Università degli Studi di Padova

Machine Learning
2024/2025

AMCO
Artificial Intelligence, Machine Learning and Control Research Group

Lecture #05 Multi-dimensional Data Visualization

Gian Antonio Susto

Before starting: Lab

Audio is not the best! We are trying to find a solution... in the meantime: bring your own headphones!

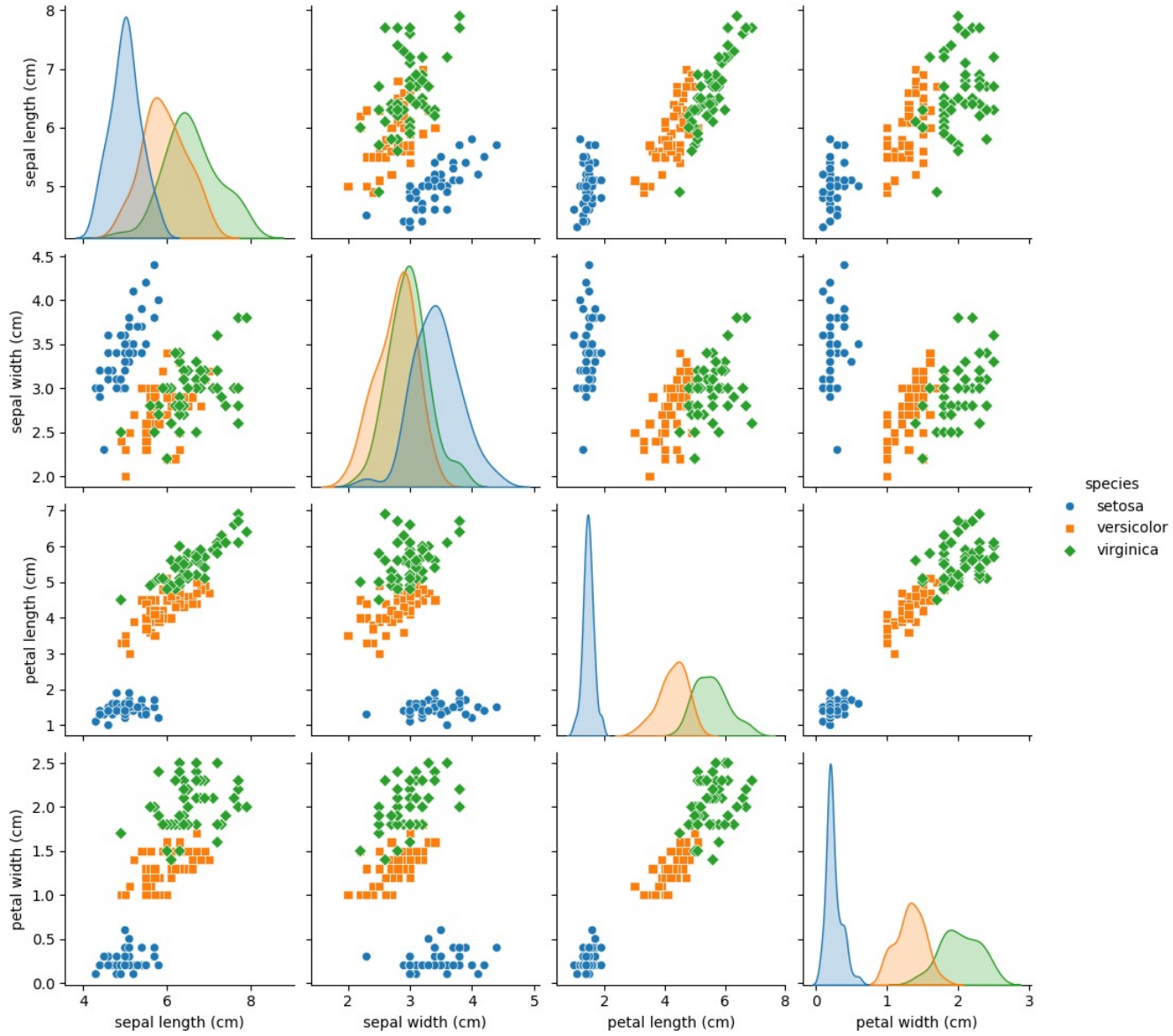
Recap – Tabular Data (the 'design matrix') - *x*

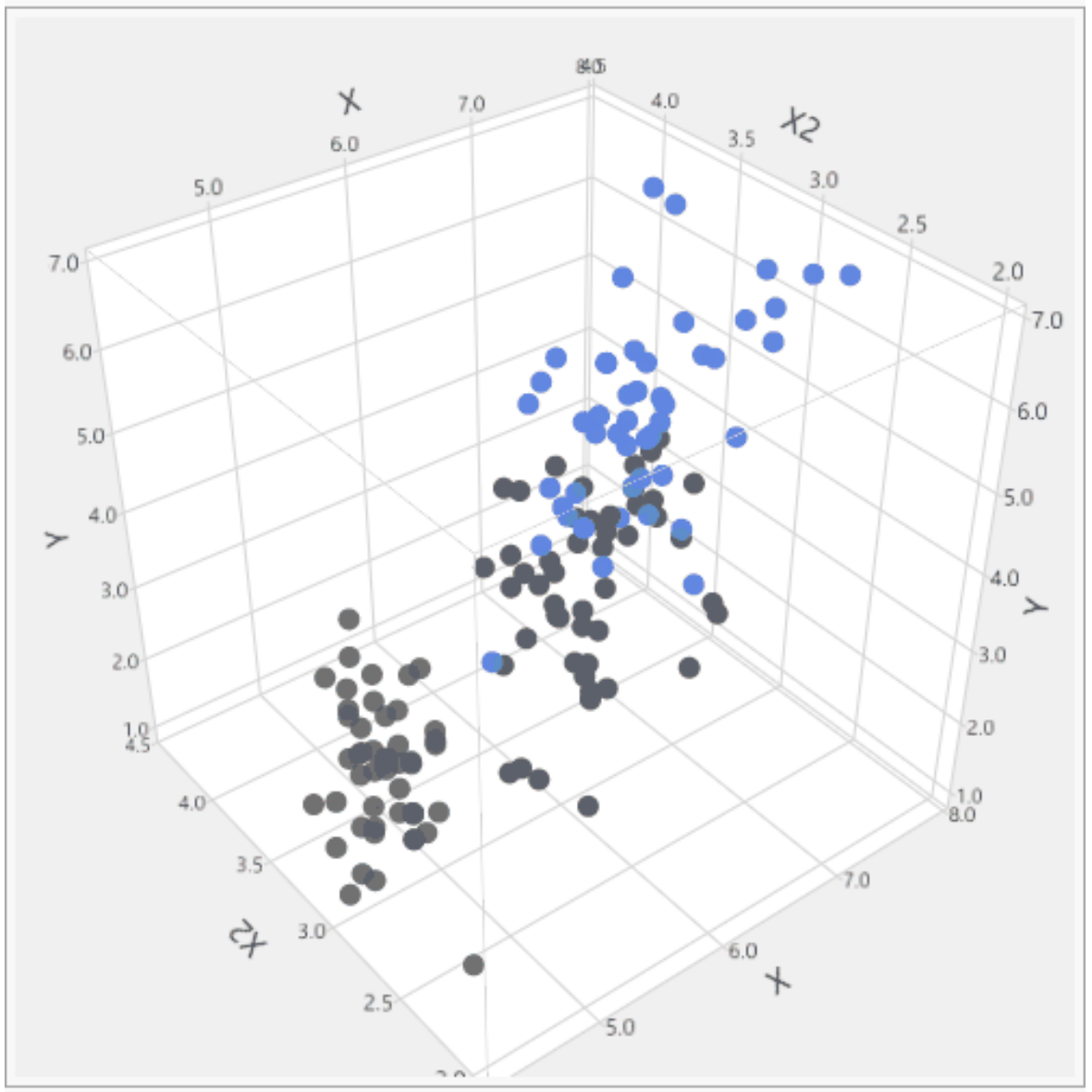
p attributes (variables, features) potentially related to the phenomenon under examination

n observations: the number of times the phenomenon we need to 'describe' is available in our data through historical examples

39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States	<=50K
49	Private	160187	9th	5	Married-spouse-absent	Other-service	Not-in-family	Black	Female	0	0	16	Jamaica	<=50K
52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	45	United-States	>50K
31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Female	14084	0	50	United-States	>50K
42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	5178	0	40	United-States	>50K
37	Private	280464	Some-college	10	Married-civ-spouse	Exec-managerial	Husband	Black	Male	0	0	80	United-States	>50K
30	State-gov	141297	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	Asian-Pac-Islander	Male	0	0	40	India	>50K
23	Private	122272	Bachelors	13	Never-married	Adm-clerical	Own-child	White	Female	0	0	30	United-States	<=50K
32	Private	205019	Assoc-acdm	12	Never-married	Sales	Not-in-family	Black	Male	0	0	50	United-States	<=50K
40	Private	121772	Assoc-voc	11	Married-civ-spouse	Craft-repair	Husband	Asian-Pac-Islander	Male	0	0	40	?	>50K
34	Private	245487	7th-8th	4	Married-civ-spouse	Transport-moving	Husband	Amer-Indian-Eskimo	Male	0	0	45	Mexico	<=50K
25	Self-emp-not-inc	176756	HS-grad	9	Never-married	Farming-fishing	Own-child	White	Male	0	0	35	United-States	<=50K
32	Private	186824	HS-grad	9	Never-married	Machine-op-inspct	Unmarried	White	Male	0	0	40	United-States	<=50K
38	Private	28887	11th	7	Married-civ-spouse	Sales	Husband	White	Male	0	0	50	United-States	<=50K
43	Self-emp-not-inc	292175	Masters	14	Divorced	Exec-managerial	Unmarried	White	Female	0	0	45	United-States	>50K
40	Private	193524	Doctorate	16	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	0	60	United-States	>50K
54	Private	302146	HS-grad	9	Separated	Other-service	Unmarried	Black	Female	0	0	20	United-States	<=50K
35	Federal-gov	76845	9th	5	Married-civ-spouse	Farming-fishing	Husband	Black	Male	0	0	40	United-States	<=50K
43	Private	117037	11th	7	Married-civ-spouse	Transport-moving	Husband	White	Male	0	2042	40	United-States	<=50K
59	Private	109015	HS-grad	9	Divorced	Tech-support	Unmarried	White	Female	0	0	40	United-States	<=50K

Recap – 1D, 2D, 3D Plots… and then?

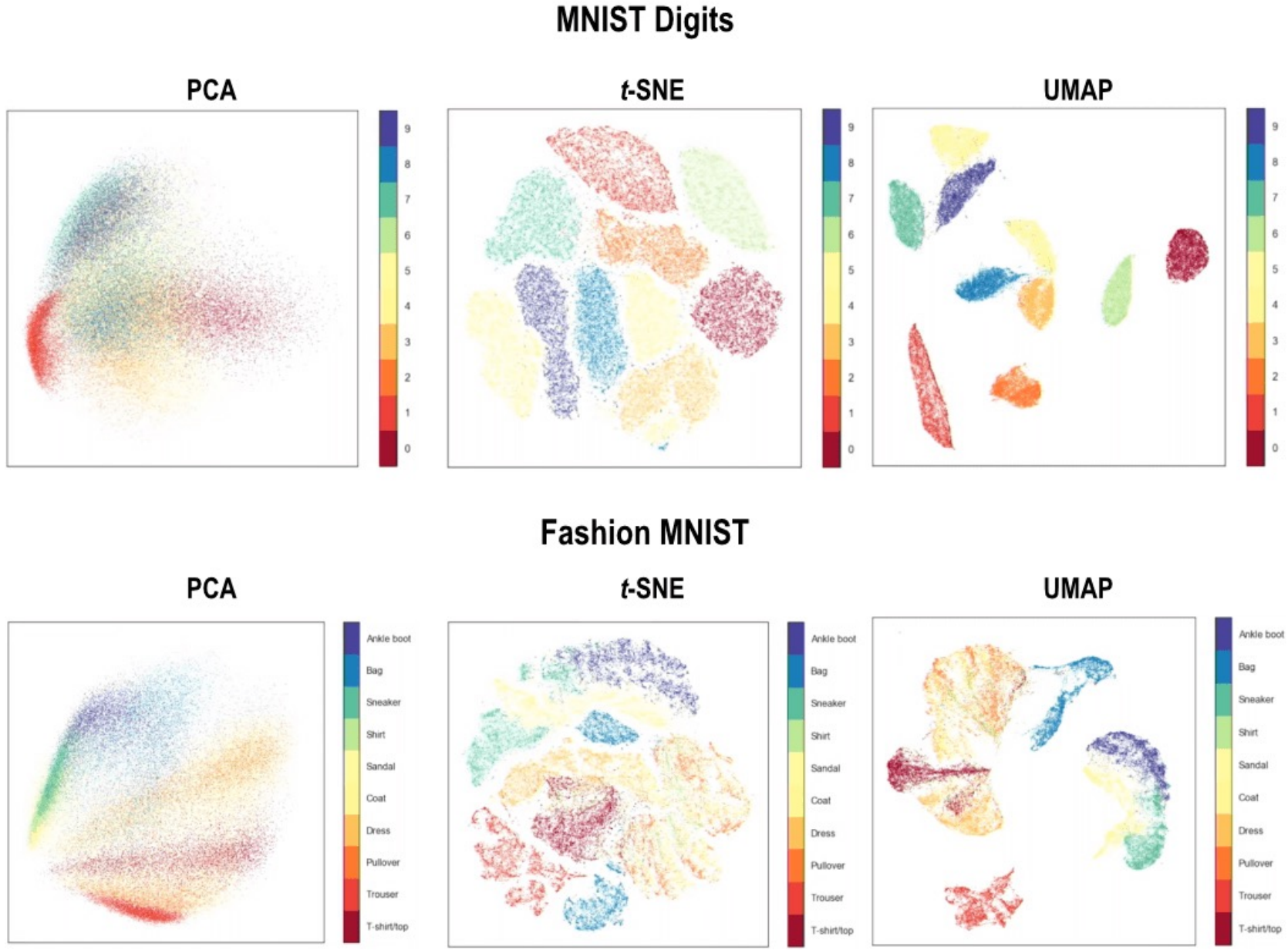




Recap – 1D, 2D, 3D Plots… and then?

There are several techniques that allow to visualize multi-dimensional data (p > 3)!

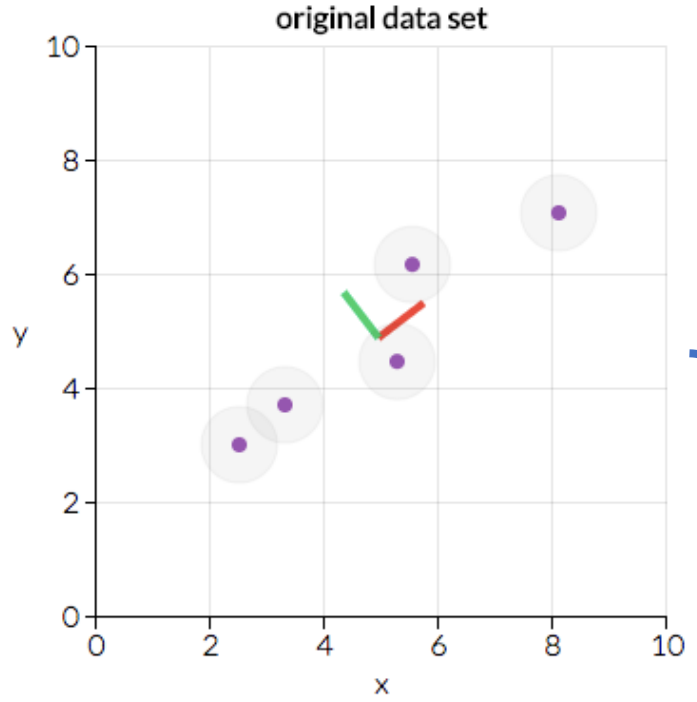
We will see today:
- PCA
- t-SNE
- UMAP

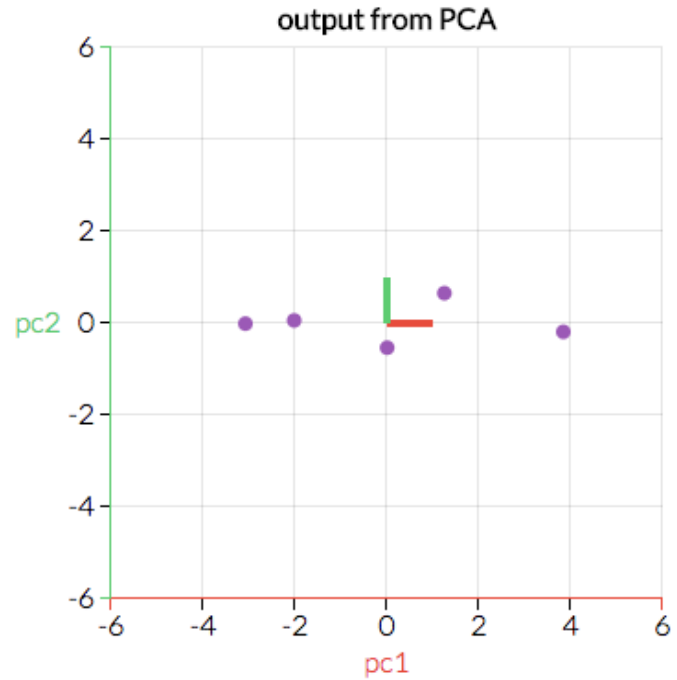


Principal Component Analysis (PCA)

Huge kudos to Joshua Starmer!

Principal Component Analysis (PCA)





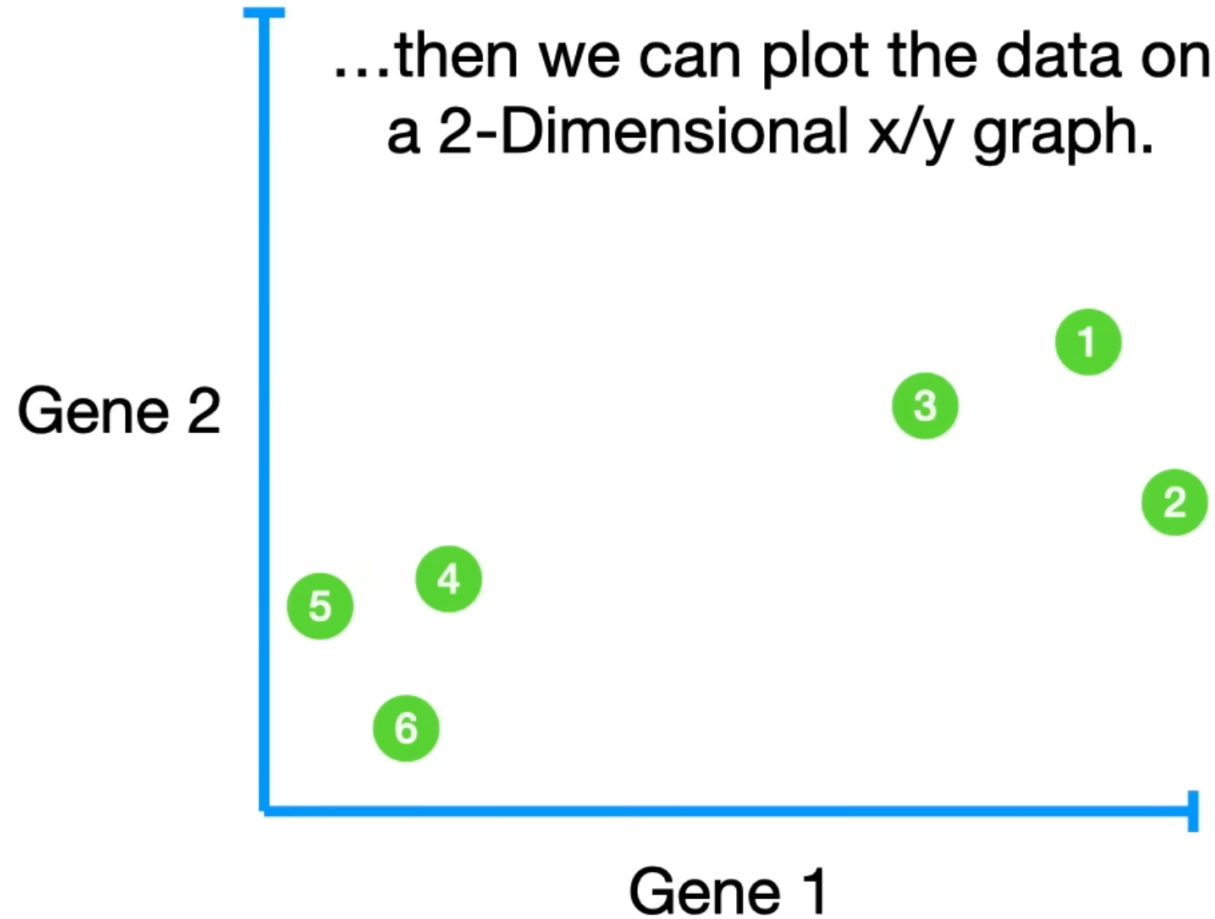
- **PCA** computes the sequence of mutually orthogonal vectors that best fit the data, aka the **principal components**.
- We can use the first 2 or 3 PCs to visualize the data... **lossy** compression!
- PCA preserves **global structure**

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

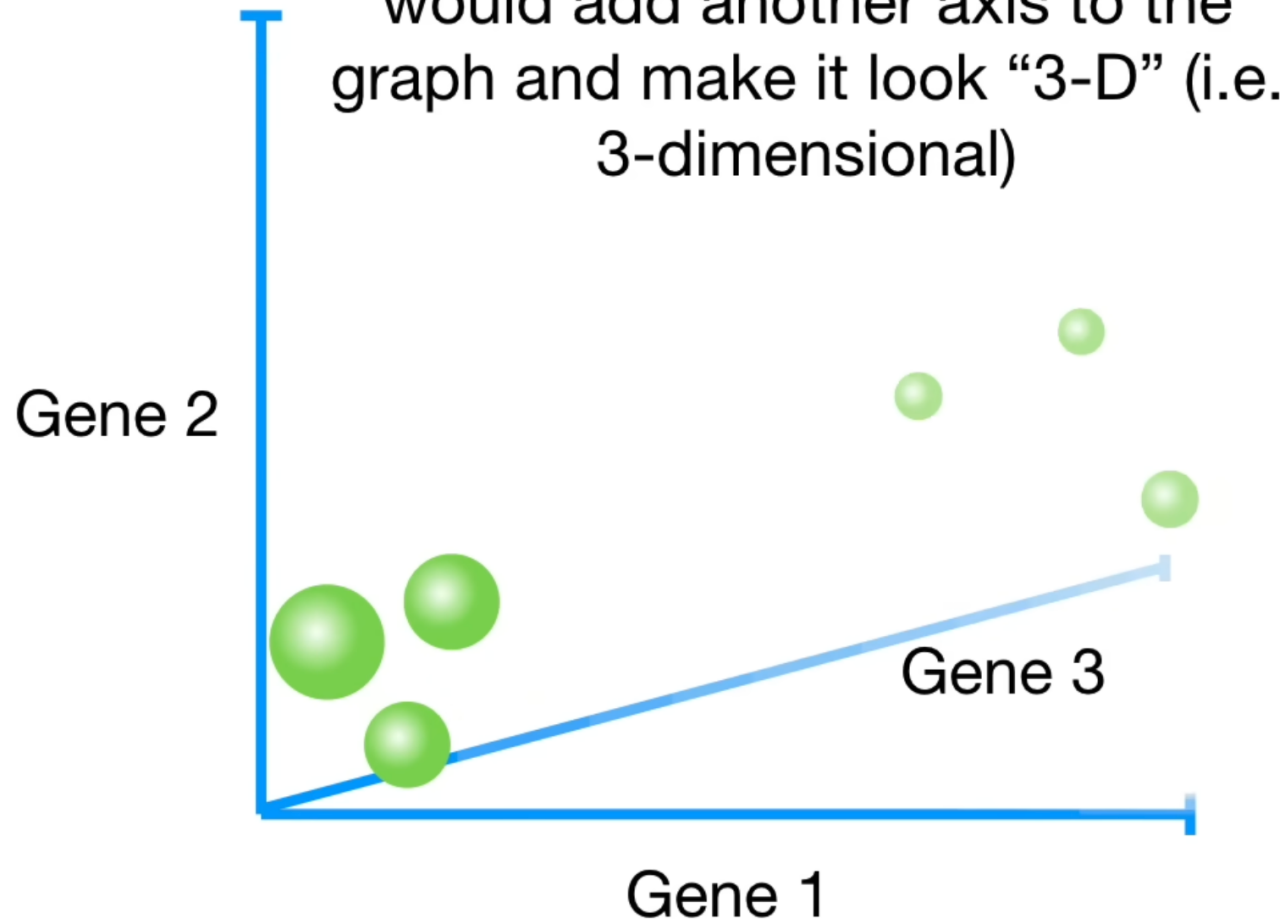
If we only measure 1 gene, we can plot the data on a number line…

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

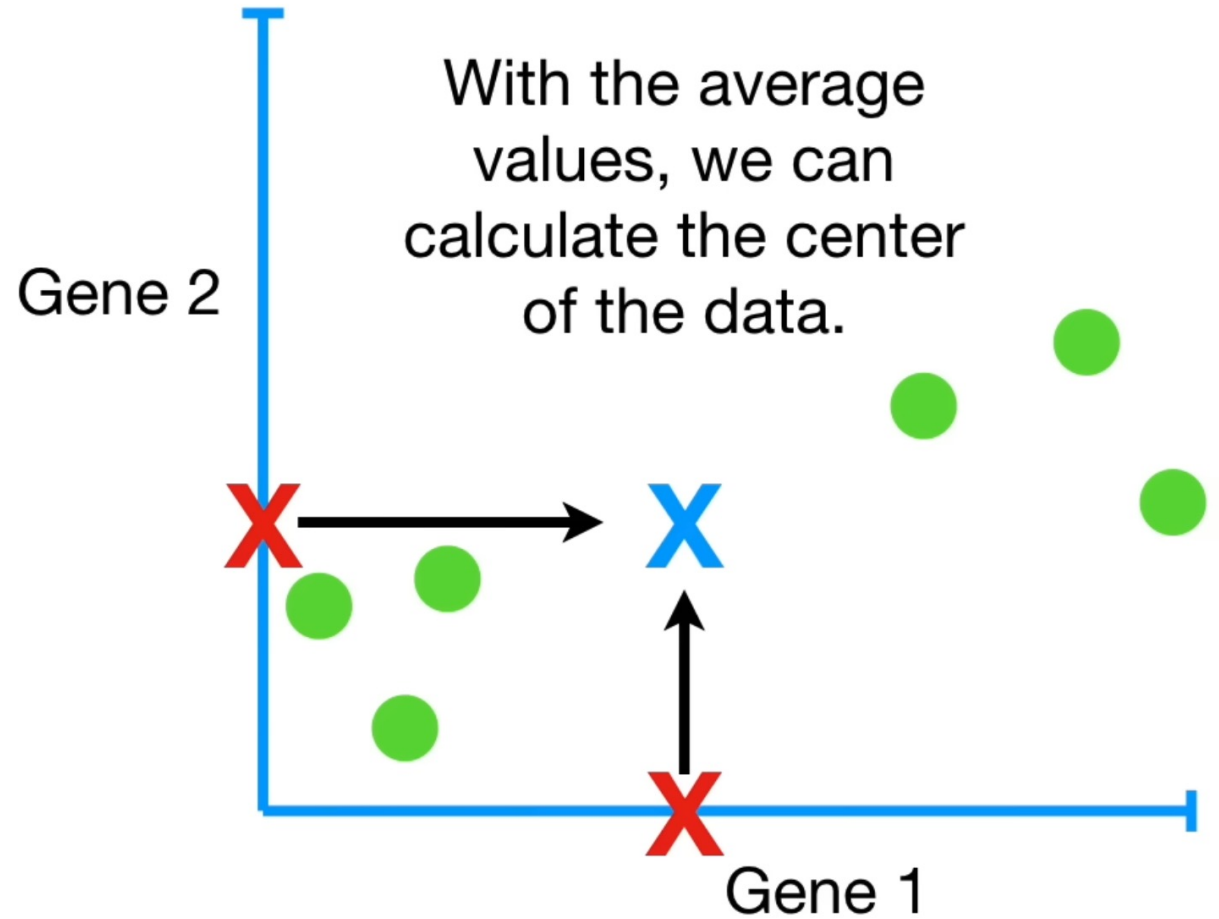


	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1
Gene 3	12	9	10	2.5	1.3	2

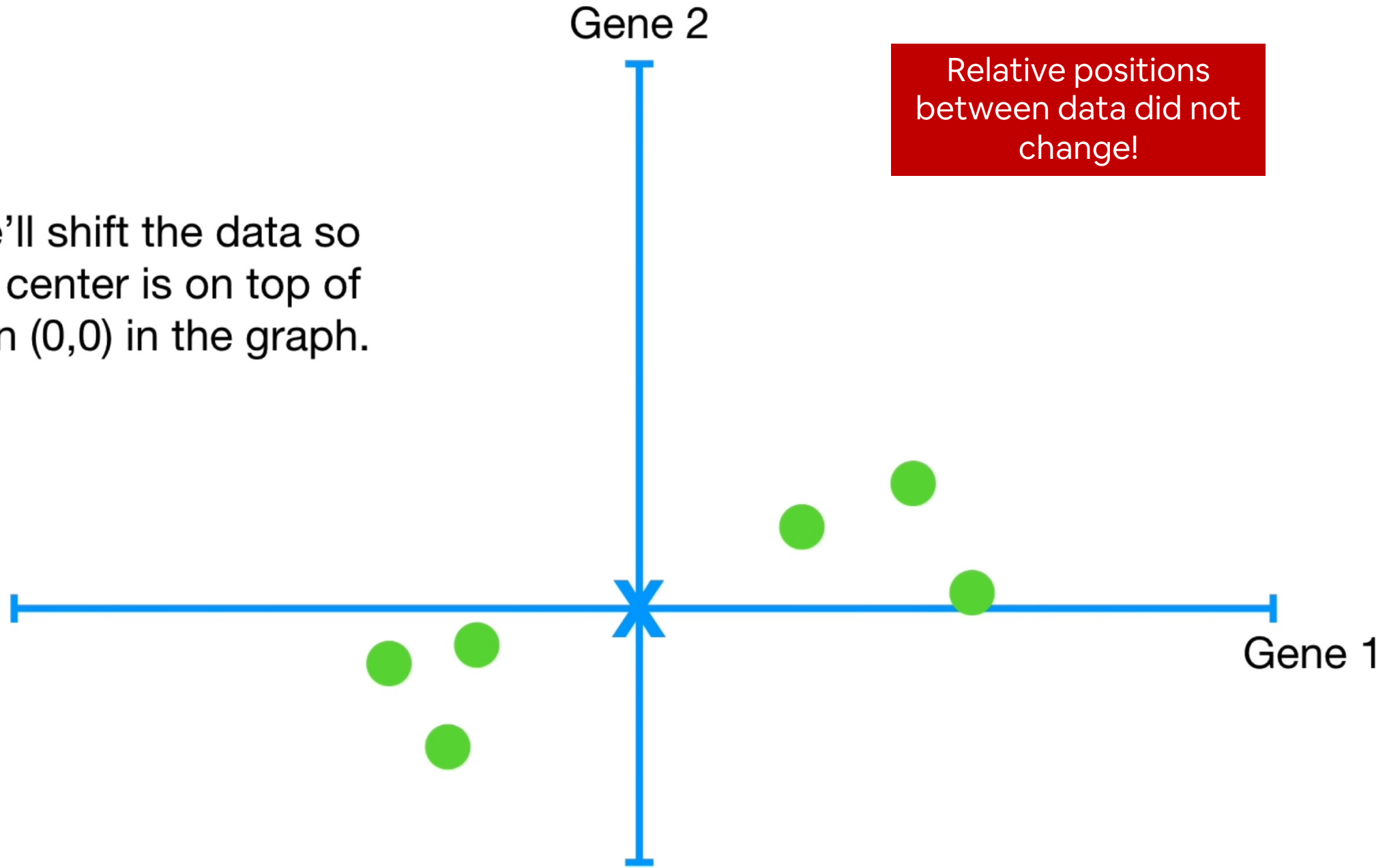
If we measured 3 genes, we would add another axis to the graph and make it look “3-D” (i.e. 3-dimensional)



	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



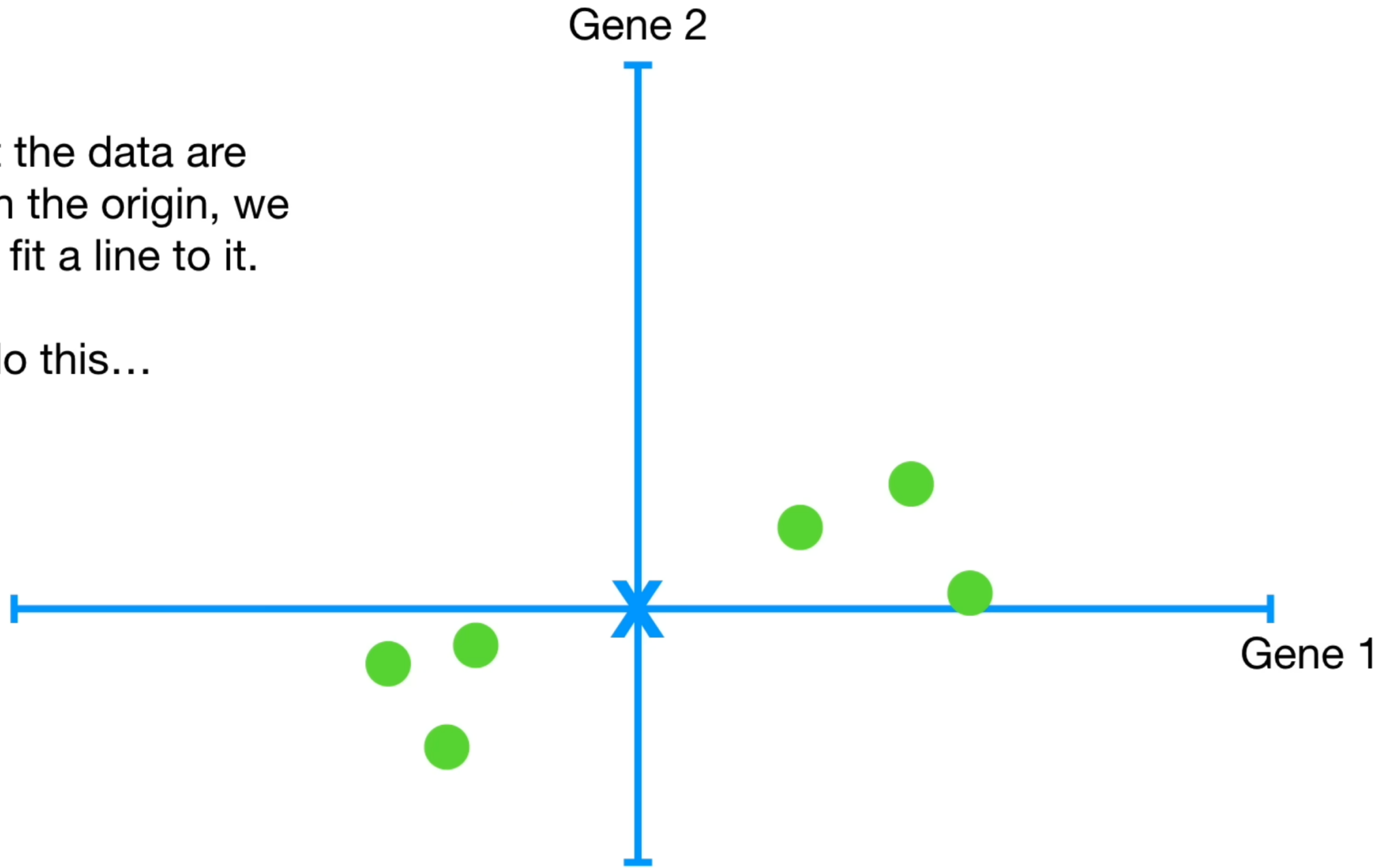
Now we'll shift the data so that the center is on top of the origin (0,0) in the graph.



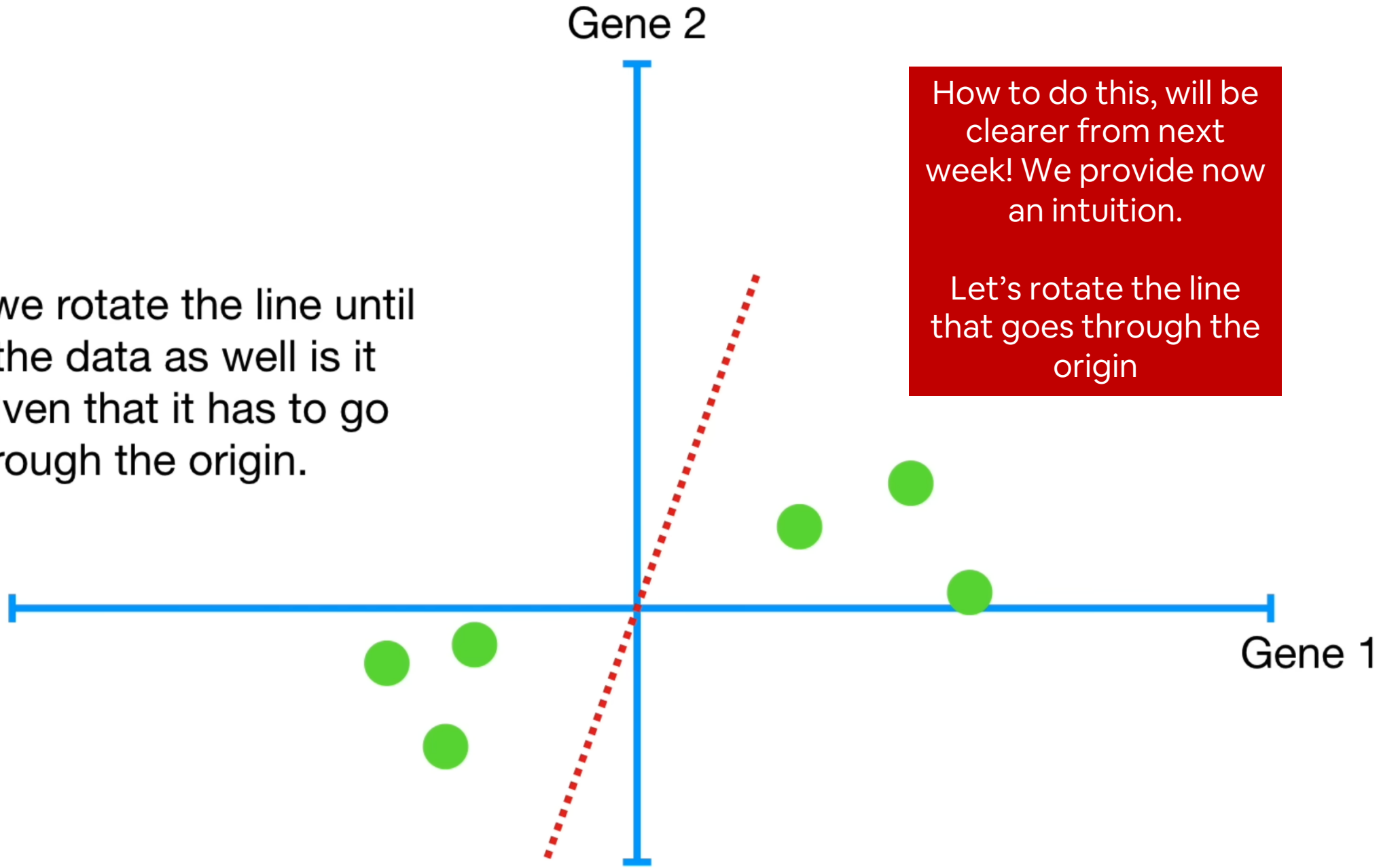
Relative positions between data did not change!

