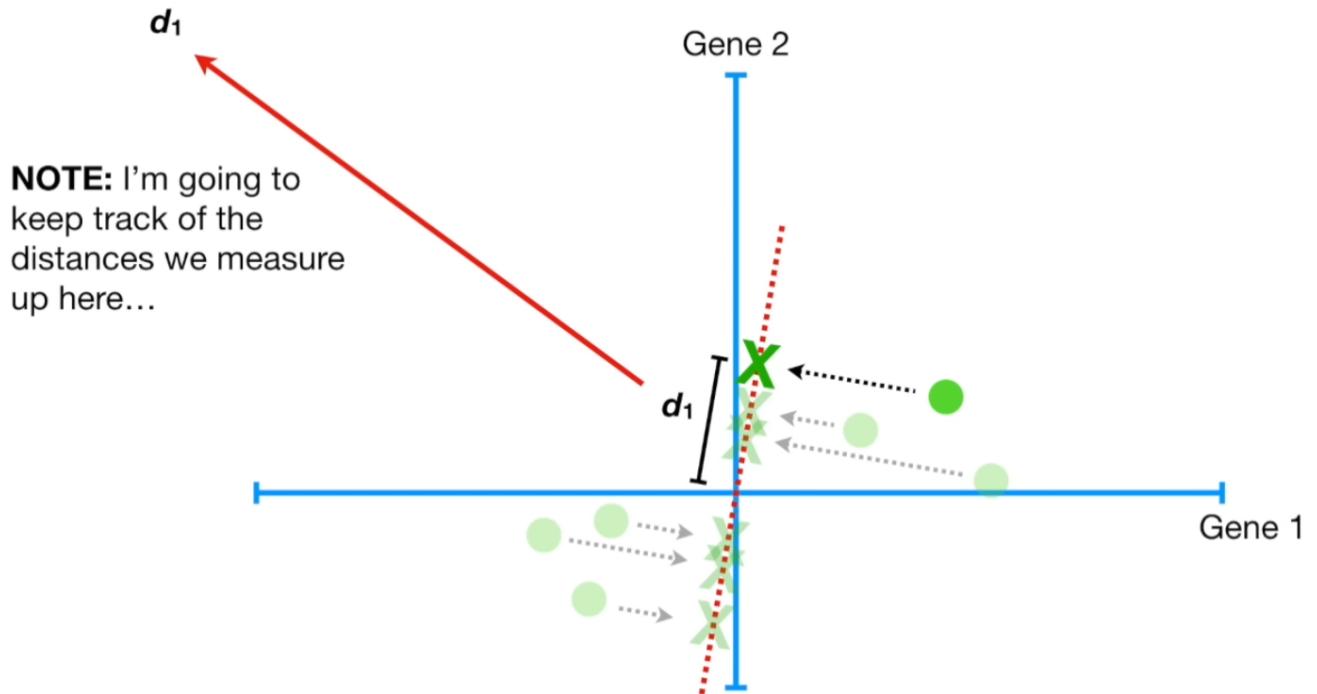
The reason I'm making such a fuss about this is that, intuitively, it makes sense to minimize **b**, (the distance from the point to the line...)



~~...but it's actually easier to calculate  $c$ ,  
the distance from the projected point  
to the origin, so PCA finds the best  
fitting line (by maximizing the sum of  
the squared distances from the  
projected points to the origin.)~~







$$d_1 \quad d_2 \quad d_3 \quad d_4 \quad d_5 \quad d_6$$

Here are all 6 distances  
that we measured.

Gene 2



Gene 1

$$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2$$

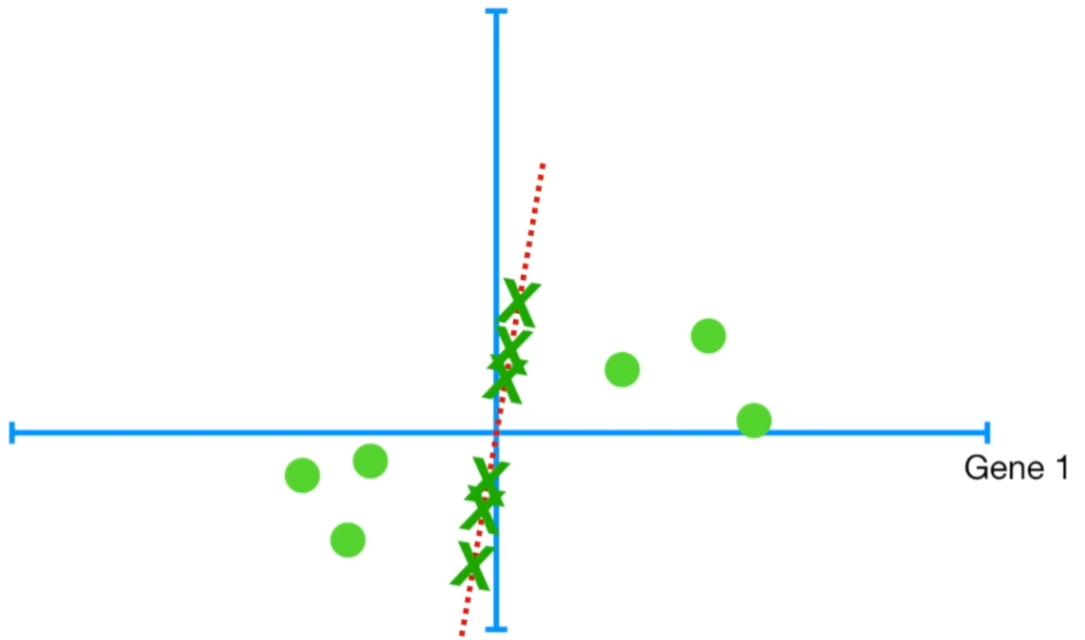
Then we sum up all  
these squared  
distances...

Gene 2



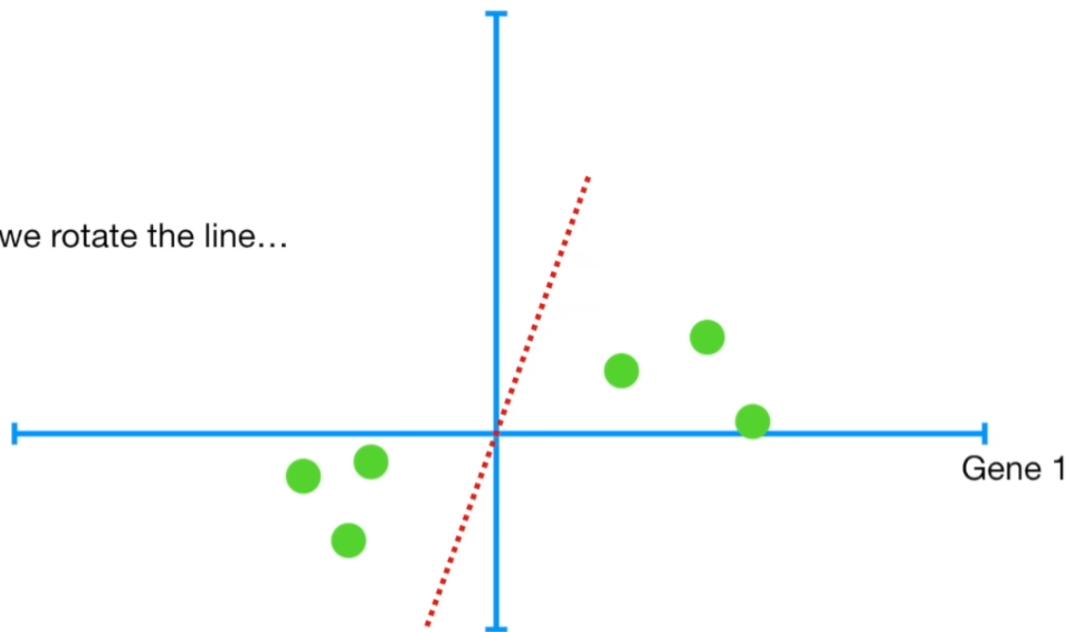
Gene 1

$$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2 = \text{sum of squared distances} = \text{SS(distances)}$$



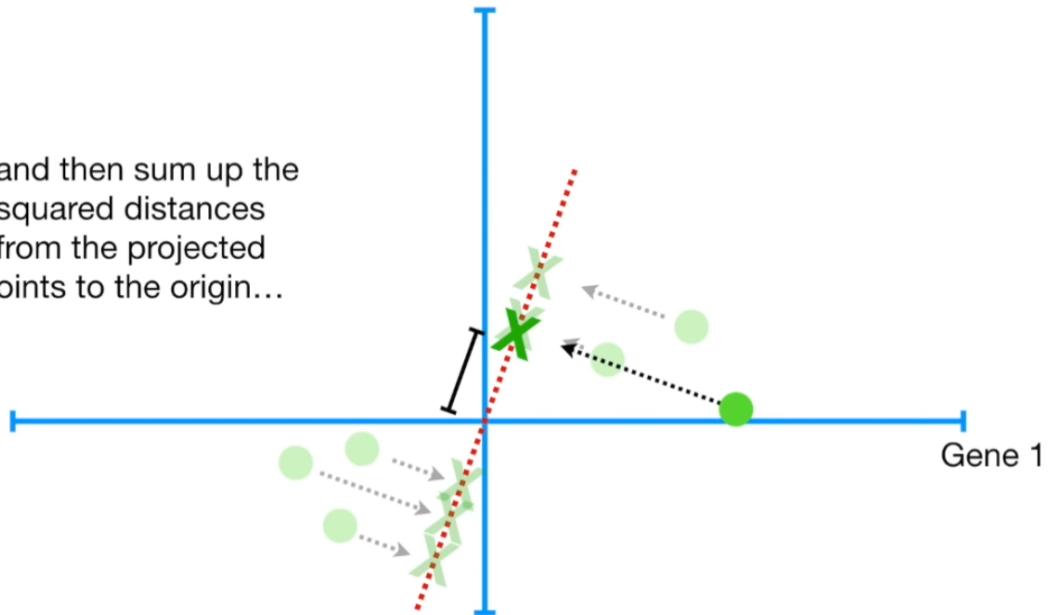
Gene 2

Now we rotate the line...



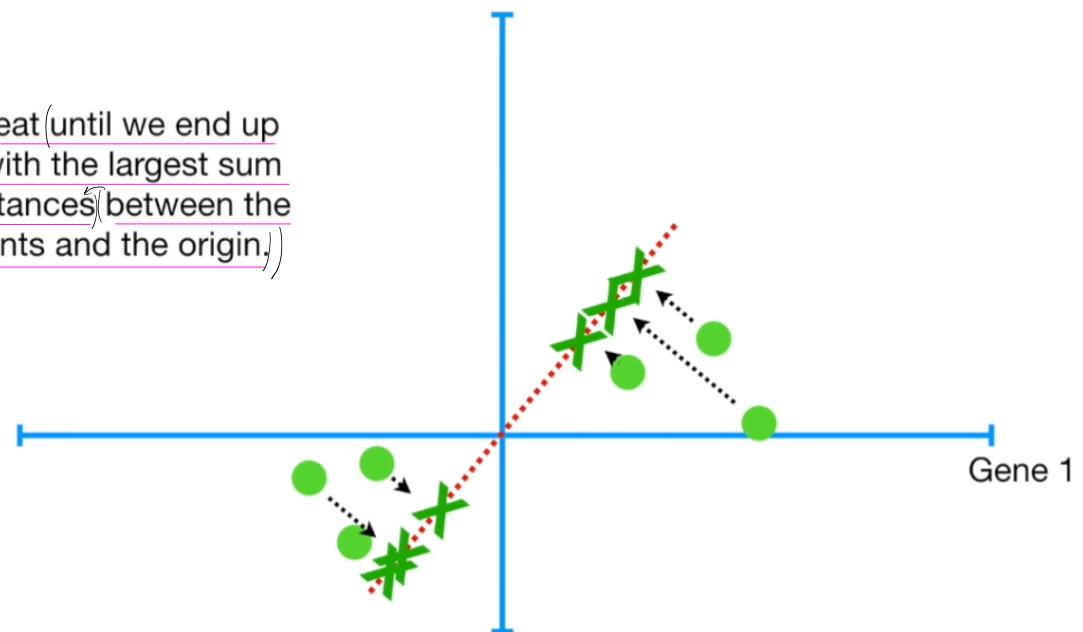
$$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2 = \text{sum of squared distances} = \text{SS}(distances)$$

...and then sum up the squared distances from the projected points to the origin...



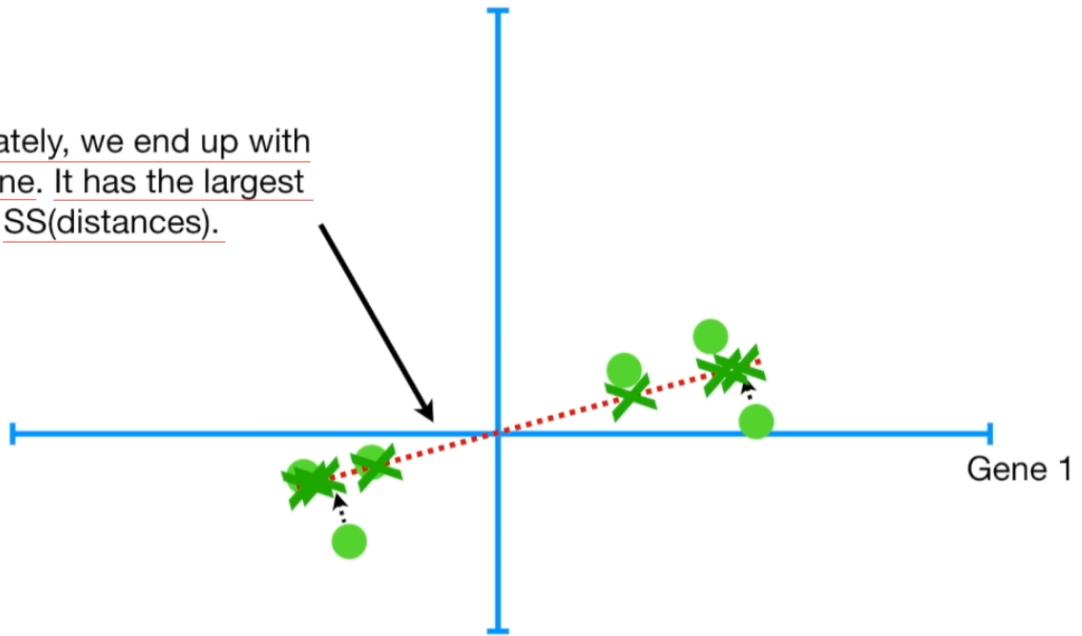
$$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2 = \text{sum of squared distances} = \text{SS}(distances)$$

~~...and we repeat until we end up with the line with the largest sum of squared distances between the projected points and the origin.)~~