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

To do this…

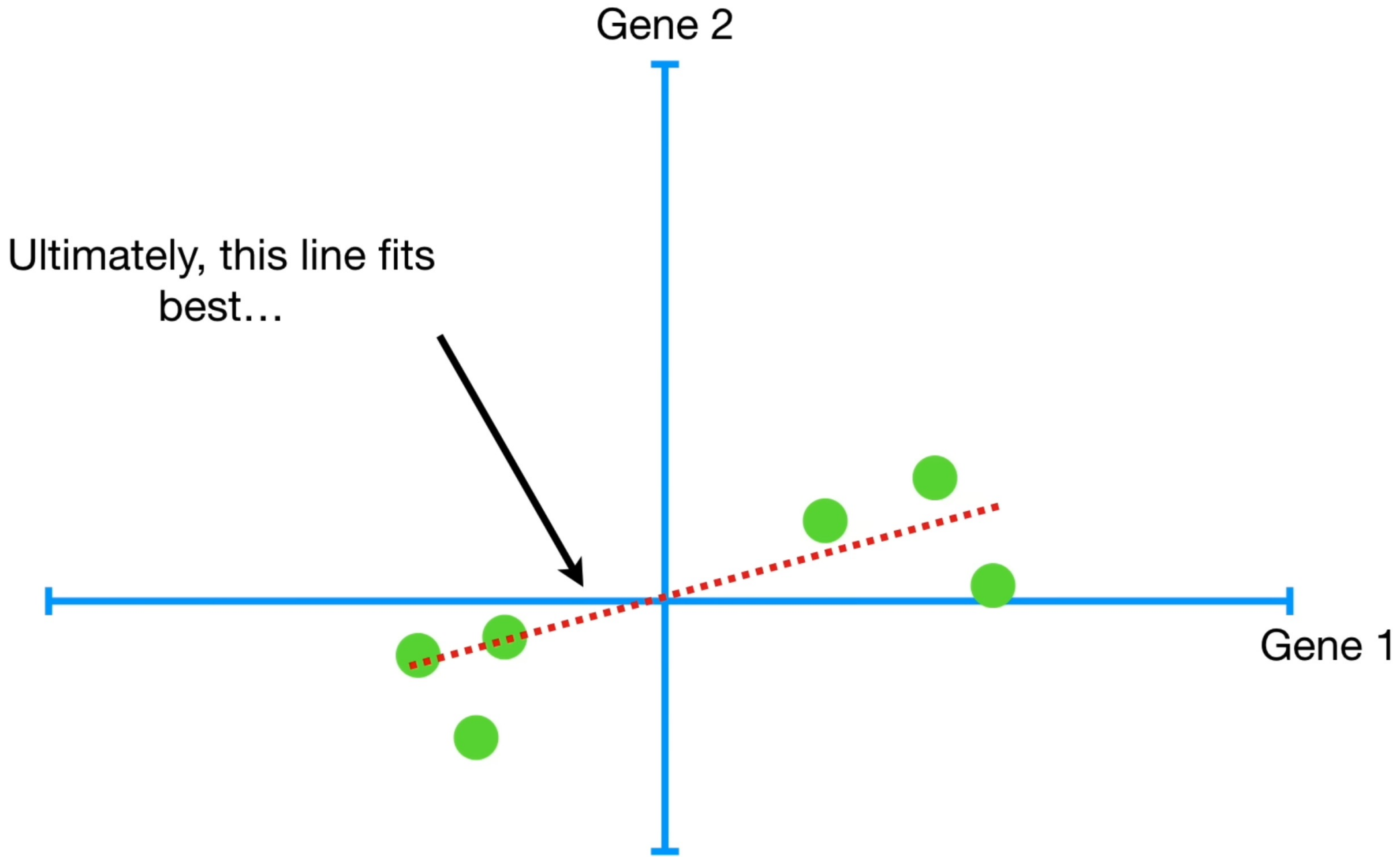


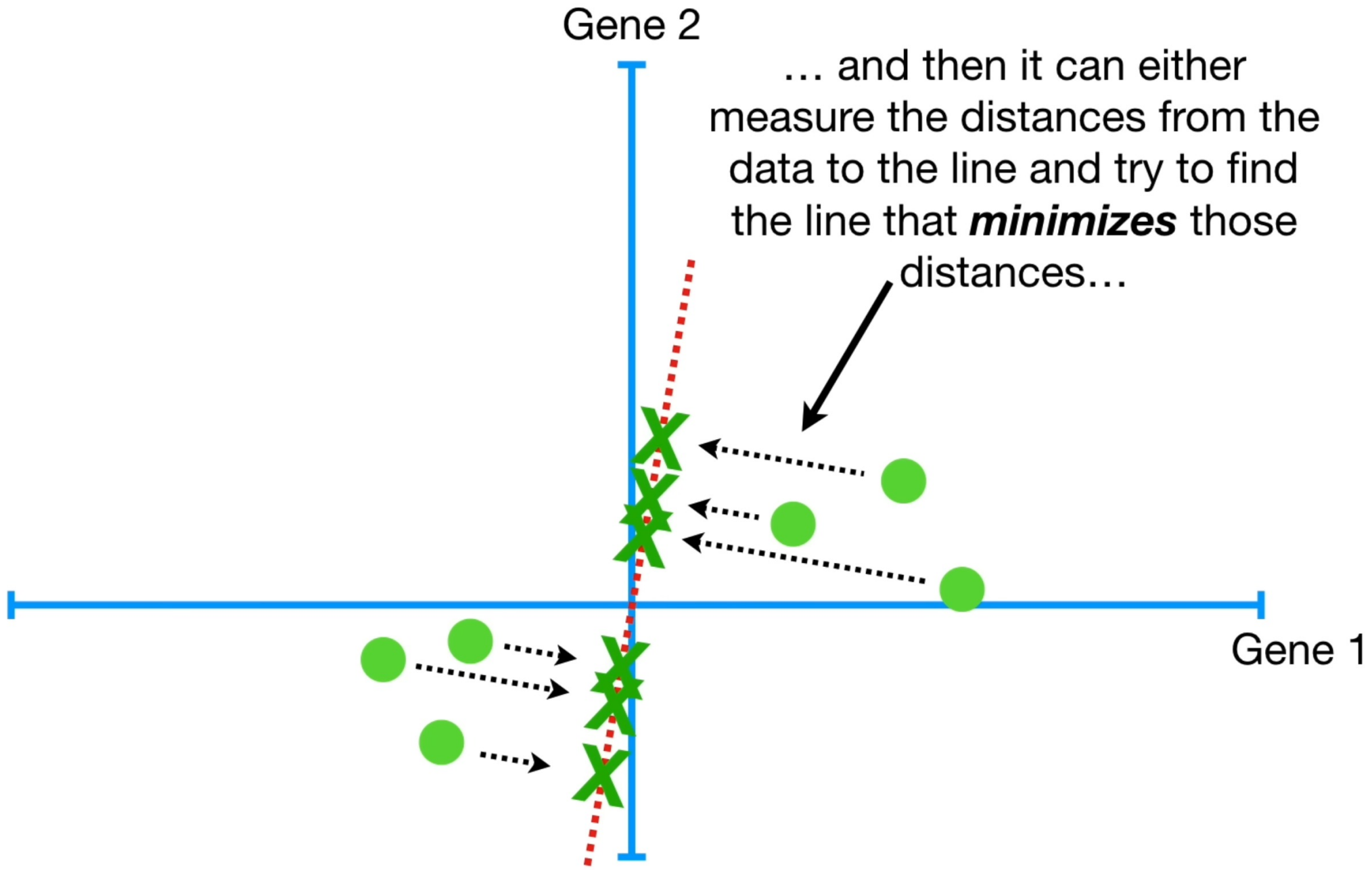
…then we rotate the line until it fits the data as well is it can, given that it has to go through the origin.



How to do this, will be clearer from next week! We provide now an intuition.

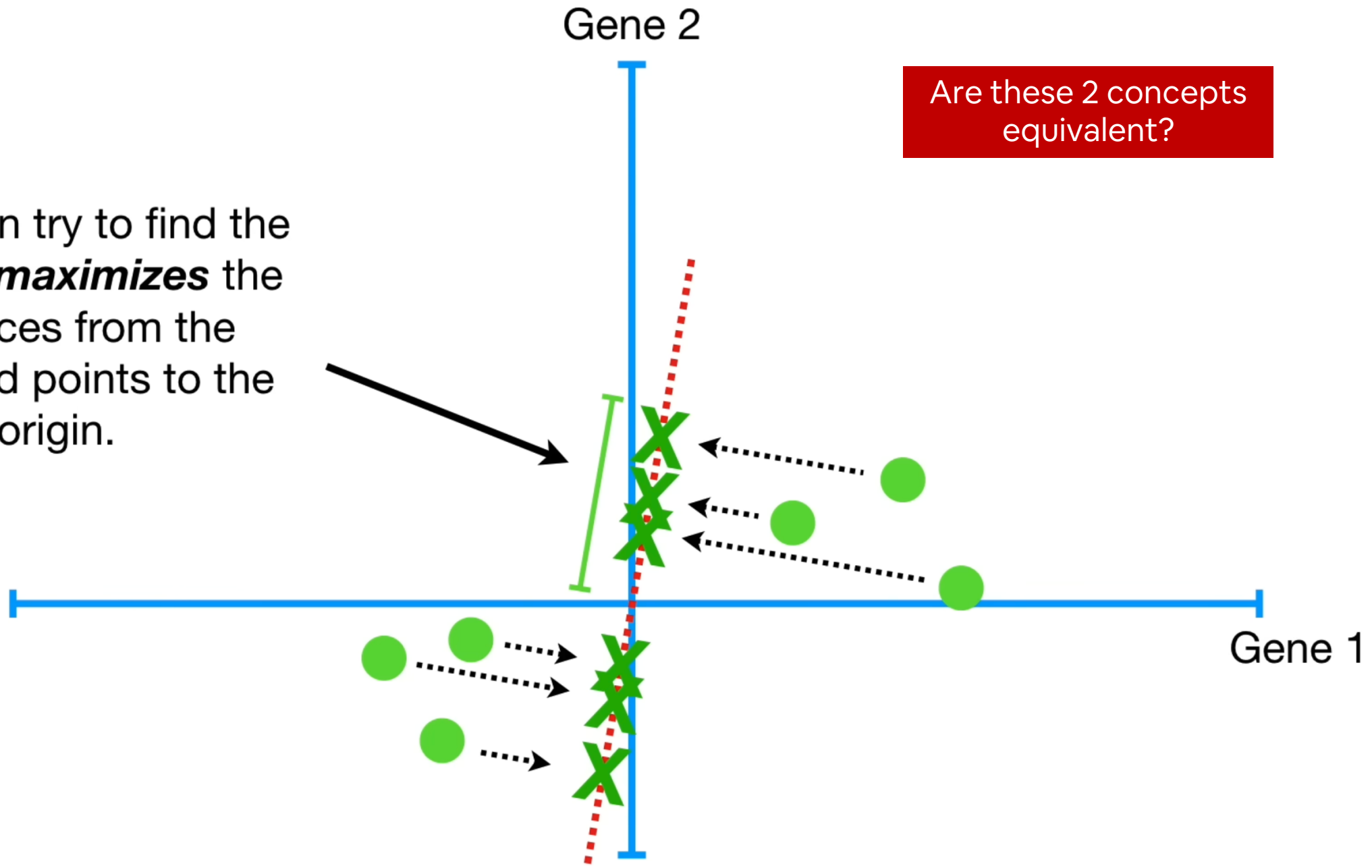
Let’s rotate the line that goes through the origin

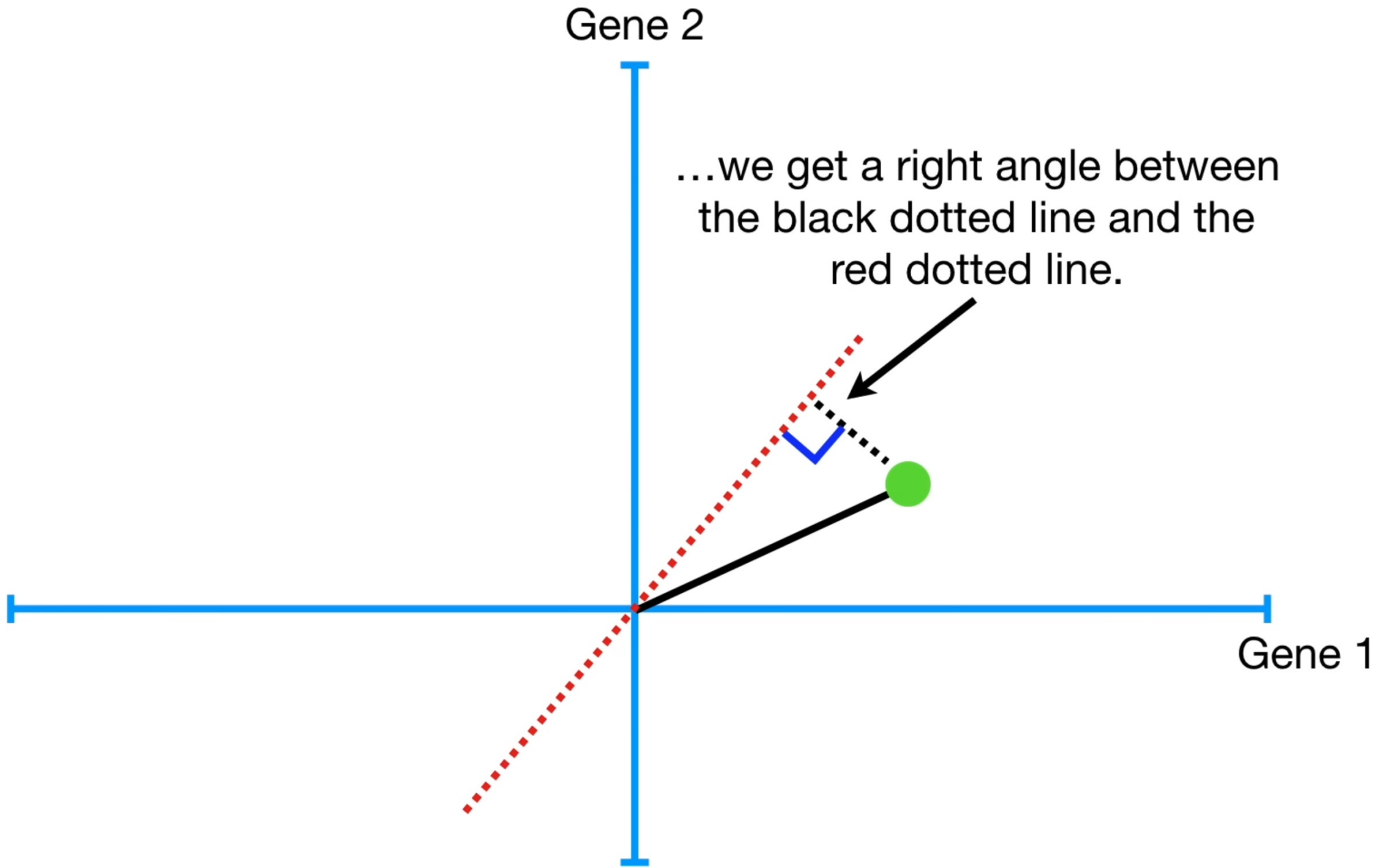

Gene 2
Ultimately, this line fits best…
Gene 1


Gene 2
… and then it can either measure the distances from the data to the line and try to find the line that ***minimizes*** those distances…
Gene 1

Are these 2 concepts equivalent?

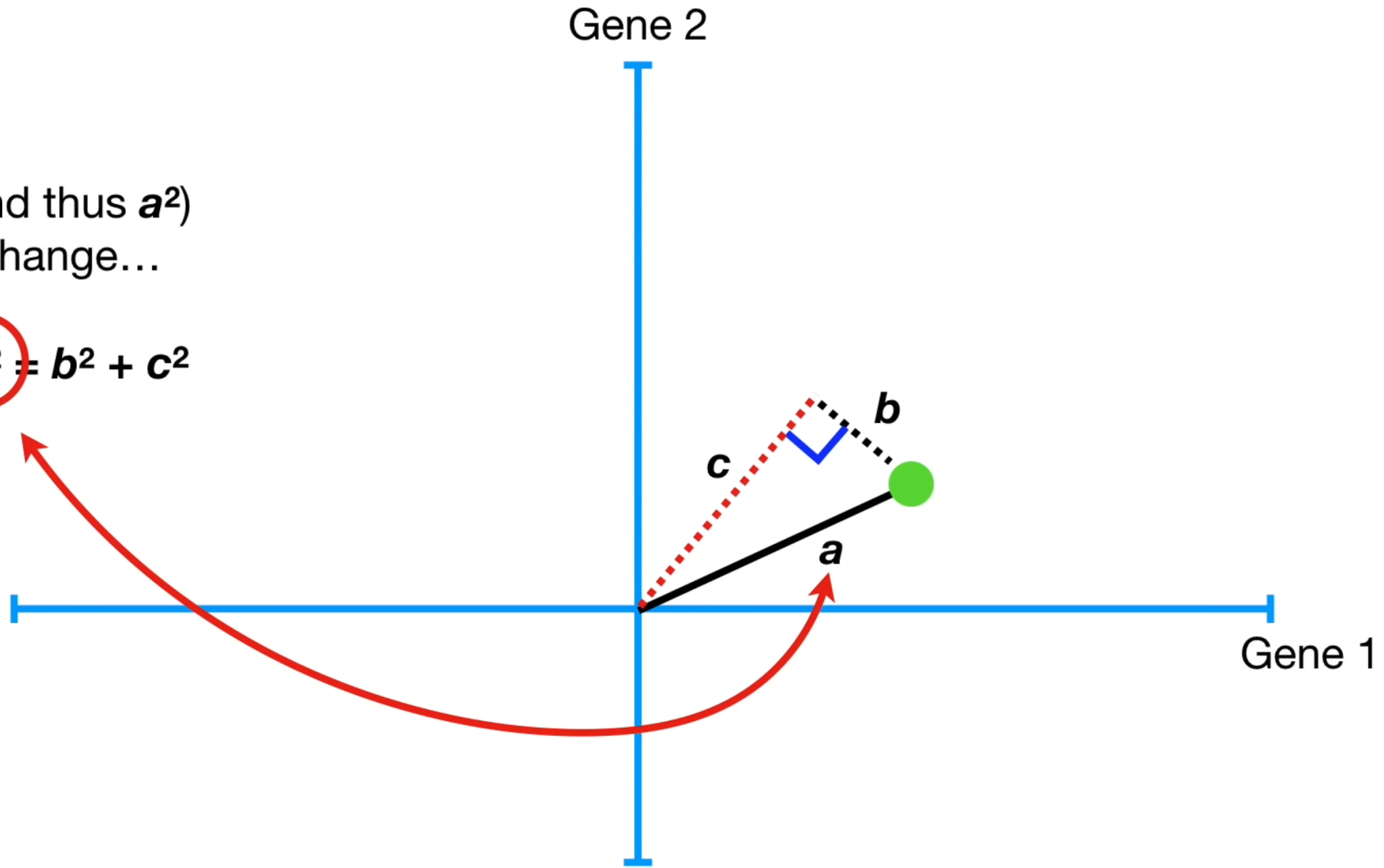
…or it can try to find the line that ***maximizes*** the distances from the projected points to the origin.

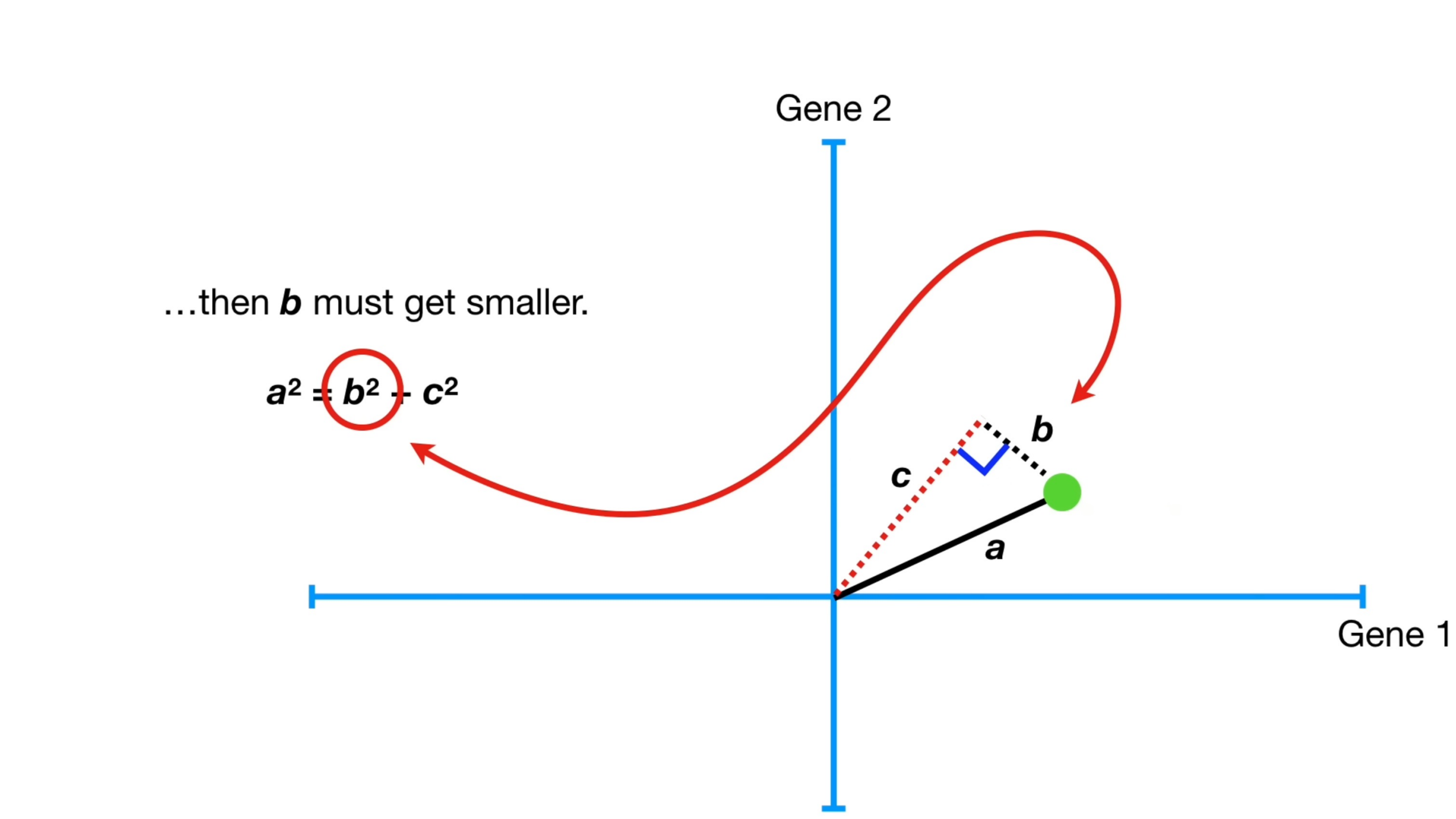



Gene 2
…we get a right angle between the black dotted line and the red dotted line.
Gene 1

Since ***a*** (and thus a^2) doesn't change…

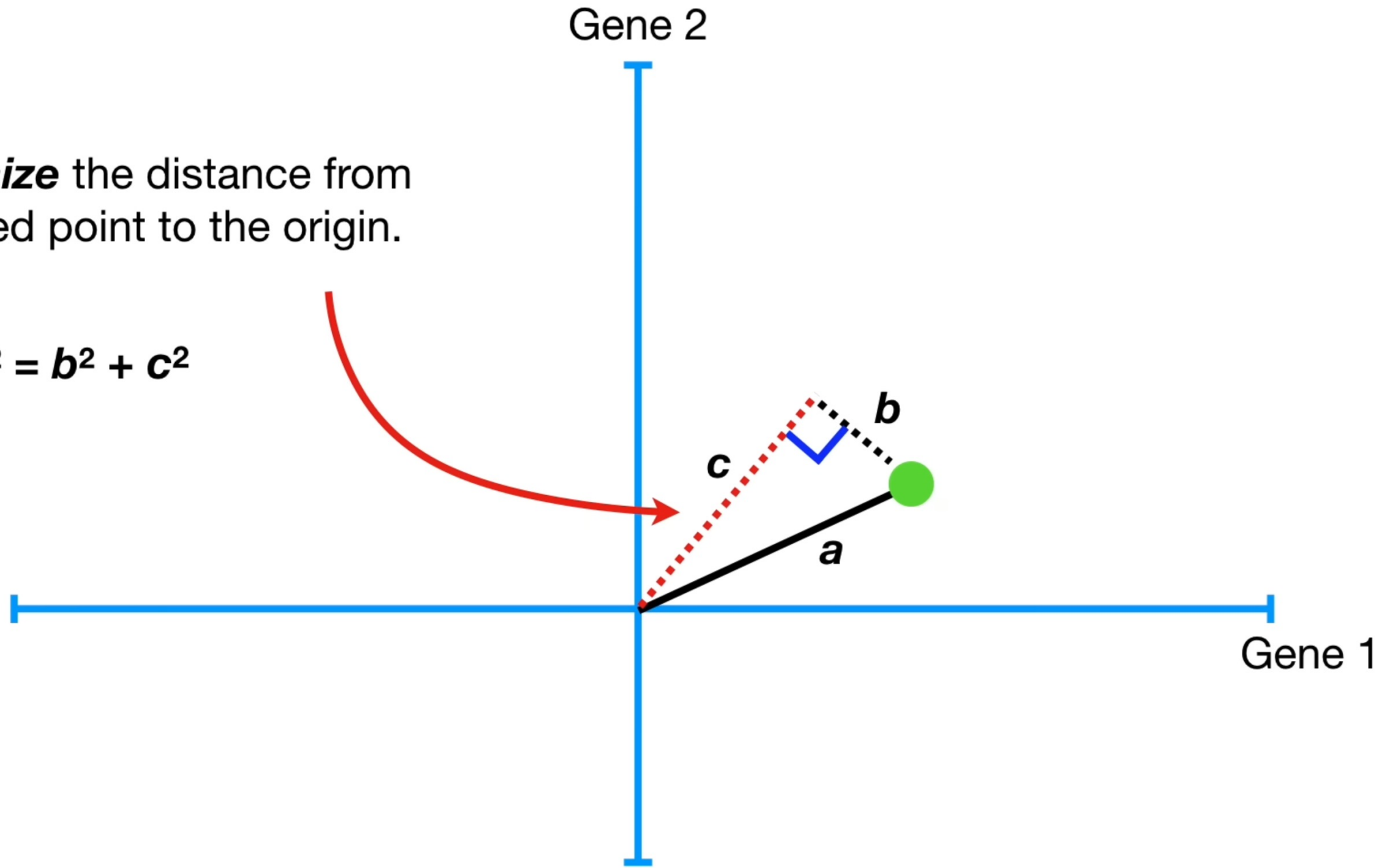
$a^2 = b^2 + c^2$



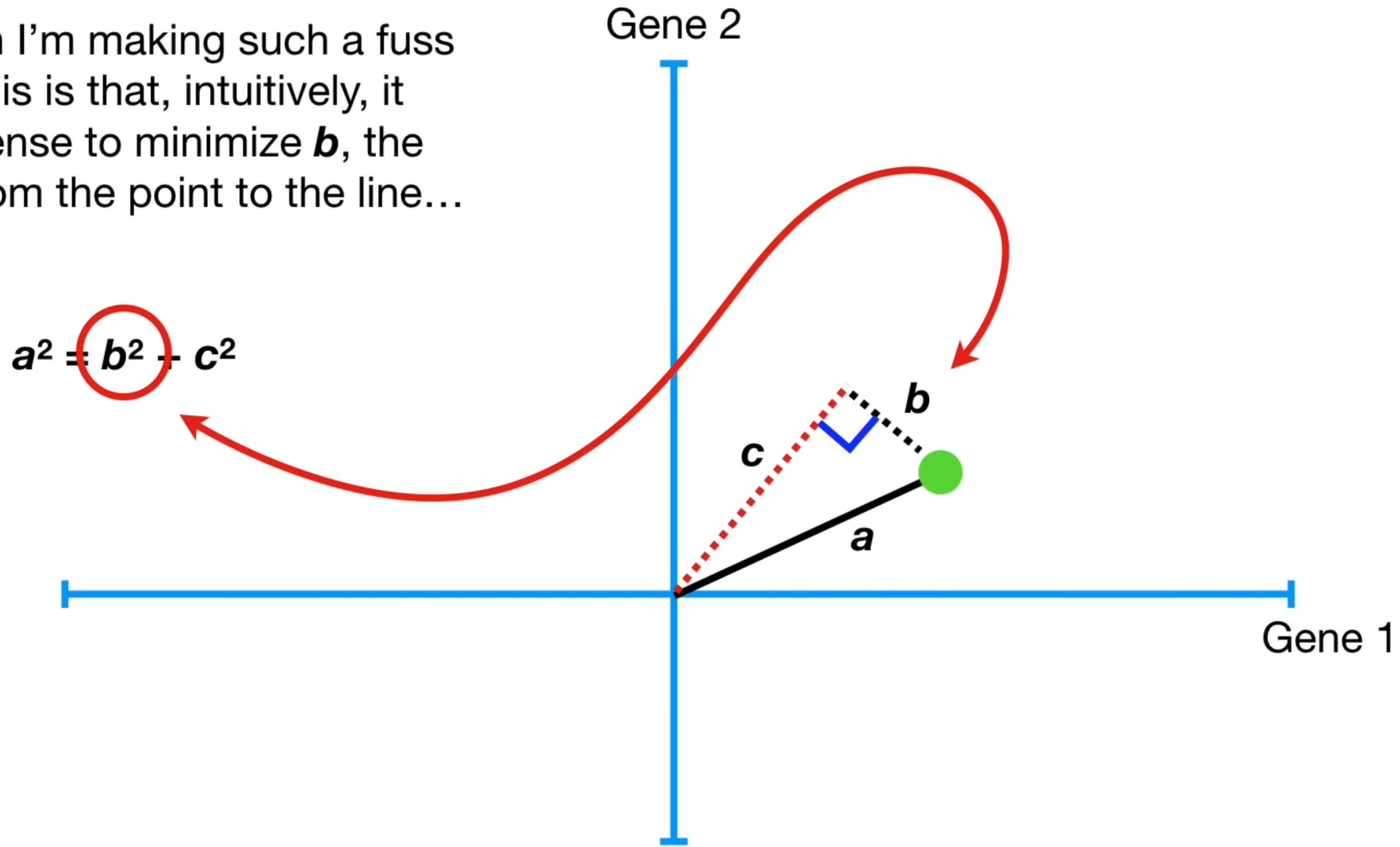

Gene 2
…then b must get smaller.
$a^2 = b^2 + c^2$
b
c
a
Gene 1

…or ***maximize*** the distance from the projected point to the origin.

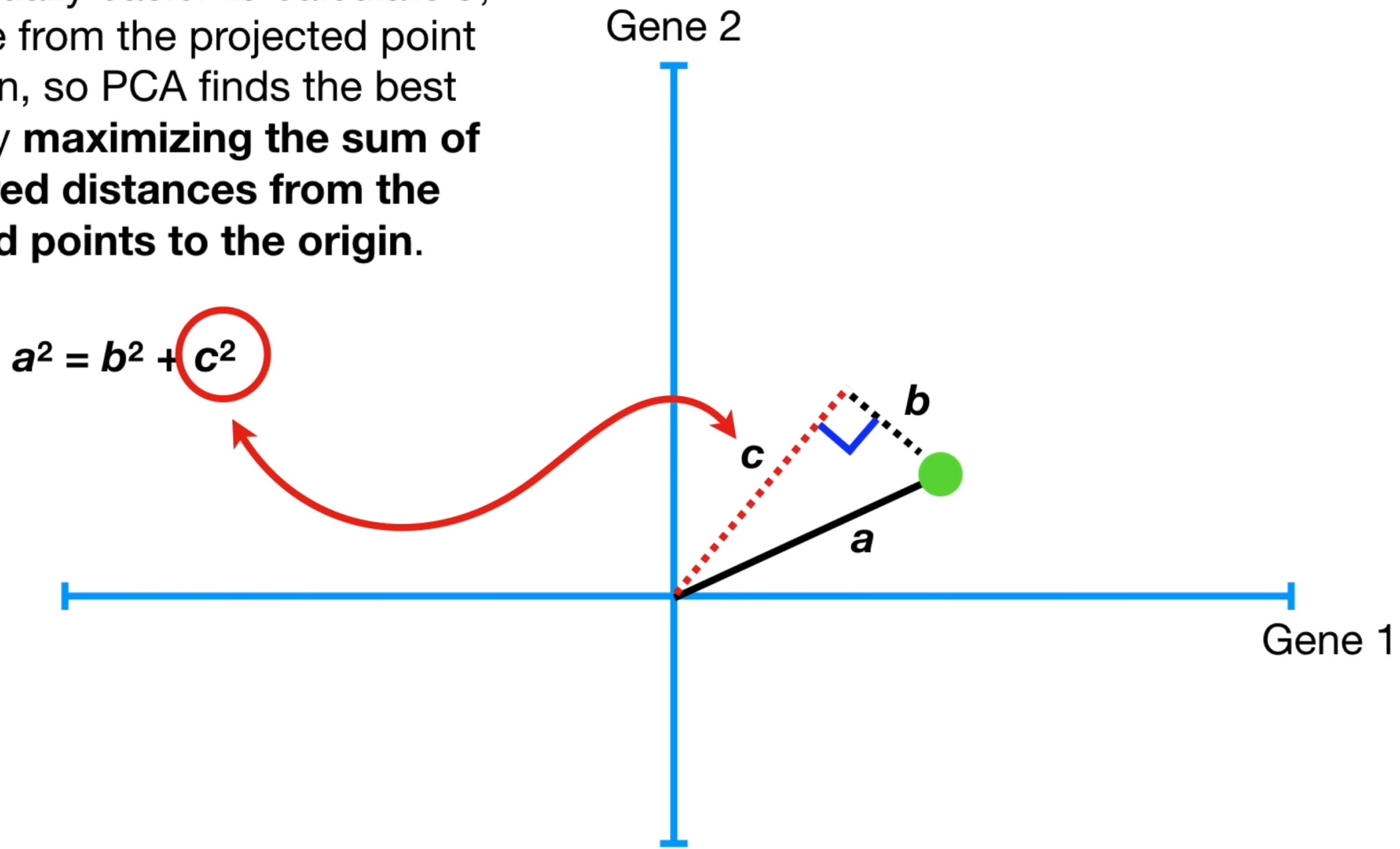
$$a^2 = b^2 + c^2$$

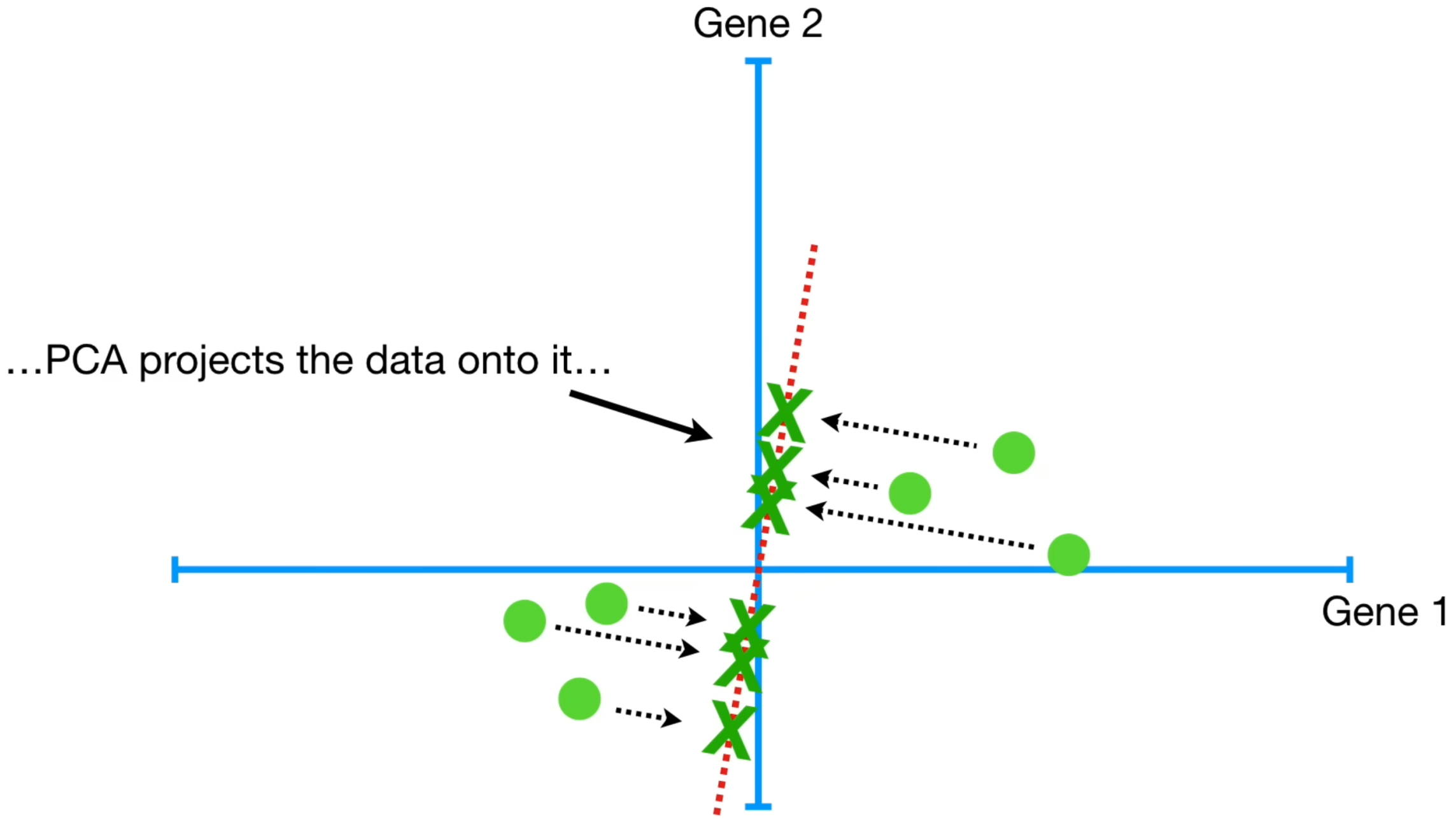


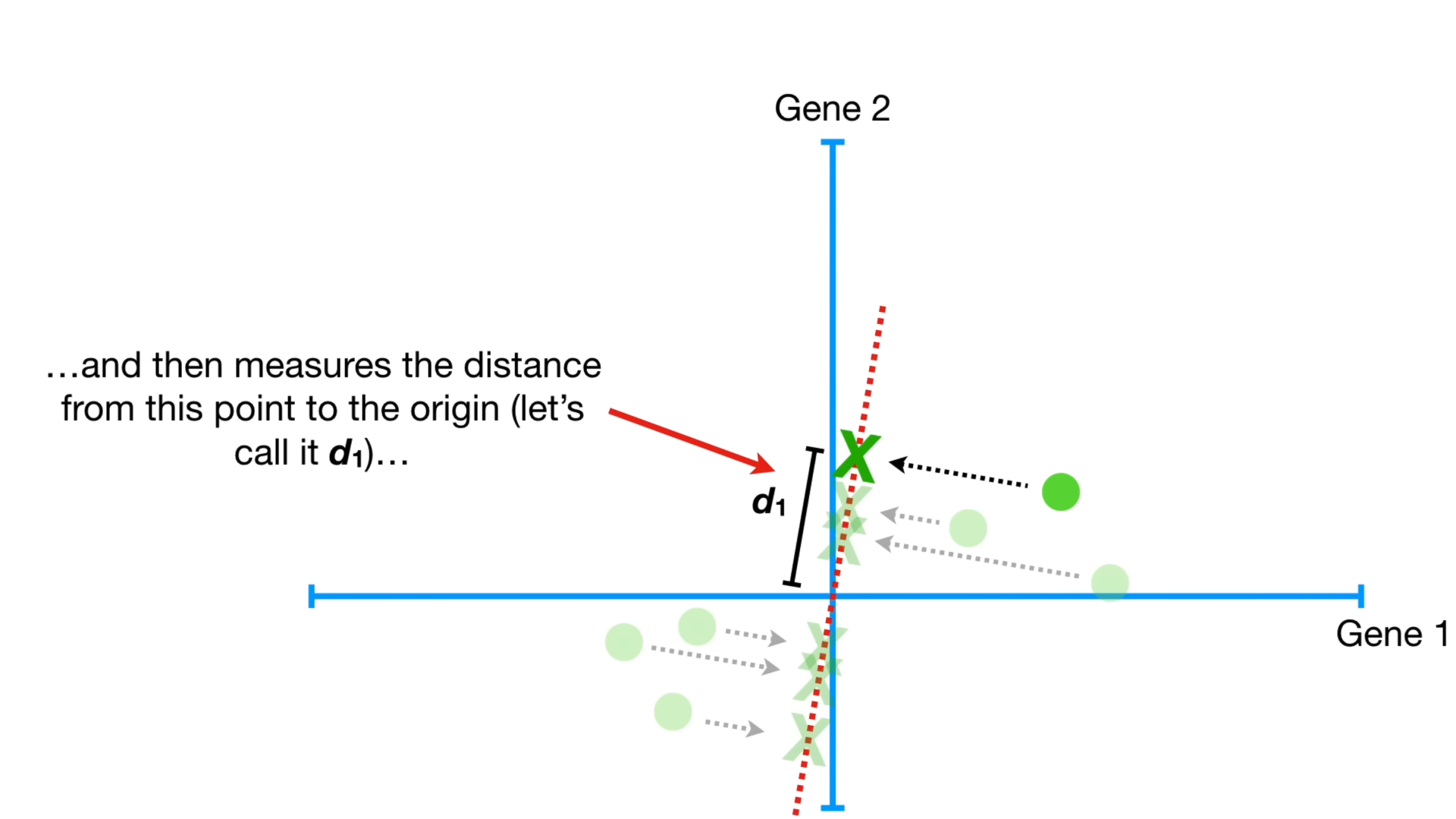
The reason I'm making such a fuss about this is that, intuitively, it makes sense to minimize ***b***, the distance from the point to the line…

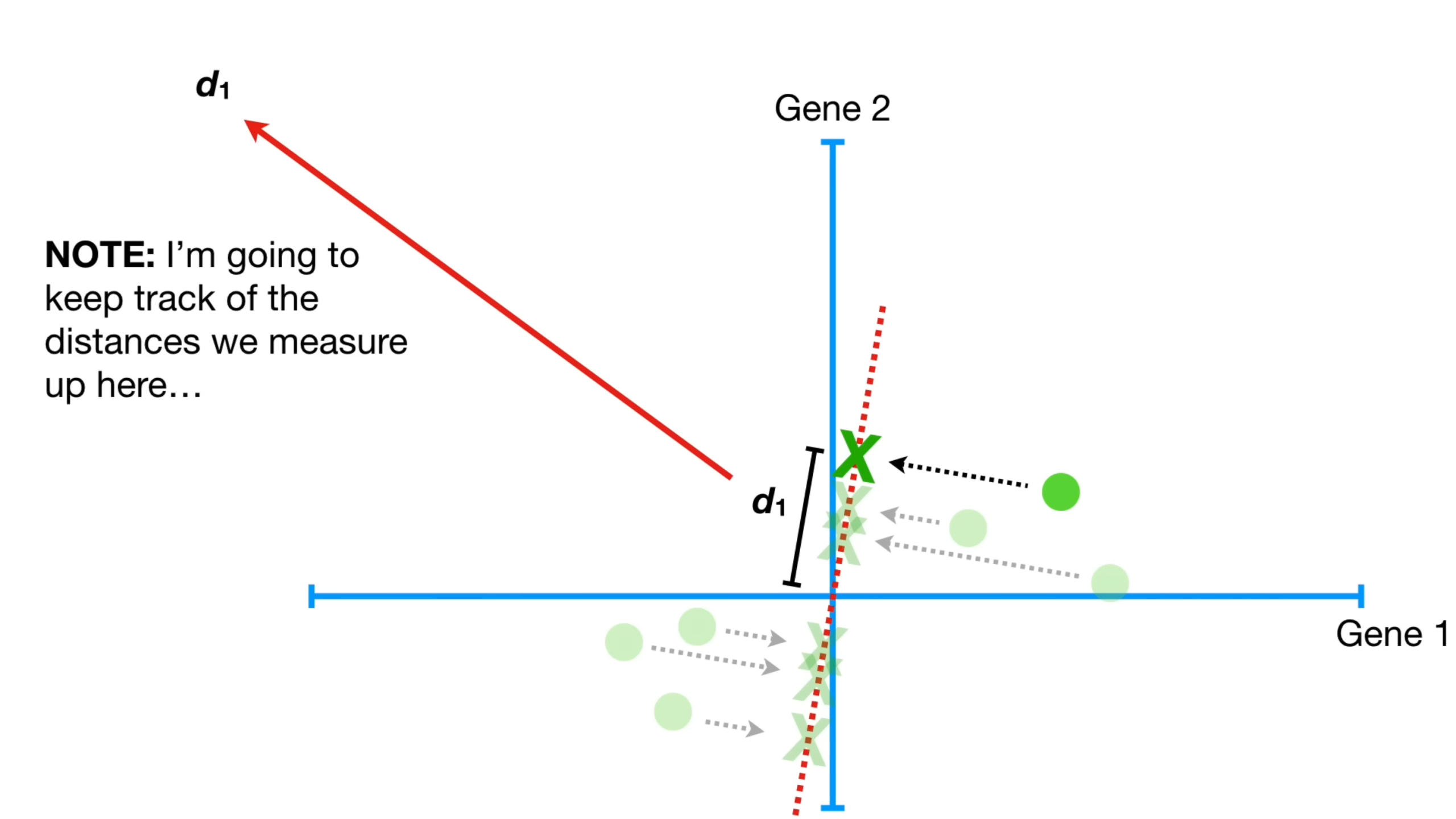


…but it's actually easier to calculate ***c***, the distance from the projected point to the origin, so PCA finds the best fitting line by **maximizing the sum of the squared distances from the projected points to the origin**.



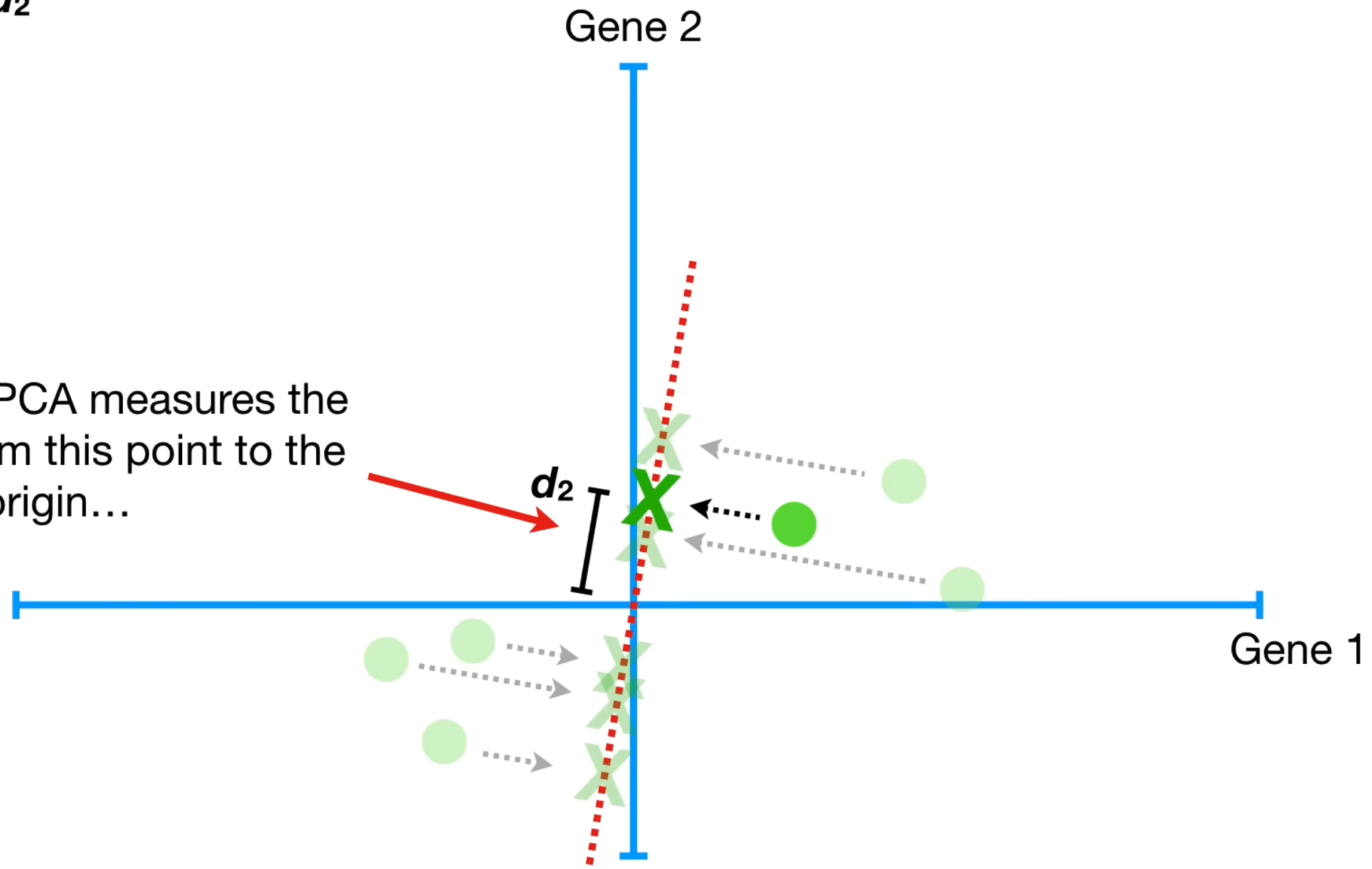

Gene 2
…PCA projects the data onto it…
Gene 1


Gene 2
…and then measures the distance from this point to the origin (let's call it d_1)…
d_1
Gene 1

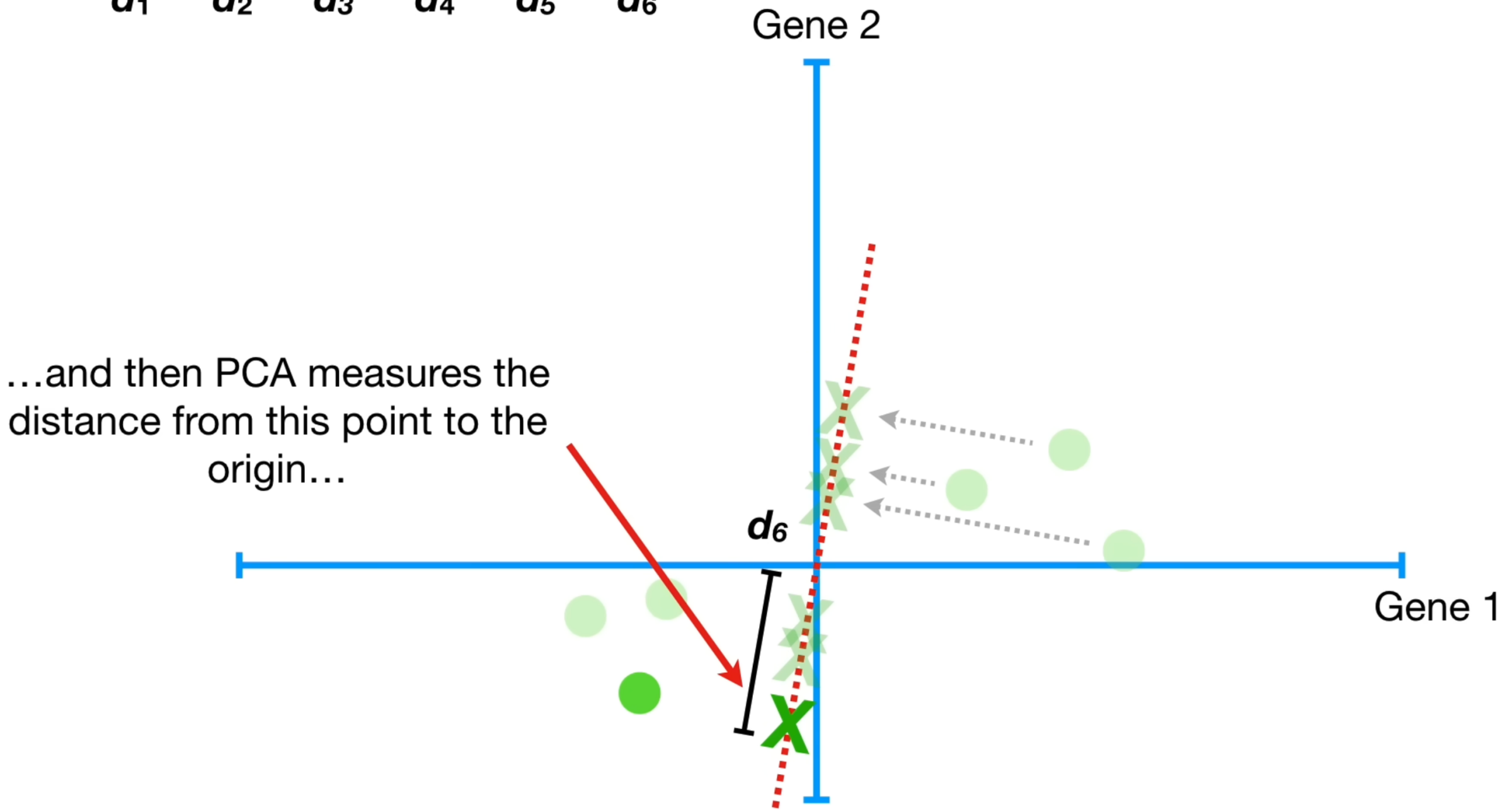

d_1
Gene 2
NOTE: I'm going to keep track of the distances we measure up here…
d_1
Gene 1

d_1 d_2

…and then PCA measures the distance from this point to the origin…

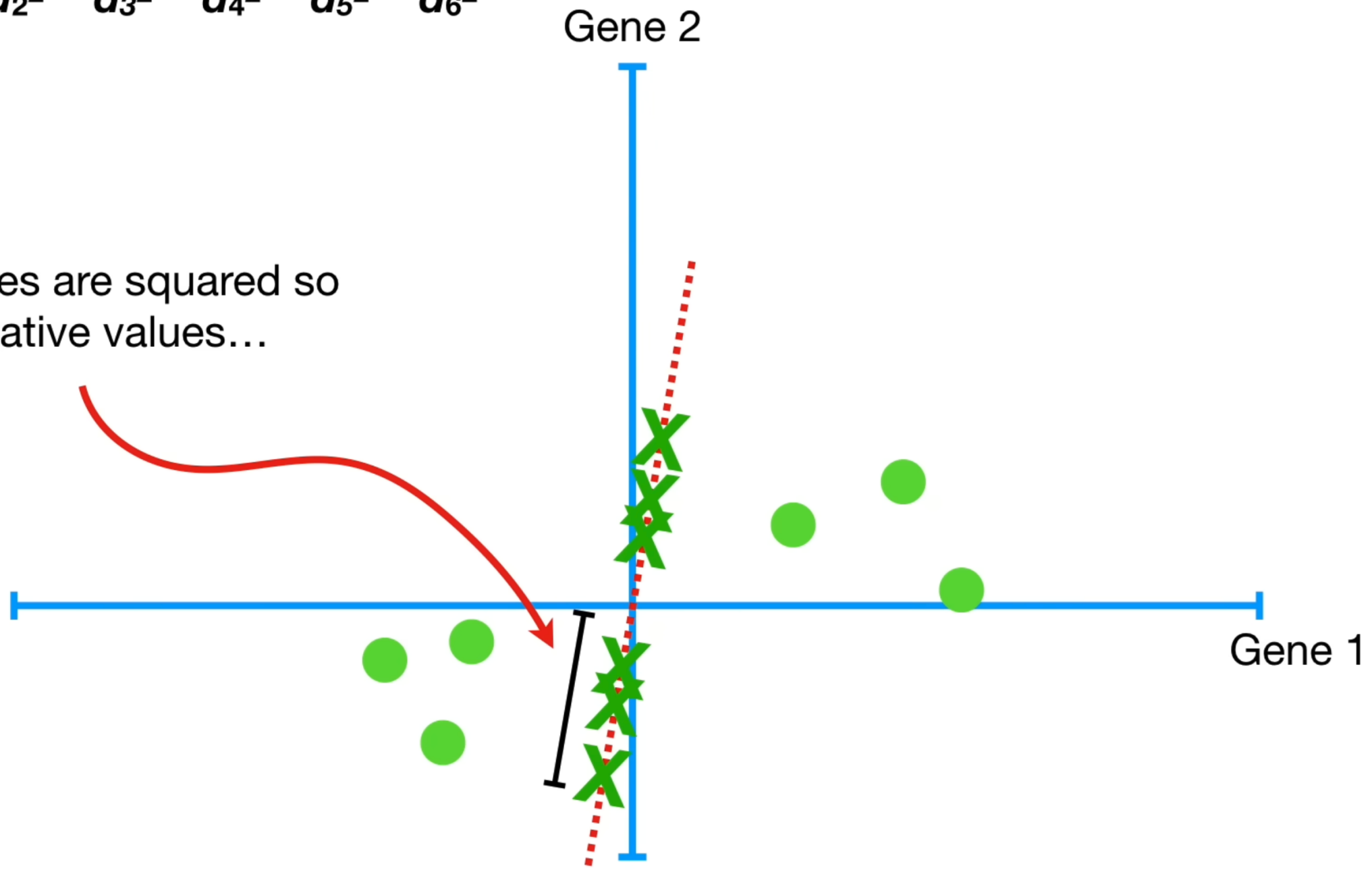


d_1 d_2 d_3 d_4 d_5 d_6



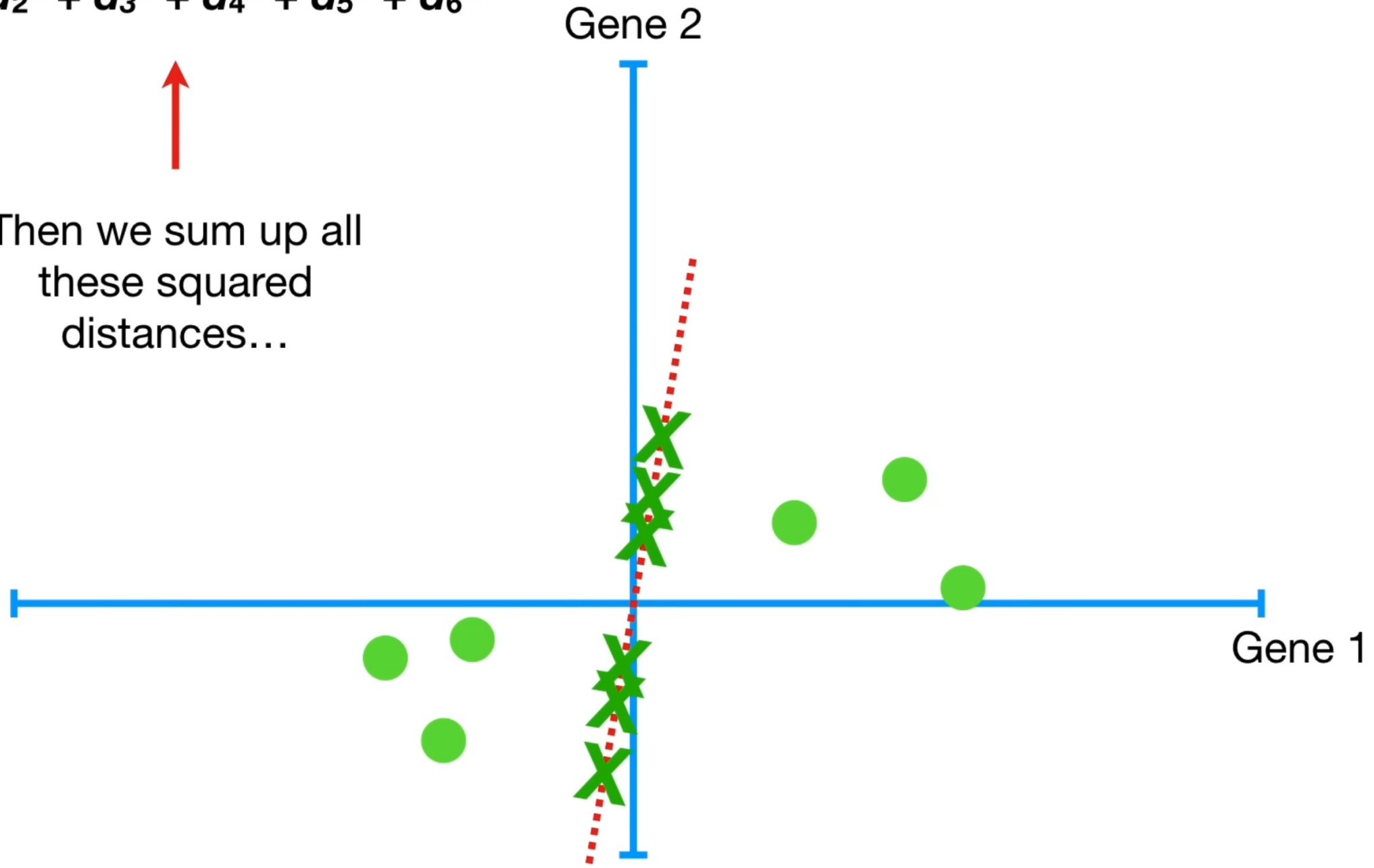
d_1^2 d_2^2 d_3^2 d_4^2 d_5^2 d_6^2

The distances are squared so that negative values…

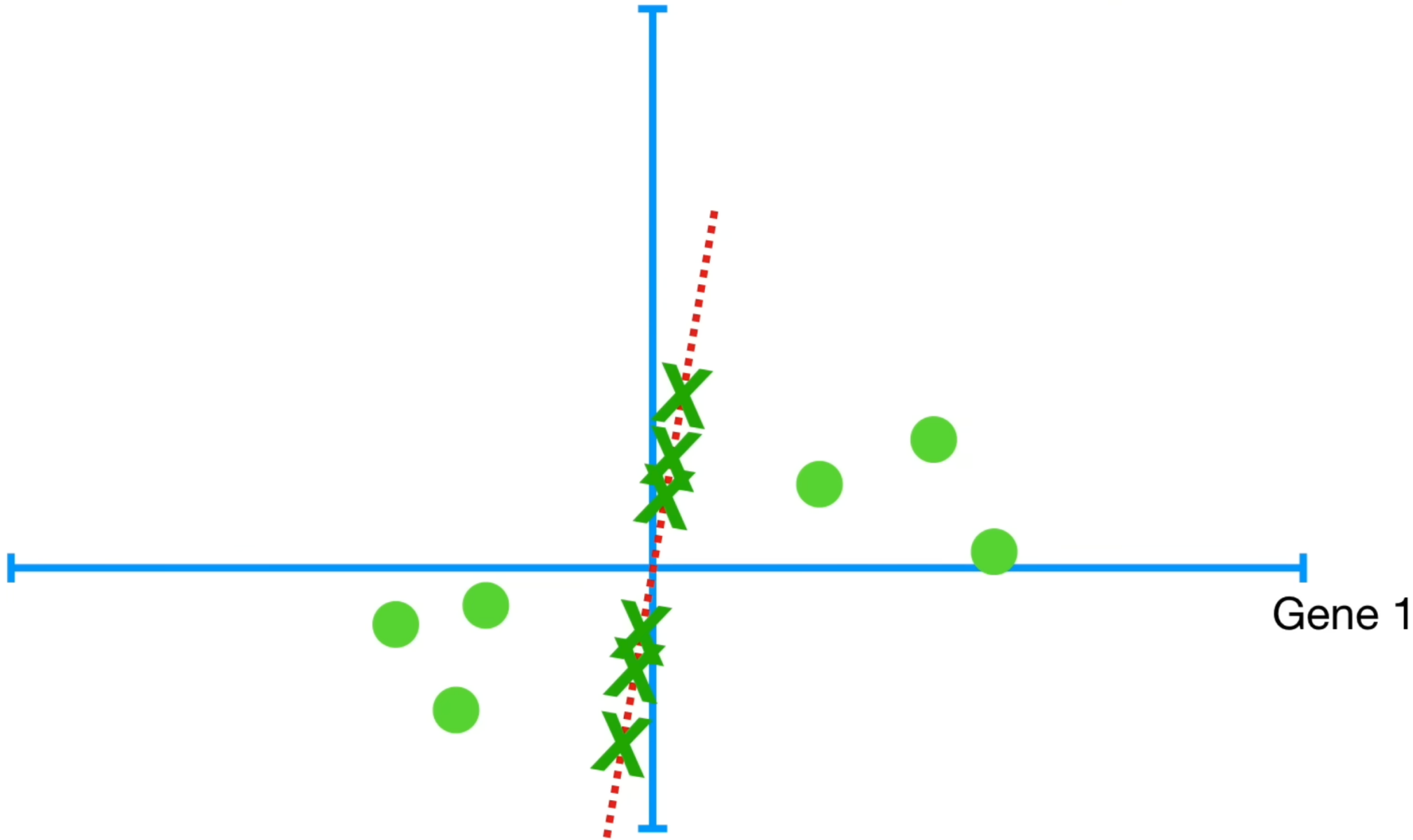


$$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2$$

Then we sum up all these squared distances…

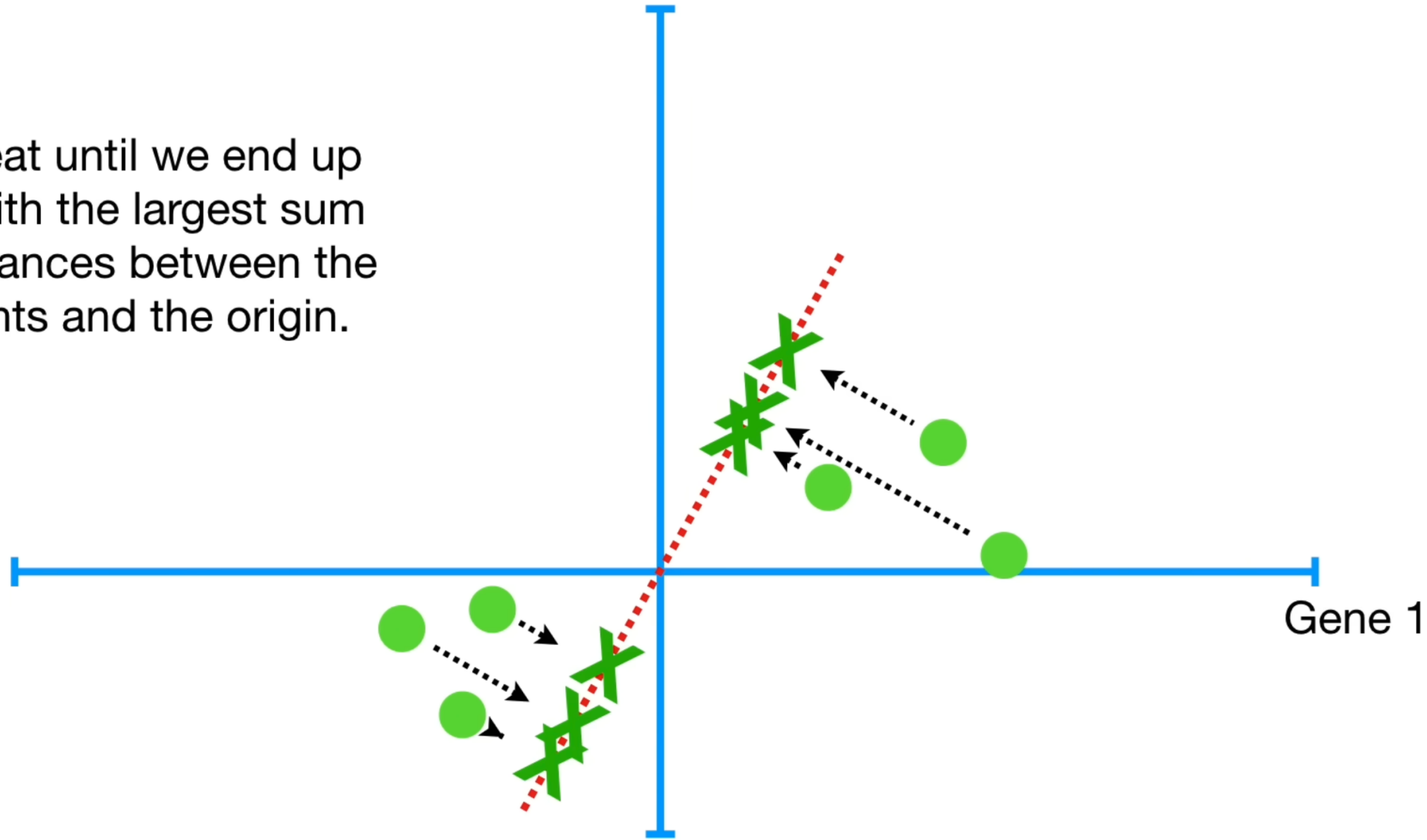


$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2$ = sum of squared distances = SS(distances)



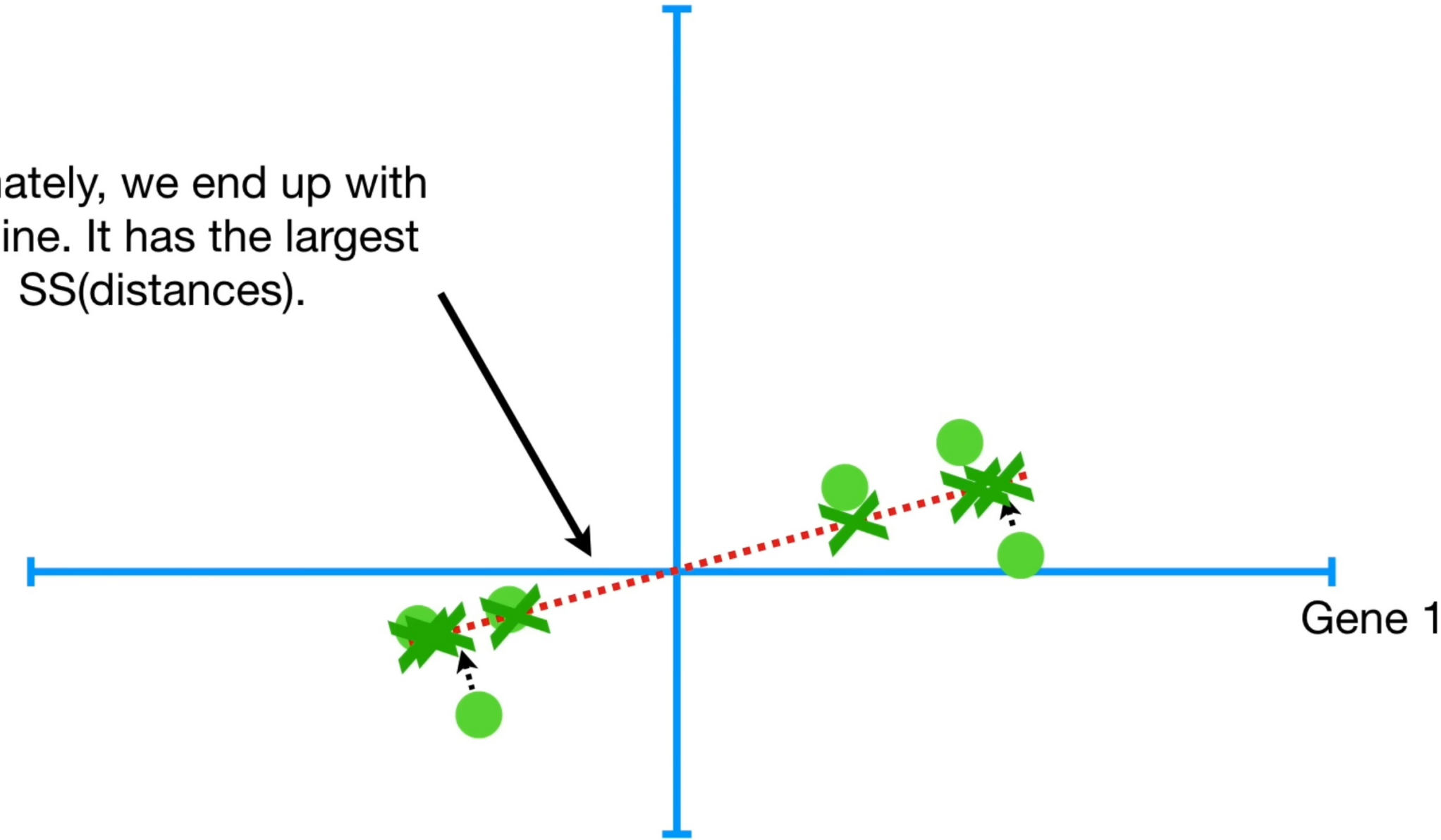
$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2$ = sum of squared distances = SS(distances)

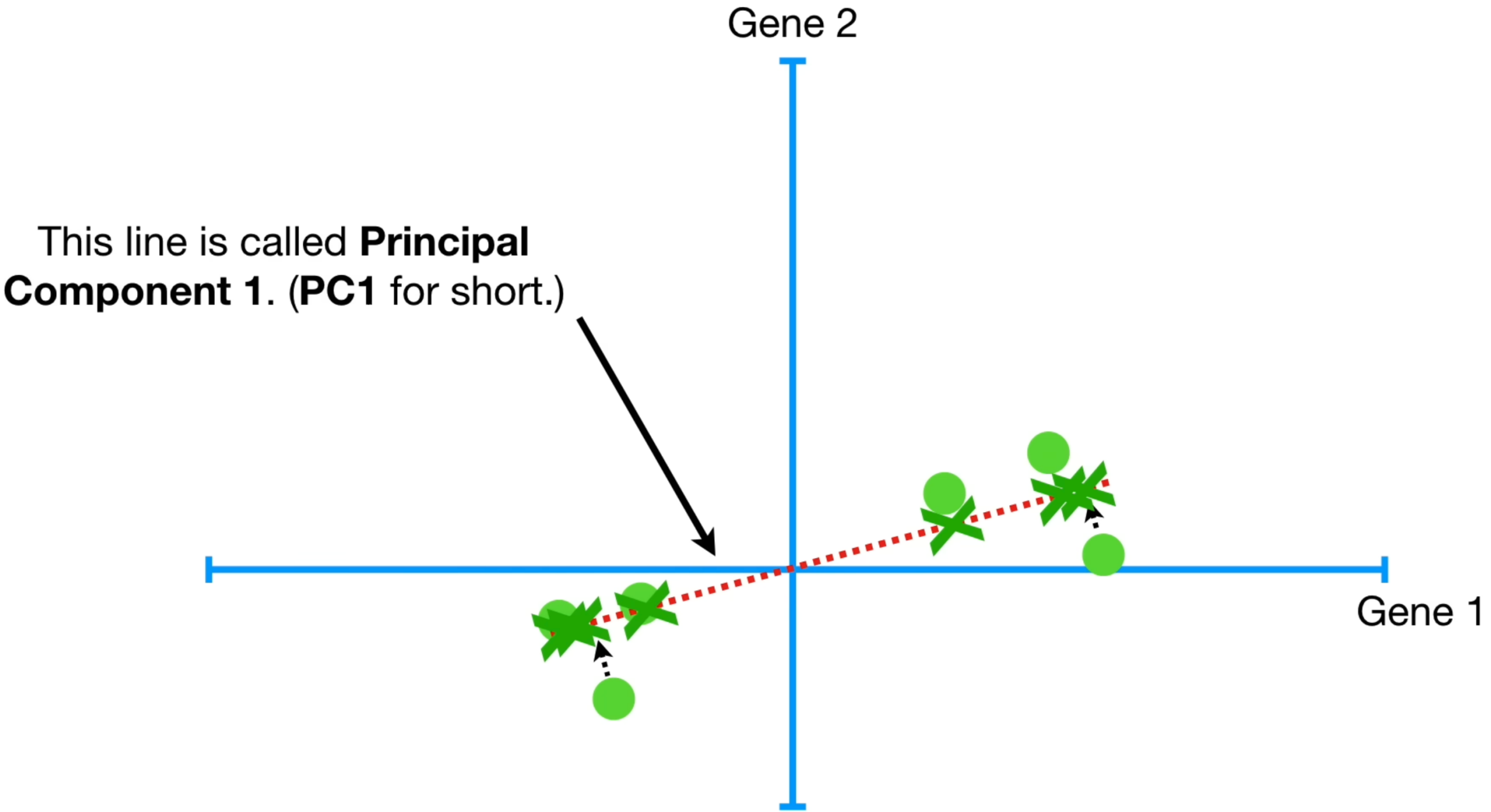
…and we repeat until we end up with the line with the largest sum of squared distances between the projected points and the origin.