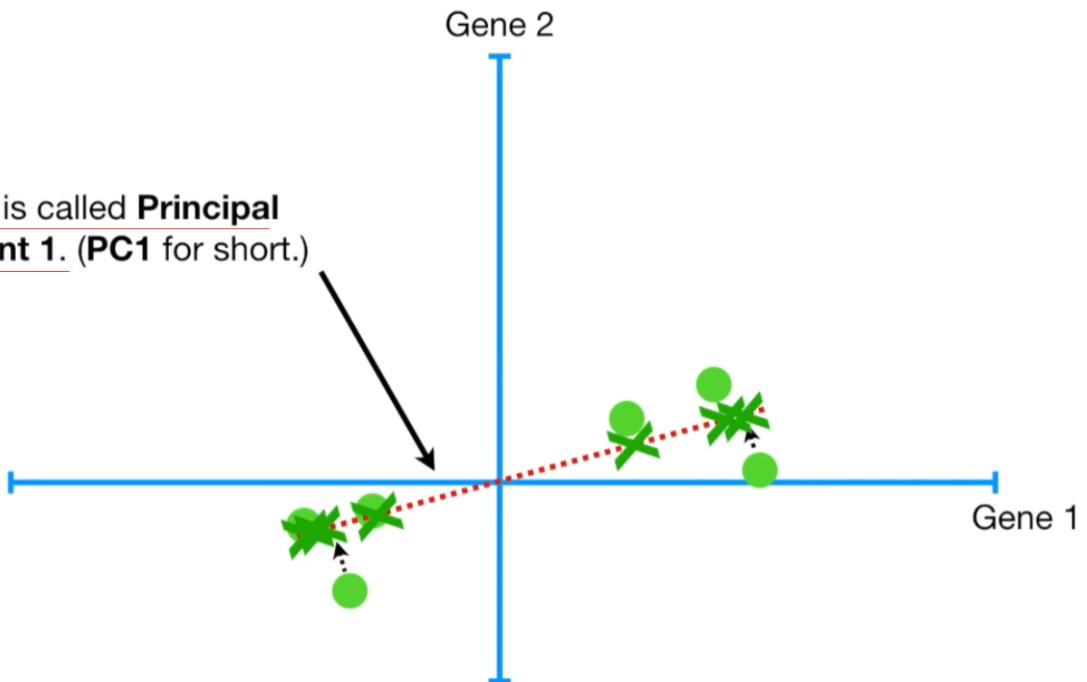


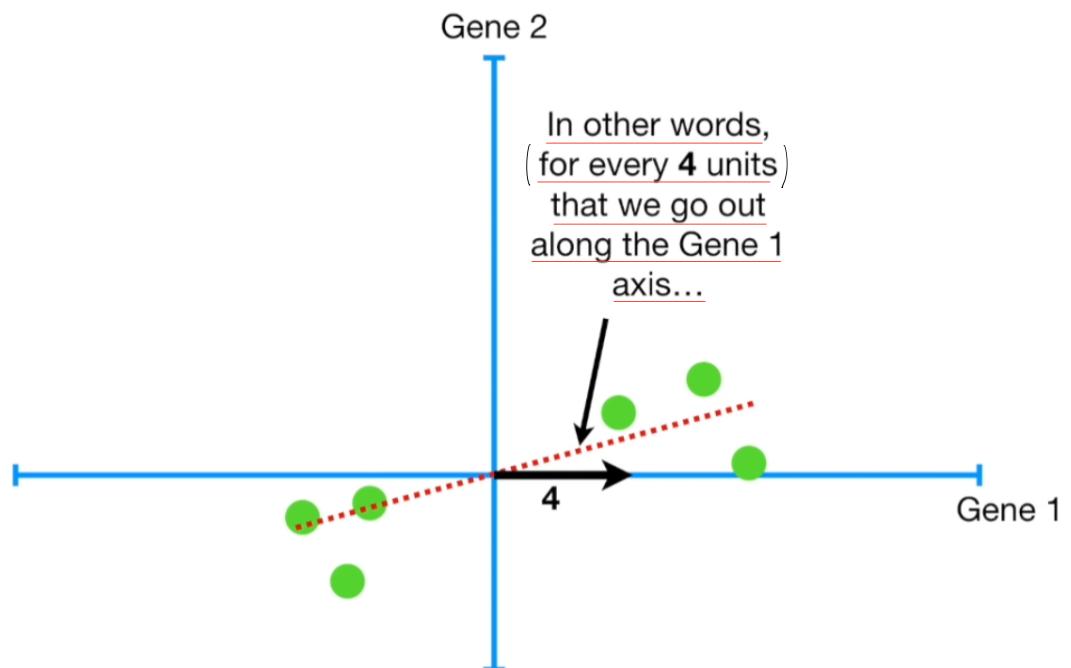
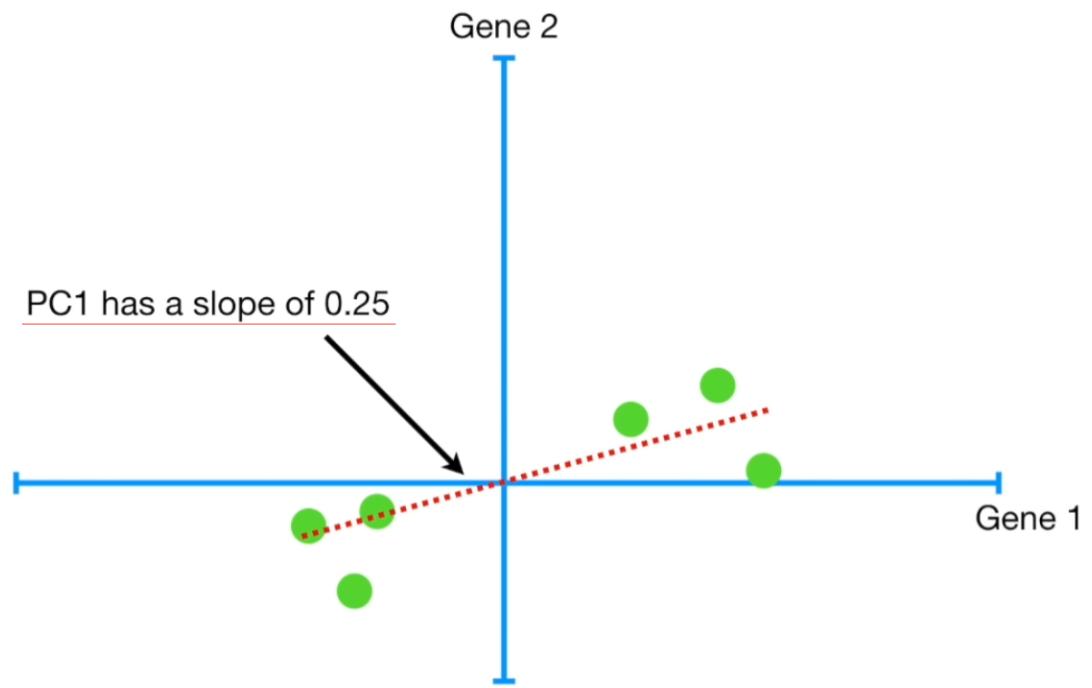
$$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2 = \text{sum of squared distances} = \text{SS}(distances)$$

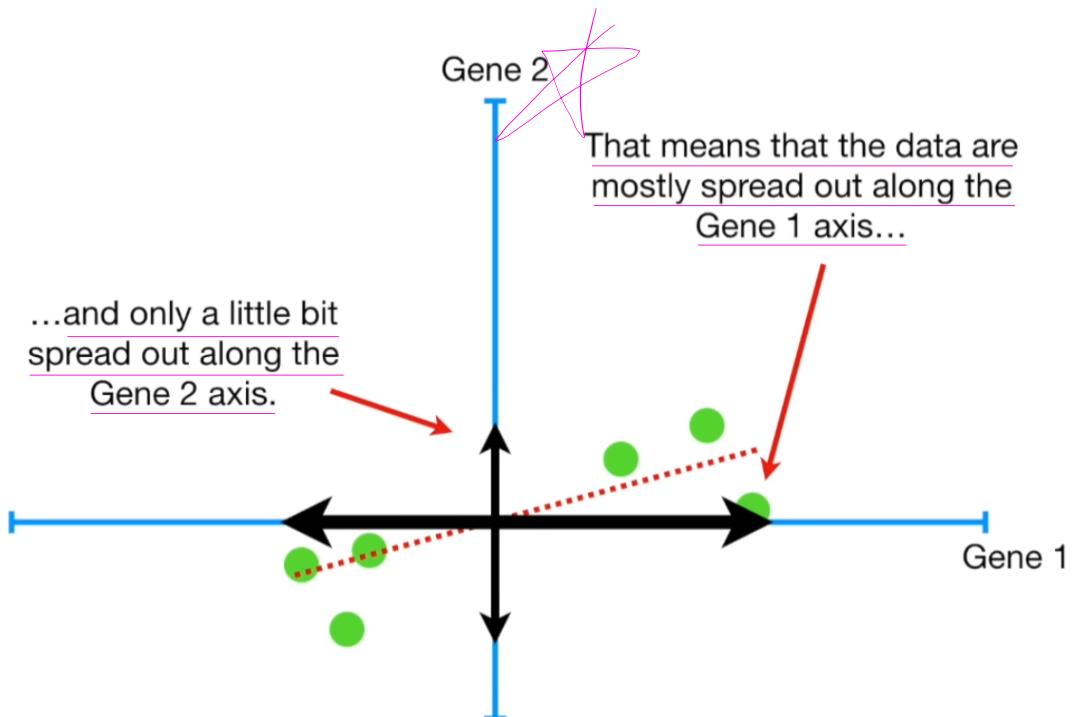
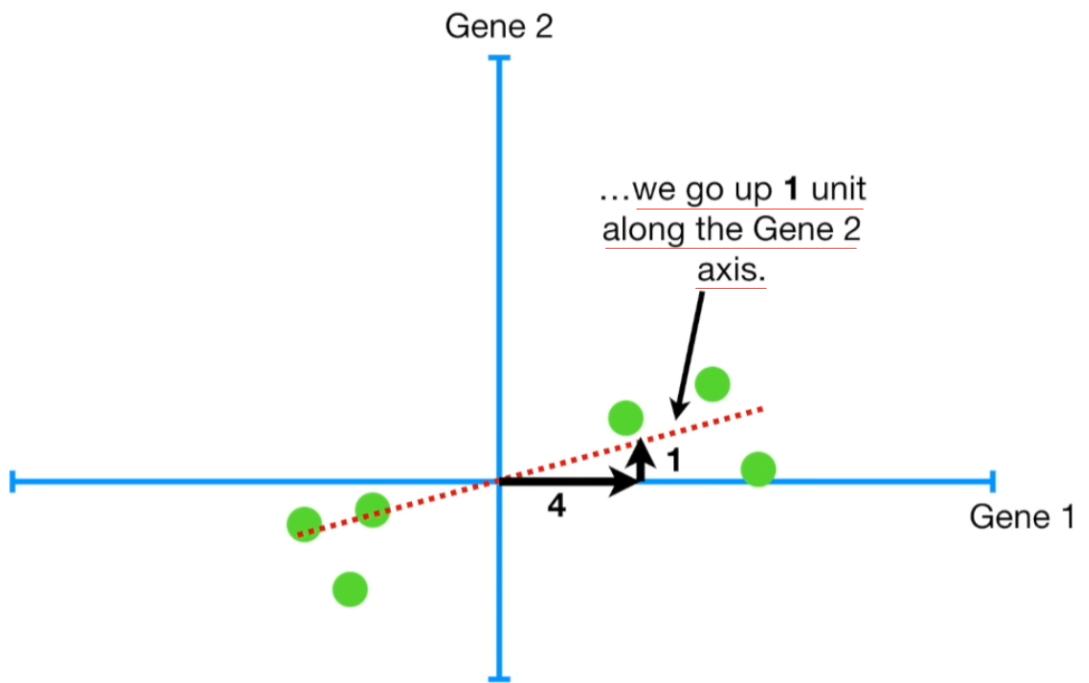
Ultimately, we end up with this line. It has the largest SS(distances).



This line is called **Principal Component 1**. (PC1 for short.)



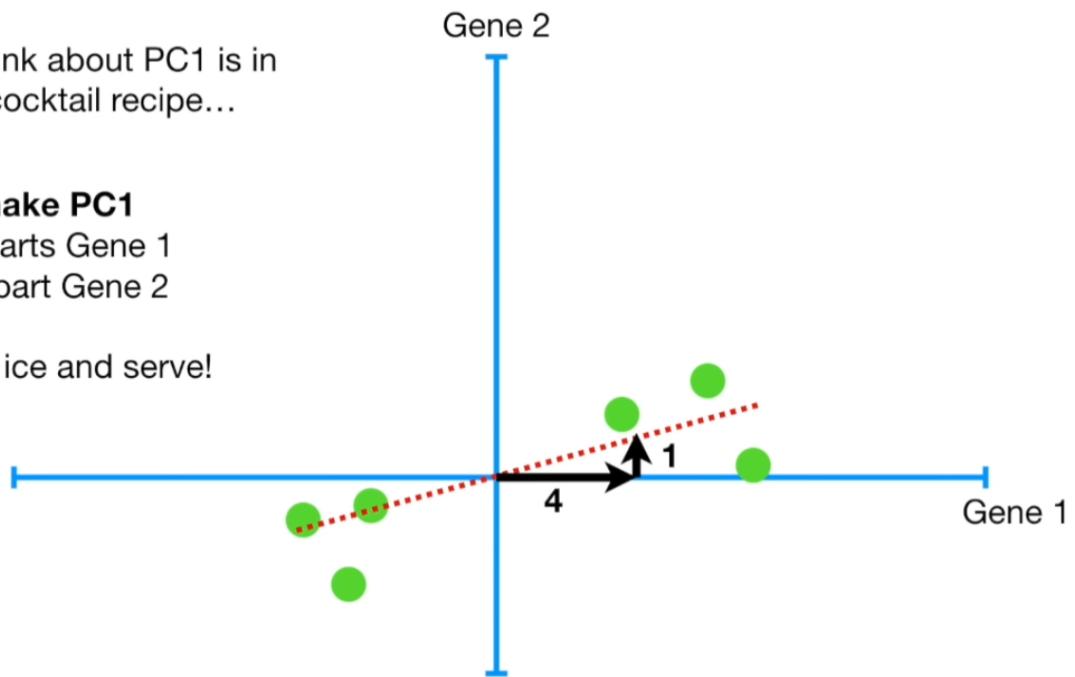




One way to think about PC1 is in terms of a cocktail recipe...