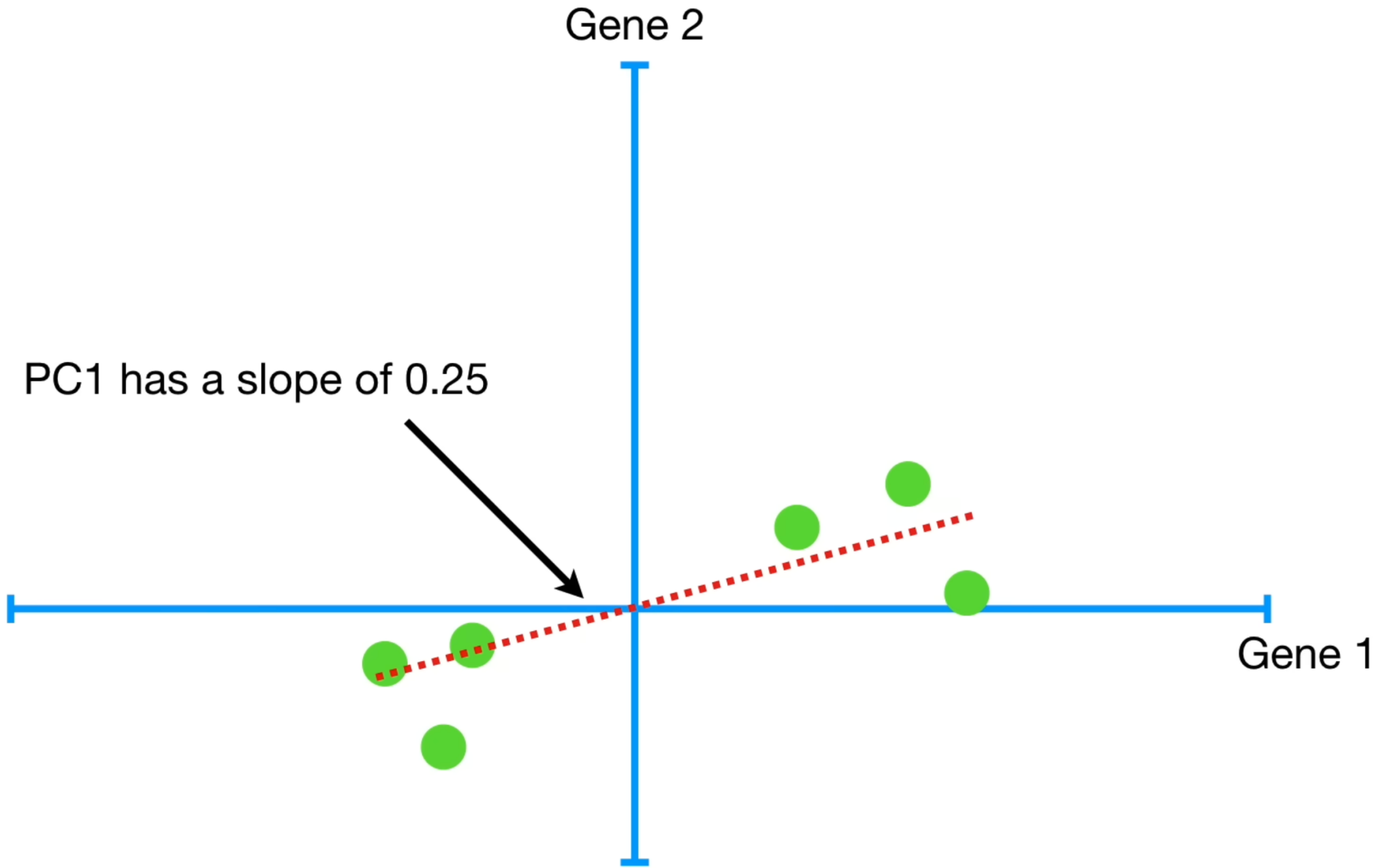


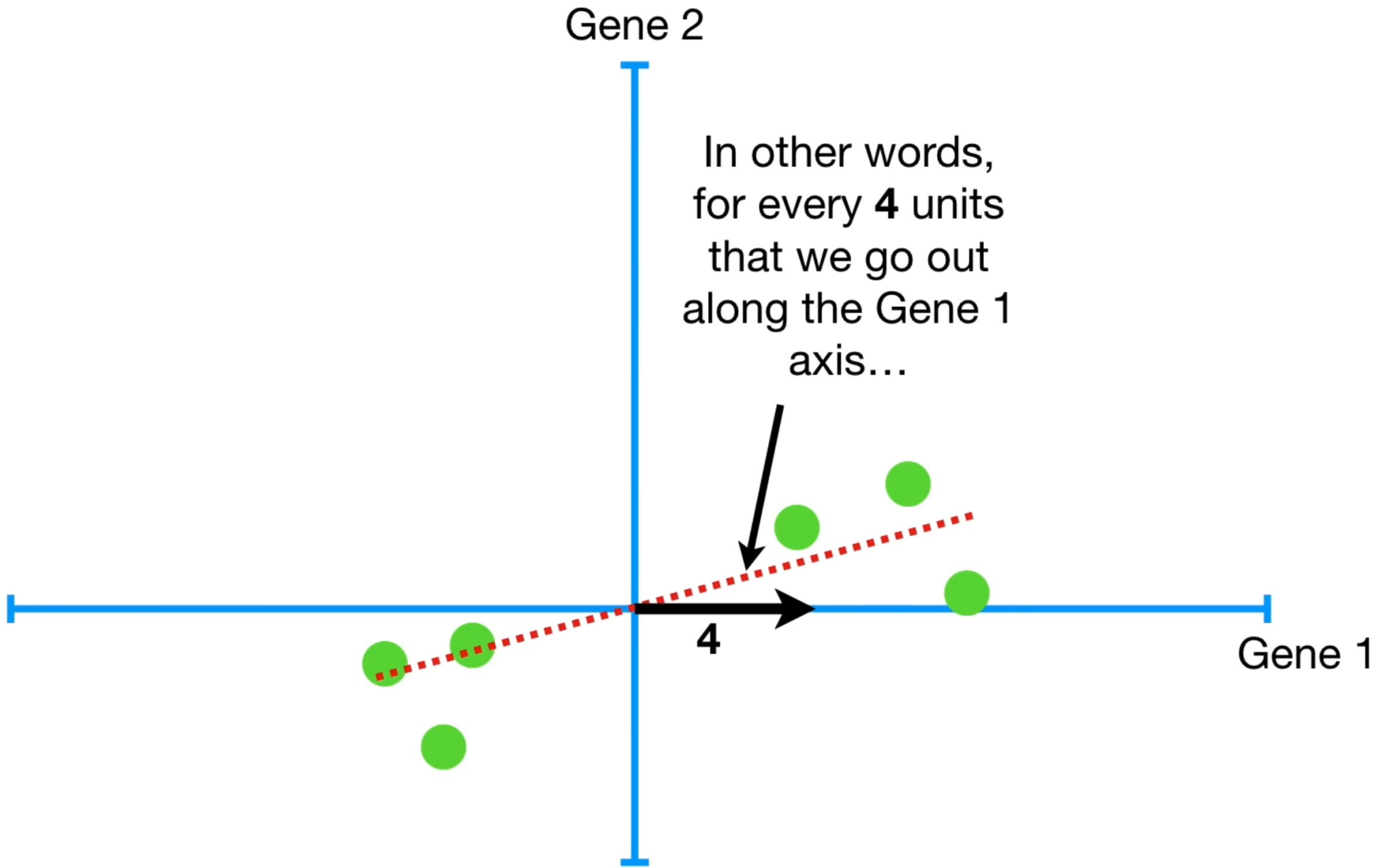
$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2$ = sum of squared distances = SS(distances)

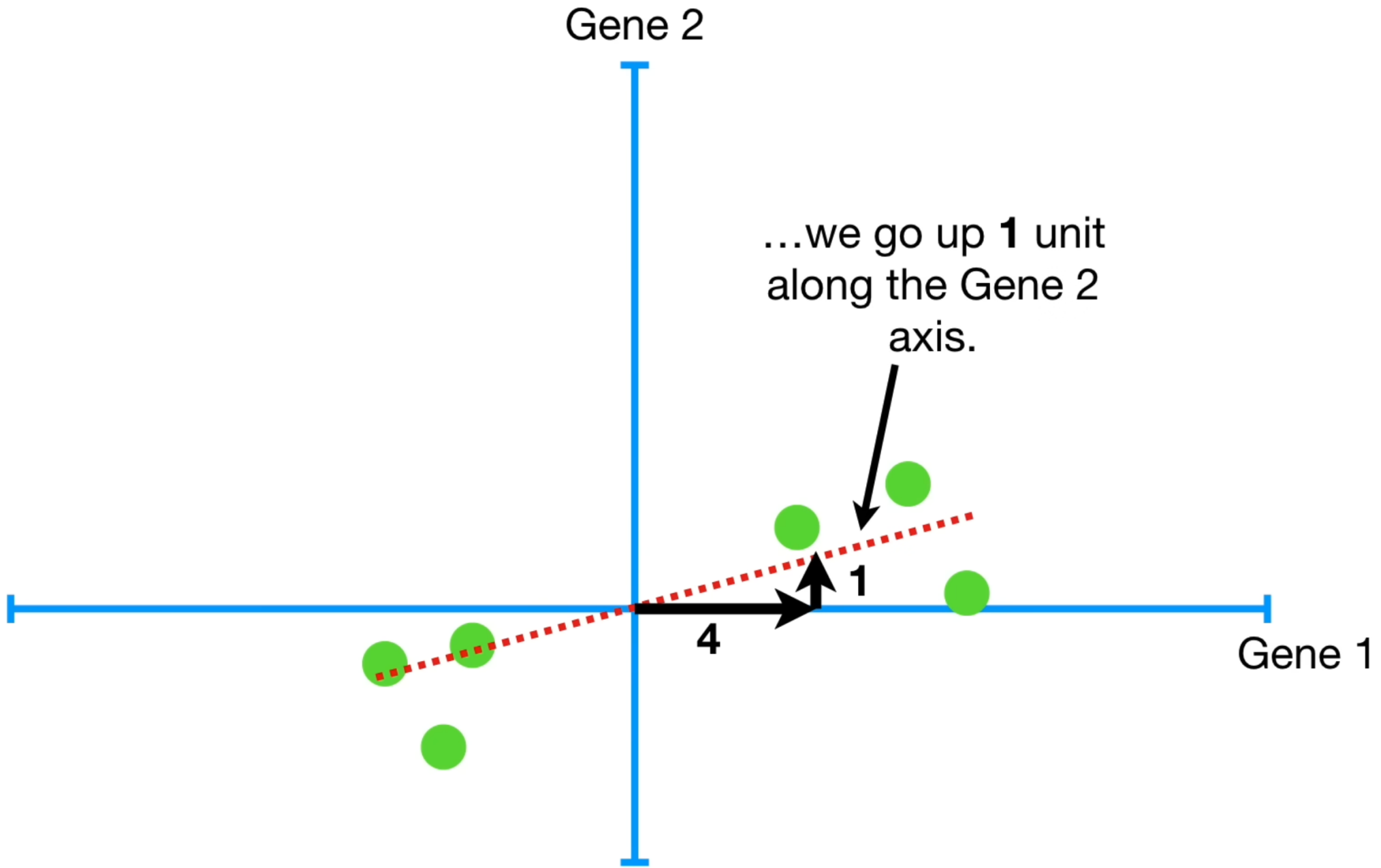
Ultimately, we end up with this line. It has the largest SS(distances).

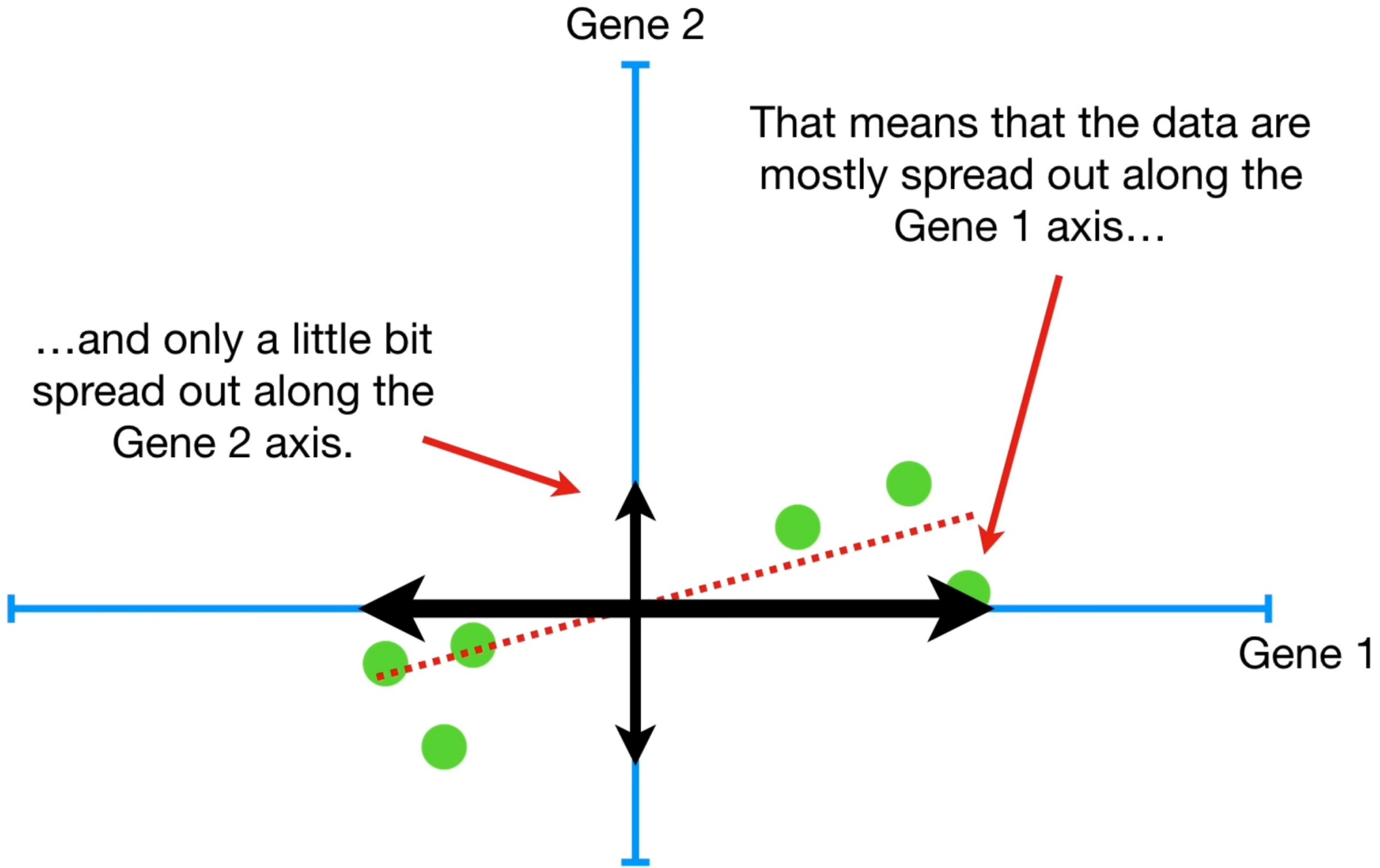



Gene 2
This line is called **Principal Component 1**. (**PC1** for short.)
Gene 1


Gene 2
PC1 has a slope of 0.25
Gene 1


Gene 2
In other words,
for every 4 units
that we go out
along the Gene 1
axis…
4
Gene 1

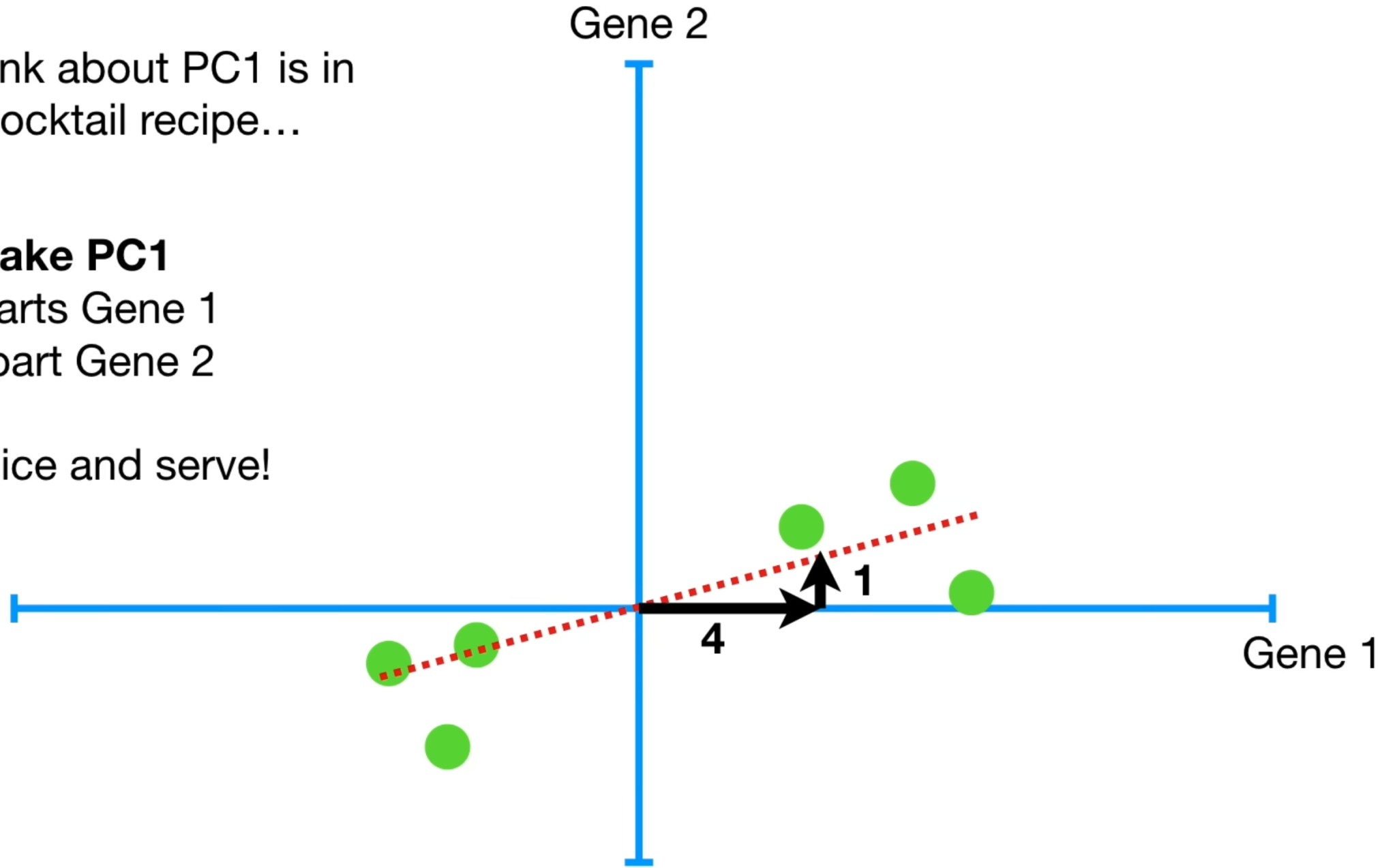

Gene 2
…we go up 1 unit along the Gene 2 axis.
1
4
Gene 1


Gene 2
That means that the data are mostly spread out along the Gene 1 axis…
…and only a little bit spread out along the Gene 2 axis.
Gene 1

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

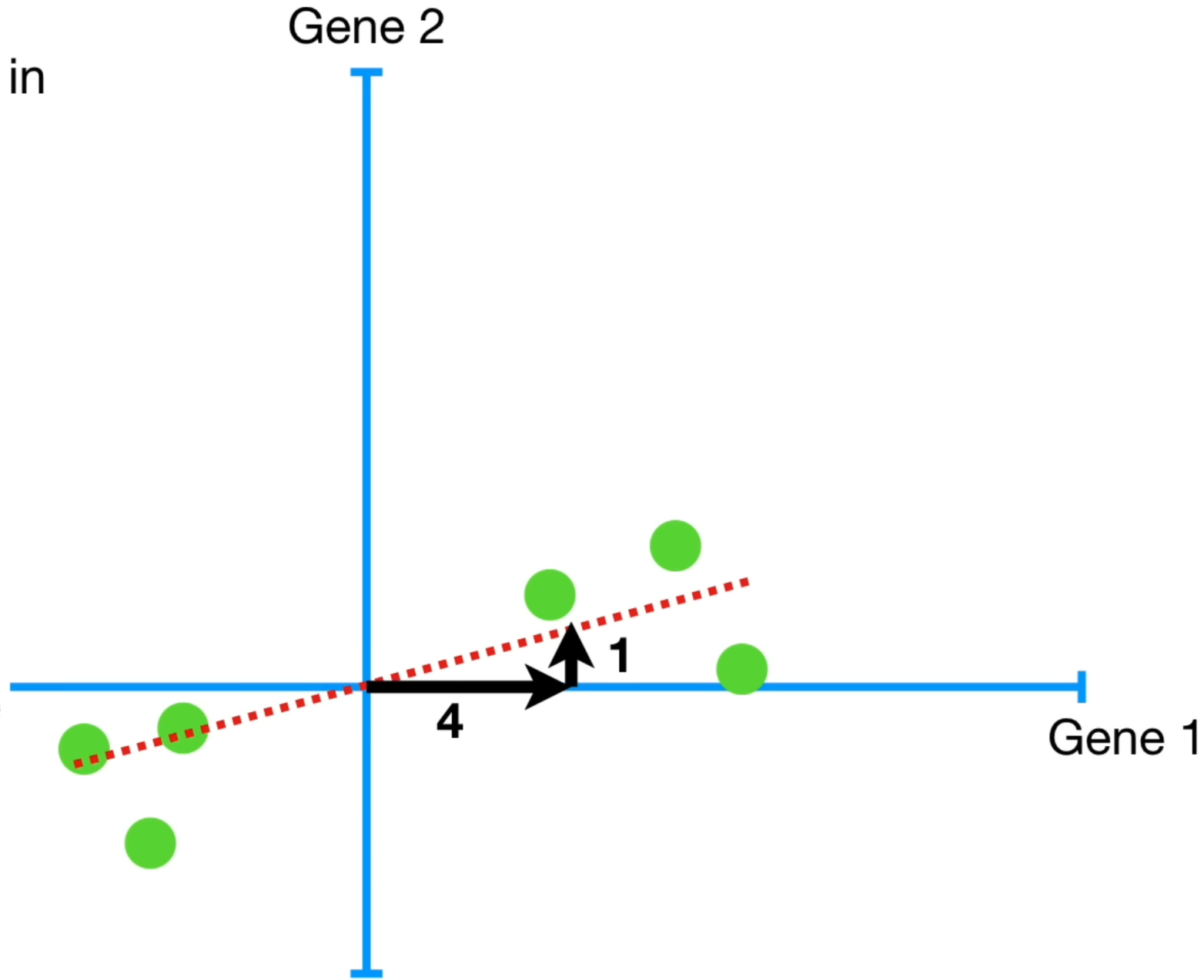
Pour over ice and serve!



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

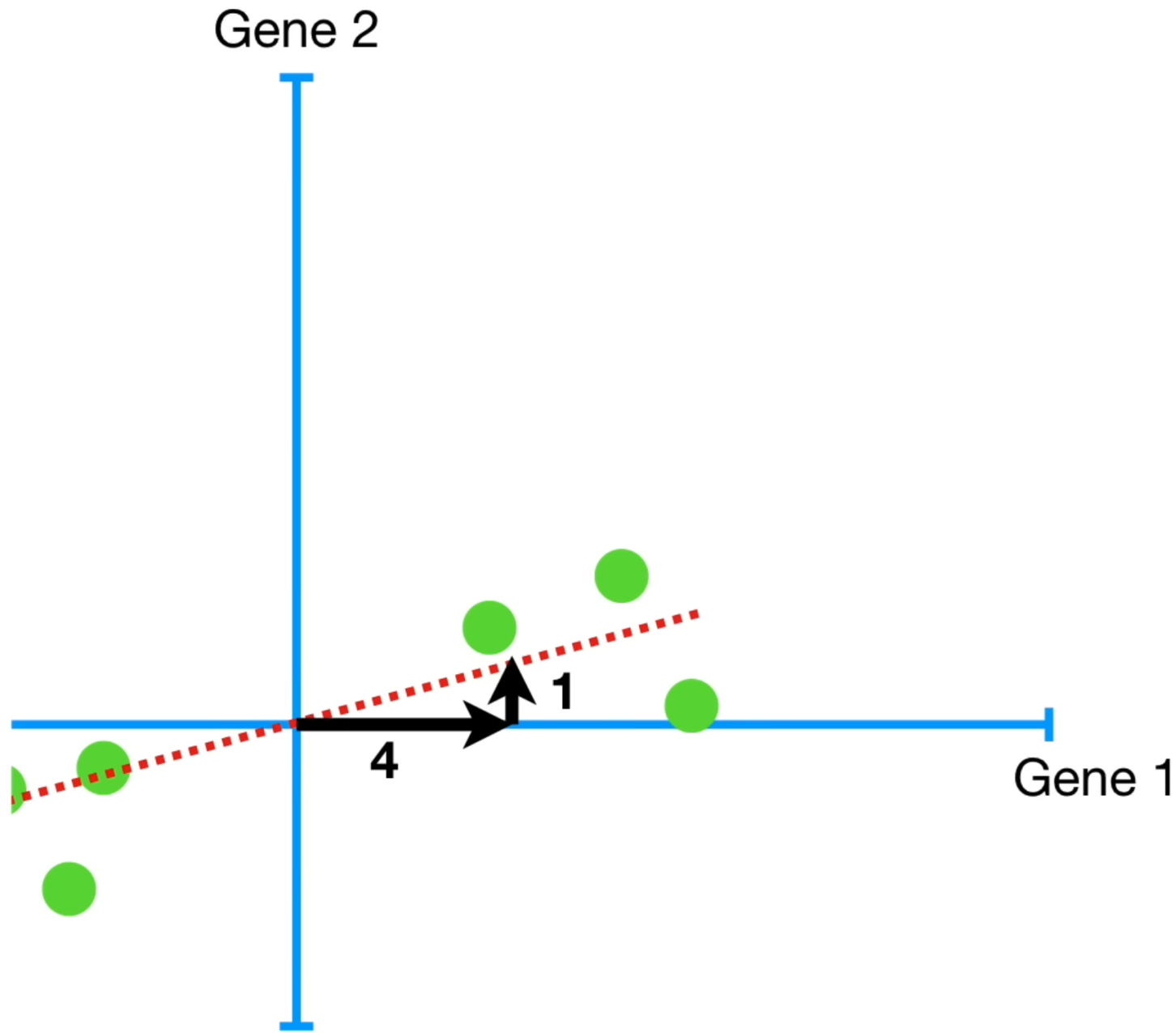
To make PC1
Mix **4** parts Gene 1
with **1** part Gene 2

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..

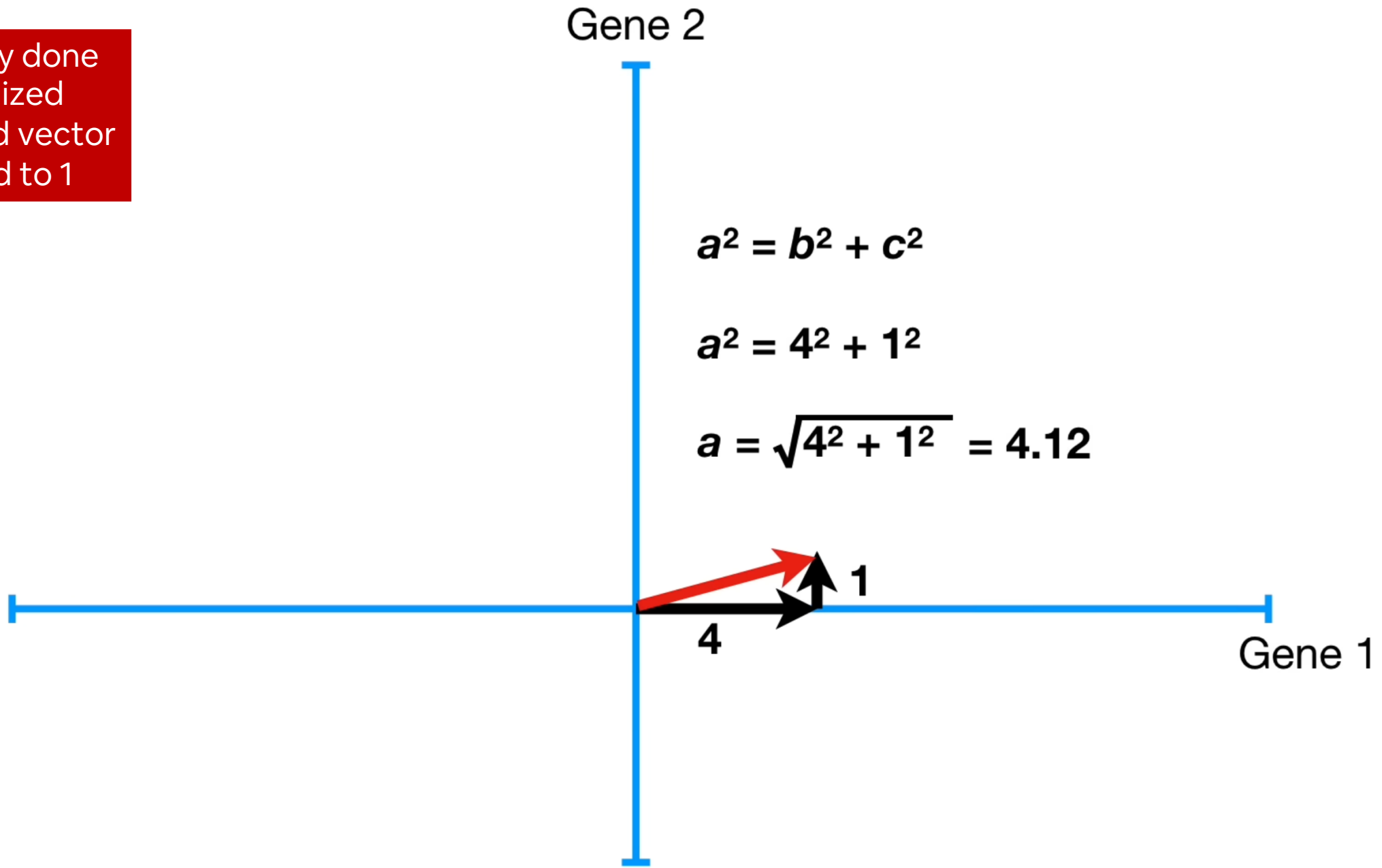


To make PC1
Mix **4** parts Gene 1
with **1** part Gene 2

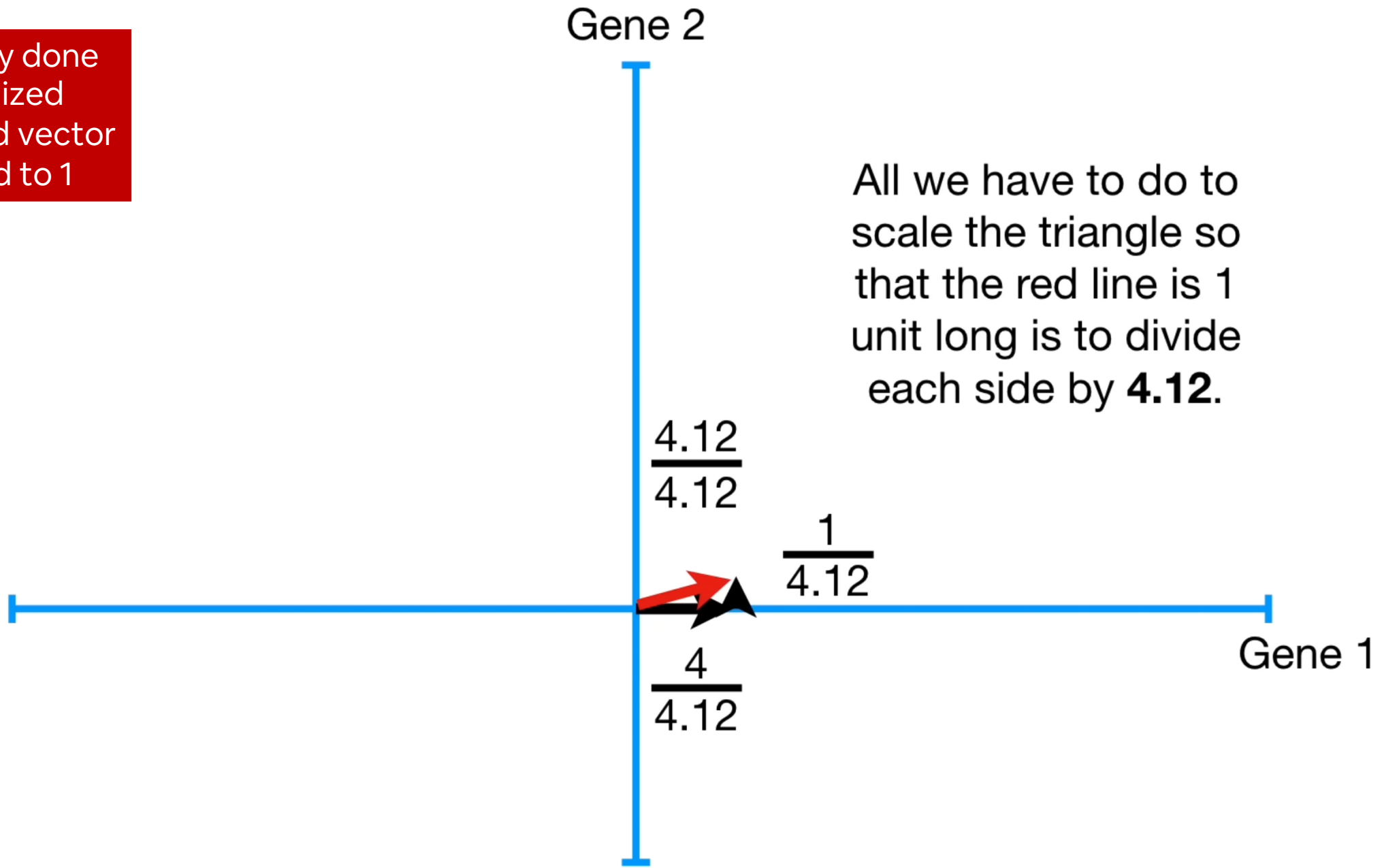
Terminology Alert!!!!
Mathematicians call this cocktail recipe a "*linear combination*" of Genes 1 and 2.



PCA is formally done with normalized vectors: the red vector is normalized to 1



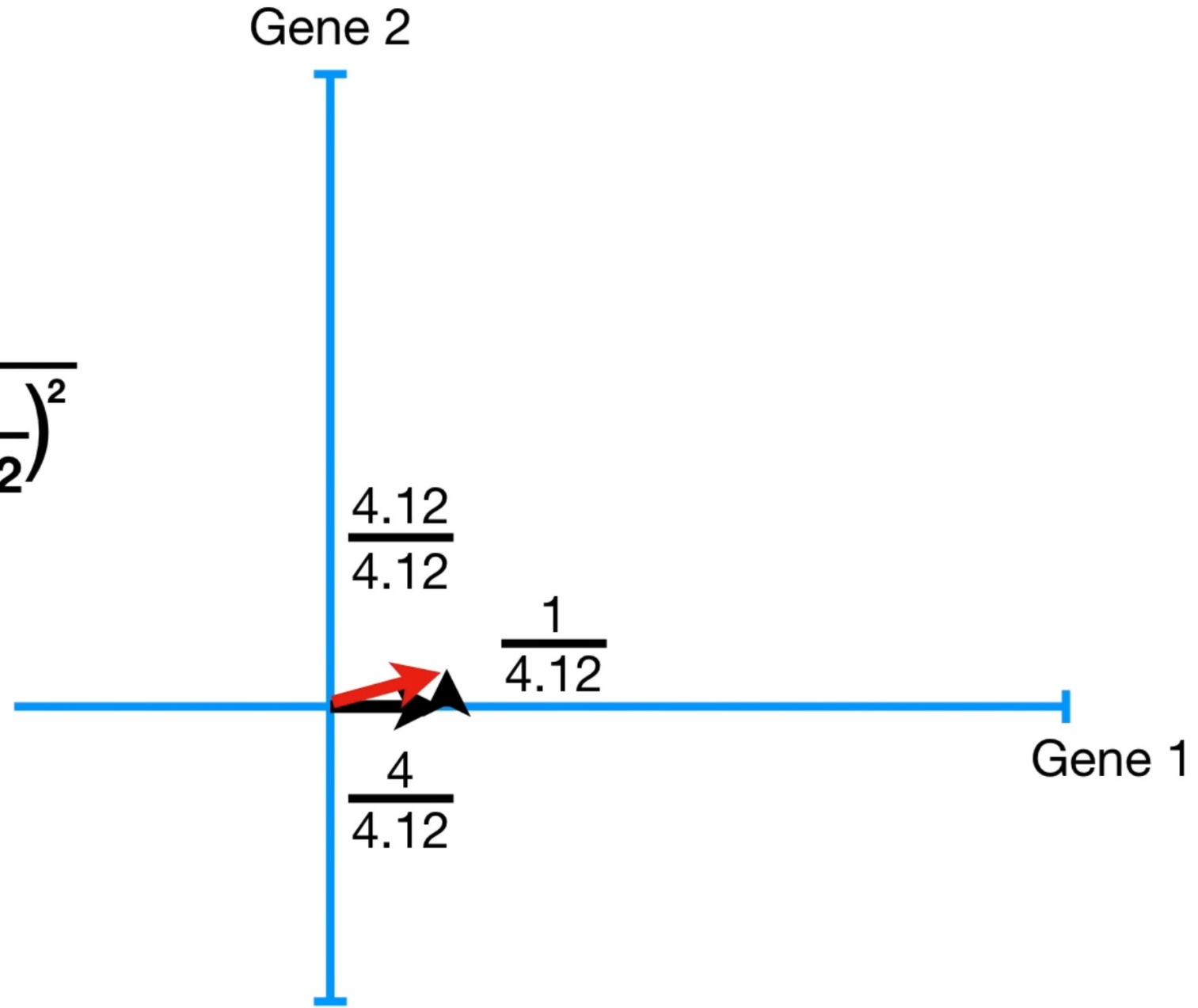
PCA is formally done with normalized vectors: the red vector is normalized to 1



$$\frac{4.12}{4.12} = \frac{\sqrt{4^2 + 1^2}}{4.12} = \sqrt{\left(\frac{4^2 + 1^2}{4.12^2}\right)}$$

$$= \sqrt{\left(\frac{4}{4.12}\right)^2 + \left(\frac{1}{4.12}\right)^2}$$

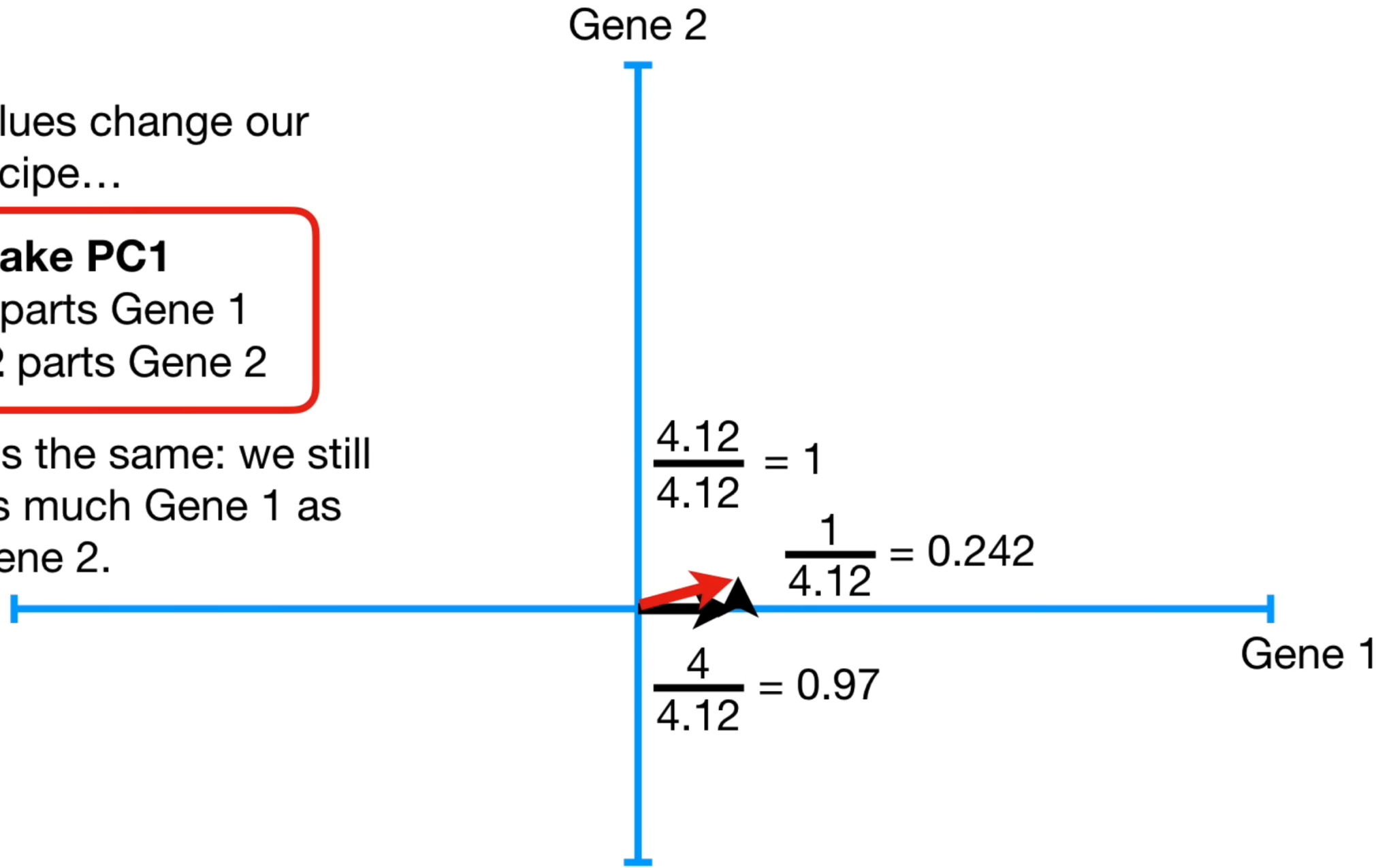
For those of you keeping score, here's the math worked out that shows that all we need to do is divide all 3 sides by **4.12**.

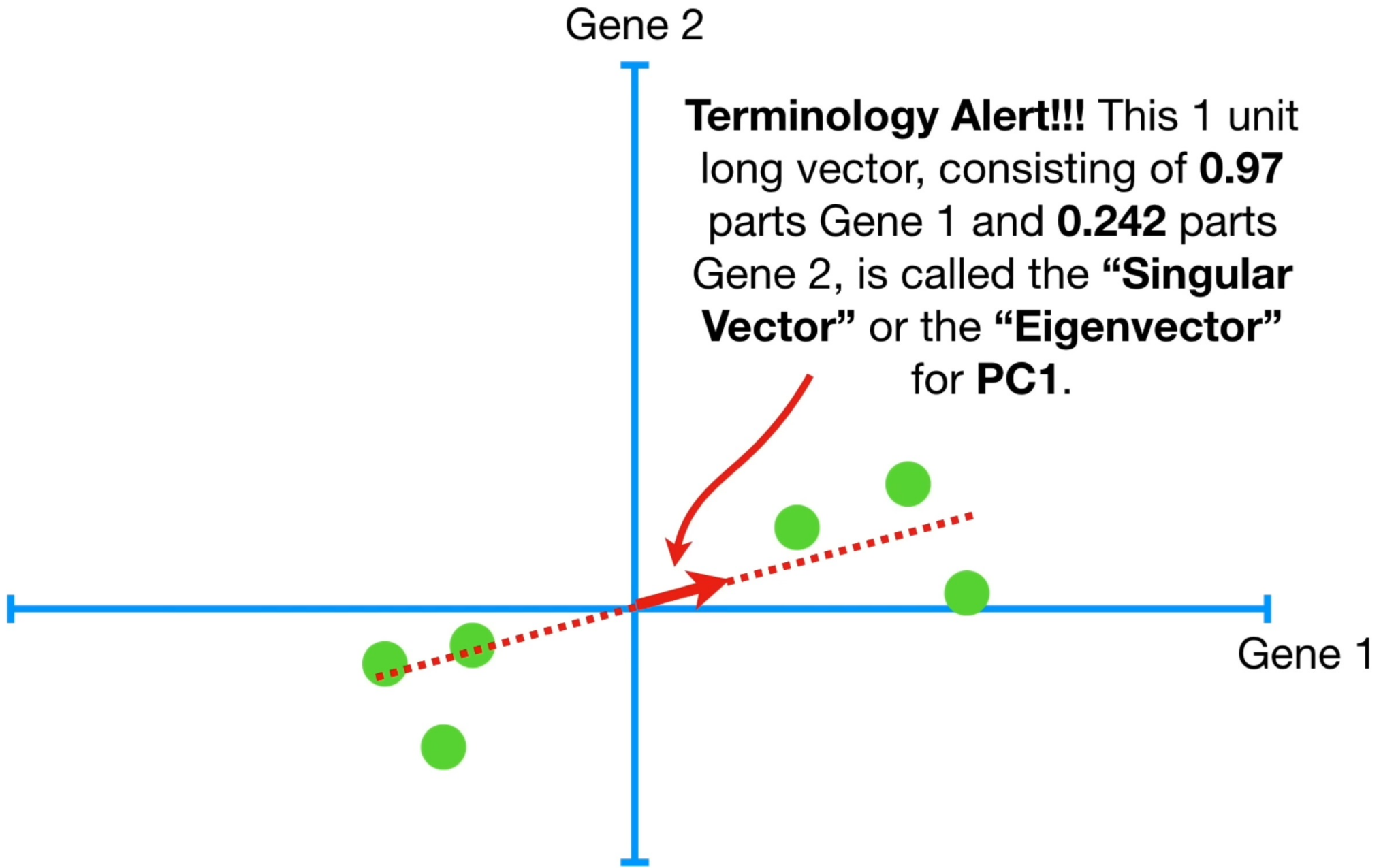


The new values change our recipe…

To make PC1
Mix **0.97** parts Gene 1
with **0.242** parts Gene 2

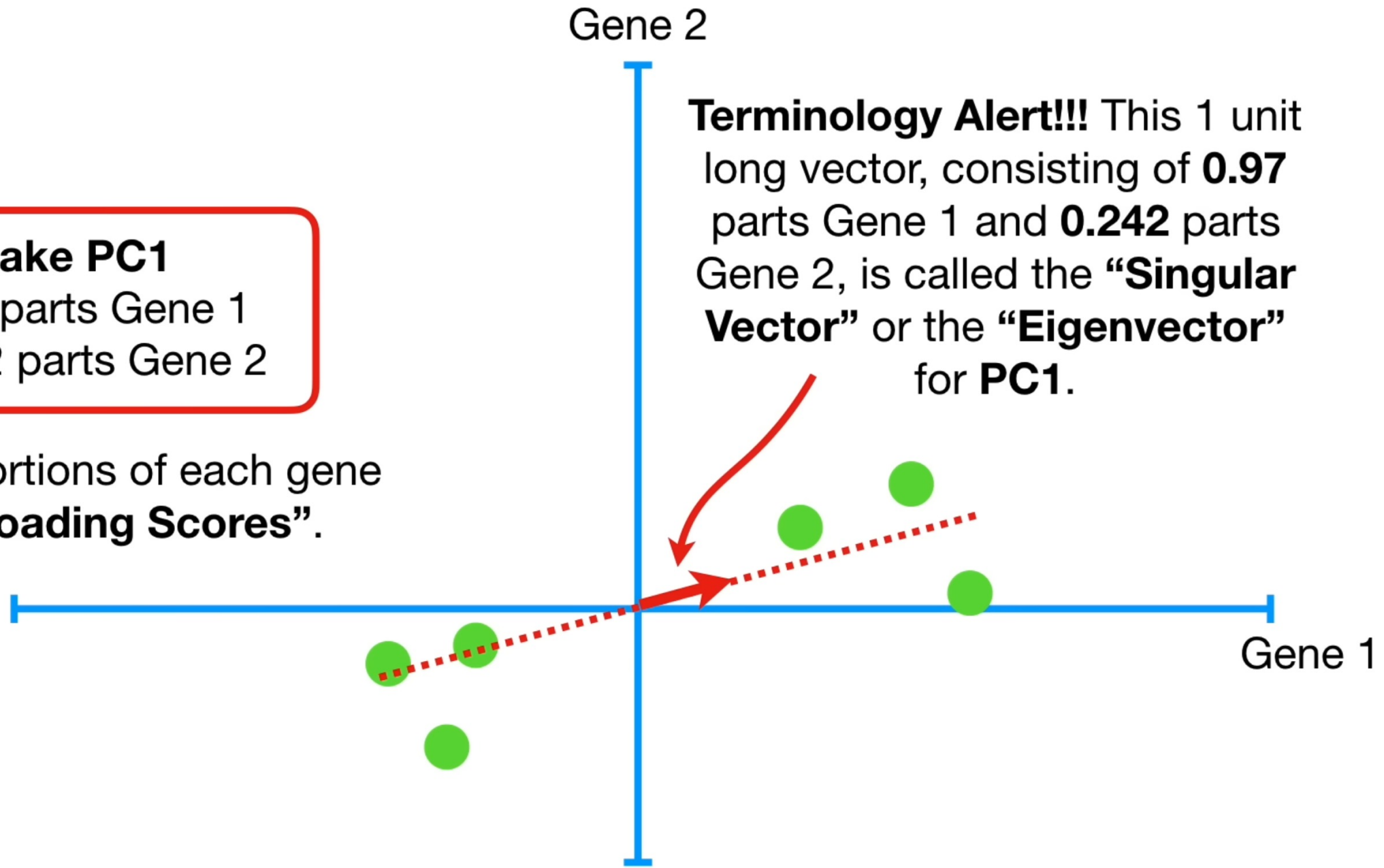
…but the ratio is the same: we still use 4 times as much Gene 1 as Gene 2.




Gene 2
Terminology Alert!!! This 1 unit long vector, consisting of **0.97** parts Gene 1 and **0.242** parts Gene 2, is called the **"Singular Vector"** or the **"Eigenvector"** for **PC1**.
Gene 1

To make PC1
Mix 0.97 parts Gene 1
with 0.242 parts Gene 2

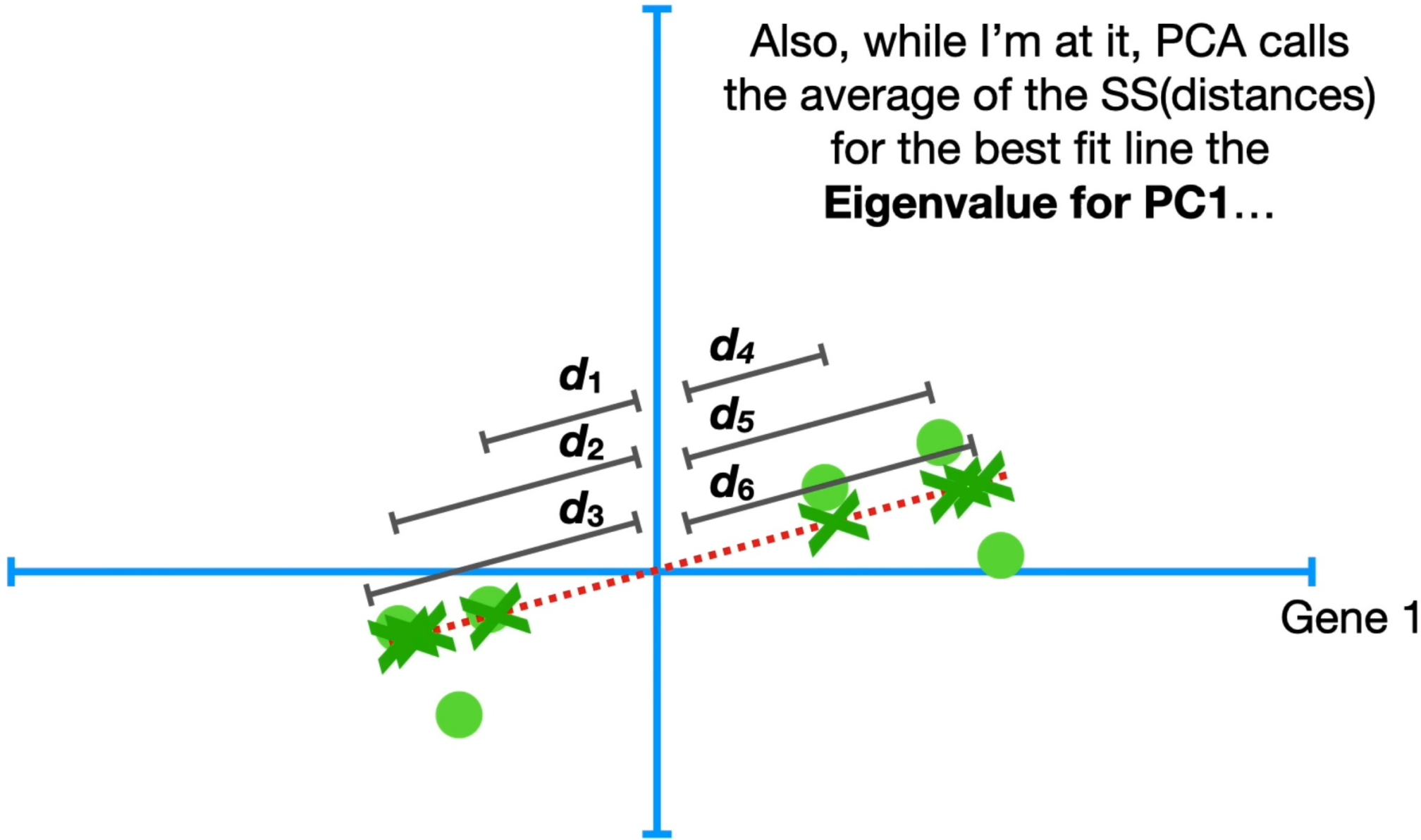
…and the proportions of each gene are called **“Loading Scores”**.



Terminology Alert!!! This 1 unit long vector, consisting of **0.97** parts Gene 1 and **0.242** parts Gene 2, is called the **“Singular Vector”** or the **“Eigenvector”** for **PC1**.

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

Also, while I'm at it, PCA calls the average of the SS(distances) for the best fit line the **Eigenvalue for PC1**…

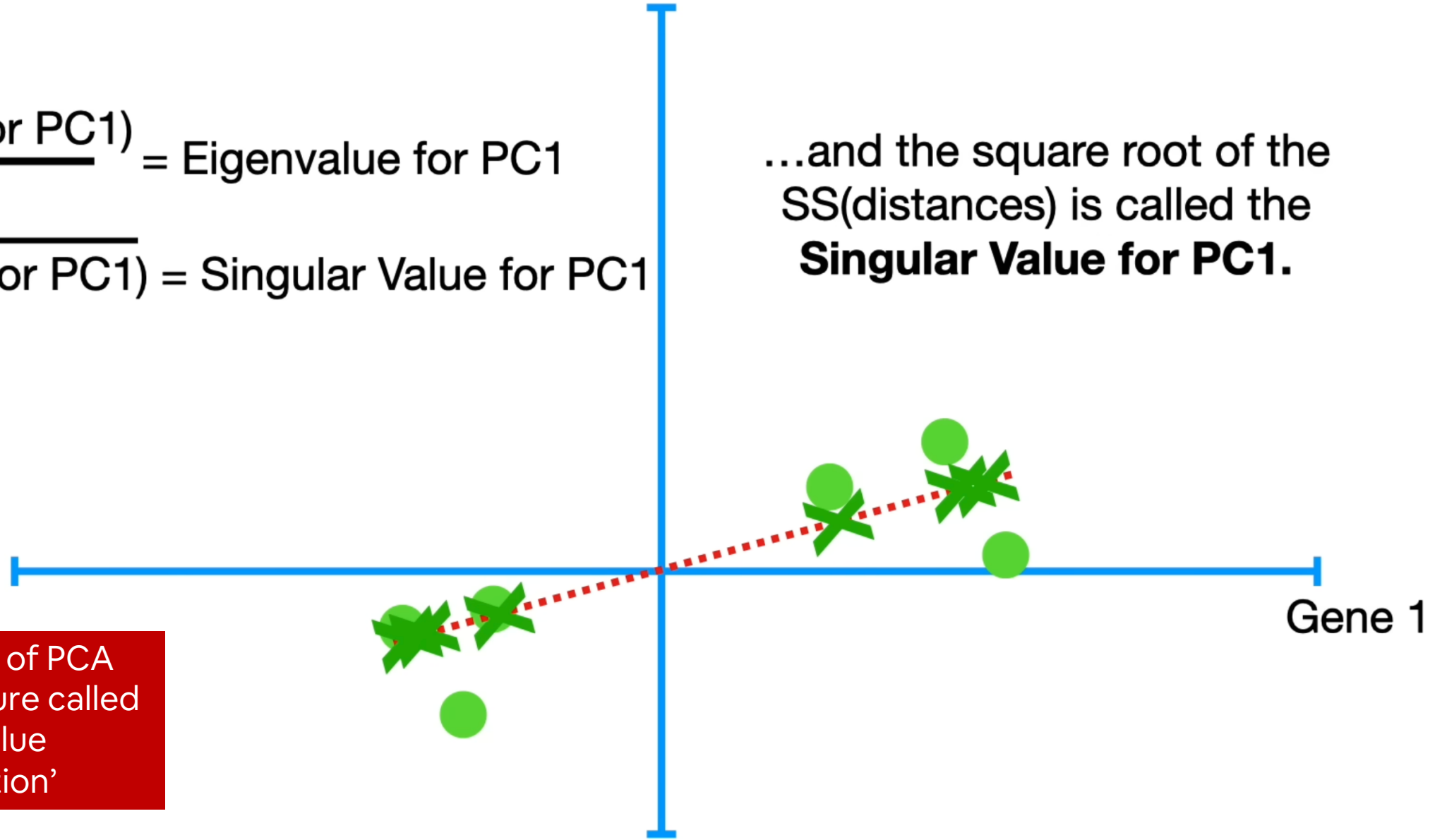


$$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2 = \text{sum of squared distances} = \text{SS(distances)}$$

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

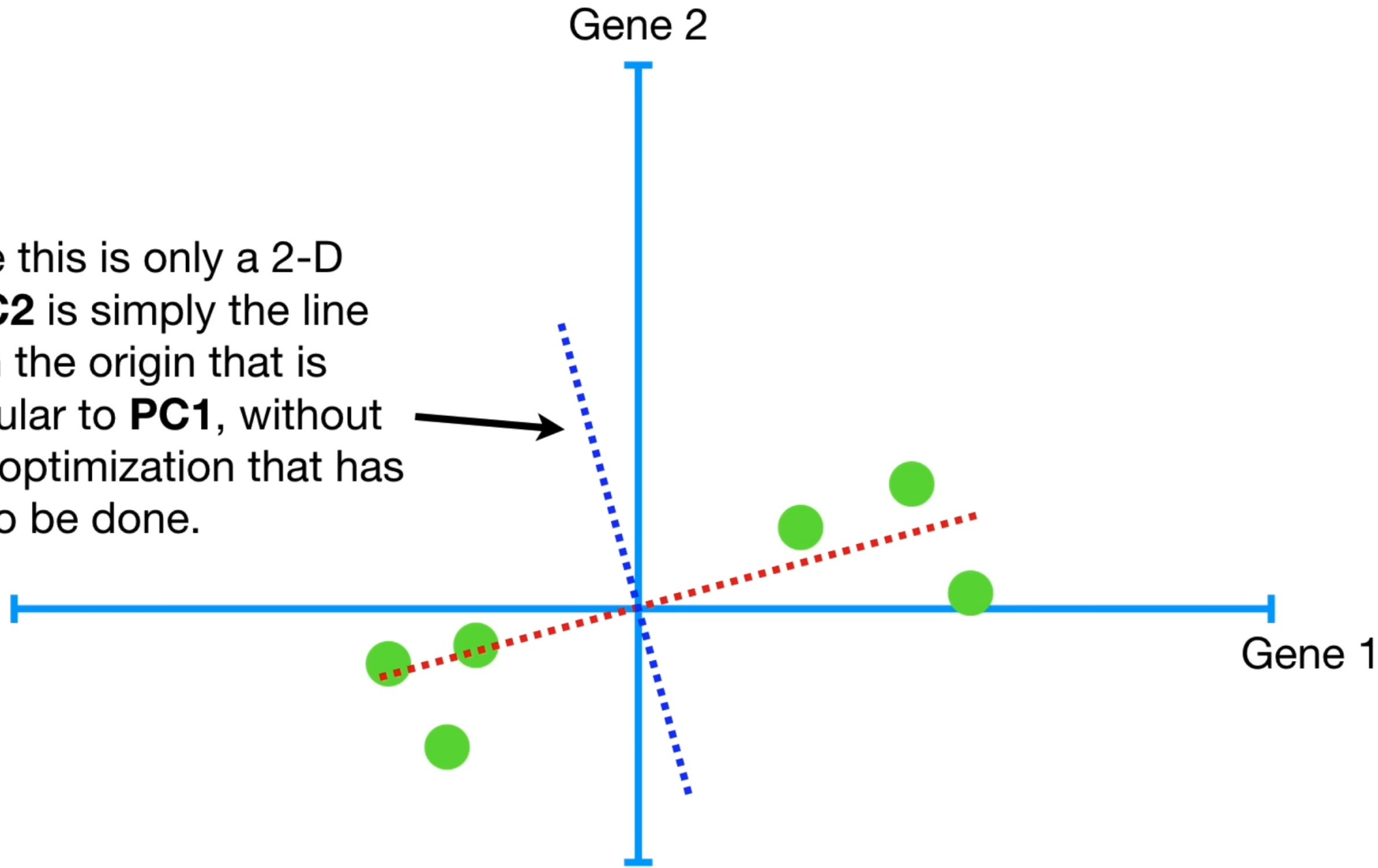
$$\sqrt{\text{SS(distances for PC1)}} = \text{Singular Value for PC1}$$

…and the square root of the SS(distances) is called the **Singular Value for PC1.**



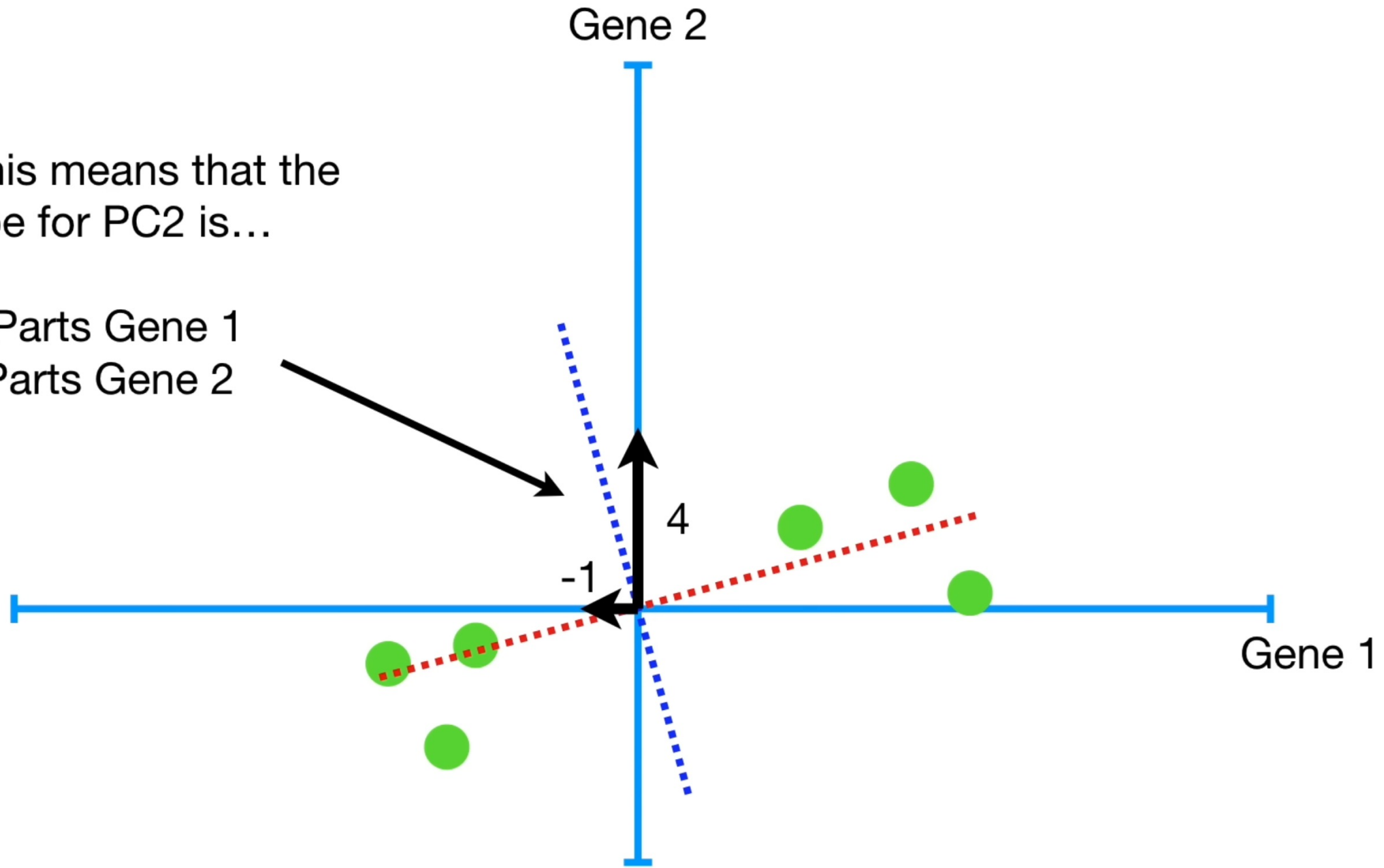
Under the hood of PCA there is a procedure called 'Singular Value Decomposition'

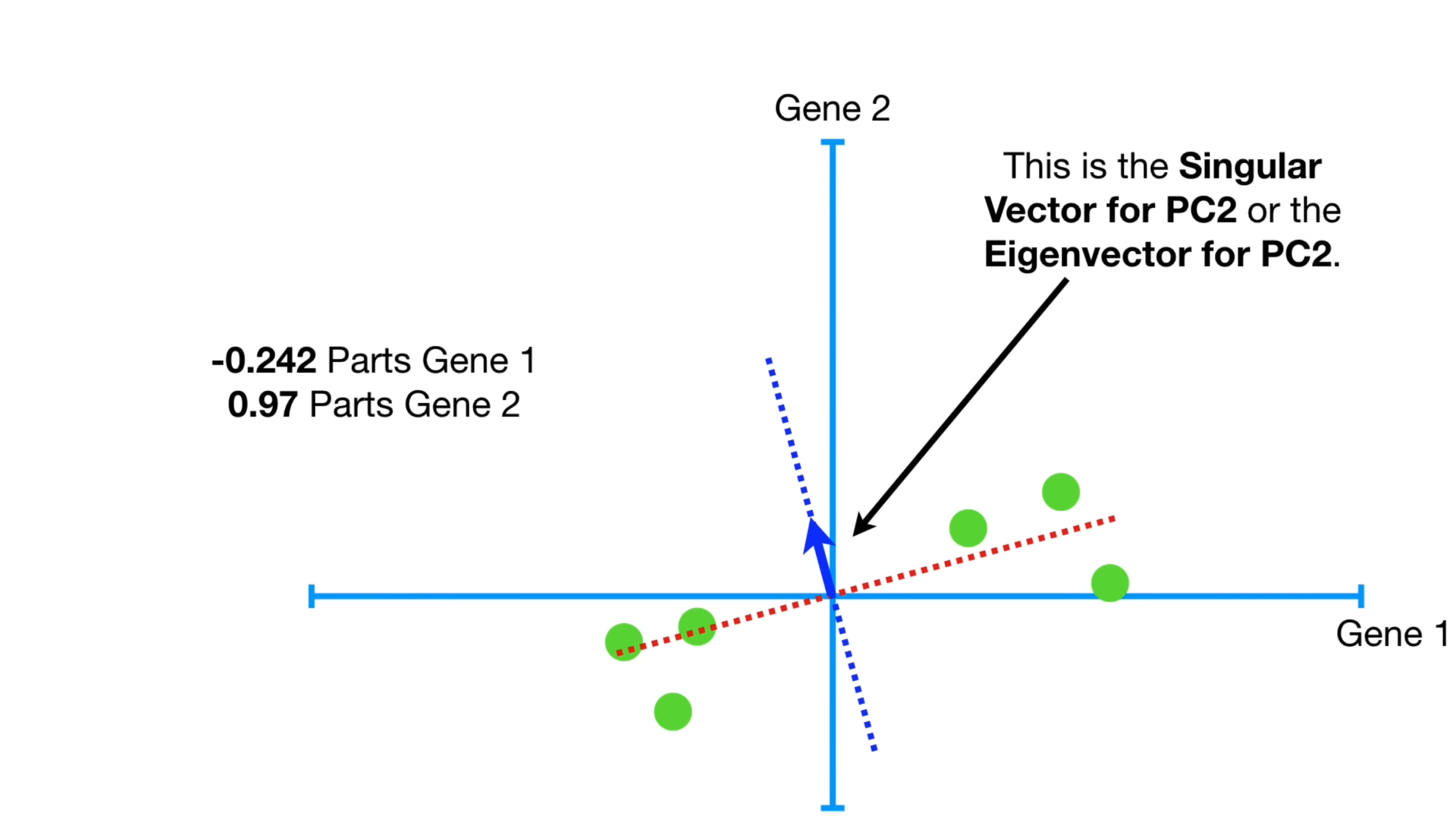
Because this is only a 2-D graph, **PC2** is simply the line through the origin that is perpendicular to **PC1**, without any further optimization that has to be done.



…and this means that the recipe for PC2 is…

-1 Parts Gene 1
4 Parts Gene 2

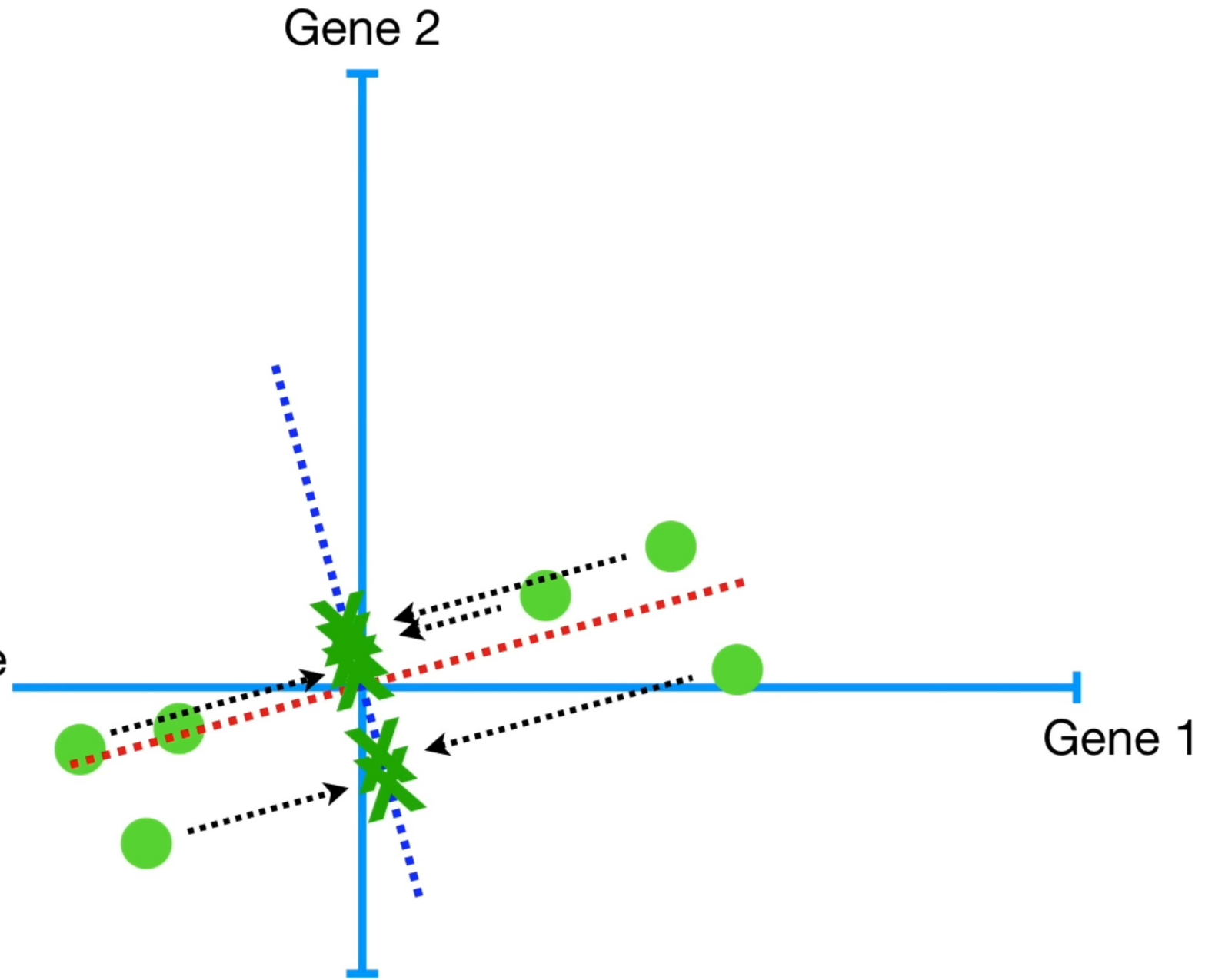



Gene 2
This is the **Singular Vector for PC2** or the **Eigenvector for PC2**.
-0.242 Parts Gene 1
0.97 Parts Gene 2
Gene 1

These are the **Loading Scores for PC2**.

-0.242 Parts Gene 1
0.97 Parts Gene 2

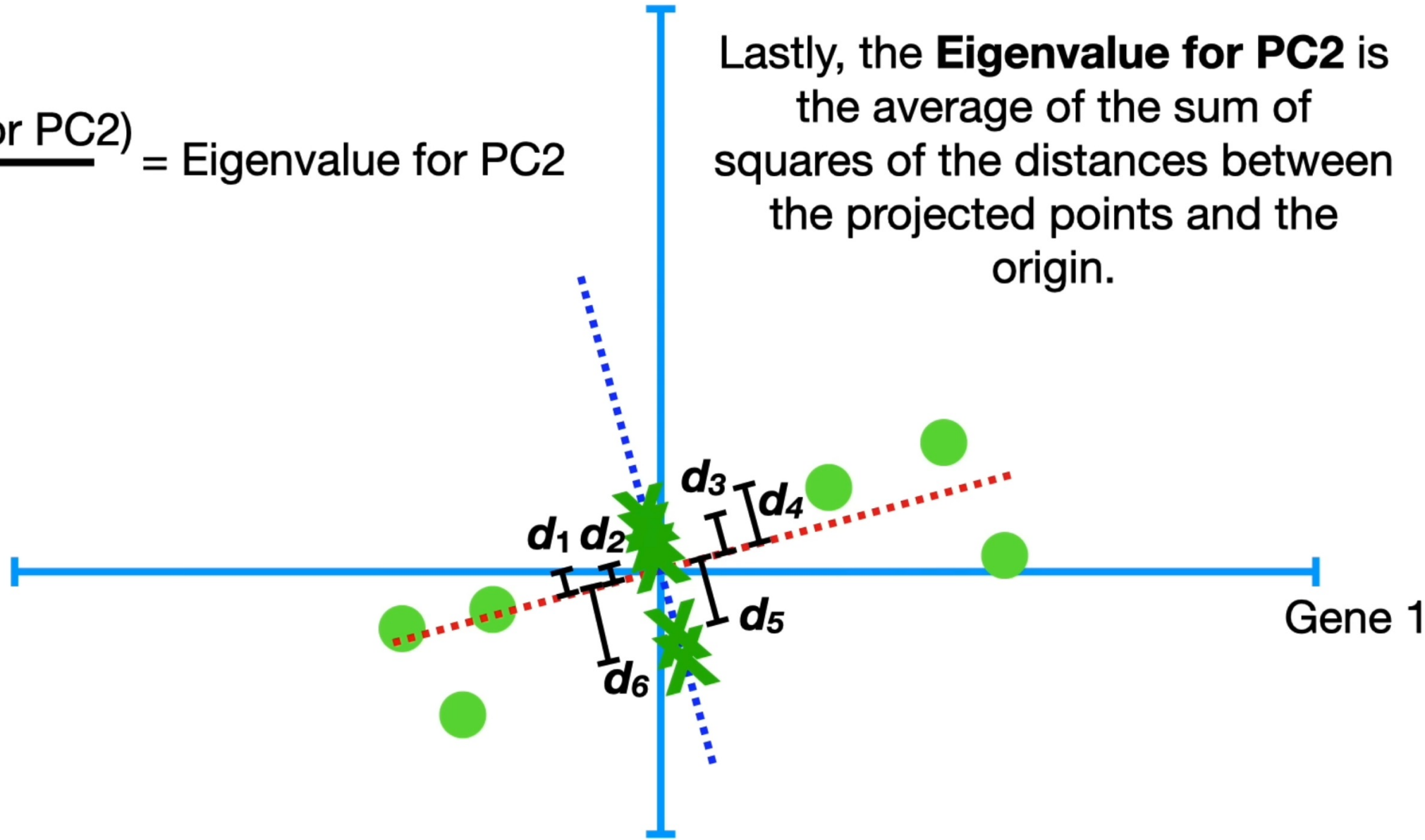
They tell us that, in terms of how the values are projected onto PC2, Gene 2 is 4 times as important as Gene 1.



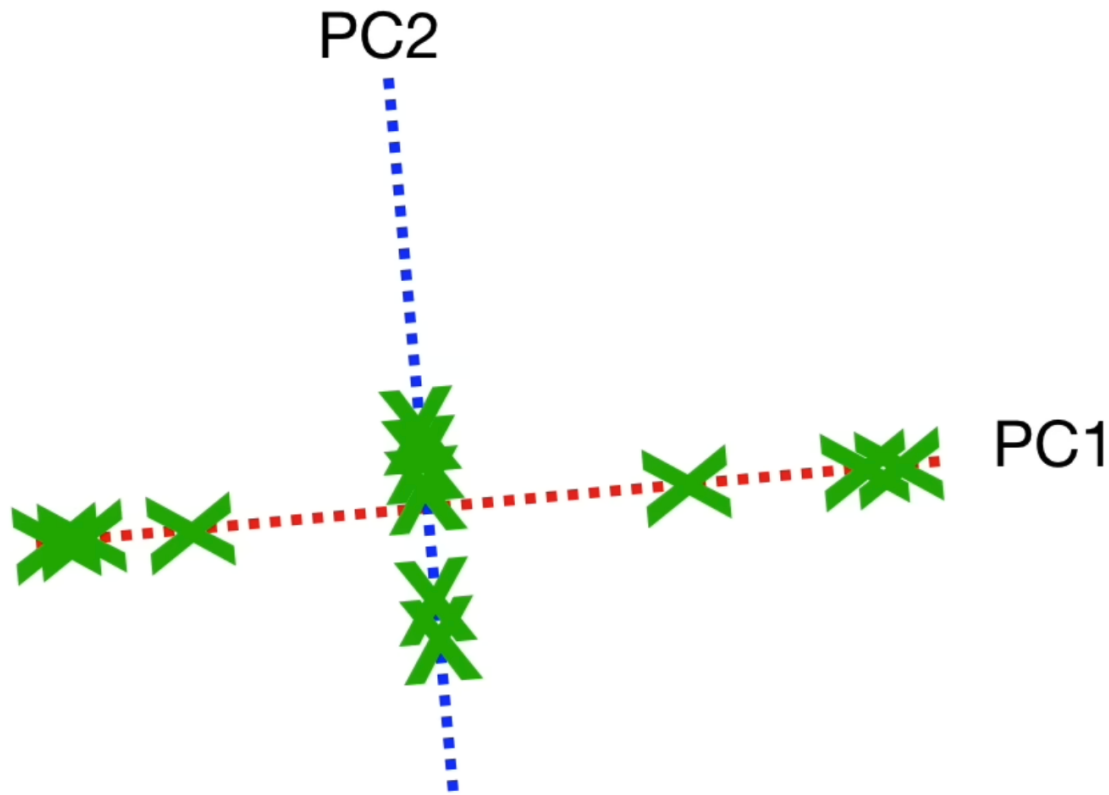
$$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2 = \text{sum of squared distances} = \text{SS(distances)}$$

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

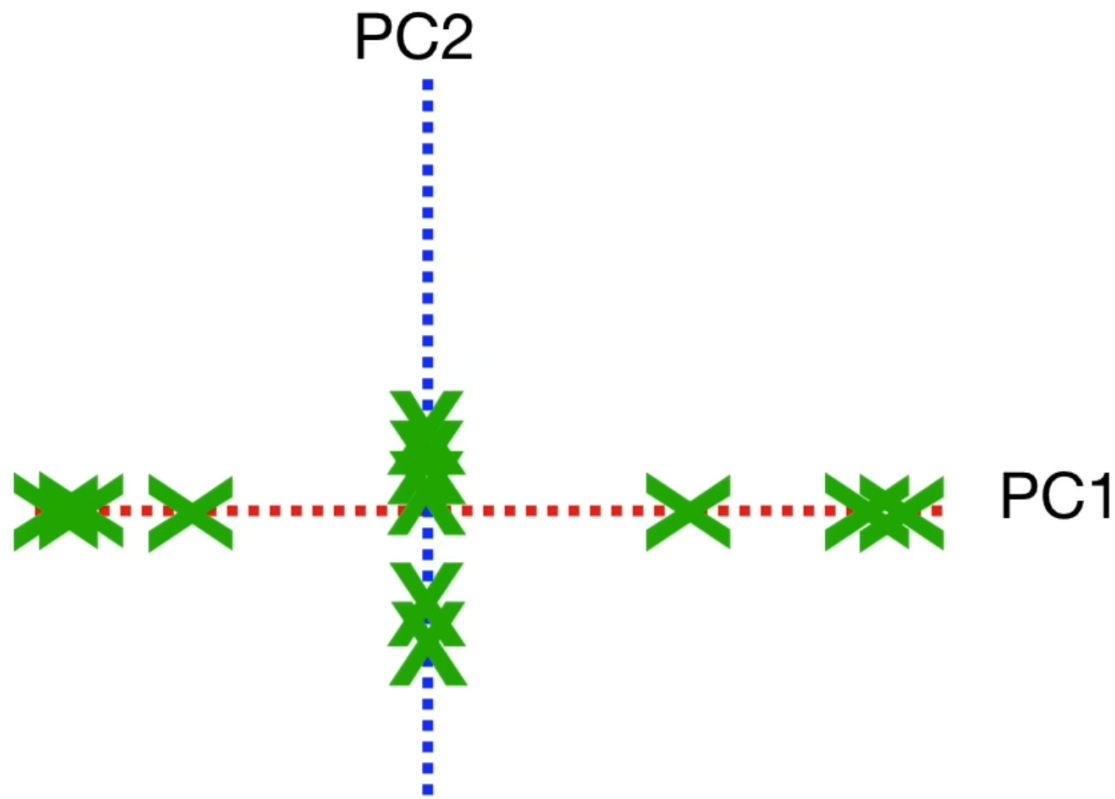
Lastly, the **Eigenvalue for PC2** is the average of the sum of squares of the distances between the projected points and the origin.