**To make PC1**  
Mix **4** parts Gene 1  
with **1** part Gene 2

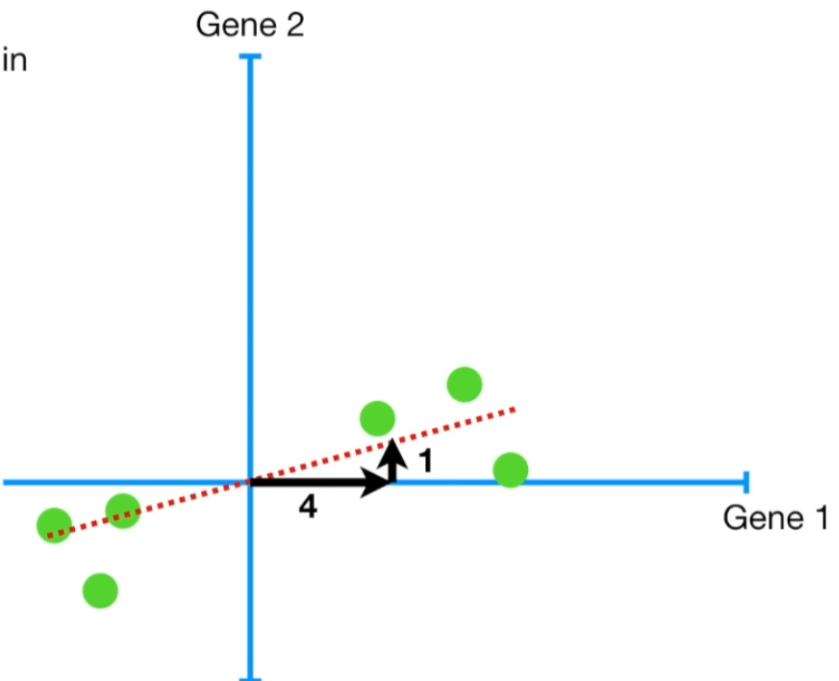
Pour over ice and serve!



One way to think about PC1 is in terms of a cocktail recipe...

**To make PC1**  
Mix **4** parts Gene 1  
with **1** part Gene 2

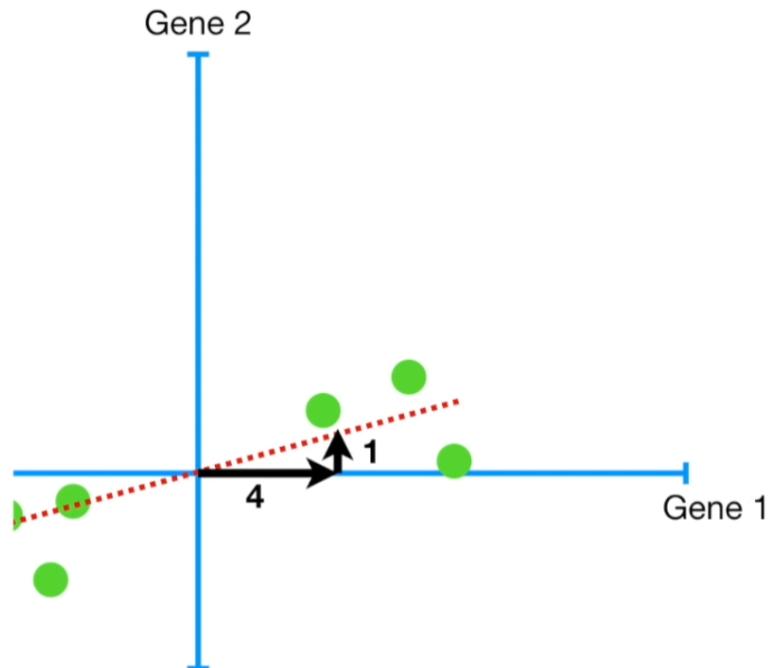
The ratio of Gene 1 to Gene 2 tells you that Gene 1 is more important (when it comes to describing how the data are spread out..)



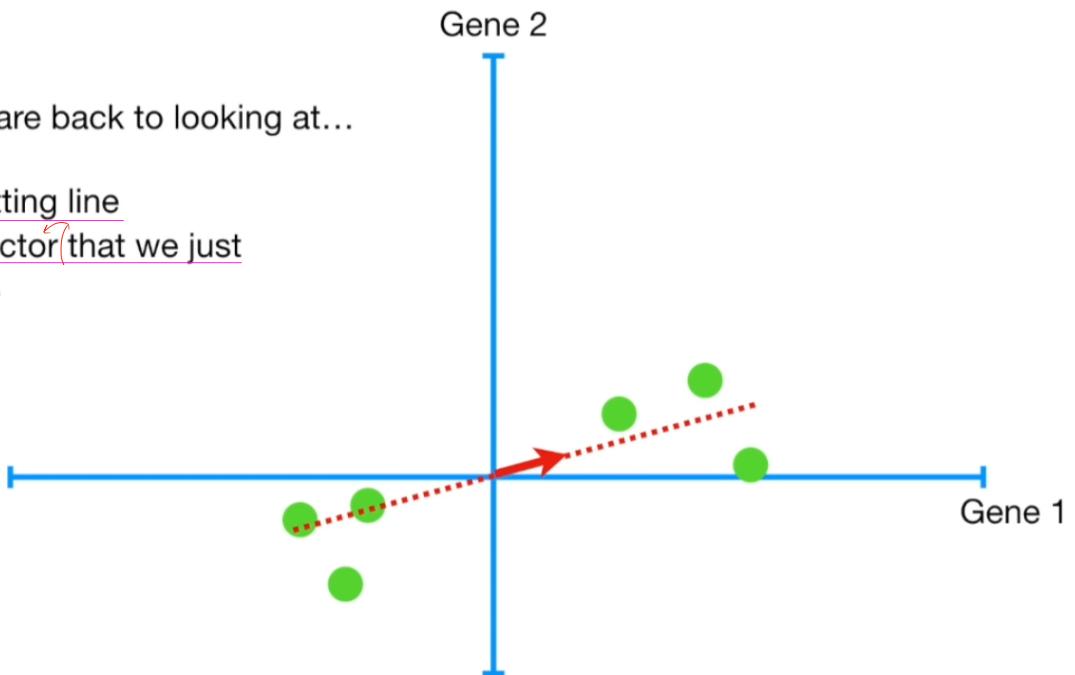
**To make PC1**  
Mix 4 parts Gene 1  
with 1 part Gene 2

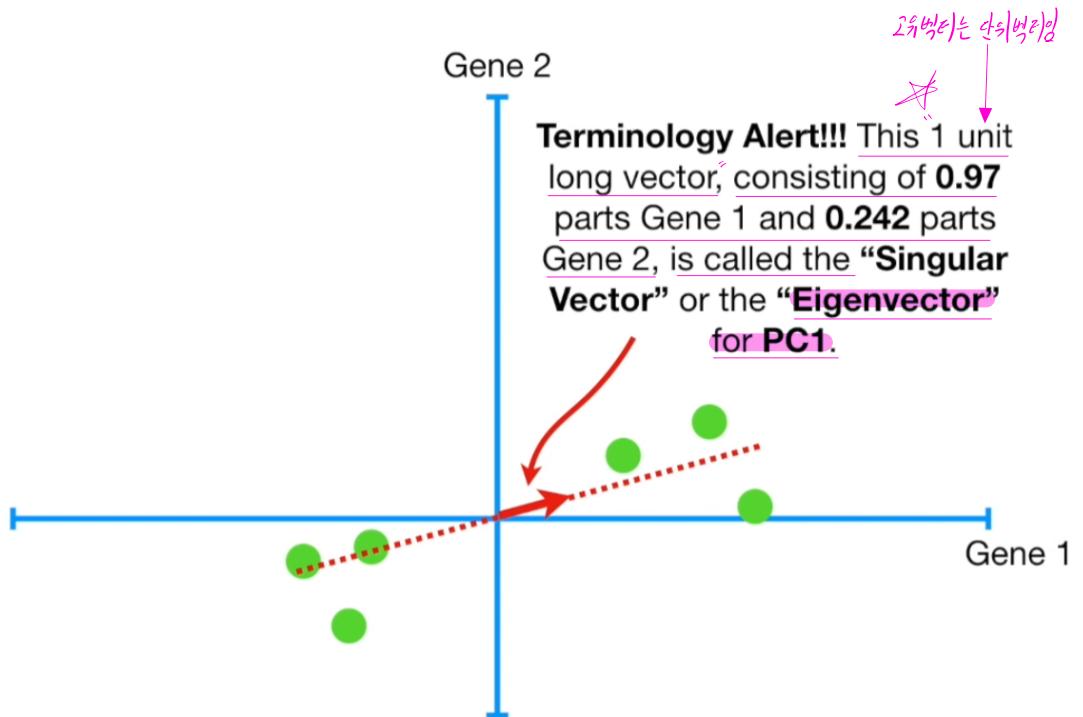
**Terminology Alert!!!!**  
Mathematicians call this cocktail recipe a “*linear combination*” of Genes 1 and 2.

[feature 벡터) 경우) 선형결합.]



So now we are back to looking at...  
 • The data  
 • The best fitting line  
 • The unit vector that we just calculated.)

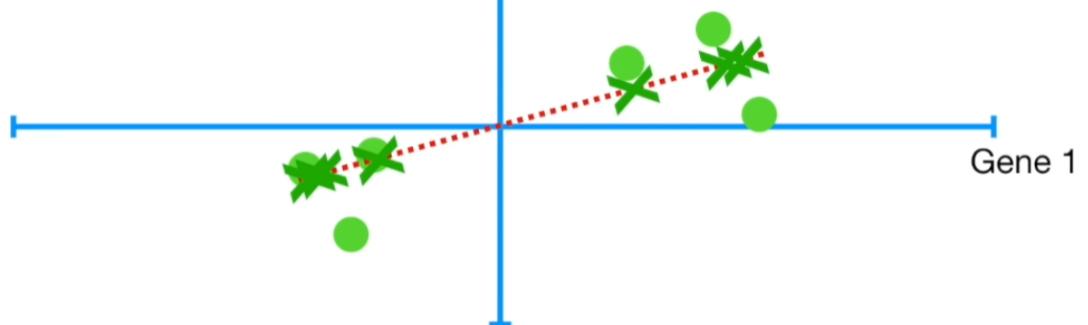




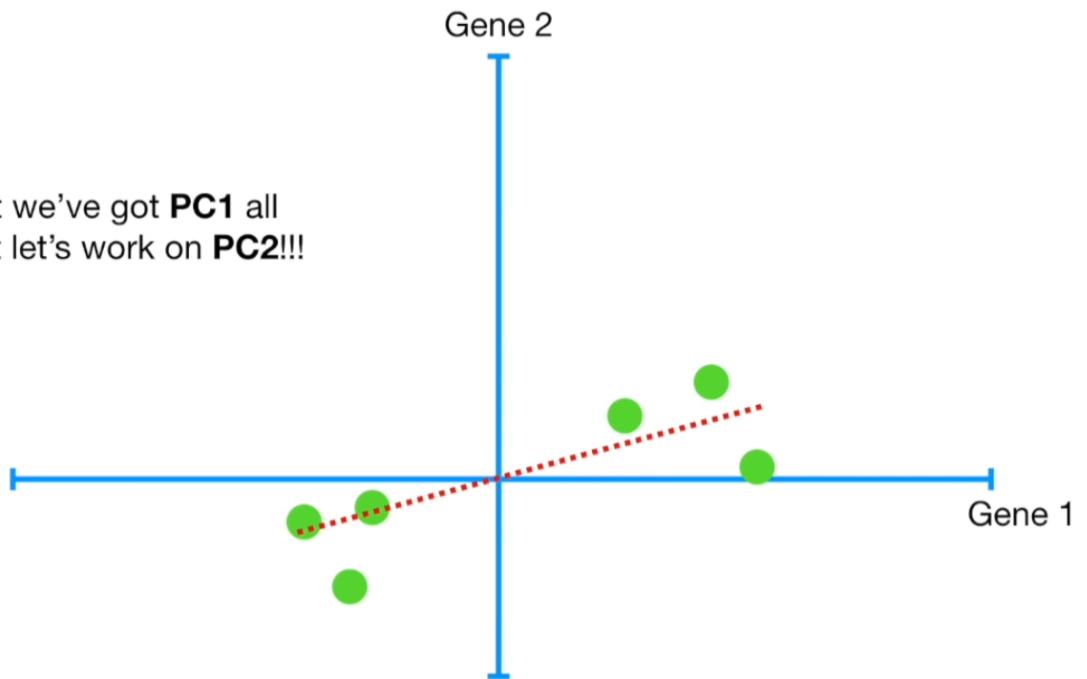
SS(distances for PC1) = Eigenvalue for PC1

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

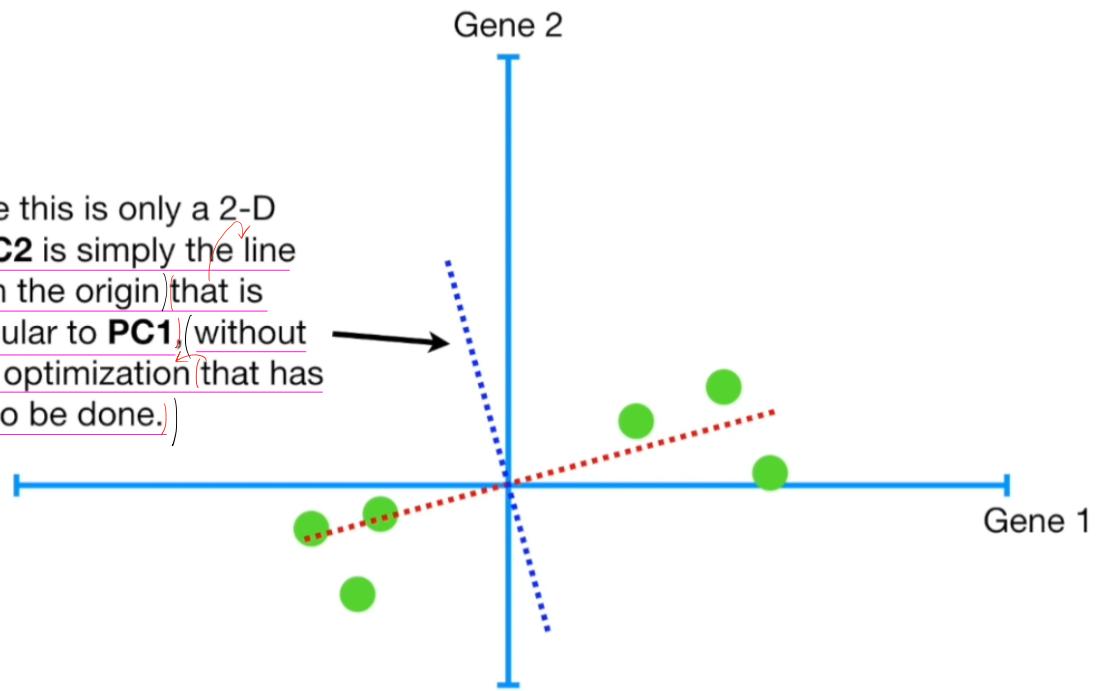
...and the square root of the **Eigenvalue for PC1** is called the **Singular Value for PC1**.



Now that we've got **PC1** all figured out let's work on **PC2!!!**

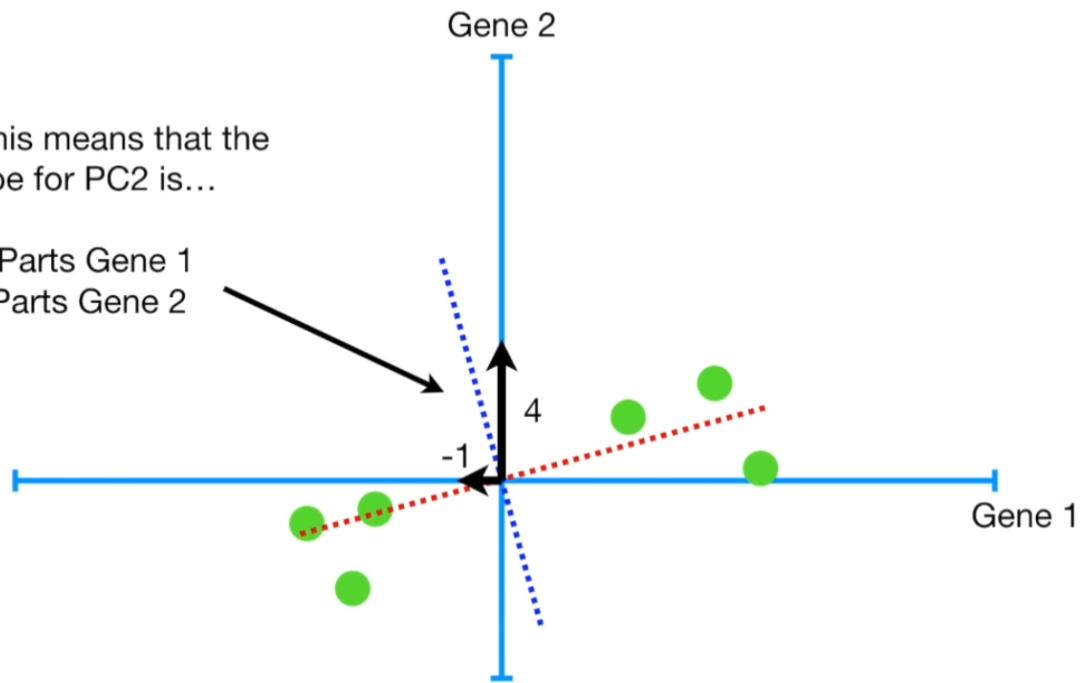


Because this is only a 2-D graph, **PC2** is simply the line (through the origin) that is perpendicular to **PC1** (without any further optimization that has to be done.)



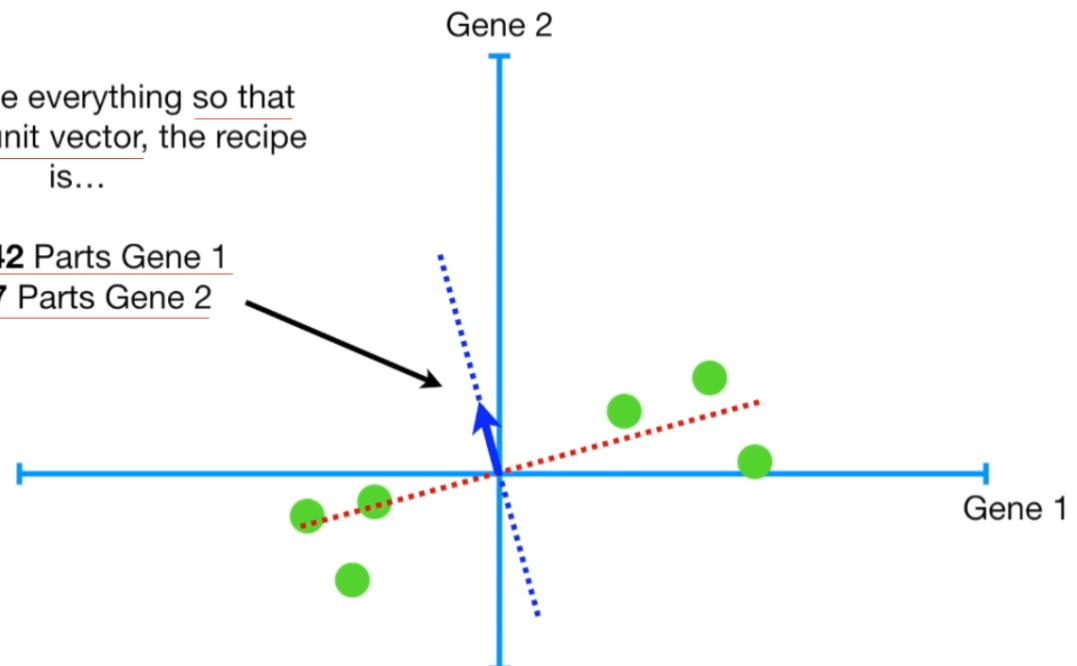
...and this means that the recipe for PC2 is...

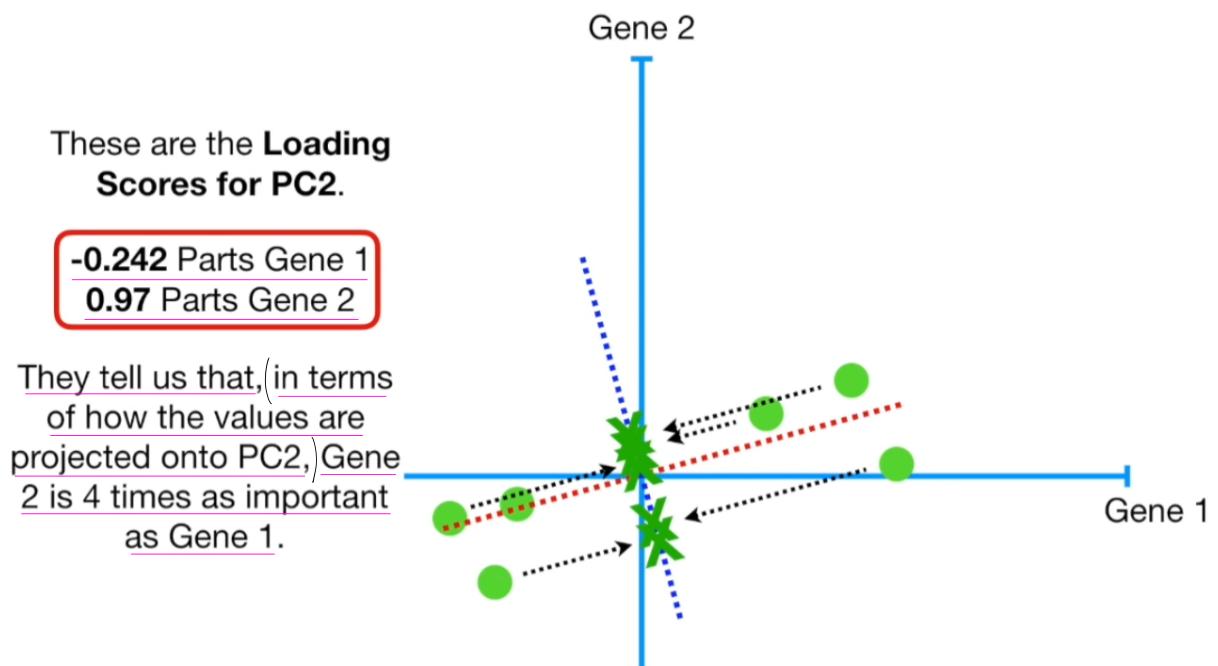
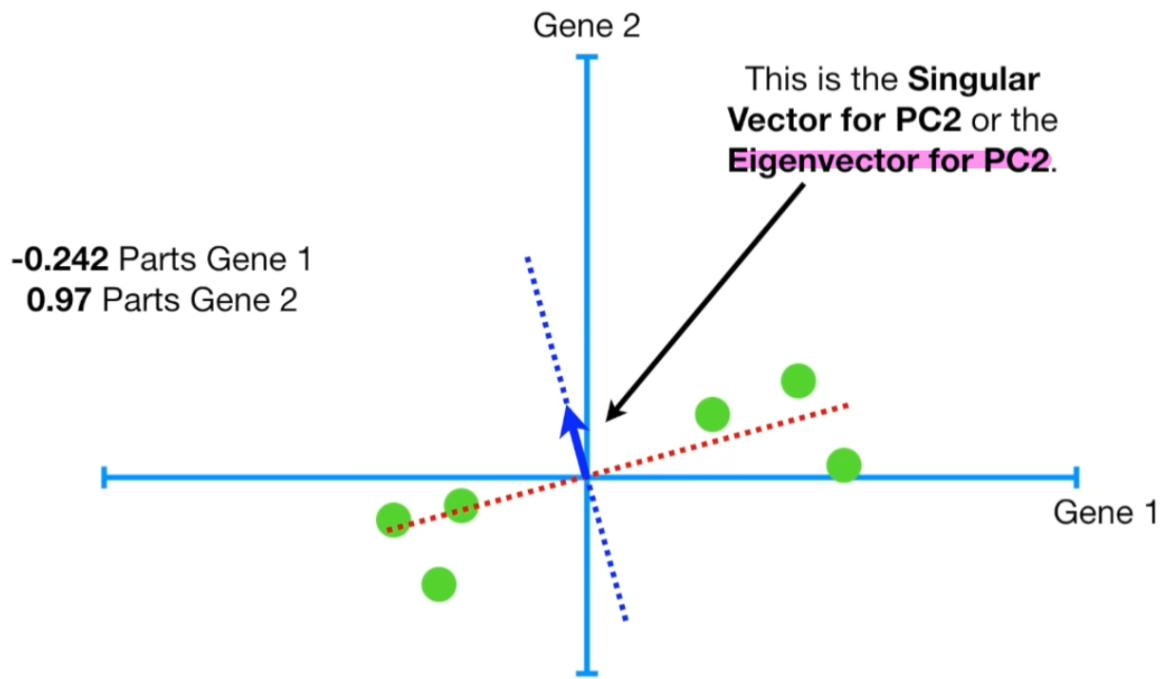
**-1** Parts Gene 1  
**4** Parts Gene 2



If we scale everything so that  
we get a unit vector, the recipe  
is...

**-0.242** Parts Gene 1  
**0.97** Parts Gene 2

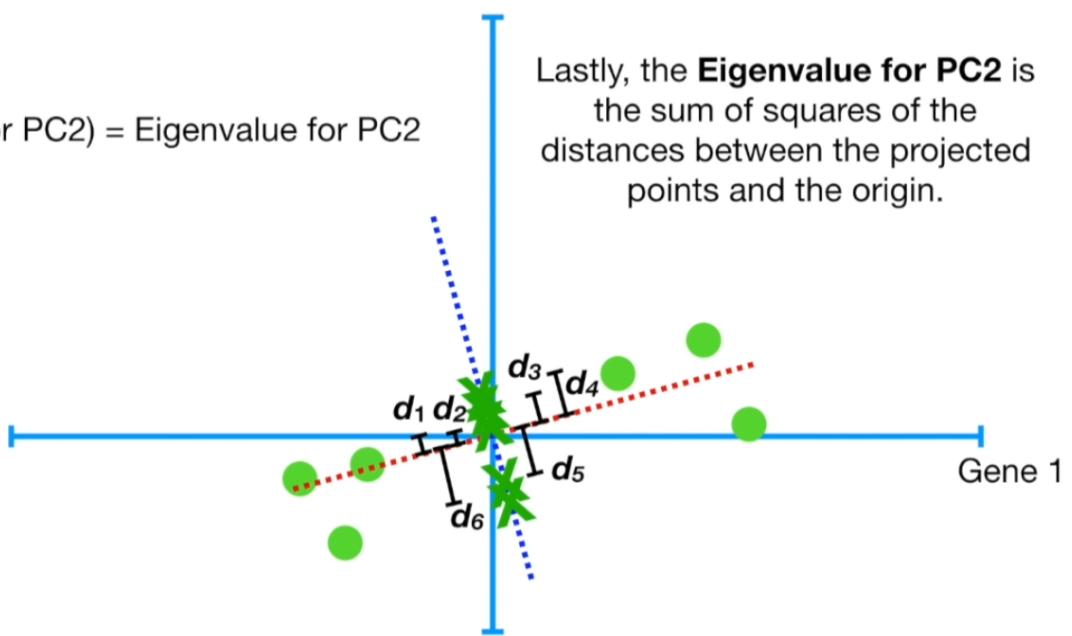




$$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2 = \text{sum of squared distances} = \text{SS}(distances)$$

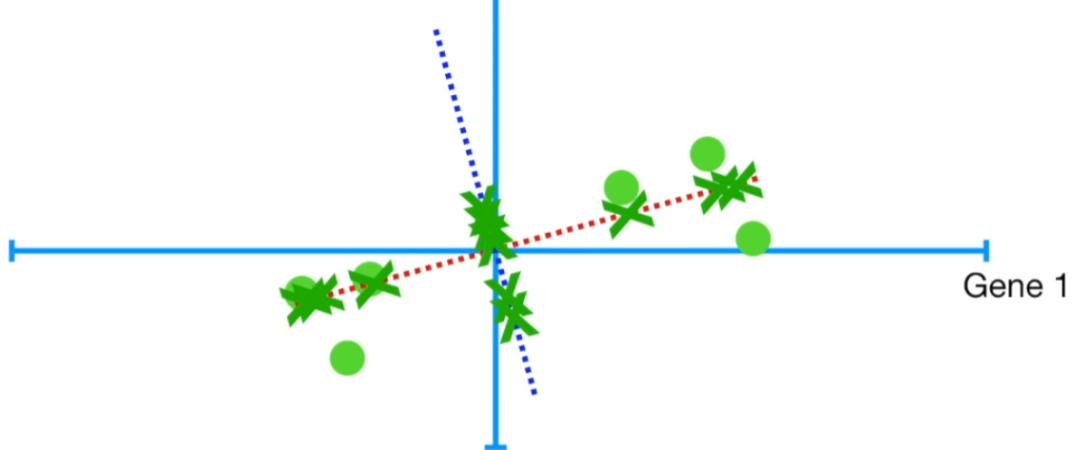
$\text{SS}(\text{distances for PC2}) = \text{Eigenvalue for PC2}$

Lastly, the **Eigenvalue for PC2** is the sum of squares of the distances between the projected points and the origin.

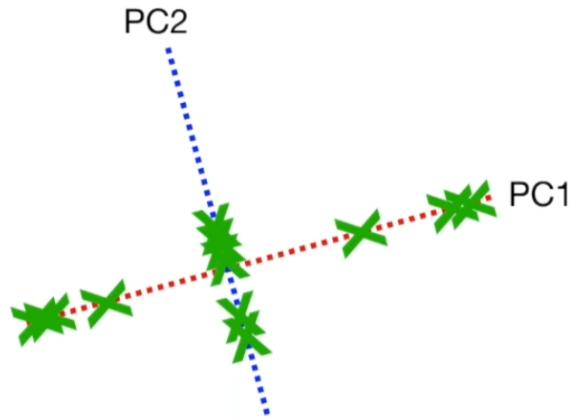


Hooray!!!