We simply rotate everything so that PC1 is horizontal…



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

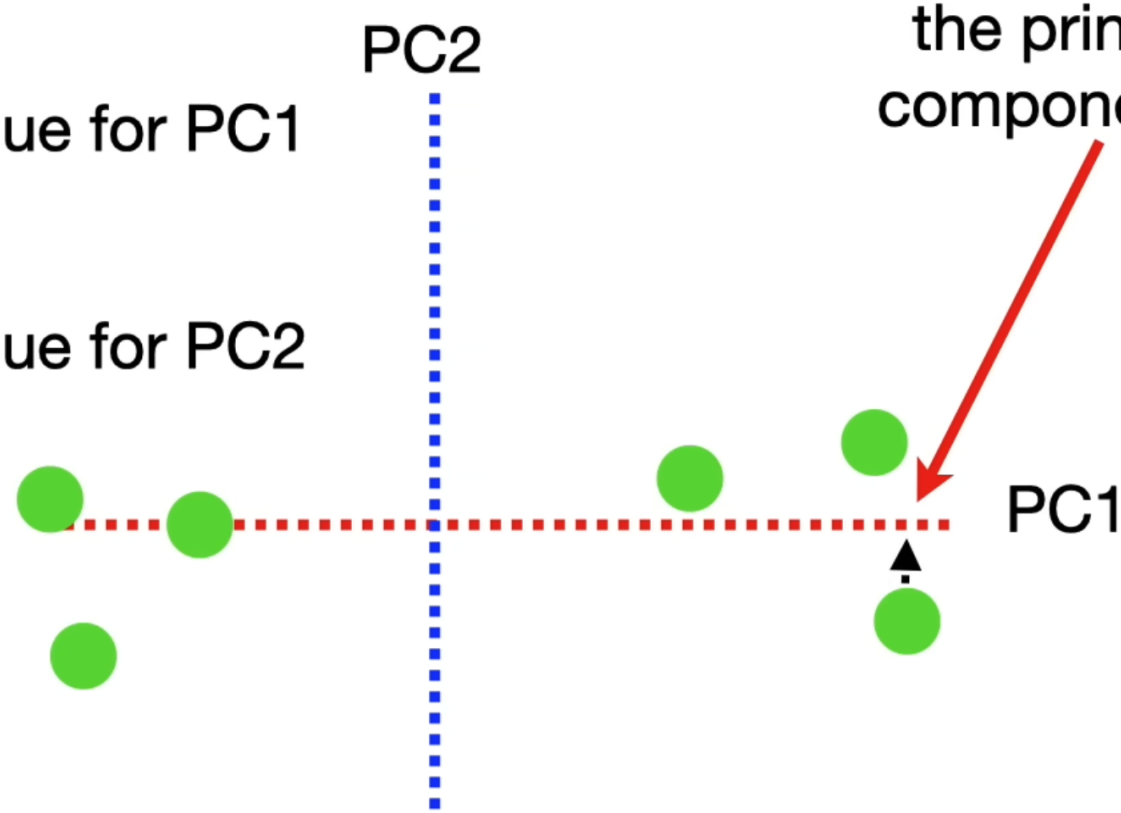


Remember the eigenvalues?

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

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

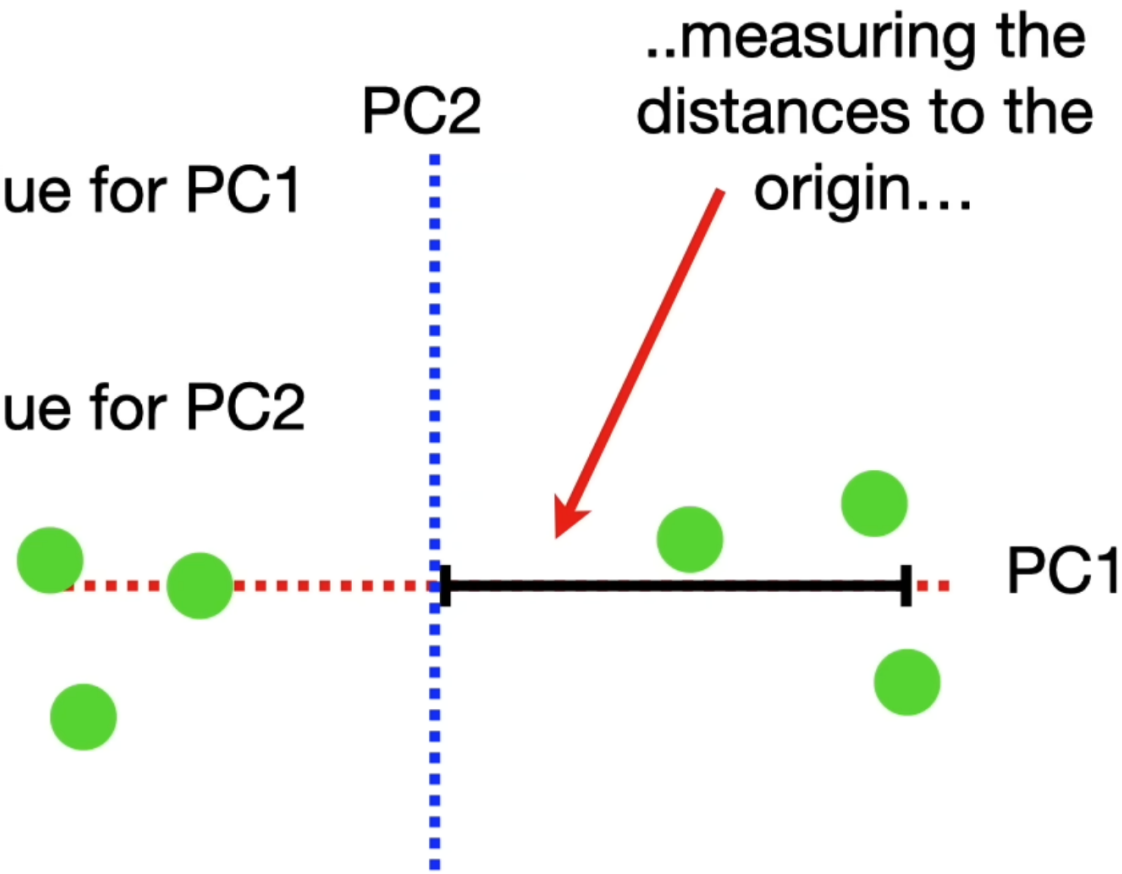
We got those by projecting the data onto the principal components…



Remember the eigenvalues?

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

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

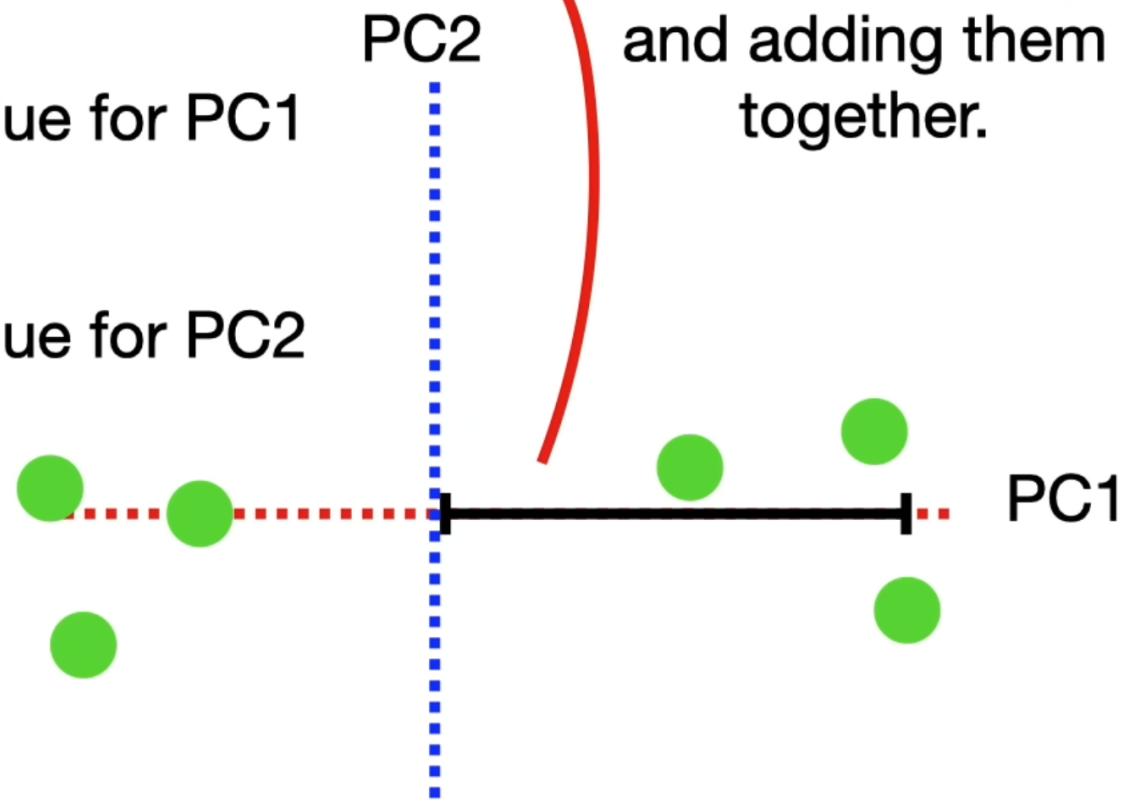


Remember the eigenvalues?

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

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

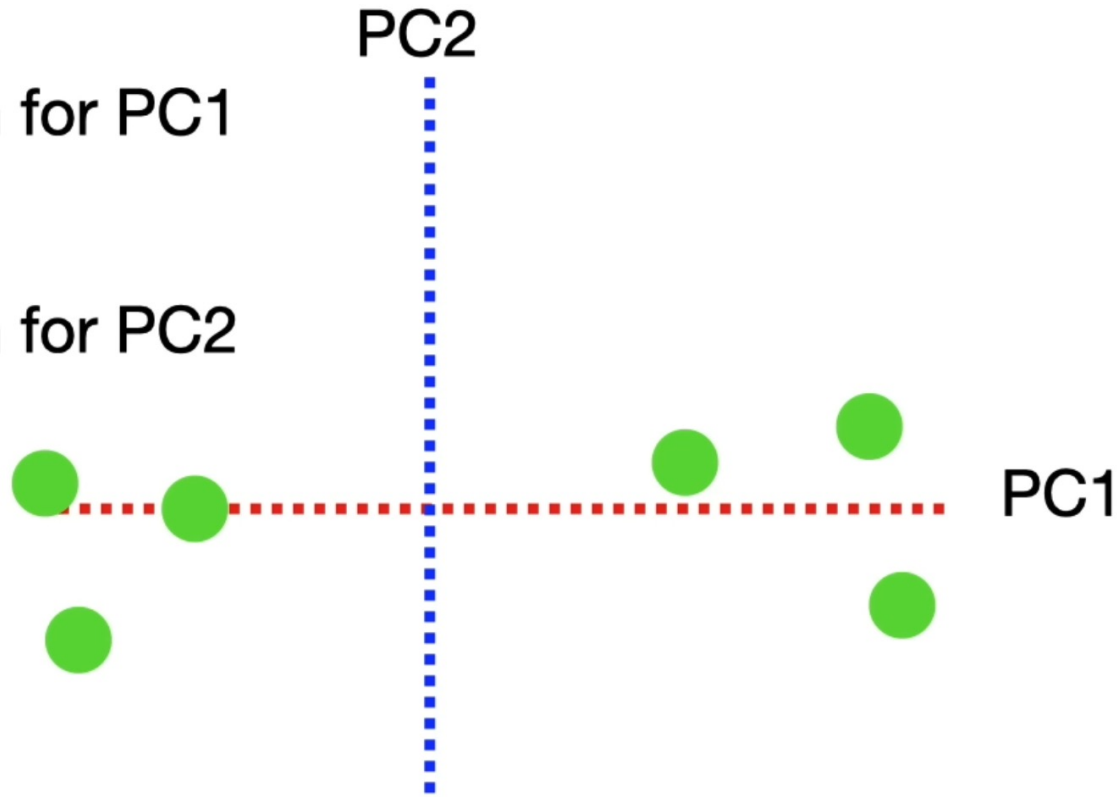
…then squaring and adding them together.



Well, if you are familiar with the equation for variation, you will notice **Eigenvalues** are just measures of variation.

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

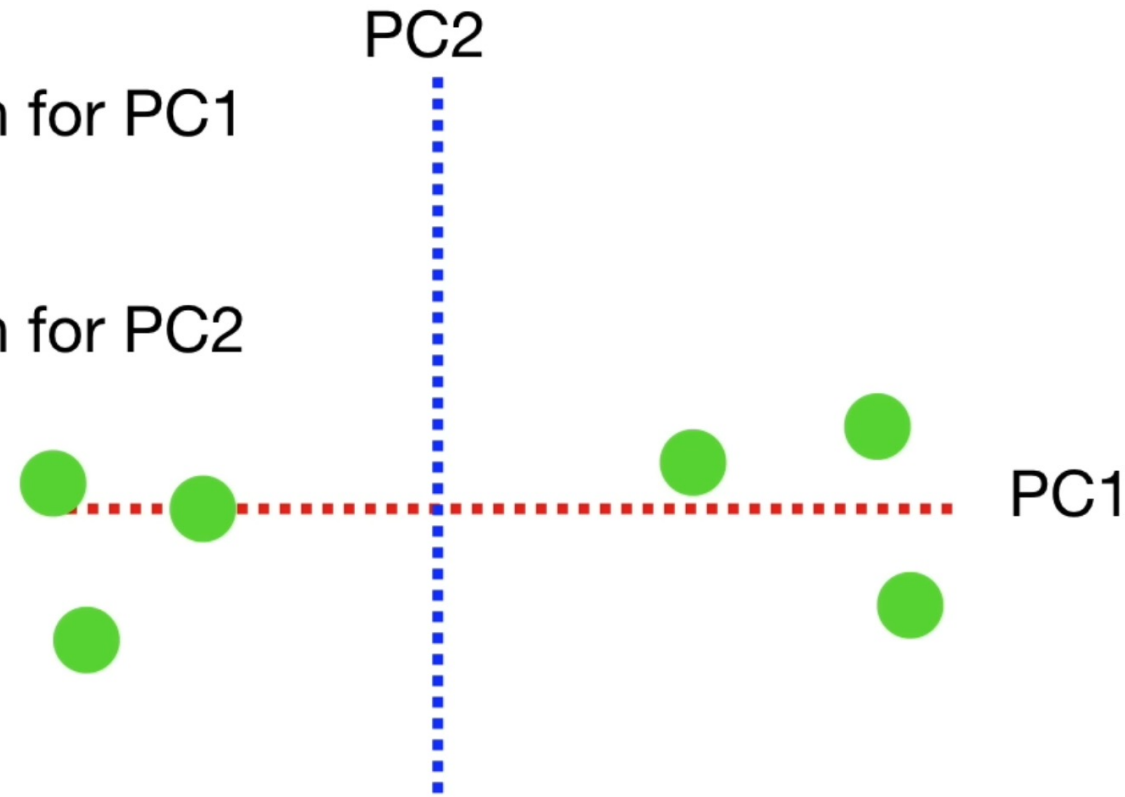
$$\frac{\text{SS(distances for PC2)}}{n - 1} = \text{Variation for PC2}$$



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

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

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

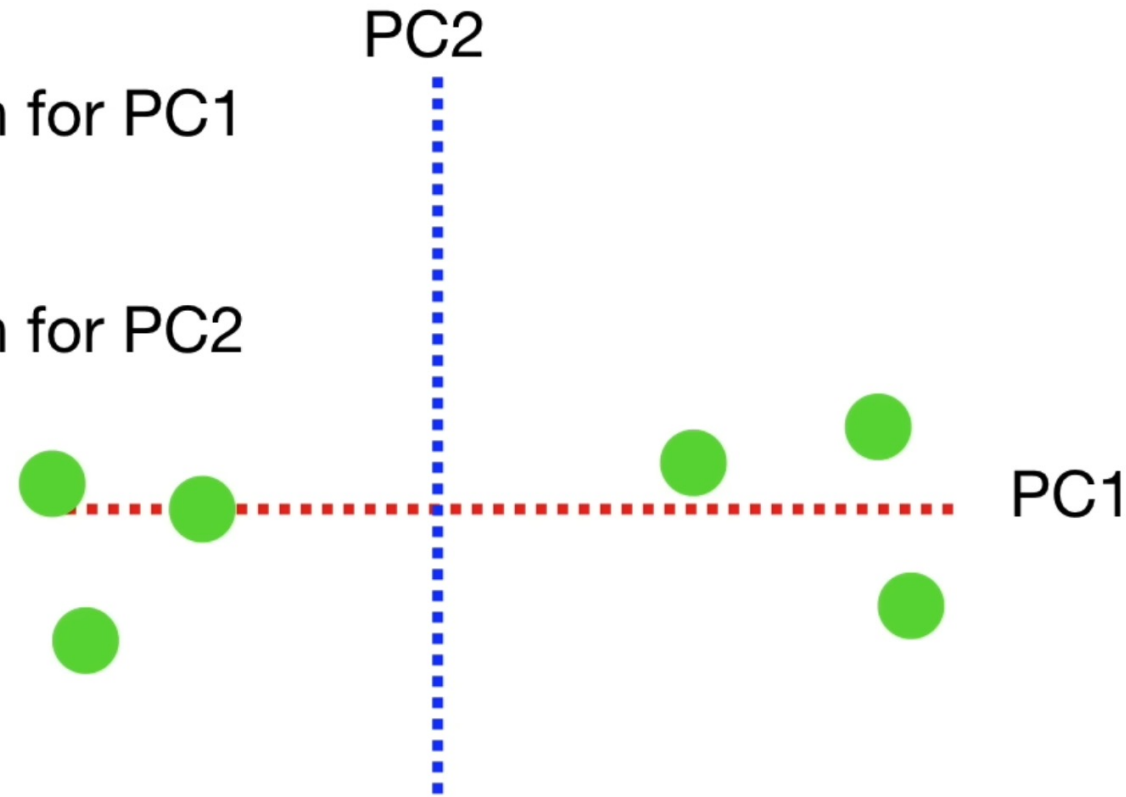


For the sake of the example, imagine that the Variation for **PC1 = 15**, and the variation for **PC2 = 3**.

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

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

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

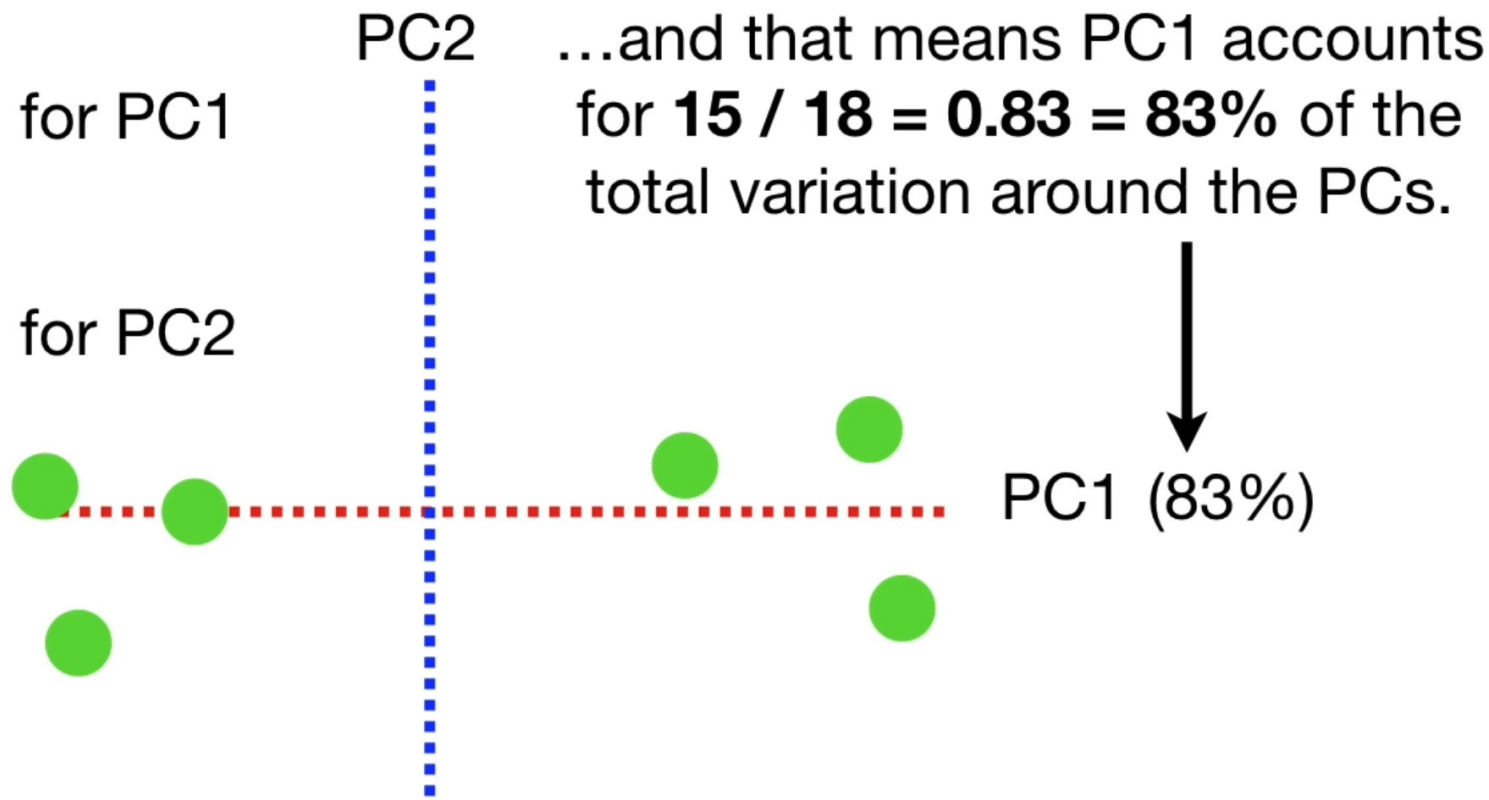


For the sake of the example, imagine that the Variation for **PC1 = 15**, and the variation for **PC2 = 3**.

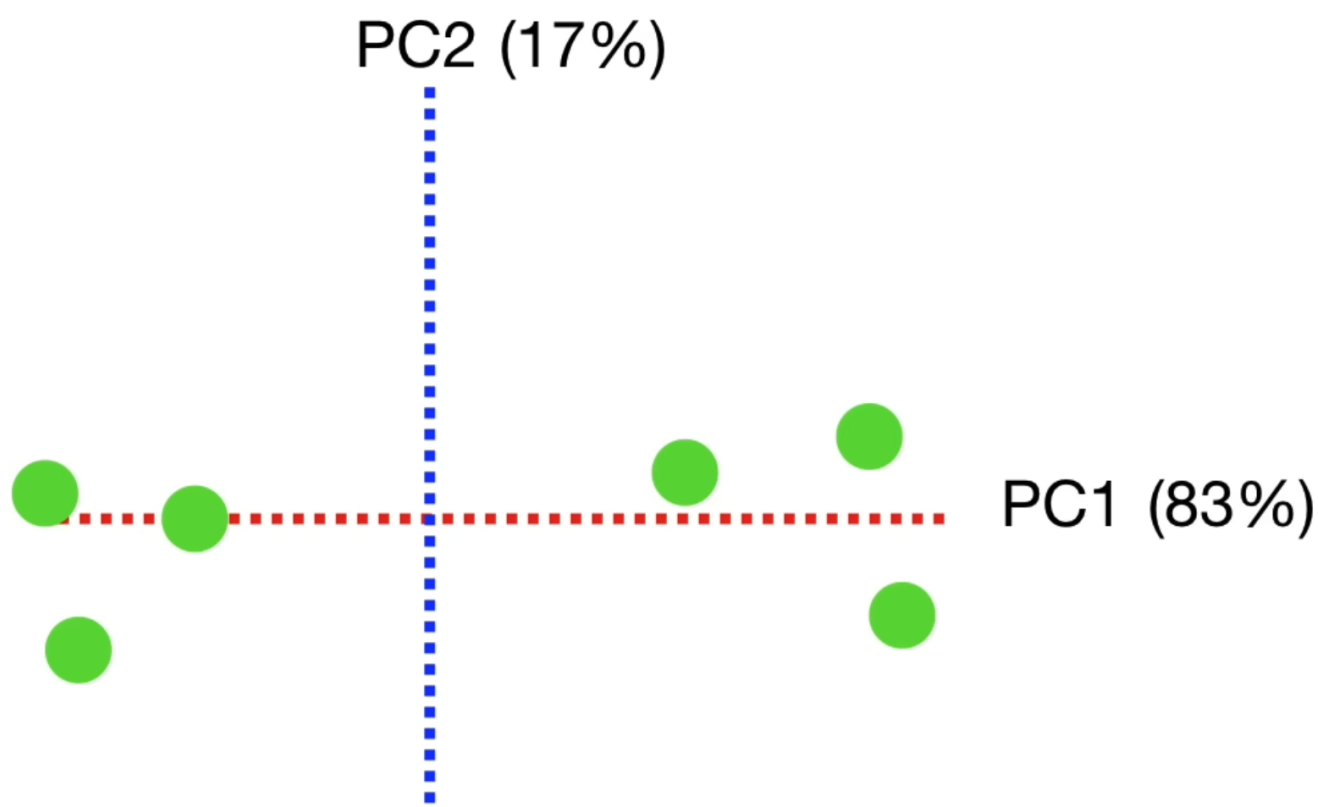
That means that the total variation around both PCs is **15 + 3 = 18**…

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

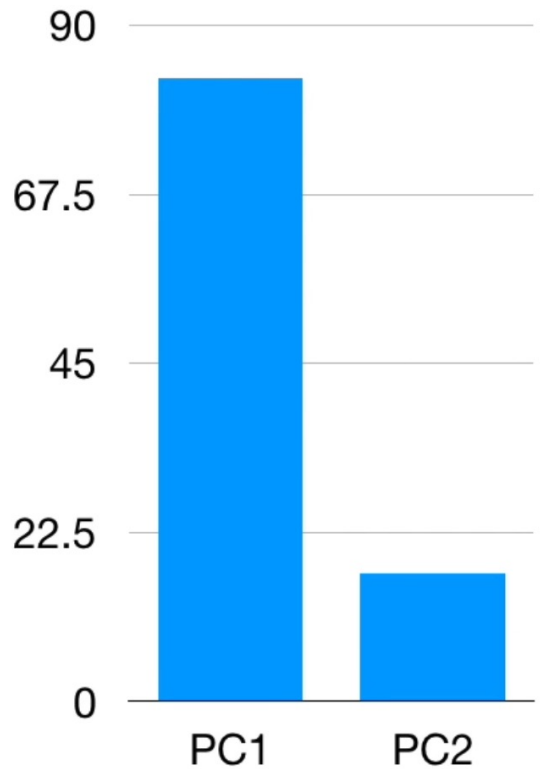
$$\frac{SS(\text{distances for PC2})}{n - 1} = \text{Variation for PC2}$$

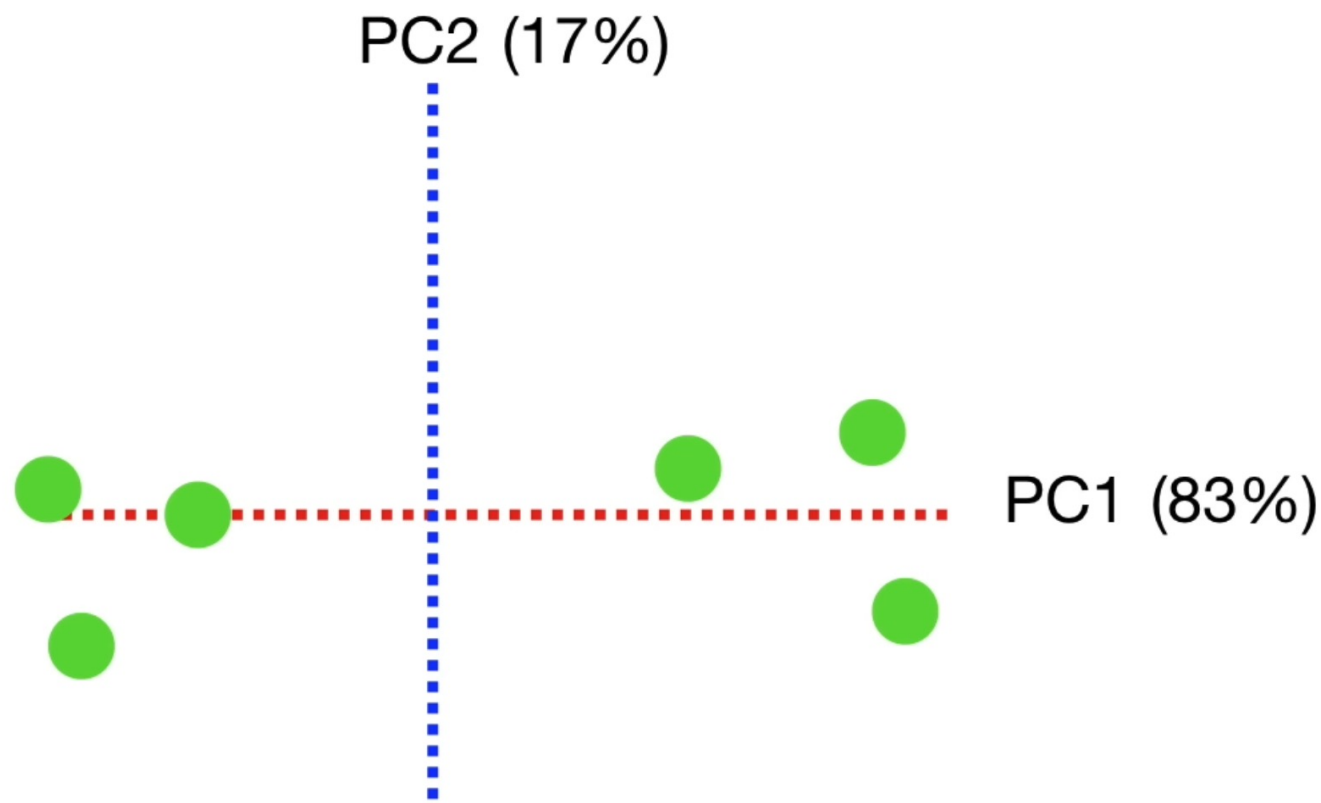


PC2 accounts for **3 / 18 = 0.17 = 17%** of the total variation around the PCs.



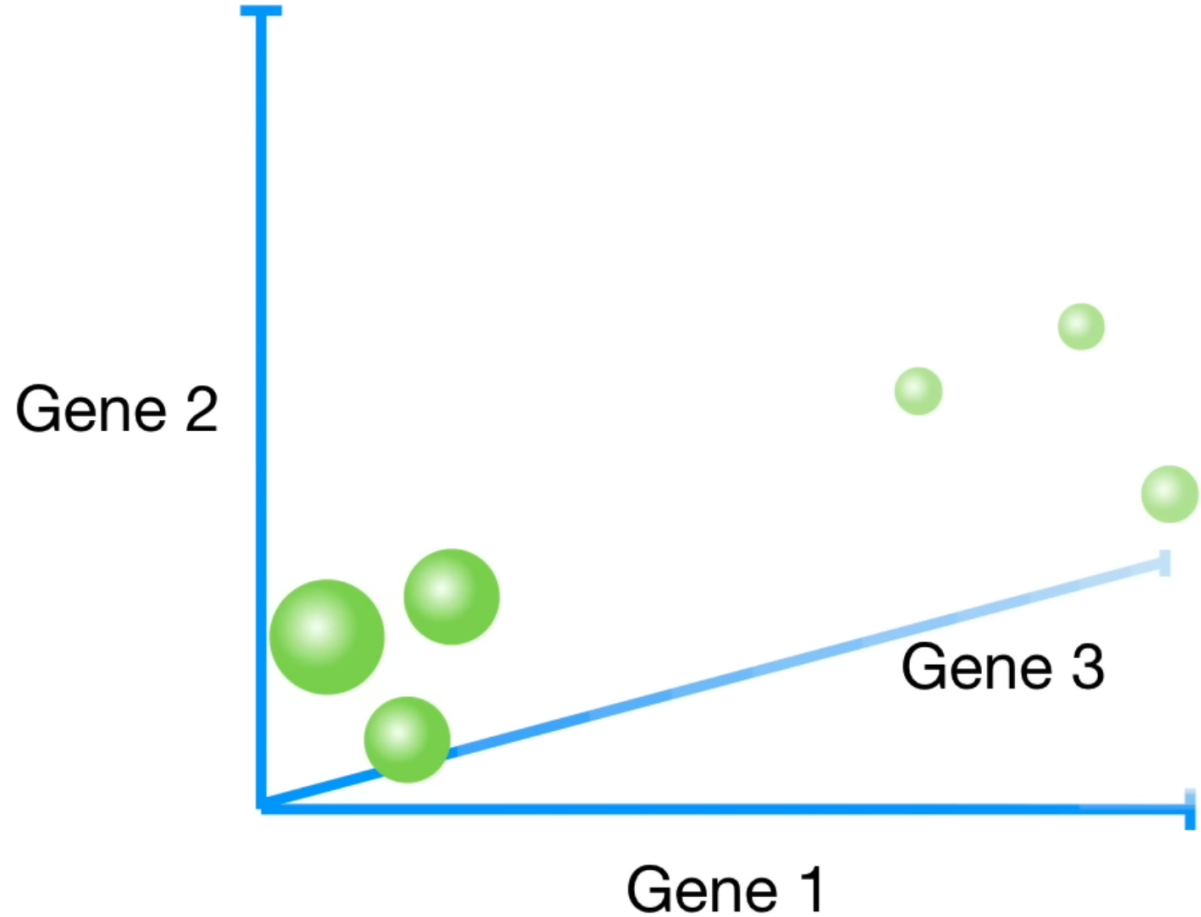
TERMINOLOGY ALERT!!!! A **Scree Plot** is a graphical representation of the percentages of variation that each PC accounts for.