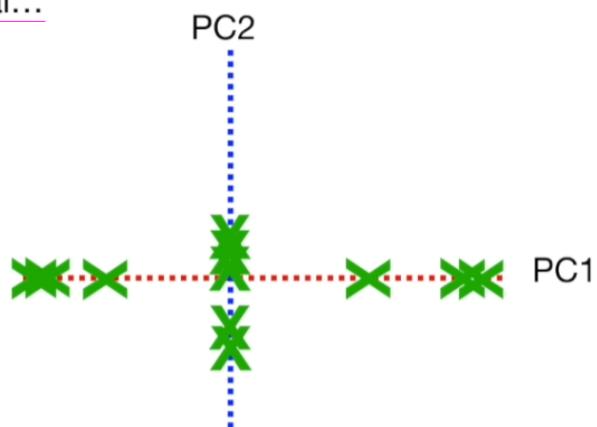
We've worked out PC1 and PC2!!!



To draw the final PCA plot...

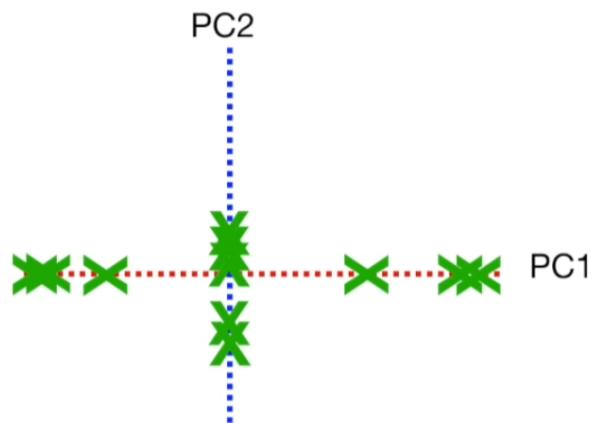


We simply rotate everything so  
that PC1 is horizontal...

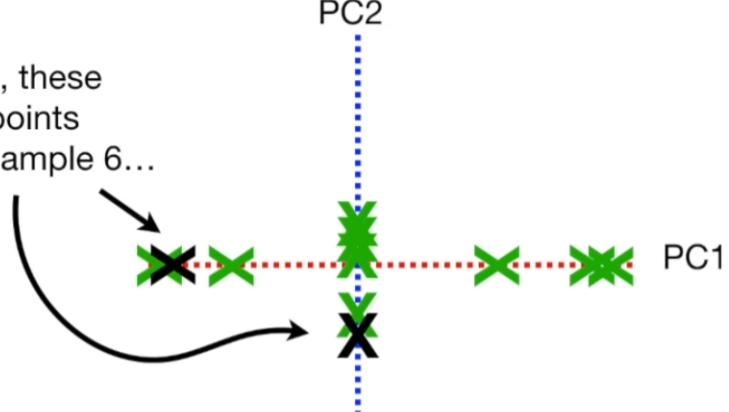


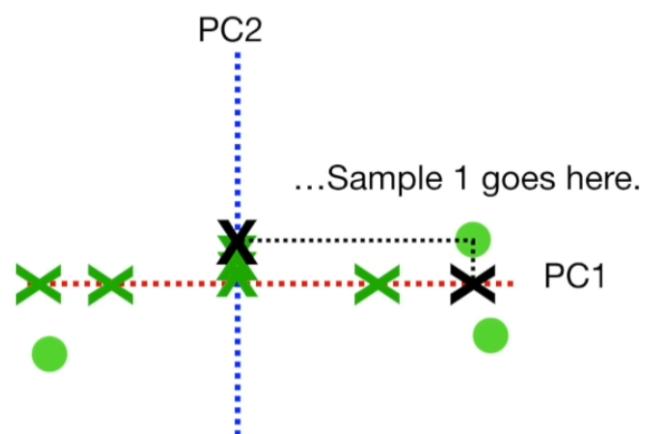
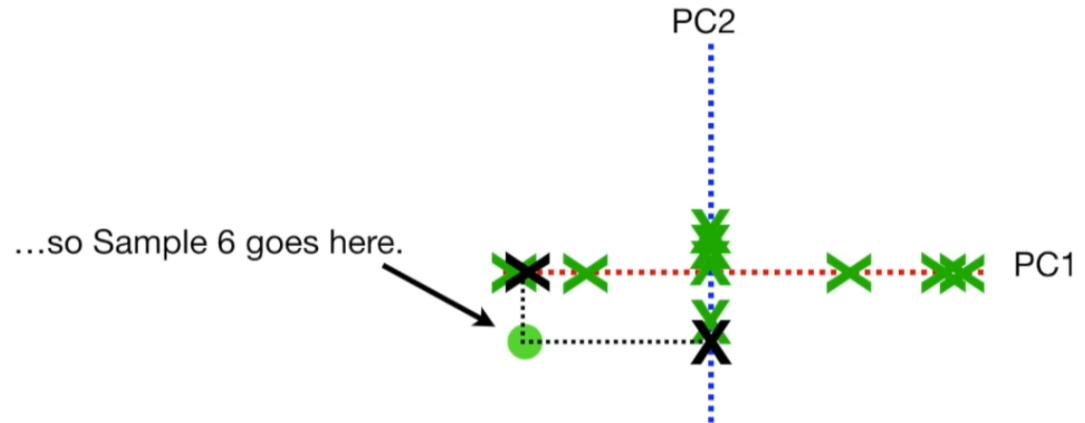
②

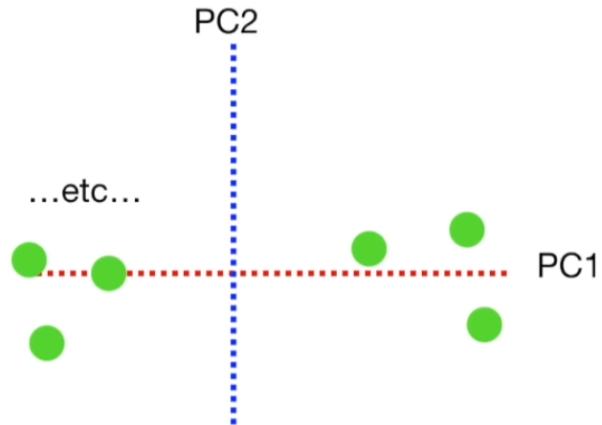
...then we use the projected points  
to find where the samples go in  
the PCA plot.



For example, these  
projected points  
correspond to Sample 6...

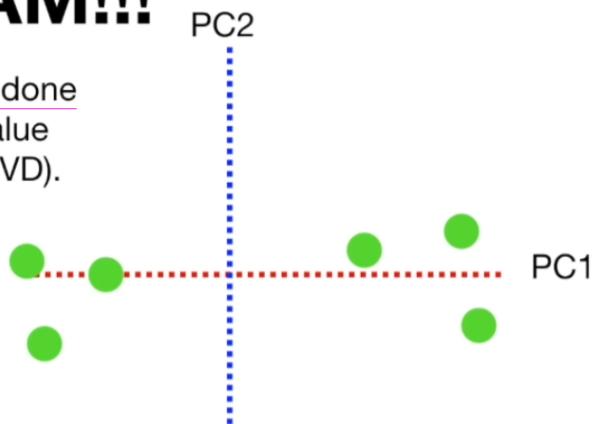






## Double BAM!!!

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

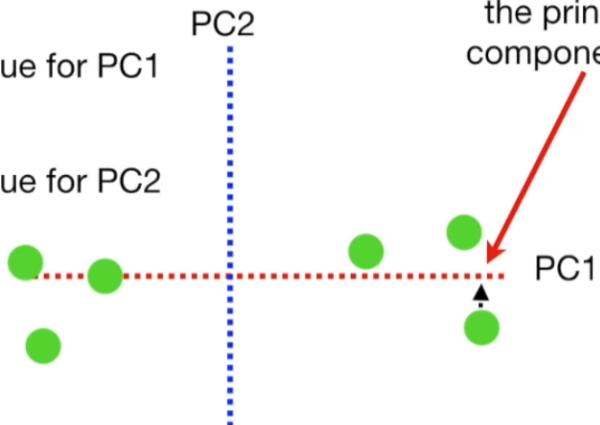


Remember the eigenvalues?

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

$$\text{SS(distances for PC2)} = \text{Eigenvalue for PC2}$$

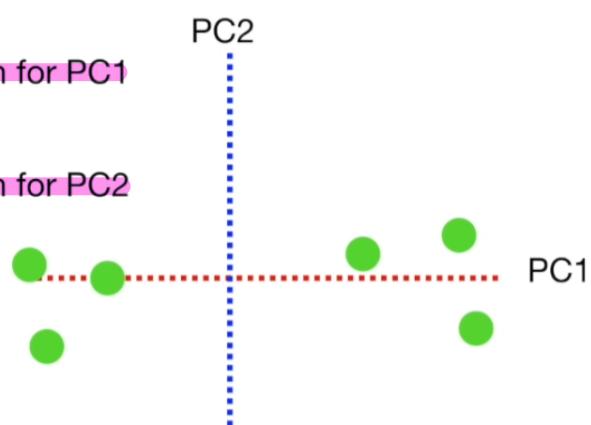
We got those (by projecting the data onto the principal components...)



~~We can convert them into variation (around the origin  $(0, 0)$ ) by dividing by the sample size minus 1 (i.e.  $n - 1$ ).~~

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

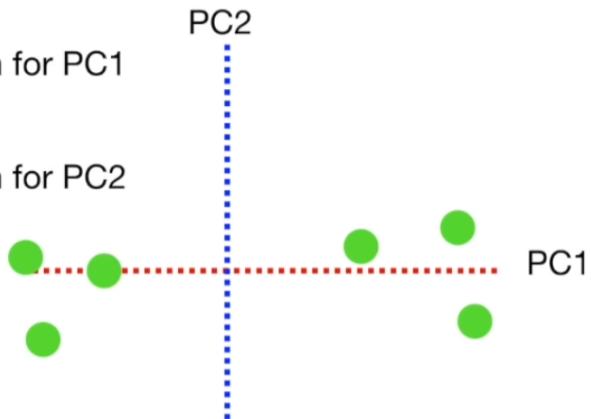
$$\frac{\text{SS(distances for PC2)}}{n - 1} = \text{Variation for PC2}$$



For the sake of the example, imagine that the Variation for **PC1 = 15**, and the variation for **PC2 = 3**.

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

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

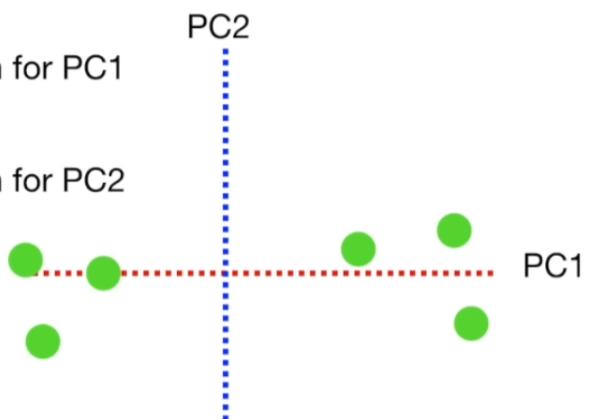


For the sake of the example, imagine that the Variation for **PC1 = 15**, and the variation for **PC2 = 3**.

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

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

*(That means that the total variation around both PCs is **15 + 3 = 18...**)*



For the sake of the example, imagine that the Variation for **PC1 = 15**, and the variation for **PC2 = 3**.

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

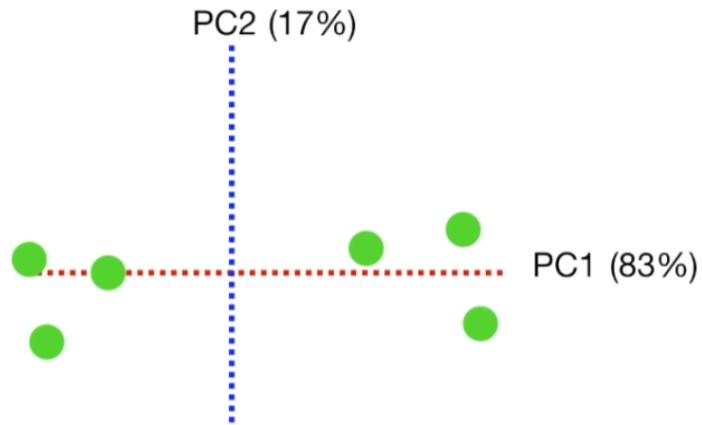
$$\frac{\text{SS(distances for PC2)}}{n - 1} = \text{Variation for PC2}$$

That means that the total variation around both PCs is  **$15 + 3 = 18$** ...

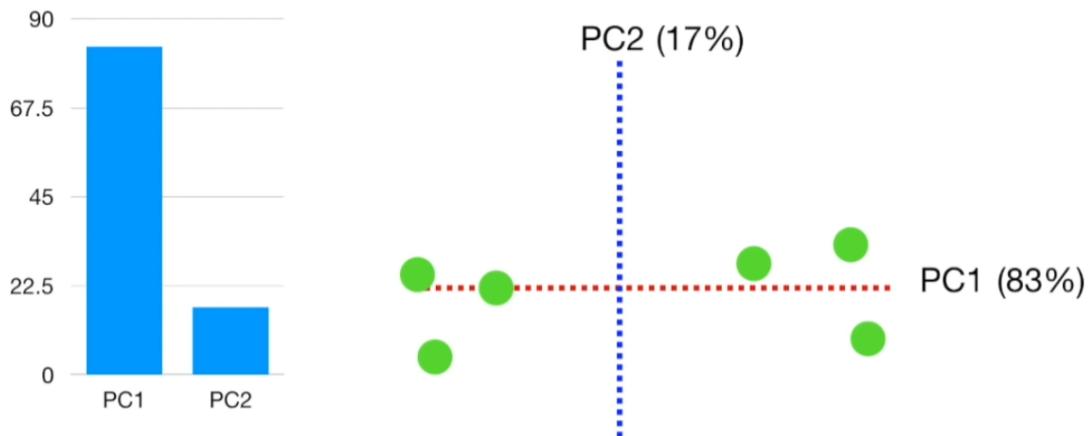
PC2 ...and that means ~~PC1~~ accounts for  **$15 / 18 = 0.83 = 83\%$**  of the total variation around the PCs.



PC2 accounts for  **$3 / 18 = 0.17 = 17\%$**  of the total variation around the PCs.



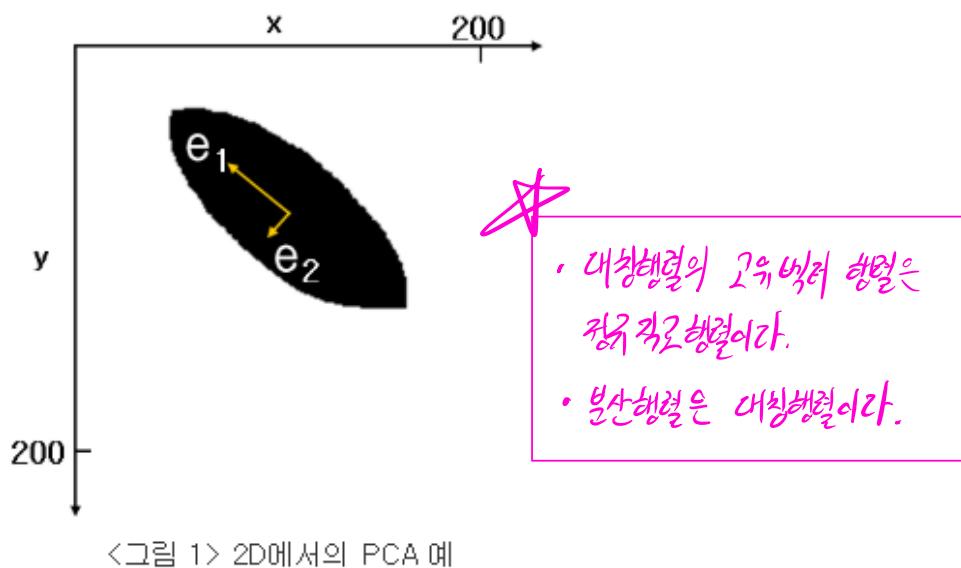
**TERMINOLOGY ALERT!!!!** A **Scree Plot** is a graphical representation of the percentages of variation that each PC accounts for.)



## 1. PCA(Principal Component Analysis)란?

PC: 특징(아이디어)의 분산·행렬에 대한 고유벡터.

PCA는 분포된 데이터들의 주성분(Principal Component)를 찾아주는 방법이다. 좀 더 구체적으로 보면 아래 그림과 같이 2차원 좌표평면에 n개의 점 데이터  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ 들이 타원형으로 분포되어 있을 때



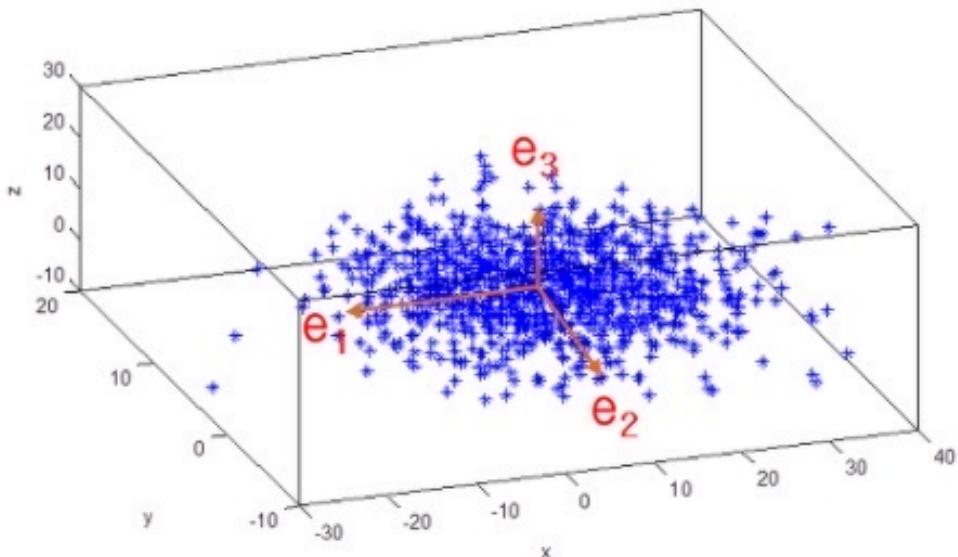
<그림 1> 2D에서의 PCA 예

“이 데이터들의 분포 특성을 2개의 벡터로 가장 잘 설명할 수 있는 방법은 무엇일까? 그 건 바로, 그림에서와 같이  $e_1, e_2$  두 개의 벡터로 데이터 분포를 설명하는 것이다.  $e_1$ 의 방향과 크기, 그리고  $e_2$ 의 방향과 크기를 알면 이 데이터 분포가 어떤 형태인지를 가장 단순하면서도 효과적으로 파악할 수 있다.”

PCA는 (데이터 하나 하나에 대한 성분을 분석하는 것이 아니라) 여러 데이터들이 모여 하나의 분포를 이루 때 이 분포의 주 성분을 분석해 주는 방법이다.

여기서 주성분이라 함은 그 방향으로 데이터들의 분산이 가장 큰 방향벡터를 의미한다. <그림 1>에서  $e_1$  방향을 따라 데이터들의 분산(흩어진 정도)이 가장 크다. 그리고  $e_1$ 에 수직이면서 그 다음으로 데이터들의 분산이 가장 큰 방향은  $e_2$ 이다.

PCA는 2차원 데이터 집합에 대해 PCA를 수행하면 2개의 서로 수직인 주성분 벡터를 반환하고, 3차원 점들에 대해 PCA를 수행하면 3개의 서로 수직인 주성분 벡터들을 반환한다. 예를 들어 3차원 데이터의 경우는 아래 그림과 같이 3개의 서로 수직인 주성분 벡터를 찾아준다.



〈그림 2〉 3D에서의 PCA

### (3) 다른 분석을 위한 사전분석 역할

- 회귀분석 등에서 독립변수간 다중공선성이 존재할 경우 상관도가 높은 변수들을 하나의 주성분 혹은 요인으로 축소하여 회귀분석 모형 개발에 활용함
- 주성분 또는 요인분석을 통해 차원을 축소한 후에 군집분석을 수행하면 군집화 결과, 연산속도 개선됨
- 기계에서 나오는 다수의 센서데이터를 주성분분석이나 요인분석을 하여 차원을 축소한 후에 시계열로 분포나 추세의 변화를 분석하면 기계의 고장(fatal failure) 징후를 사전에 파악하는데 활용

*선행상관지수*

## PCA의 원리

PCA는 어떻게 동작할까?

2차원을 1차원으로 줄이는 예를 보면서 이해해보자.

아래와 같이 2차원 공간에  $x_1$ ,  $x_2$  2개의 feature가 축으로 설정되어 있다. 그리고 7개의 점이 존재한다.

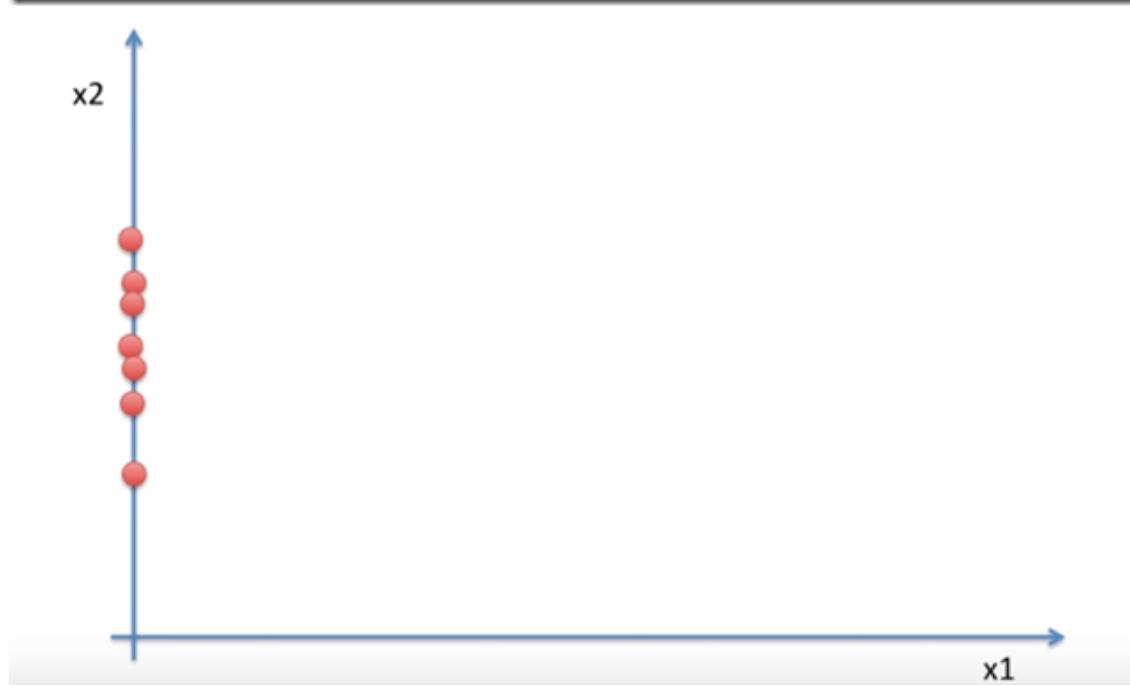
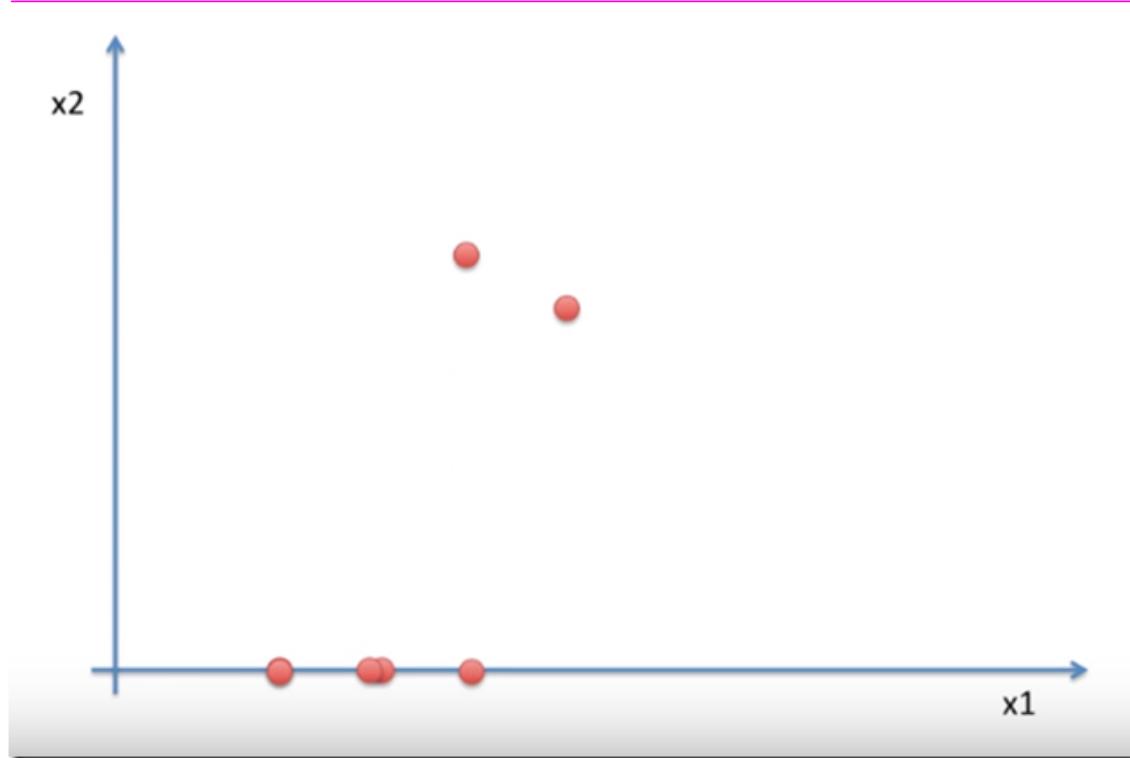


어떻게 2차원을 1차원으로 줄일 수 있을까?

가장 쉬운방법은 하나의 축을 선택하여, 점을 몰아넣는 방법이다.

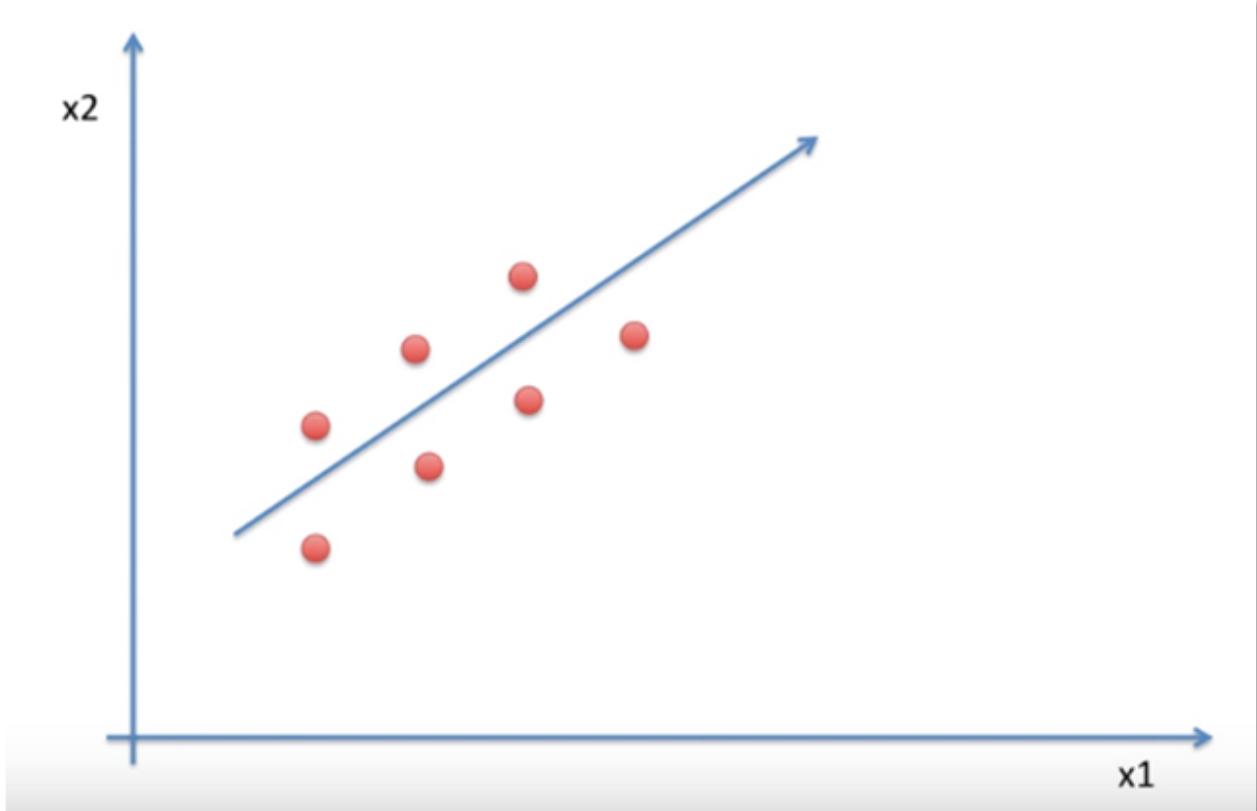


이러한 방식을 사용하면 몇개의 점은 서로 겹치게 되는 문제점이 발생한다. 이 경우, 정보의 유실이 있을 수 있다.