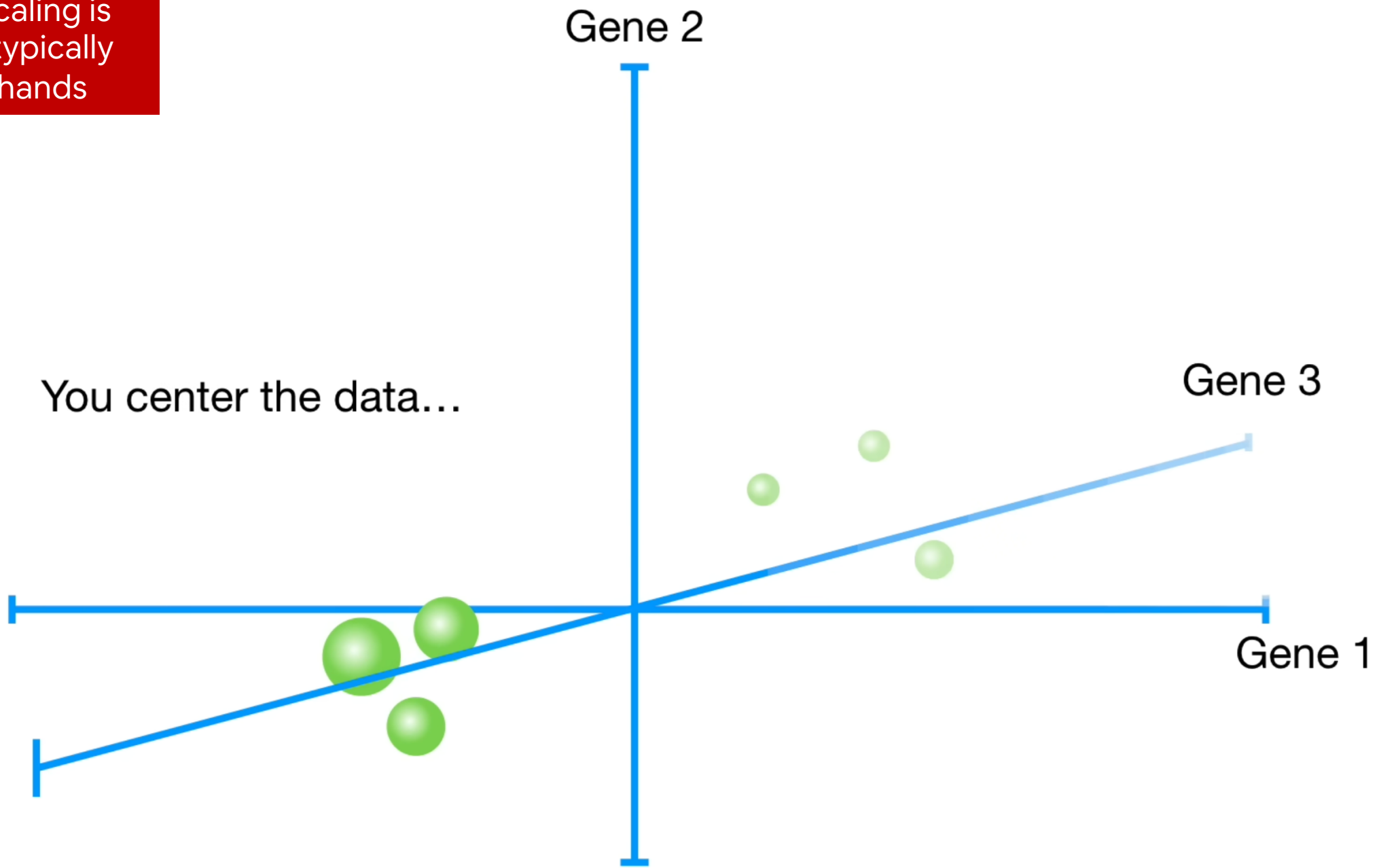


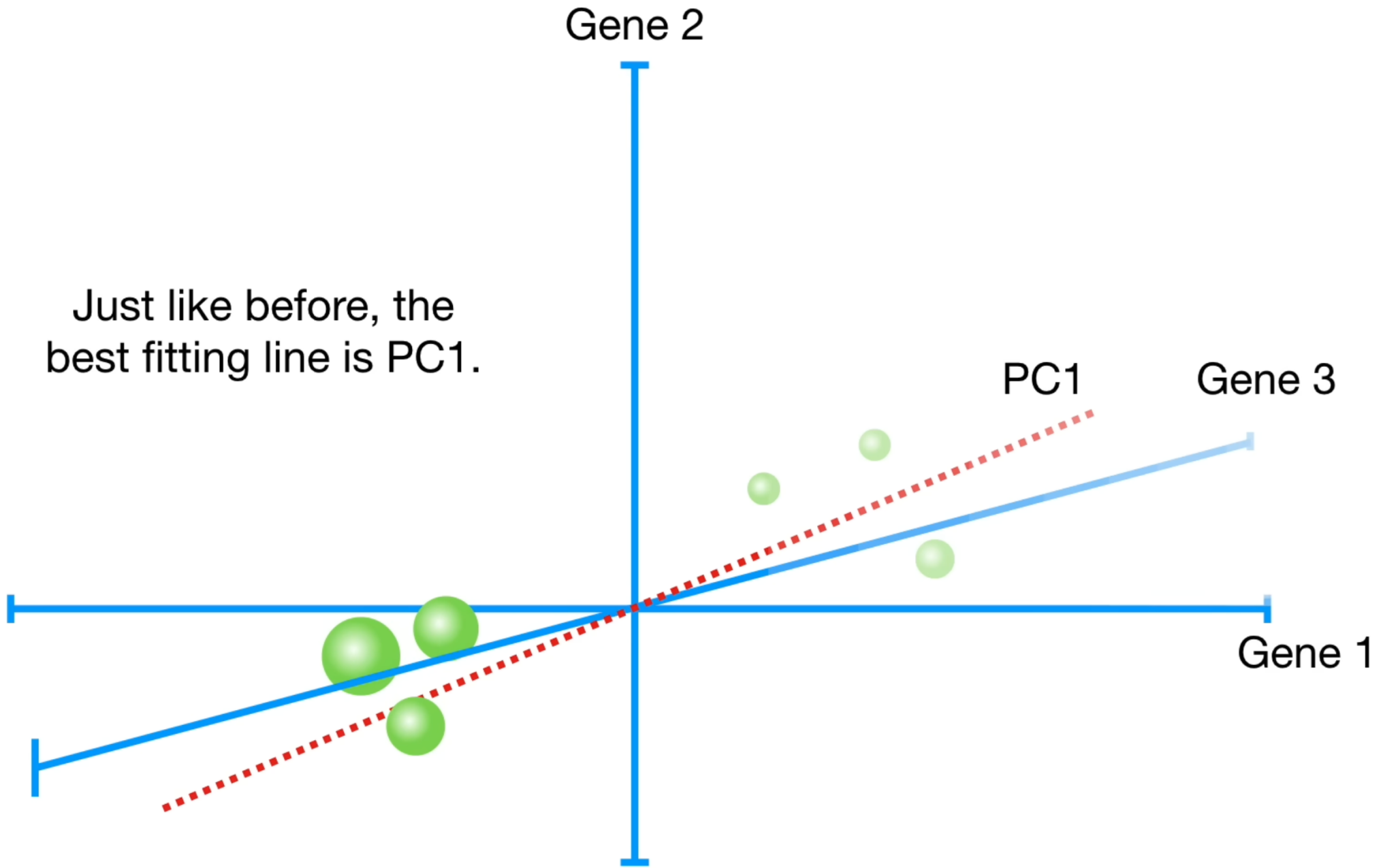
PCA with 3 variables (in this case, that means 3 genes) is pretty much the same as 2 variables…

	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
Gene 3	12	9	10	2.5	1.3	2



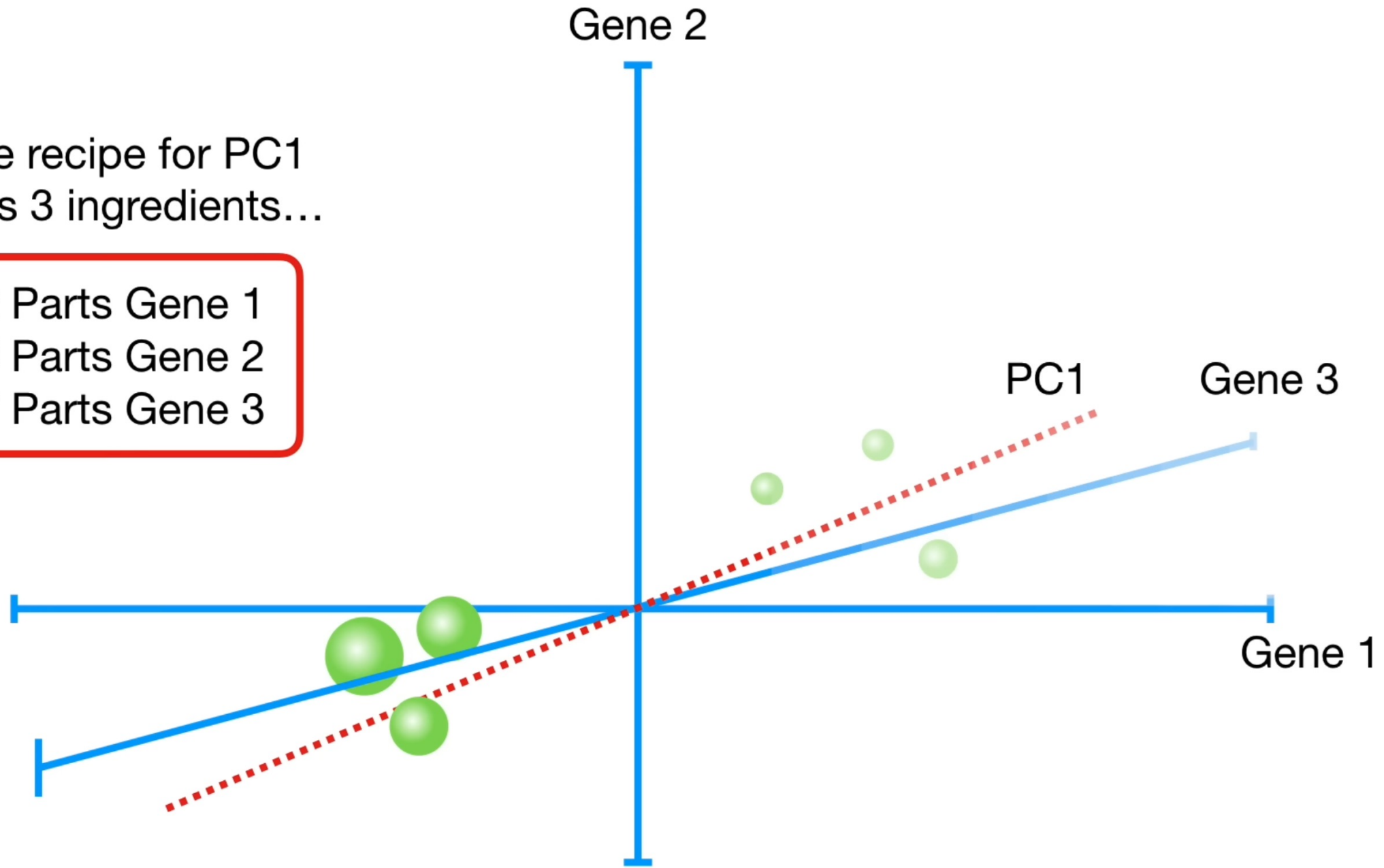
Pay attention: scaling is something you typically have to do by hands

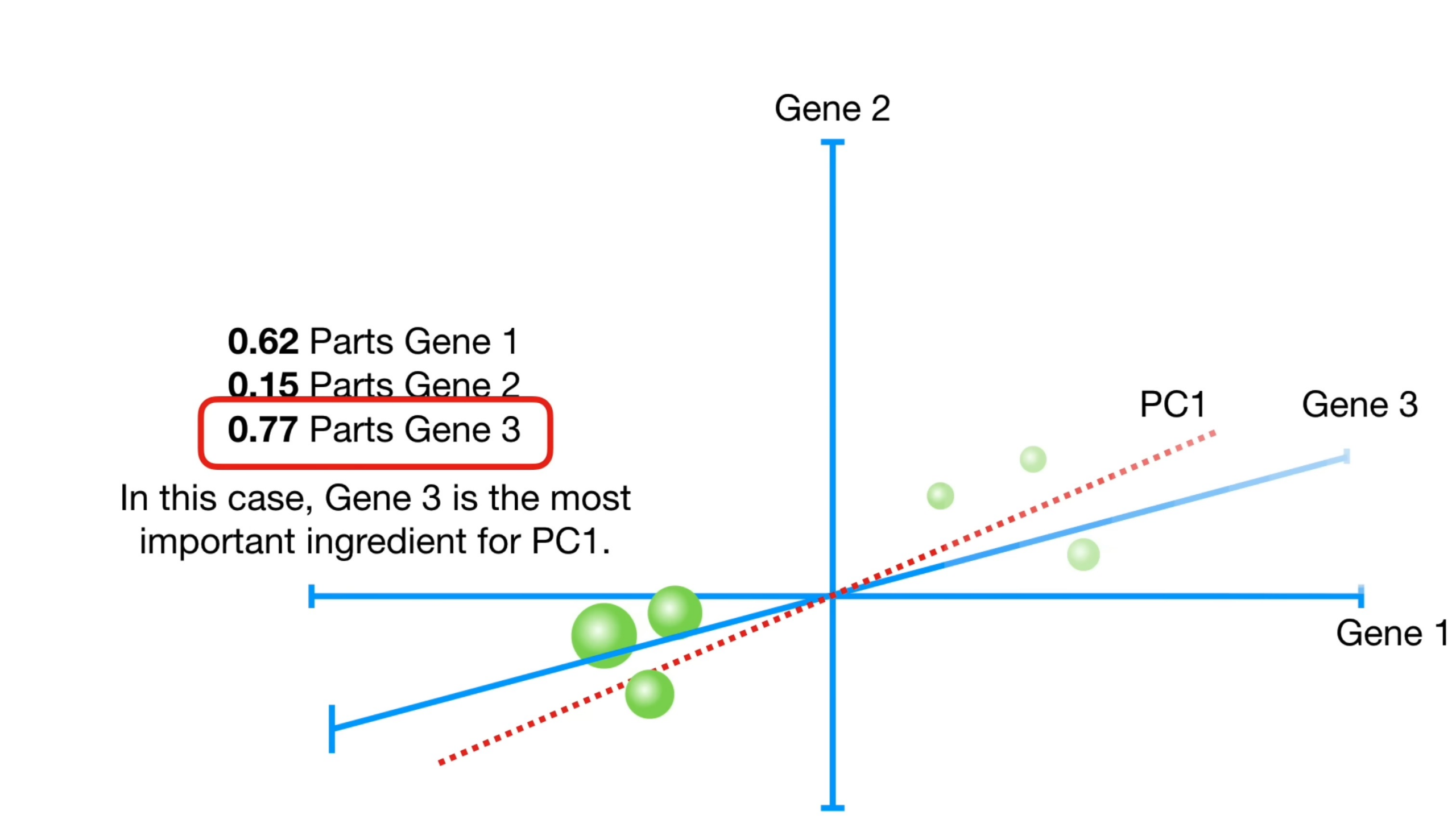



Gene 2
Just like before, the
best fitting line is PC1.
PC1
Gene 3
Gene 1

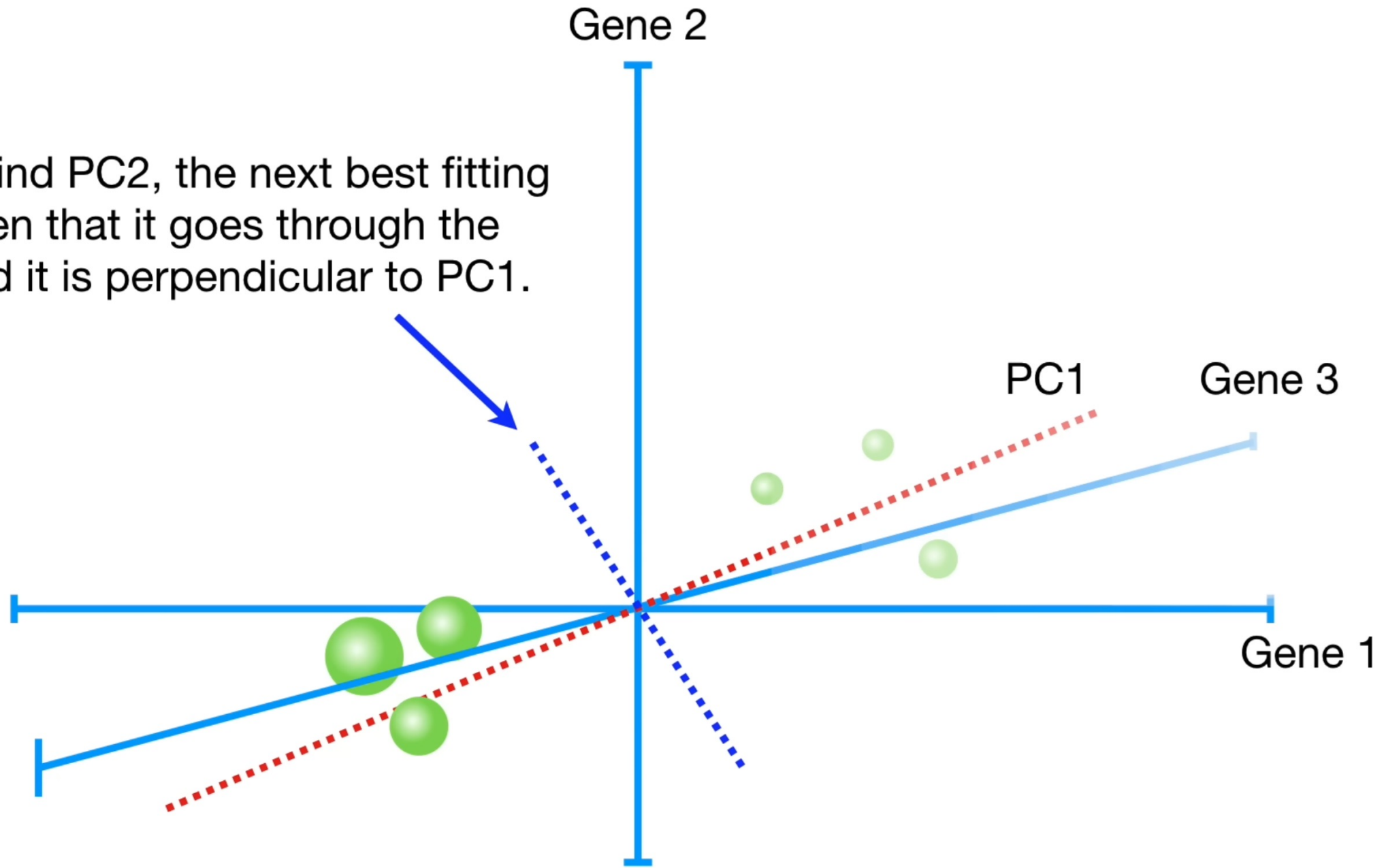
But the recipe for PC1
now has 3 ingredients…

0.62 Parts Gene 1
0.15 Parts Gene 2
0.77 Parts Gene 3



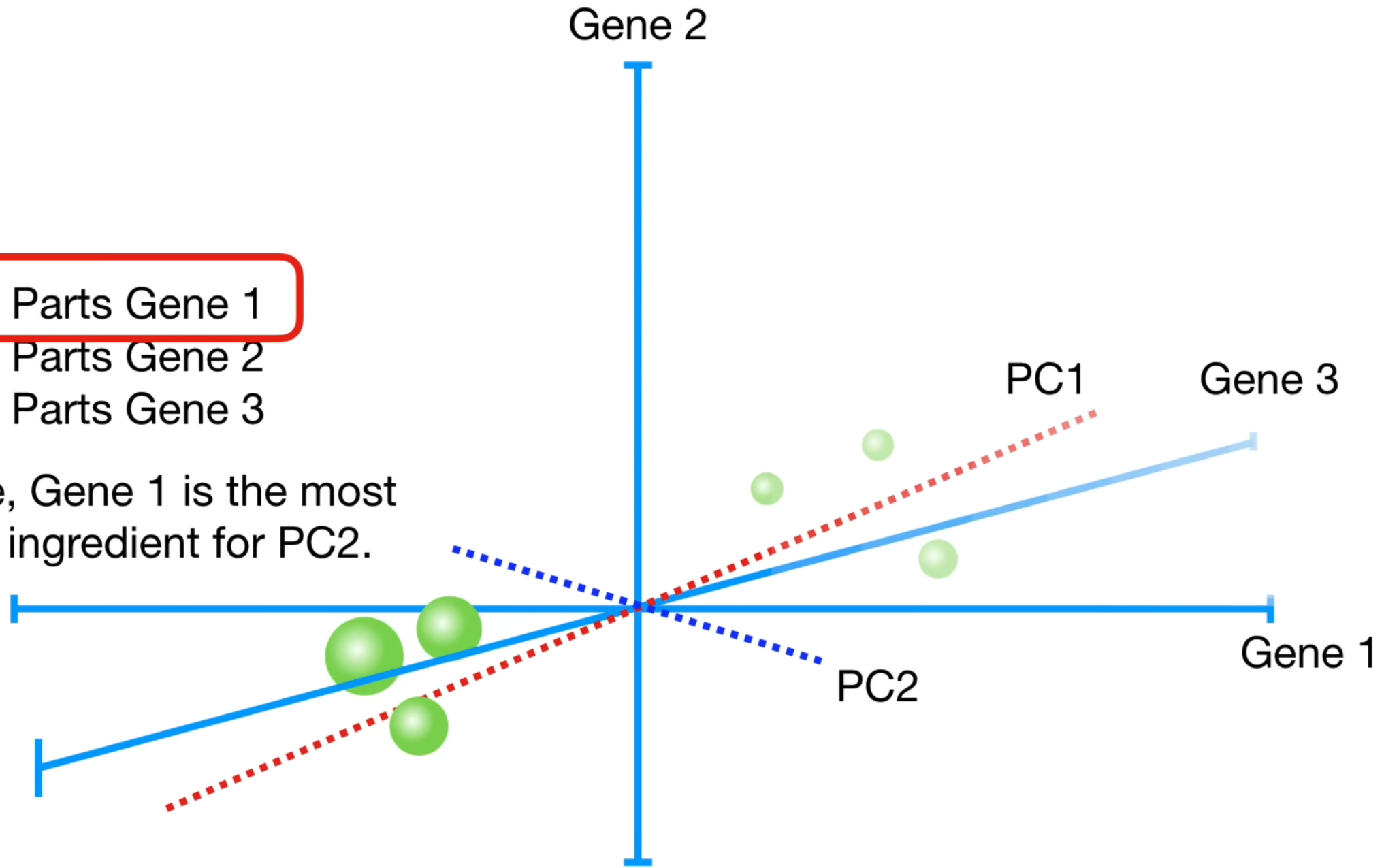

Gene 2
0.62 Parts Gene 1
0.15 Parts Gene 2
0.77 Parts Gene 3
In this case, Gene 3 is the most important ingredient for PC1.
PC1
Gene 3
Gene 1

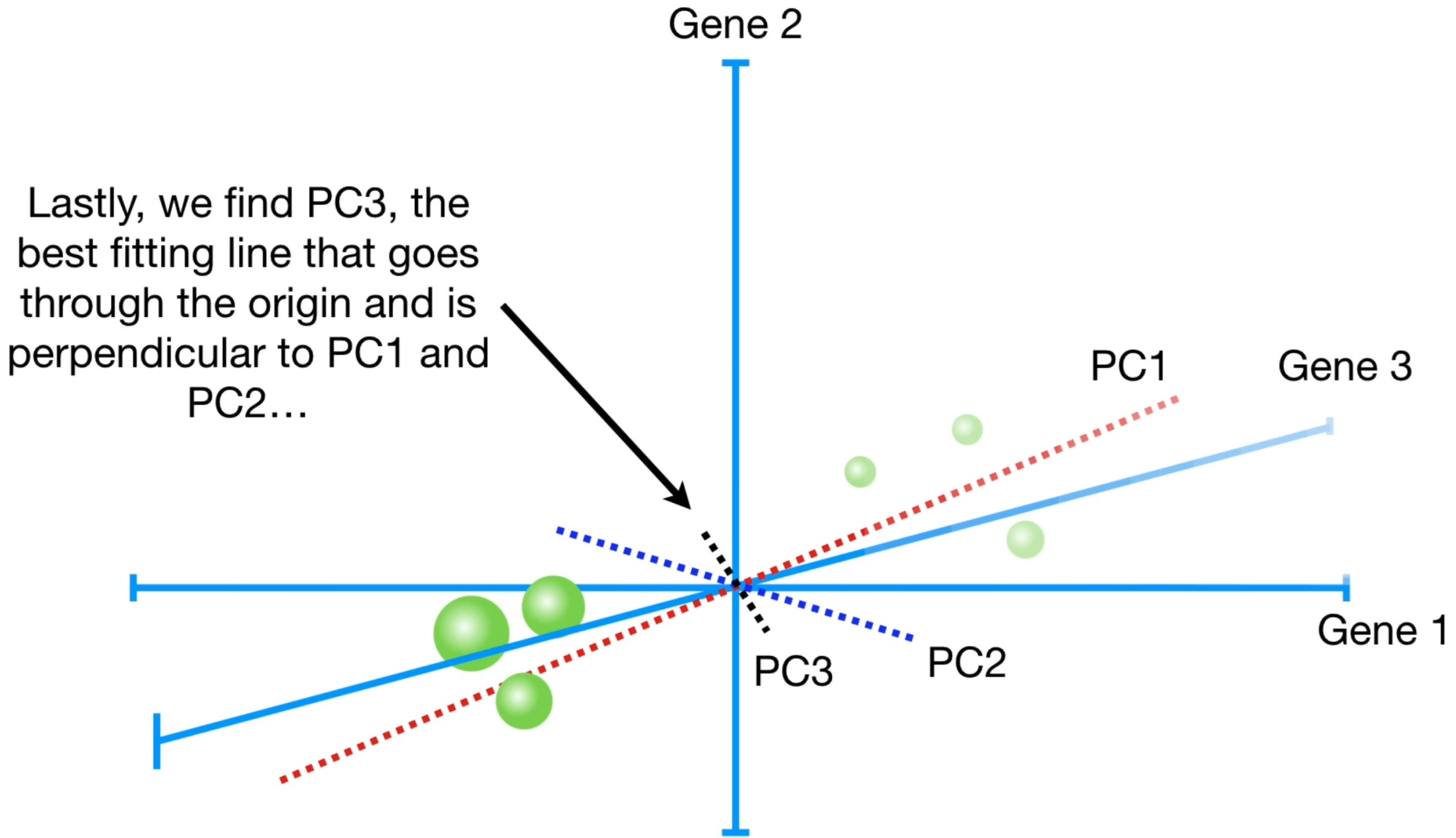
You then find PC2, the next best fitting line given that it goes through the origin and it is perpendicular to PC1.



0.77 Parts Gene 1
0.62 Parts Gene 2
0.15 Parts Gene 3

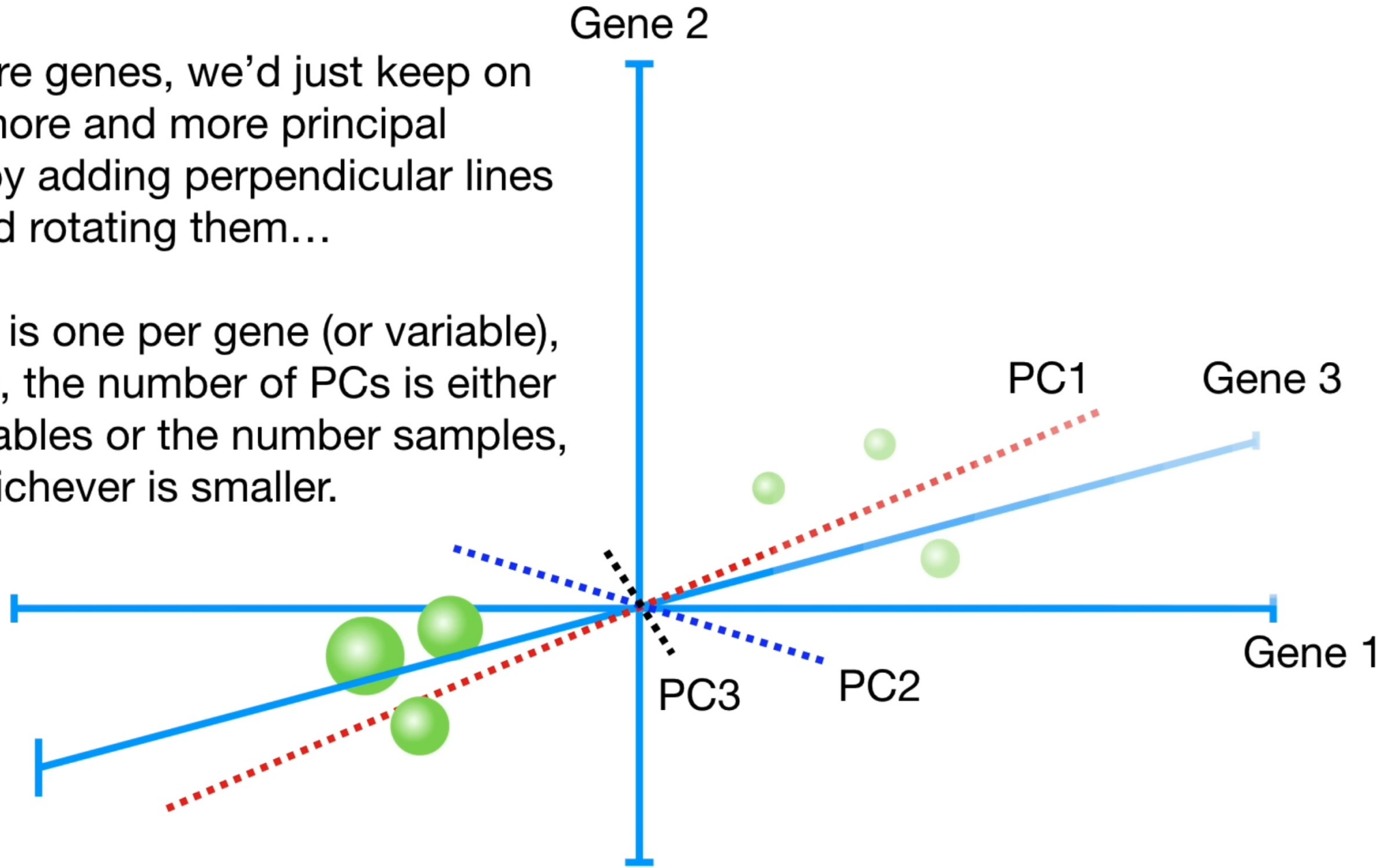
In this case, Gene 1 is the most important ingredient for PC2.



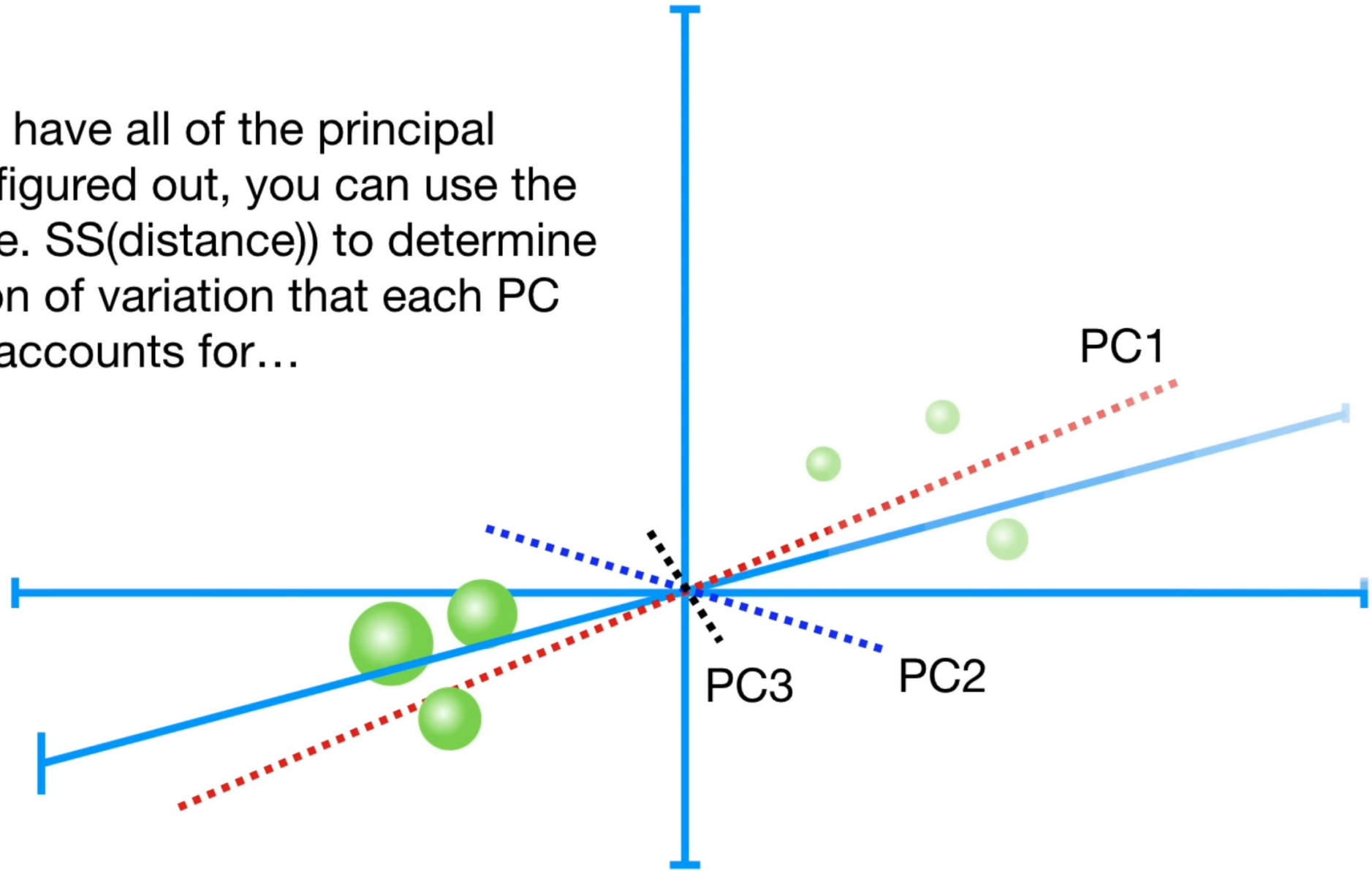

Lastly, we find PC3, the best fitting line that goes through the origin and is perpendicular to PC1 and PC2…
Gene 2
PC1
Gene 3
Gene 1
PC3
PC2

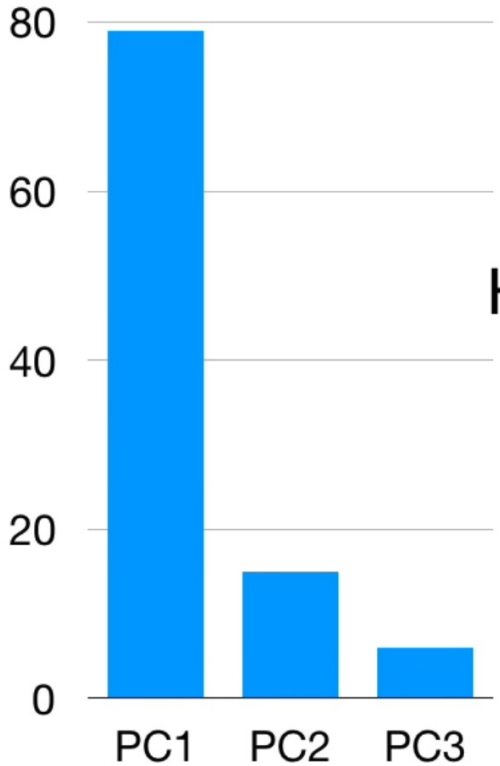
If we had more genes, we'd just keep on finding more and more principal components by adding perpendicular lines and rotating them…

In theory there is one per gene (or variable), but in practice, the number of PCs is either number of variables or the number samples, whichever is smaller.

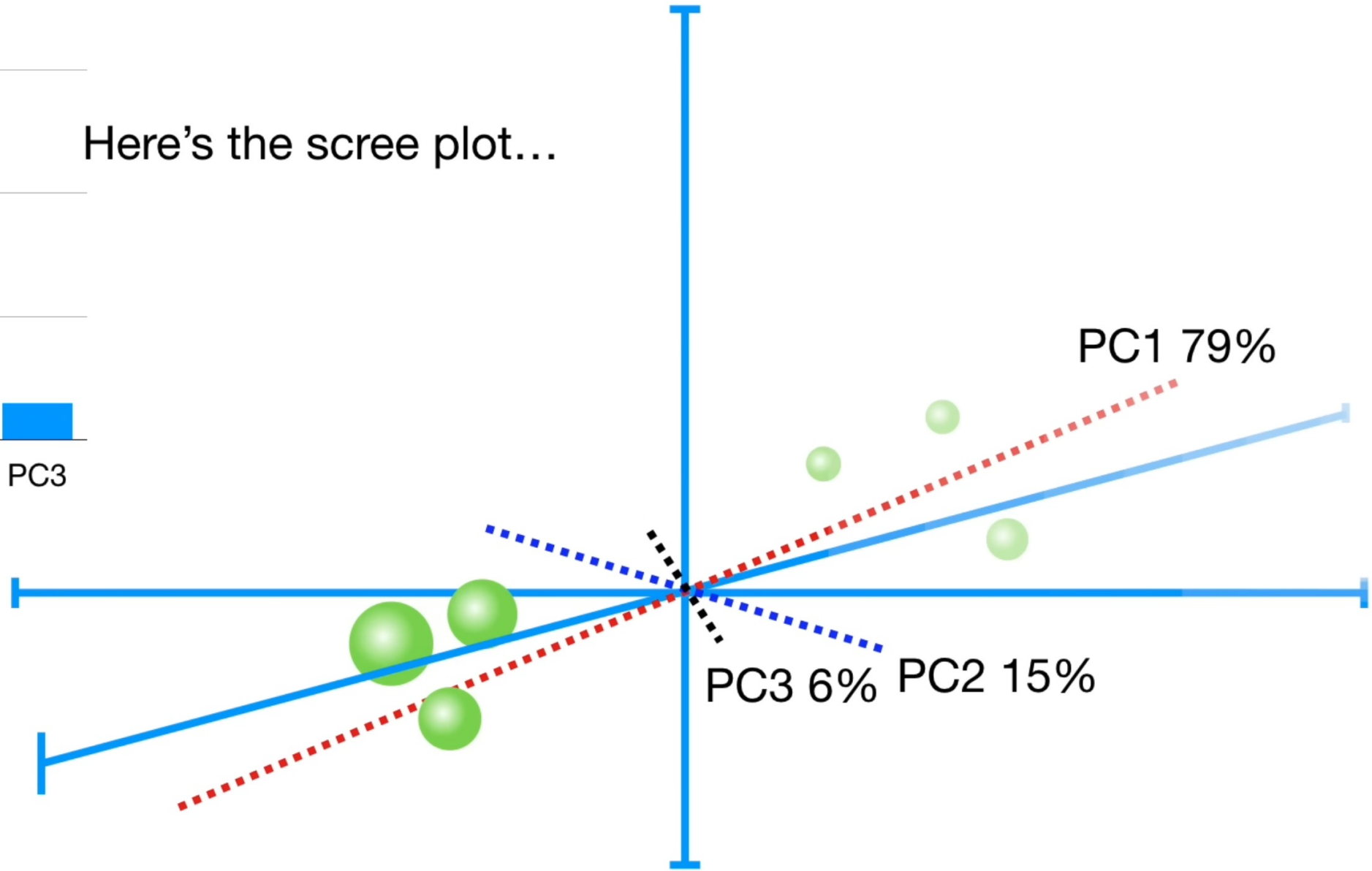


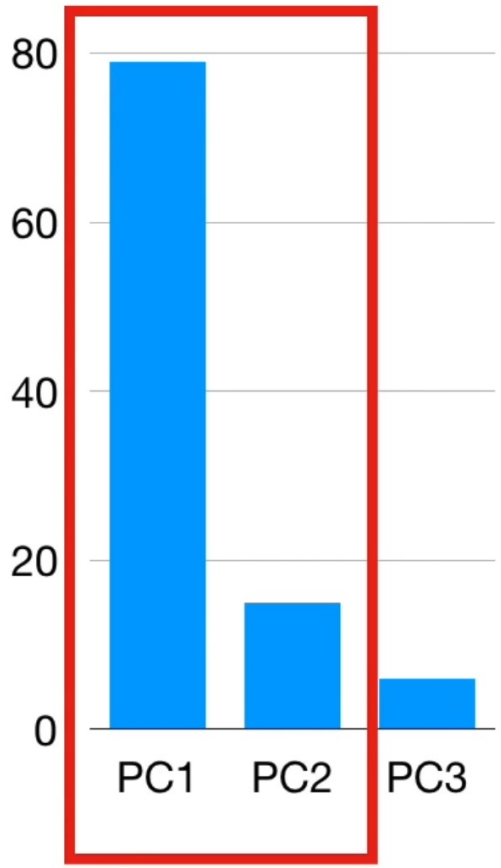
Once you have all of the principal components figured out, you can use the eigenvalues (i.e. SS(distance)) to determine the proportion of variation that each PC accounts for…