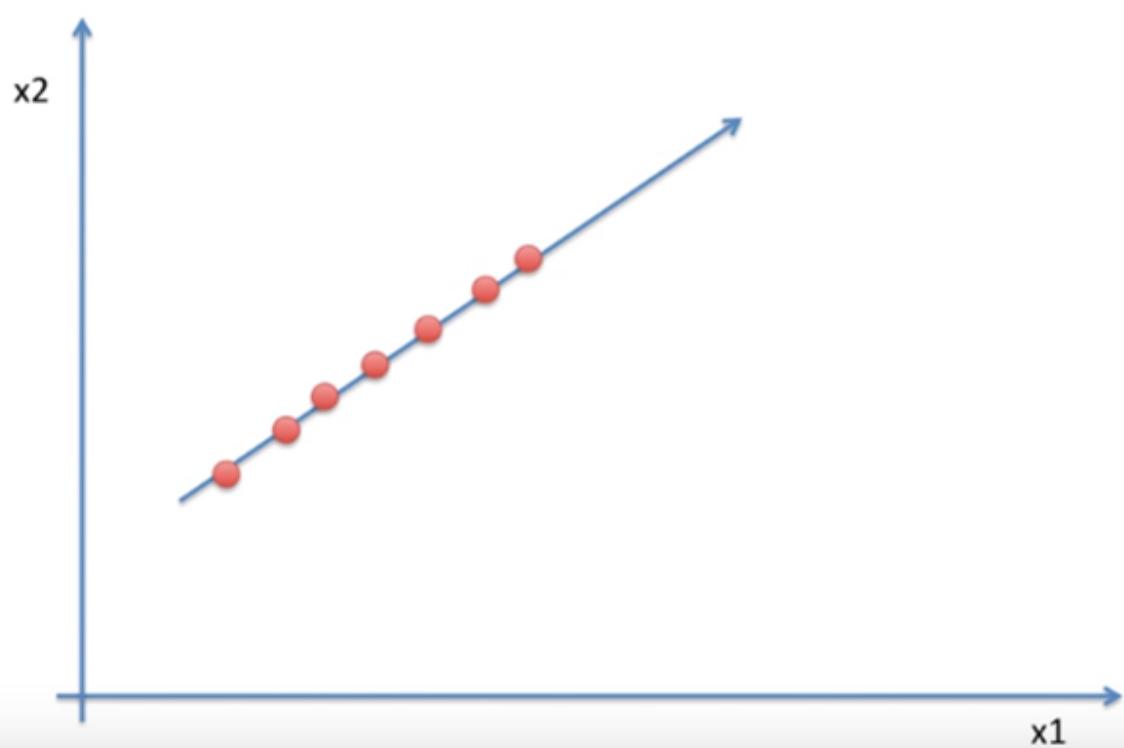
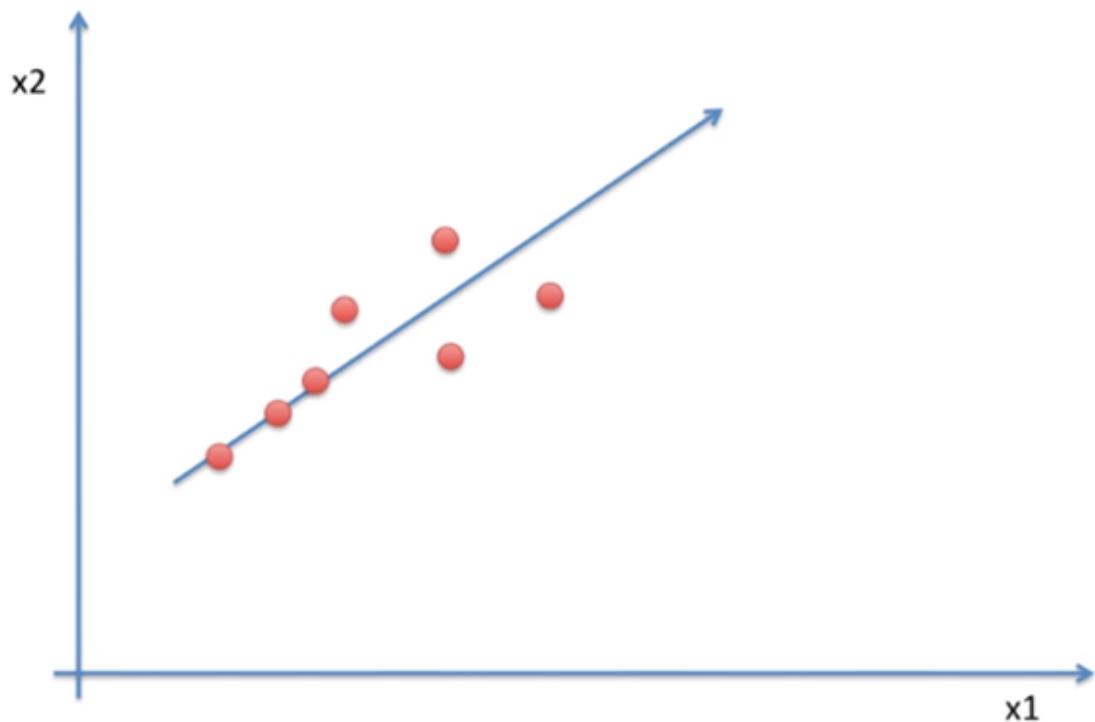


정보의 유실을 막으면서 차원을 줄이는 방법은 무엇일까?

과학적인 방법으로서, 분산이 가장 넓은 지역을 찾는 것이다.



분산이 가장 넓은 곳을 직선으로 표시하였다. 여기로 점들을 옮기게 된다면  
각 점들이 퍼져있는 정도를 지켜줄 수 있게 된다. 그 결과 점들은 겹치지 않게 된다.

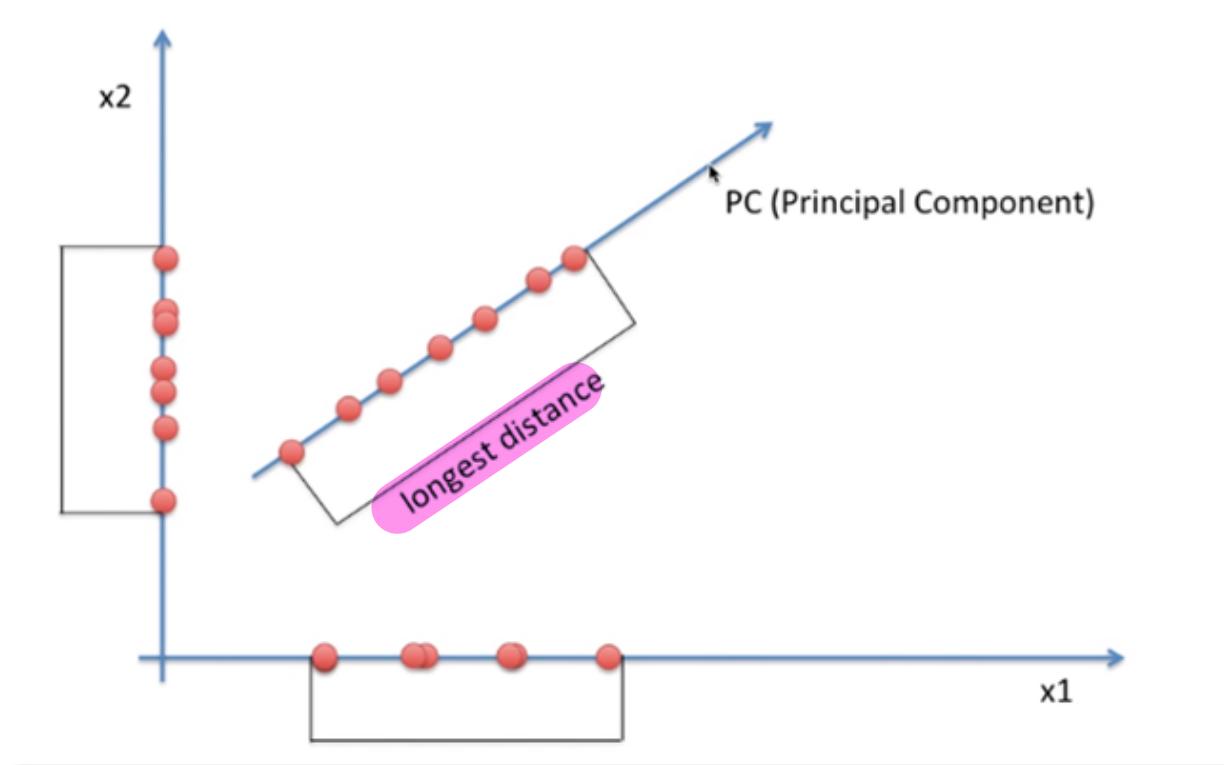


이게 바로 PCA 알고리즘이다.

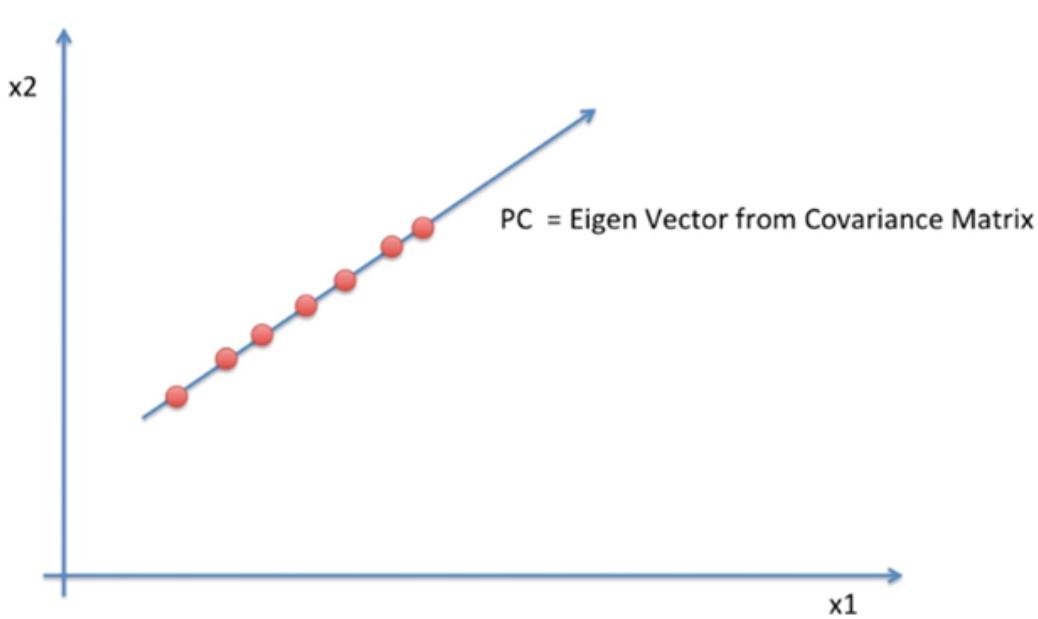
위에서 찾은 분산이 가장 넓은 지역을 PC(Princal Component)라 한다.

이 PC를 긋는 방법은 각 점들이 겹치지 않을 수 있게 살려서 긋는다.

이 때, x축으로 모은 거리나 y축으로 모은 거리보다 긴 것을 확인할 수 있다.



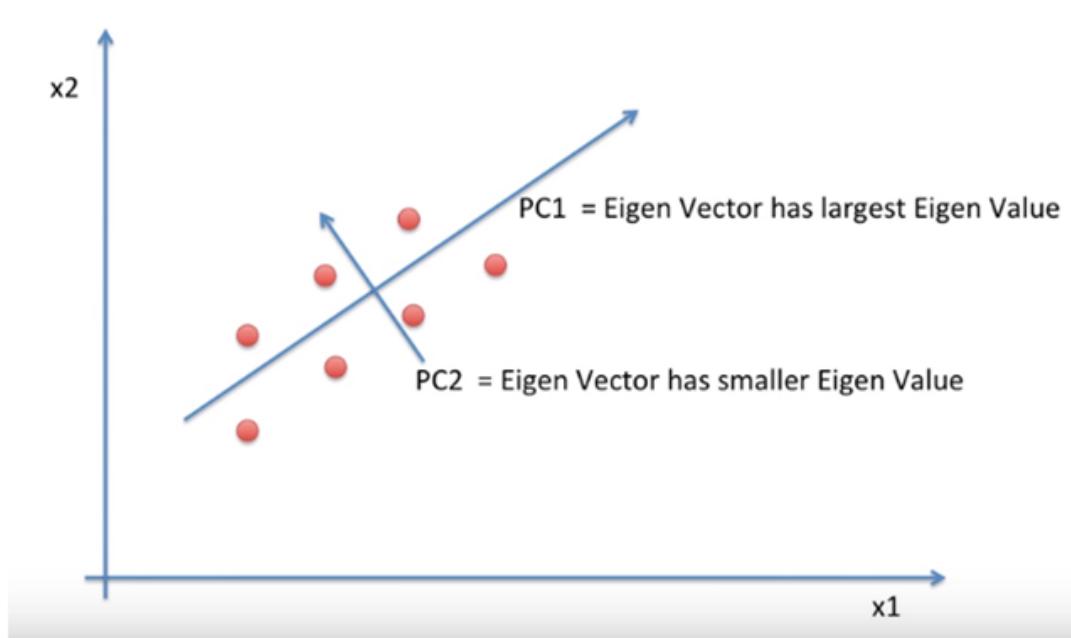
## 머신러닝에서 수학적으로 구하는 PC



수학적으로 PC를 구하는 방법은, 각 점들이 가지고 있는 feature( $x_1, x_2$ )들의 Covariance Matrix에 있는 Eigen Vector이다.

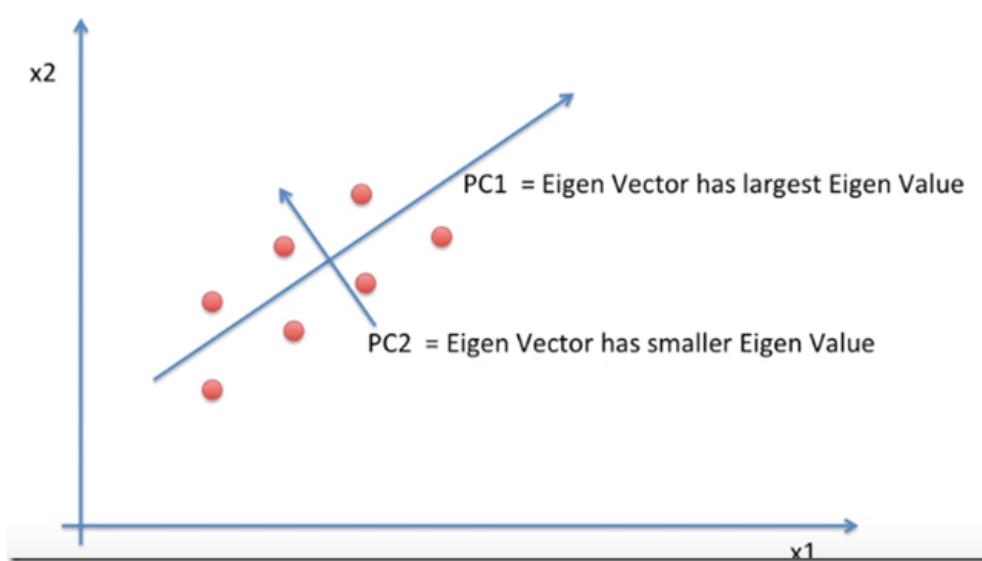
EigenVector(PC)는 2차원에서는 2개가 존재한다.

4차원이라면 4개가 존재할 것이다.

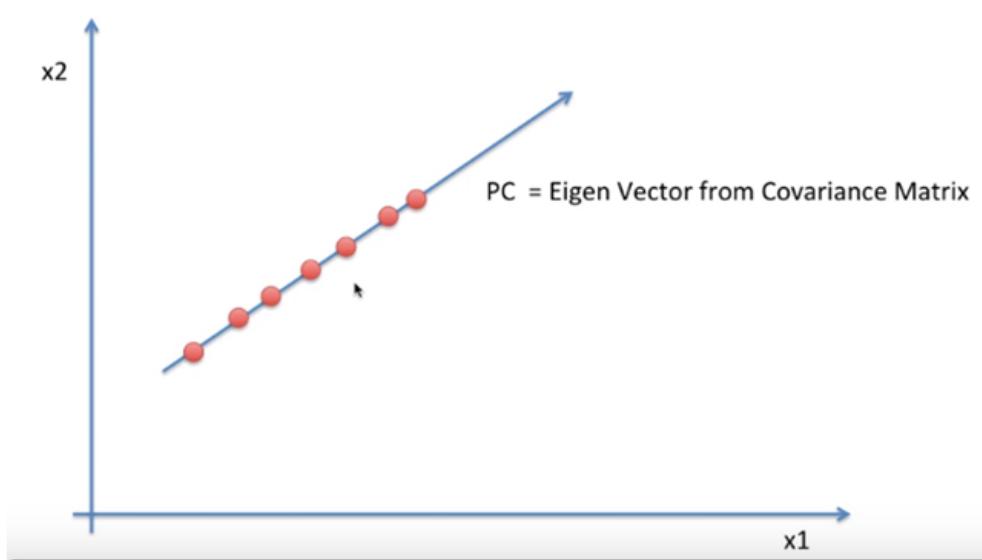


여러개의 EigenVector 중 어느 것을 선택해야 할까?

가장 넓게 퍼지게 만드는 EigenVector를 찾아야 한다. 즉, 분산이 큰 것을 의미하는 **Eigen value**가 높은 값을 찾아야 한다.

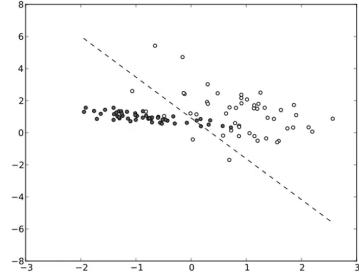


우리는 Covariance Matrix 중 Eigen value가 높은 값을 찾아 EigenVector를 찾고, 그것을 기준으로 데이터들을 옮겨 차원을 축소한다.



## PCA 란?

- Principal Component Analysis
- 여러 고차원 데이터에서 패턴을 찾는 도구 중 하나
  - ~~상관관계가 있는 변수들을 선형결합하여 변수를 축약하는 기법~~
- 변수들 간에 내재하는 상관관계, 연관성을 이용해 소수의 주성분으로 차원 축소



## PCA 예제

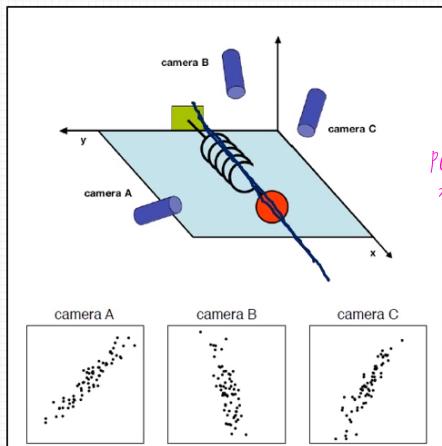
- PCA의 순서
    - ① 전체 데이터의 평균을 구한다.
    - ② 전체 데이터에 평균을 뺀다.
    - ③ 공분산 행렬을 만든다. →  $X^T X$ 로 공분산 행렬을 구함.
      - 공분산 행렬은 모든 요소의 공통된 부분을 뺀 개별적 특징이 포함된 데이터
    - ④ 공분산 행렬을 이용해 Eigen value 와 Eigen vector를 구한다.
- ☞ 각 변수의 평균을 구한다.
- ☞ 변수 내 들어있는 원소들을 해당 변수의 평균 값으로 뺄까.
- ☞ ②까지 개선 행렬( $X$ )로 공분산 행렬을 구함
- ☞ 분산과 편차의 관계
- ☞ 아래 본문의 원 방향
- ☞ 각 eigen vector들은 서로 orthogonal하다.
- ☞ 때로는 공분산 행렬 대신, “상관 행렬”을 사용할 때도 있다.

8 <직선을 벡터로 나타내면,  
'40,000차원의 벡터'  
로 표현됨.>

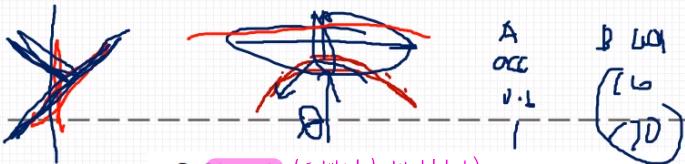
## 1. 차원 축소 (Dimensionality reduction)

대부분의 경우, 현실 세계의 문제는 가공되지 않은 데이터를 처리해야 한다. 예를 들어, 머신 러닝 모델을 이용하여 증명사진에 있는 인물의 성별을 맞추는 문제가 있을 때, 이 문제를 풀기 위해 우리는 성별이 표시된 증명사진을 머신 러닝 모델의 학습 데이터로 이용할 것이다. 하나의 사진이 200X200의 이미지라고 하면, 해당 사진은 총 40,000개의 feature를 갖는 벡터로 표현이 될 것이다. 그러나 대부분의 머신 러닝 모델은 입력 데이터의 차원이 클 경우, 차원의 저주와 학습 속도가 저하되는 문제를 갖고 있다. 이를 위해 생각해볼 수 있는 것은 이미지에서 인물에 대한 정보를 포함하지 않는 부분을 제거하여 입력 데이터의 차원을 낮추는 것이다. 예를 들어, 증명사진의 왼쪽과 오른쪽 상단은 단색의 배경이기 때문에 인물에 대한 정보를 포함하지 않으며, 배경 부분을 제거하여 모델을 학습하여도 성능에는 큰 차이가 없을 것이다. 이와 같이 데이터에서 불필요한 feature를 제거하는 작업을 차원 축소라고 한다.

# 32 PCA



A Tutorial on PCA, J. Shlens (2014)



## I. Linearity (동립변수의 선형성)

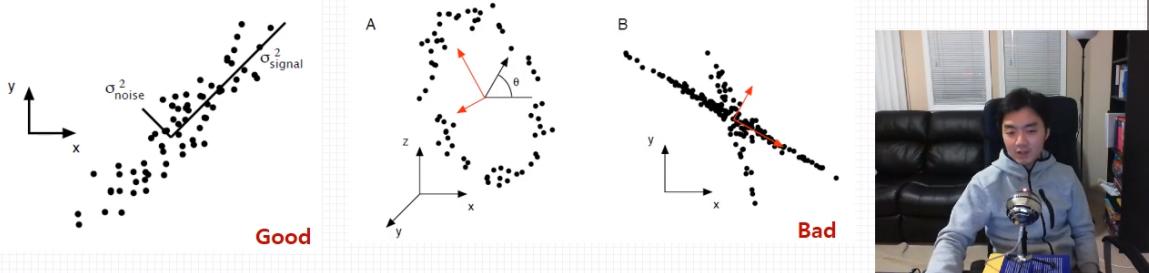
Linearity frames the problem as a change of basis. Several areas of research have explored how extending these notions to nonlinear regimes (see Discussion).

## II. Large variances have important structure.

This assumption also encompasses the belief that the data has a high SNR. Hence, principal components with larger associated variances represent interesting structure, while those with lower variances represent noise. Note that this is a strong, and sometimes, incorrect assumption (see Discussion).

## III. The principal components are orthogonal.

This assumption provides an intuitive simplification that makes PCA soluble with linear algebra decomposition techniques. These techniques are highlighted in the two following sections.



- 위의 한계점이 있다고 하더라도, 주어진 데이터셋에 PCA를 실시하는 것이 좋다.

\* Unsupervised learning의 시작은 dimensionality reduction이다.

- ↑
1. 차원을 축소시키는 알고리즘들의 공통특징은
    - 데이터들이 살고 있는 진짜 공간을 찾는 것'이다. → 'latent space'을 찾는 것'이 목적이다. (latent)
    - 또한, 차원을 축소시키는 알고리즘들은 공통적으로 불필요한 차원을 버리고 필요한 차원을 모으는 작업을 실행한다.

- PCA를 실시할 때 모든 동립변수 짝의 상관관계가 높으면 한 개의 PC로 축소할 수 있고, 모든 동립변수 짝의 상관관계가 높지 않으면 두 개 이상의 PC로 축소할 수 있다.

# 공분산 행렬의 의미 1

- 데이터 구조적 의미: 각 feature의 변동이 얼마나 닮았나?

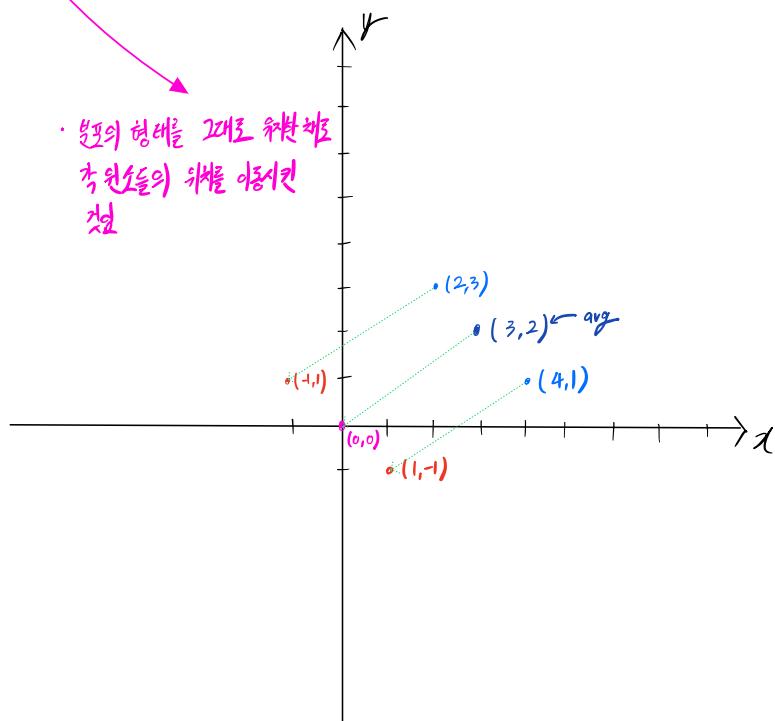
- 다음과 같은 data matrix를 생각해보자.
- 이 data matrix에서 행은 samples, 열은 features를 의미한다.

~~또, 열(features)들의 평균 값은 0이라고 하자 (중요).~~  
~~(즉, feature들의 평균값은 빼준 상태)~~

$$X = \begin{bmatrix} X_1 & X_2 & X_3 & \dots & X_d \end{bmatrix} \in \mathbb{R}^{n \times d}$$

설명:  
행수 → 표본  
열수 → 특성  
각 행은 표본이며 각 열은 특성이다.  
각 열의 평균은 0이라고 하자 (중요).  
(즉, feature들의 평균값은 빼준 상태)

• 본문의 형태를 그대로 유발해  
주원소들의 위치를 이용해  
것을



# 공분산 행렬의 의미 1

- 데이터 구조적 의미: 각 feature의 변동이 얼마나 닮았나?

- 데이터 행렬  $X$ 를 이용해 공분산 행렬을 구해보자.
- 닮은 정도를 보기 위해선 내적이 필요함.
- 그 과정은 다음과 같이 행렬의 곱으로 표현할 수 있음.

$$X^T X = \begin{pmatrix} & X_1 & \\ & X_2 & \\ \dots & & \\ & X_d & \end{pmatrix} \begin{pmatrix} | & | & & | \\ X_1 & X_2 & \dots & X_d \\ | & | & & | \end{pmatrix}$$

# 공분산 행렬의 의미 1

- 데이터 구조적 의미: 각 feature의 변동이 얼마나 닮았나?

$$\begin{aligned} X^T X &= \text{Symmetric} \quad \begin{pmatrix} & X_1 & \\ & X_2 & \\ \dots & & \\ & X_d & \end{pmatrix} \begin{pmatrix} | & | & & | \\ X_1 & X_2 & \dots & X_d \\ | & | & & | \end{pmatrix} \\ &= \begin{pmatrix} \text{↓ 부산치법} & \text{↓ 광분산치법} & & \\ \text{dot}(X_1, X_1) & \text{dot}(X_1, X_2) & \dots & \text{dot}(X_1, X_d) \\ \text{dot}(X_2, X_1) & \text{dot}(X_2, X_2) & \dots & \text{dot}(X_2, X_d) \\ \vdots & \vdots & \ddots & \vdots \\ \text{dot}(X_d, X_1) & \text{dot}(X_d, X_2) & \dots & \text{dot}(X_d, X_d) \end{pmatrix} \end{aligned}$$

## 공분산 행렬의 의미 1

- 데이터 구조적 의미: 각 feature의 변동이 얼마나 닮았나?

- $(X^T X)_{ij}$ 는 i 번째 feature와 j 번째 feature의 변동이 닮은 정도를 말해주고 있음.
- 그런데 문제점은, sample 수(n)가 많아질 수록 그 값이 커진다는 것임.
- 따라서, 이 문제점을 방지해주기 위해 값을 n으로 나눠주면, 아래와 같은 결과를 얻을 수 있음.

$$\Sigma = cov(X) \equiv \frac{X^T X}{n}$$

- 즉, covariance matrix의 i 번째 행과 j 번째 행은 i 번 feature와 j 번 feature가 서로 함께 변하는 정도를 의미하고 있음.

~~※ feature들의 평균을 0으로 만들었다는 사실을 잊으면 안됨.~~

## 공분산 행렬의 의미 2

- 수학적 의미: 선형 변환(shearing)

- 앞에서, 공분산 행렬은 feature 쌍이 서로 함께 변하는 정도를 의미하고 있다고 말했음(이 때, feature들의 평균은 0으로 만들어 준 상태임).
- 임의의 두 feature 쌍의 산점도(scatter plot)가 다음과 같다고 생각해보자.