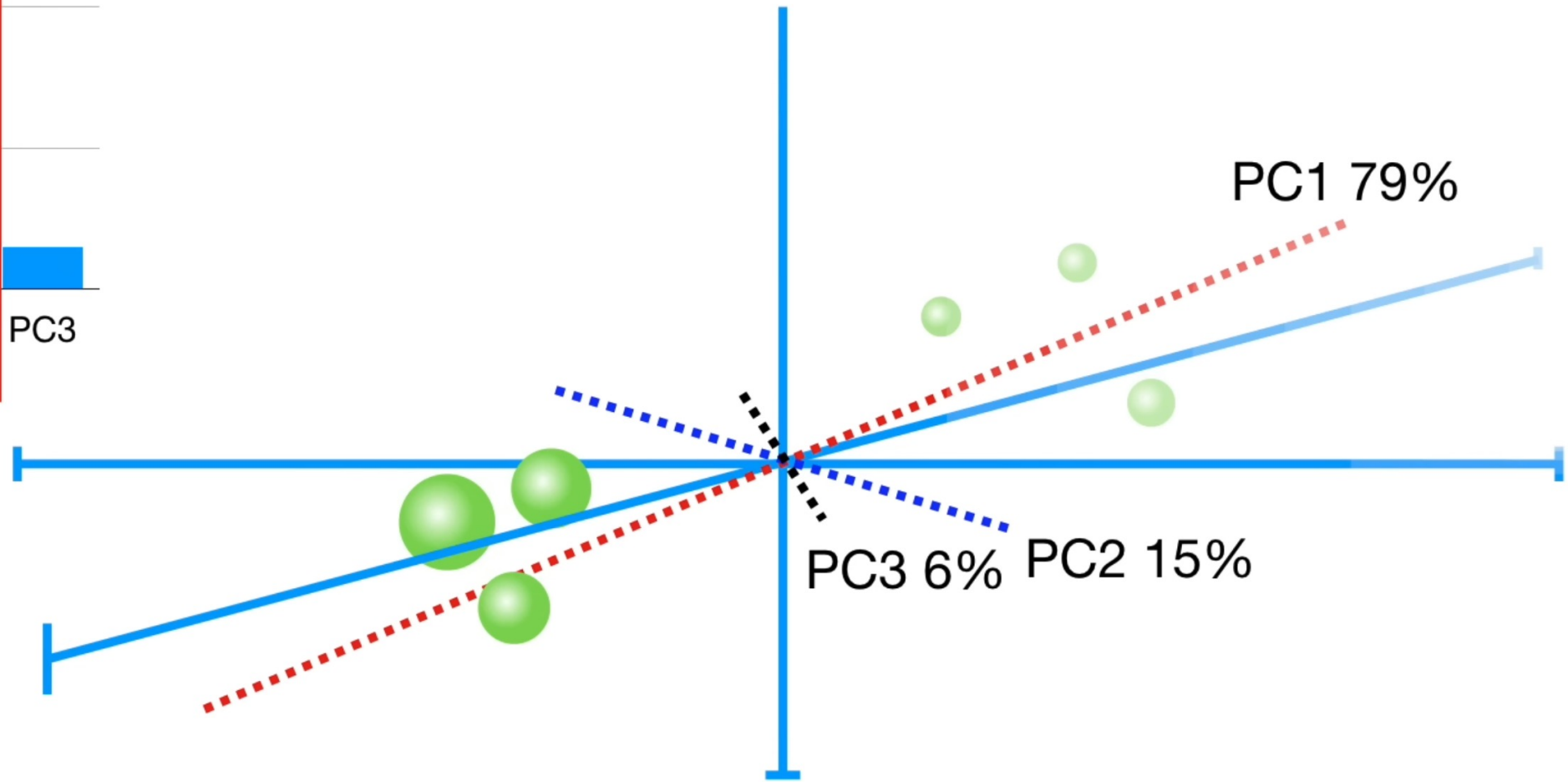


Here's the scree plot…

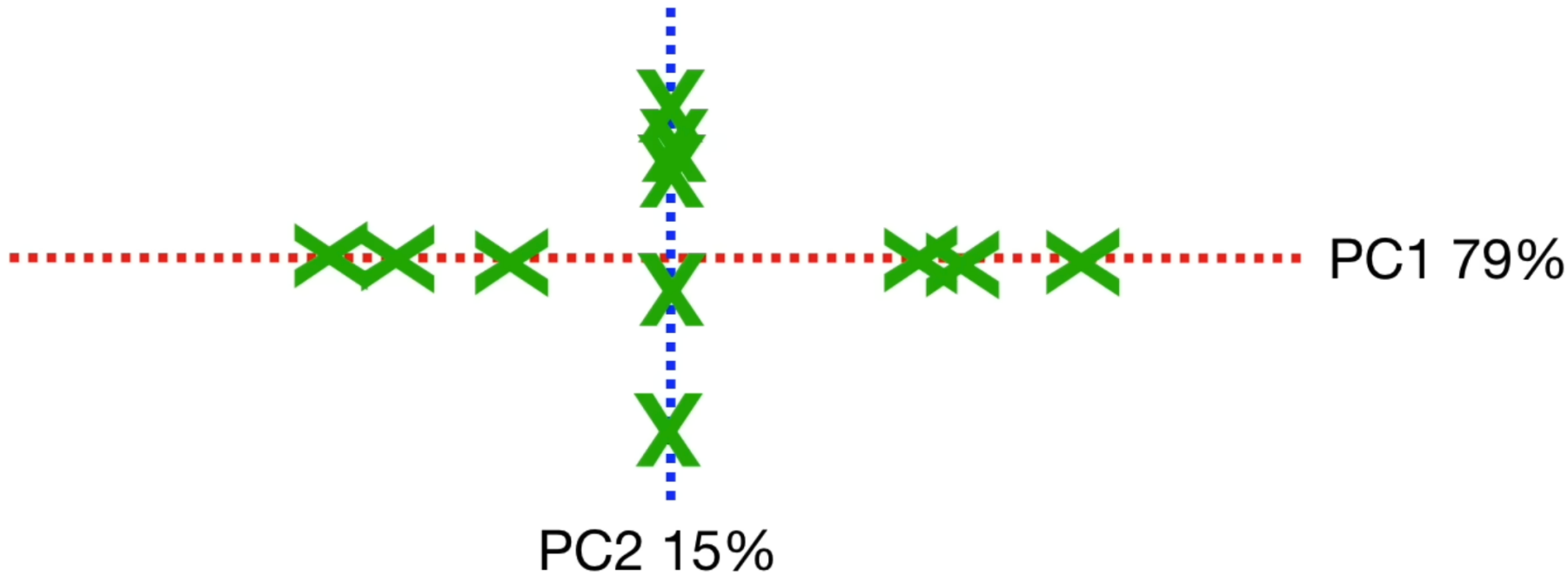


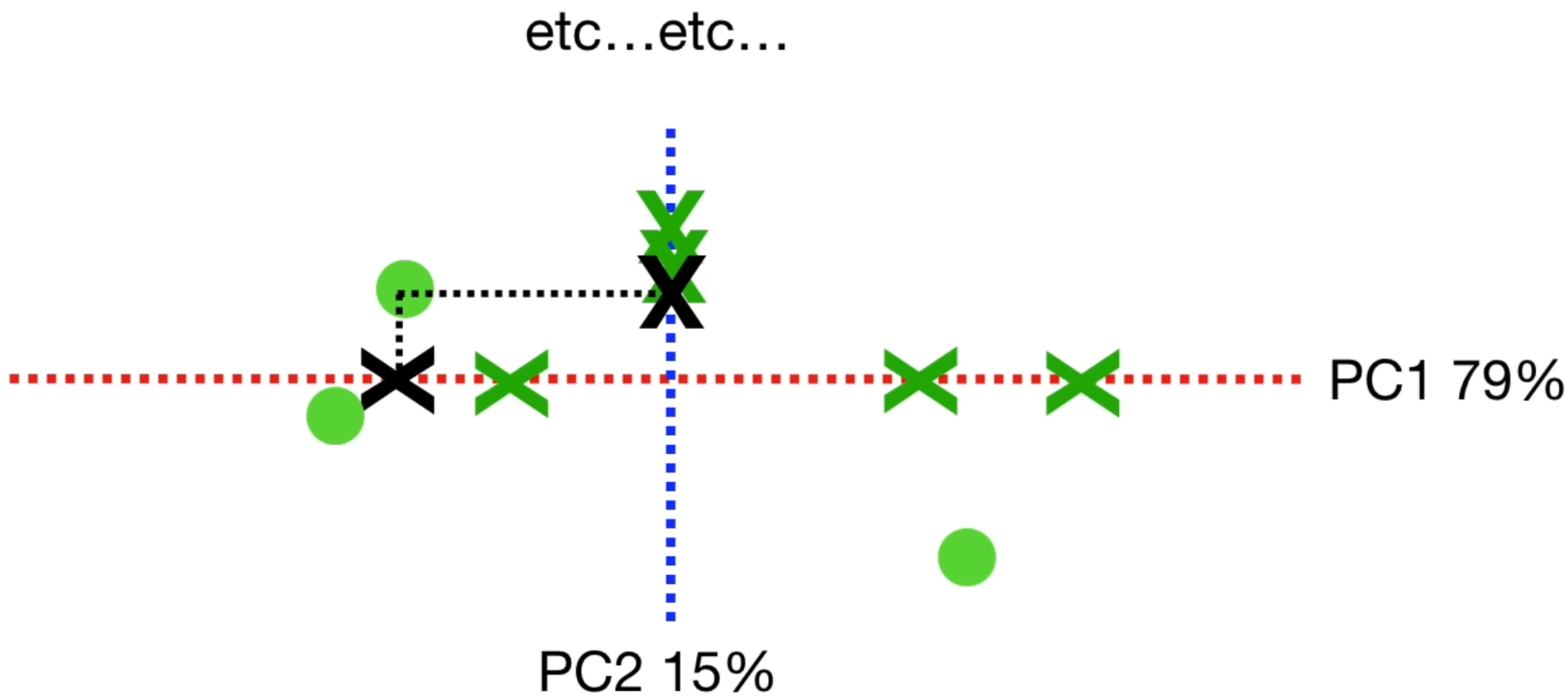


That means that a 2-D graph, using just PC1 and PC2, would be a good approximation of this 3-D graph since it would account for 94% of the variation in the data.

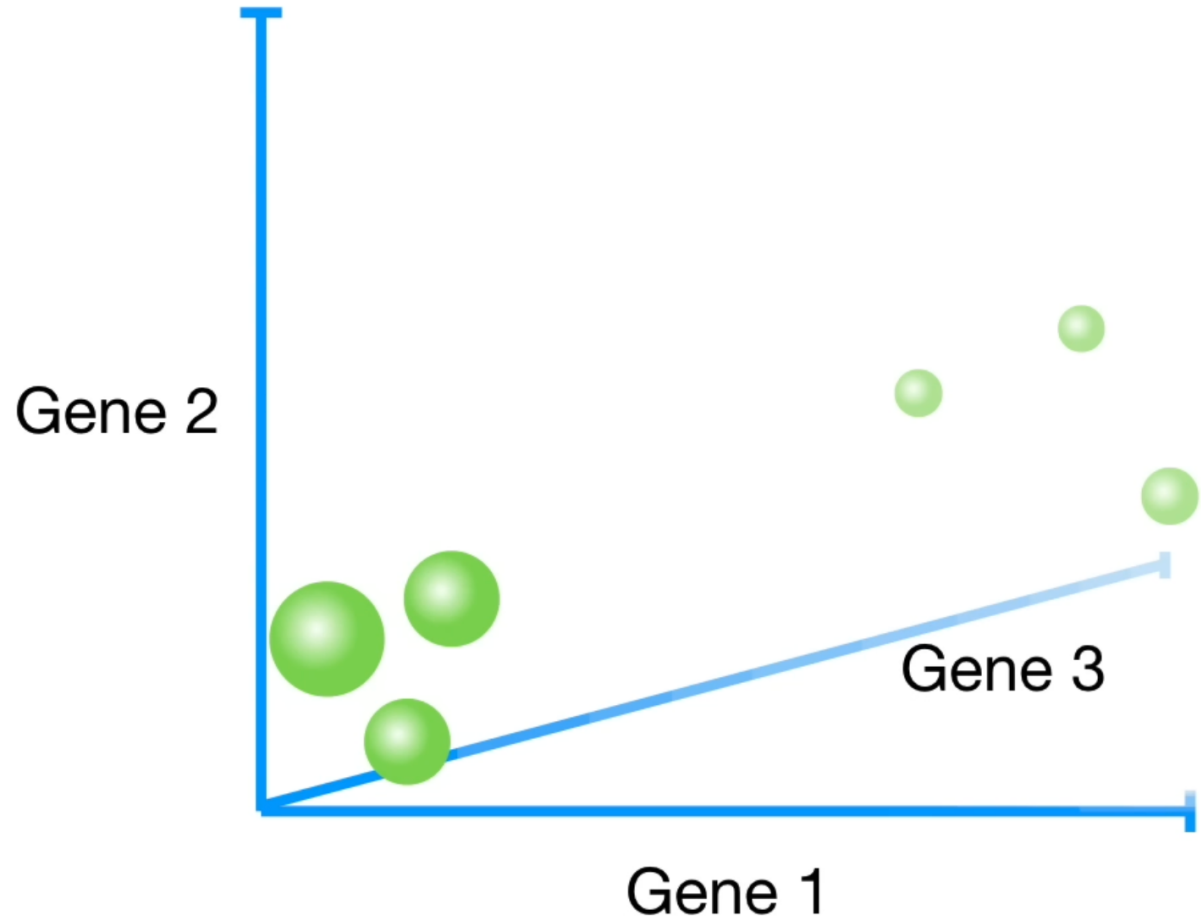


Then we rotate so that PC1 is horizontal and PC2 is vertical (this just makes it easier to look at).

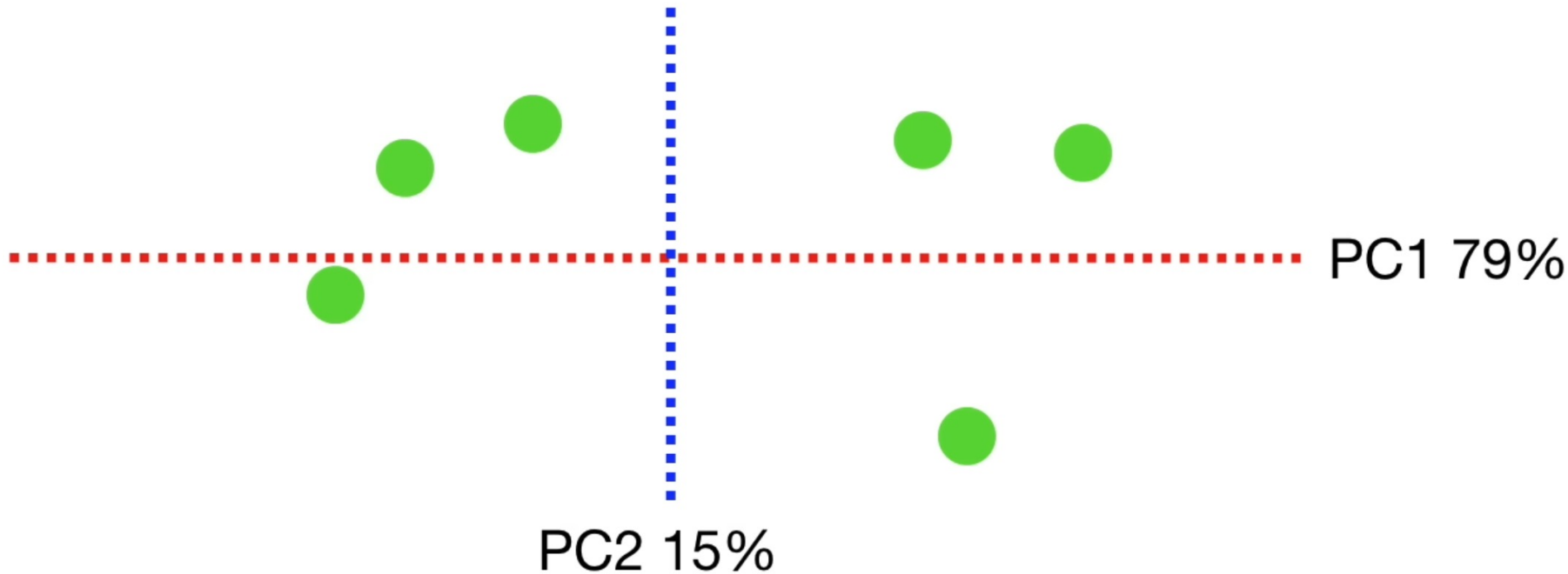



etc…etc…
PC1 79%
PC2 15%

To review, we started with an awkward 3-D graph that was kind of hard to read…

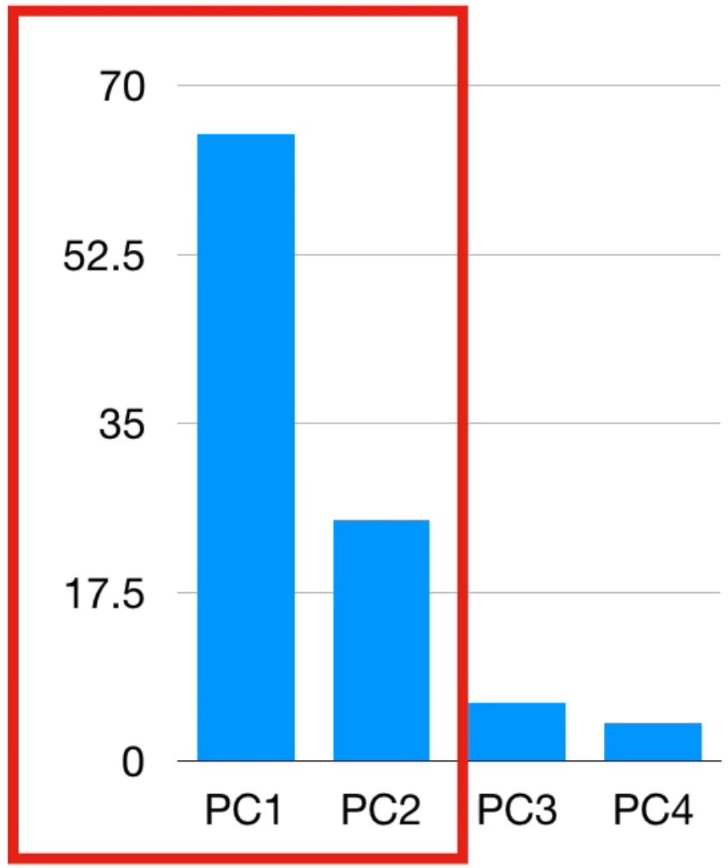


…lastly, we used PC1 and PC2 to draw a 2-Dimensional graph with the data.

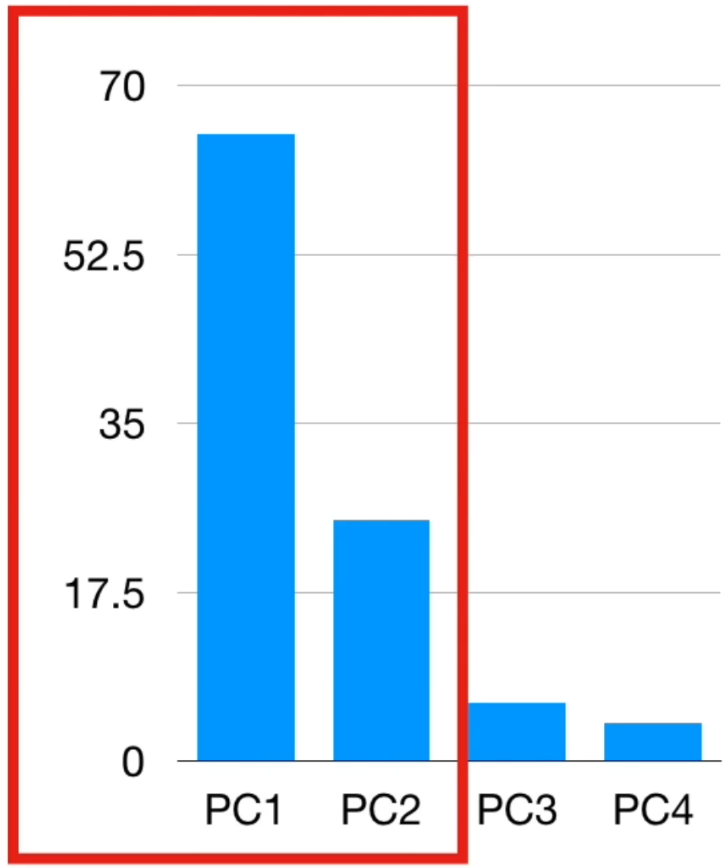


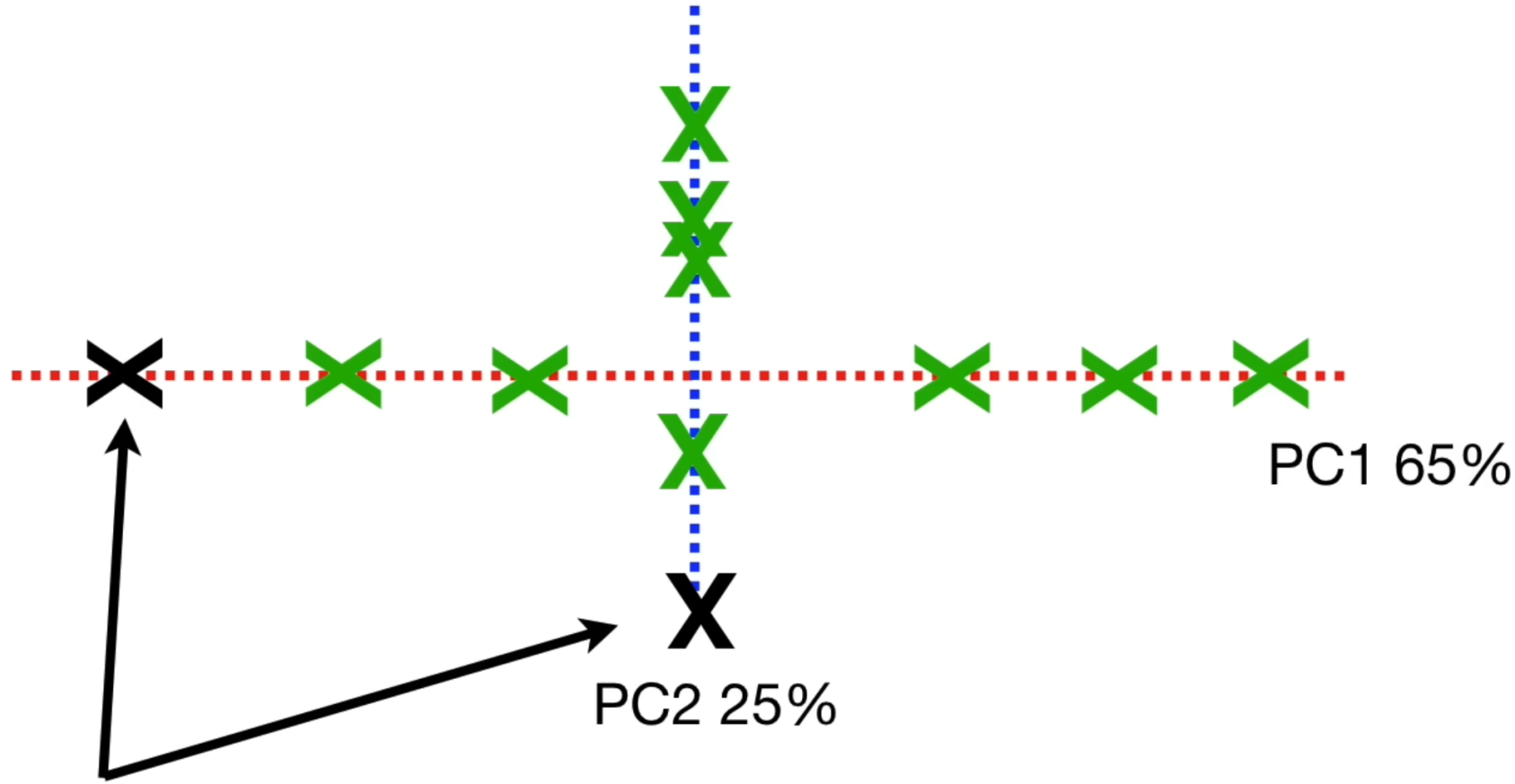
	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1
Gene 3	12	9	10	2.5	1.3	2
Gene 4	5	20	6	2	18	19

If we measured 4 genes per mouse, we would not be able to draw a 4-dimensional graph of the data…

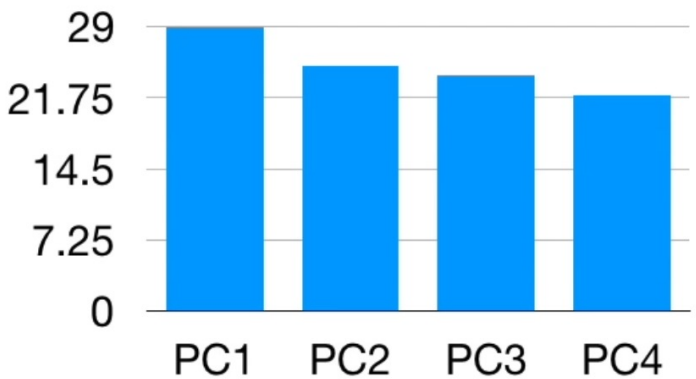


…in this case, PC1 and PC2 account for 90% of the variation, so we can just use those to draw a 2-dimensional PCA graph.



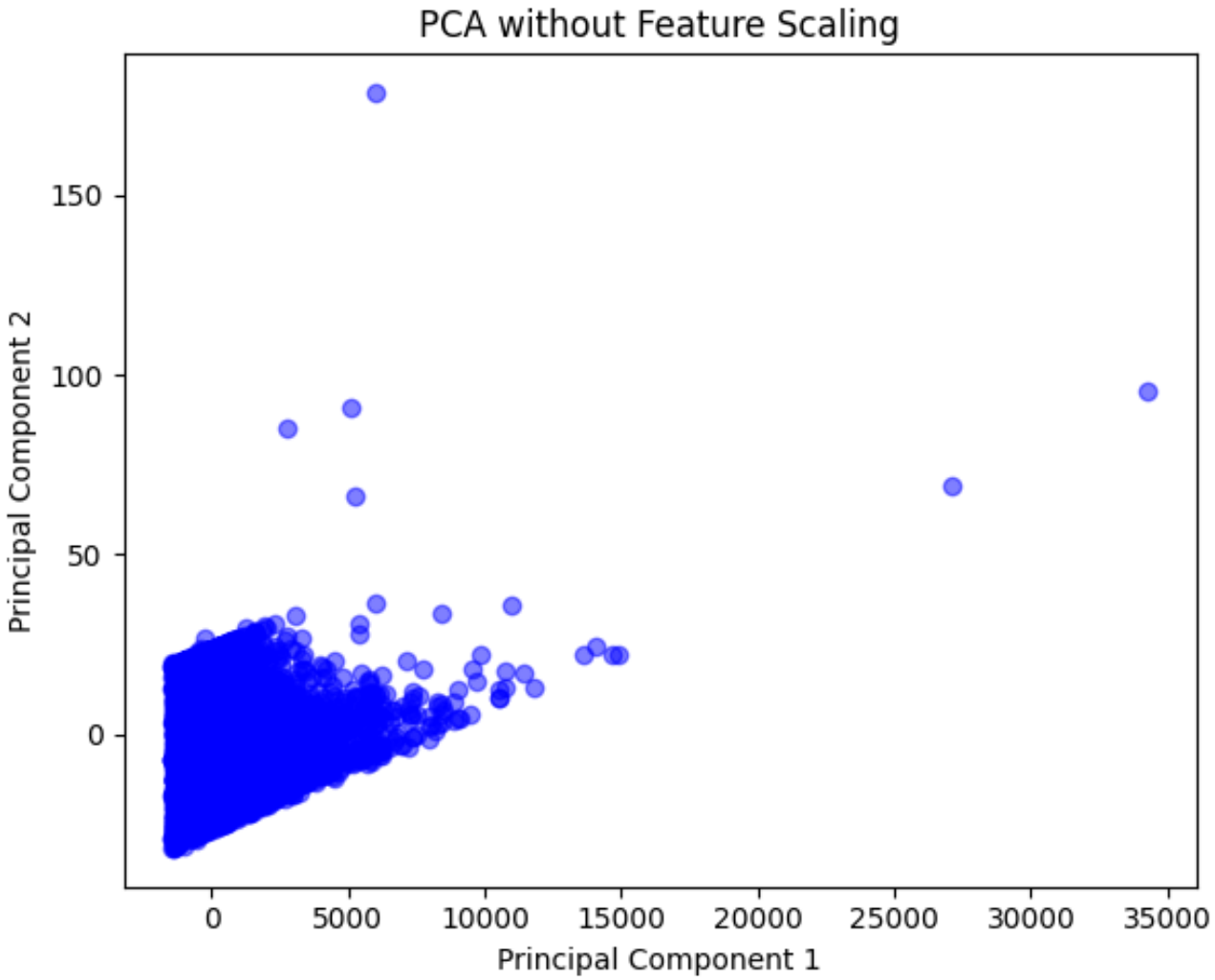


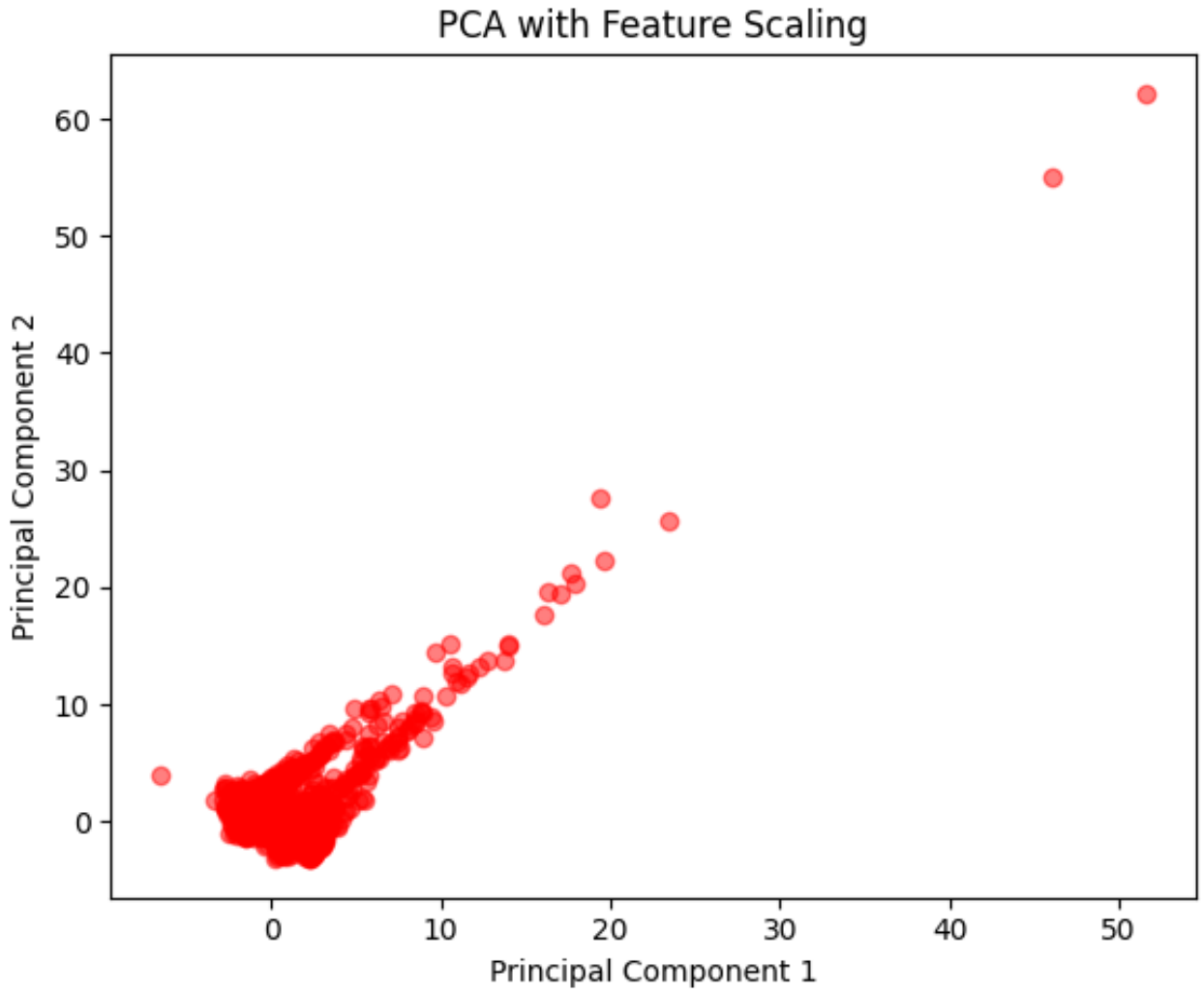
These two projected points correspond to Sample 2.



NOTE: If the scree plot looked like this, where PC3 and PC4 account for a substantial amount of variation, then just using the first 2 PCs would not create a very accurate representation of the data.

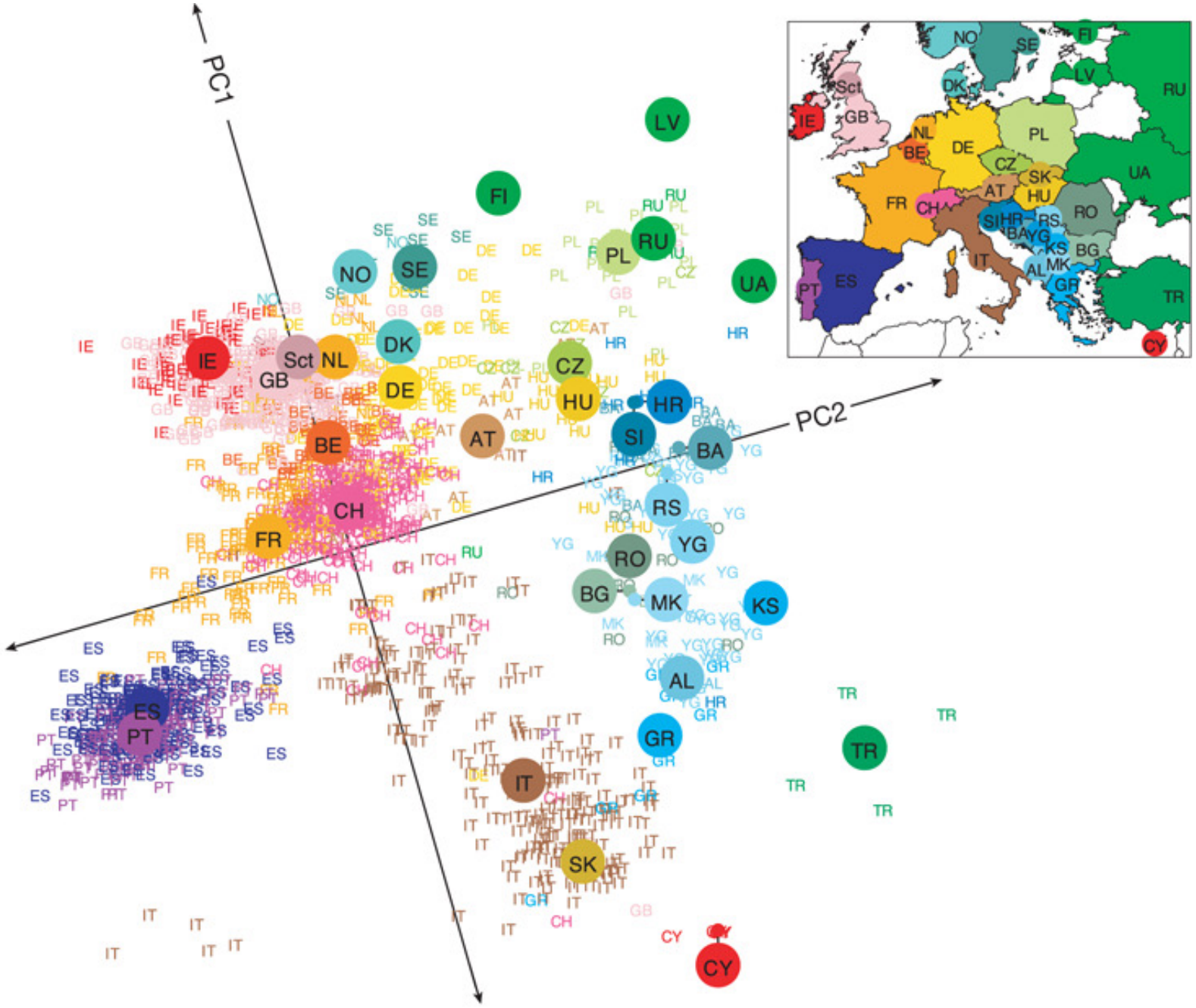
First example in which standardization matters! (California housing dataset)





A high-dimensional example

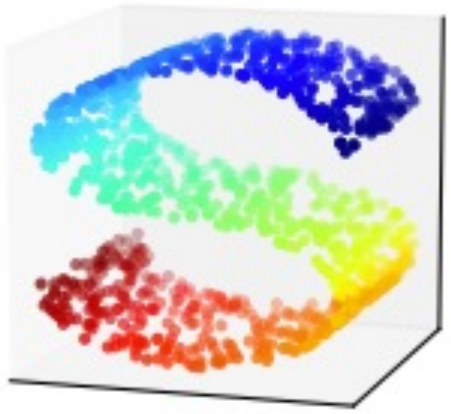
p = 197k (loci)
n = 1387 (individuals)

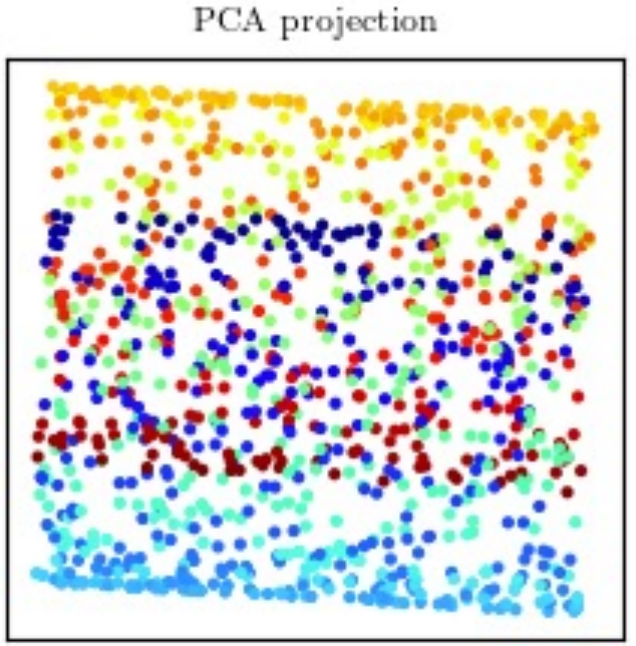


Genes mirror geography within Europe – Nature 2008 https://www.nature.com/articles/nature07331

Issues with PCA

- PCA takes care of mapping distant objects far from each other: local structure is not preserved!
- PCA does not take manifolds into account…PCA does not take manifolds into account…





T-SNE

t-Distributed Stochastic Neighbor Embedding

Huge kudos to Joshua Starmer!

T-SNE in a nutshell

- T-distributed Stochastic Neighbor Embedding (2008).
- Idea: map similar objects in the high dimensional space into close points in a low dimensional space
- t-SNE Takes care of small local distances (focus on local structure)
- Open-source code. For instance, it is available in scikit-learn.

van der Maaten, L. & Hinton, G. (2008). Visualizing Data using t-SNE . *Journal of Machine Learning Research*, 9, 2579—260, https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

Theory behind T-SNE – 1 (optional)

- Starting point: N high-dim objects $x_1, \ldots, x_N$
- We consider this "distance" in high-dim space (it only cares about local similarity):

$$p_{ij} = \frac{exp(-\|x_i - x_j\|^2/2\sigma^2)}{\sum_k \sum_{l \neq k} exp(-\|x_k - x_l\|^2/2\sigma^2)}$$

- If you look at it as a probability, it means that **the probability of picking two points is higher when they are closer**

Theory behind T-SNE – 2 (optional)

- Actually, we consider this conditional distribution:

$$p_{j|i} = \frac{exp(-\|x_i - x_j\|^2 / 2{\sigma_i}^2)}{\sum_{k \neq i} exp(-\|x_i - x_k\|^2 / 2{\sigma_i}^2)}$$

- We set the σ_i so that the conditional has a fixed perplexity (fixed number of points in the mode of the Gaussian to account for different densities of points in space)

Theory behind T-SNE – 3 (optional)

- We make the “distance” simmetric:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

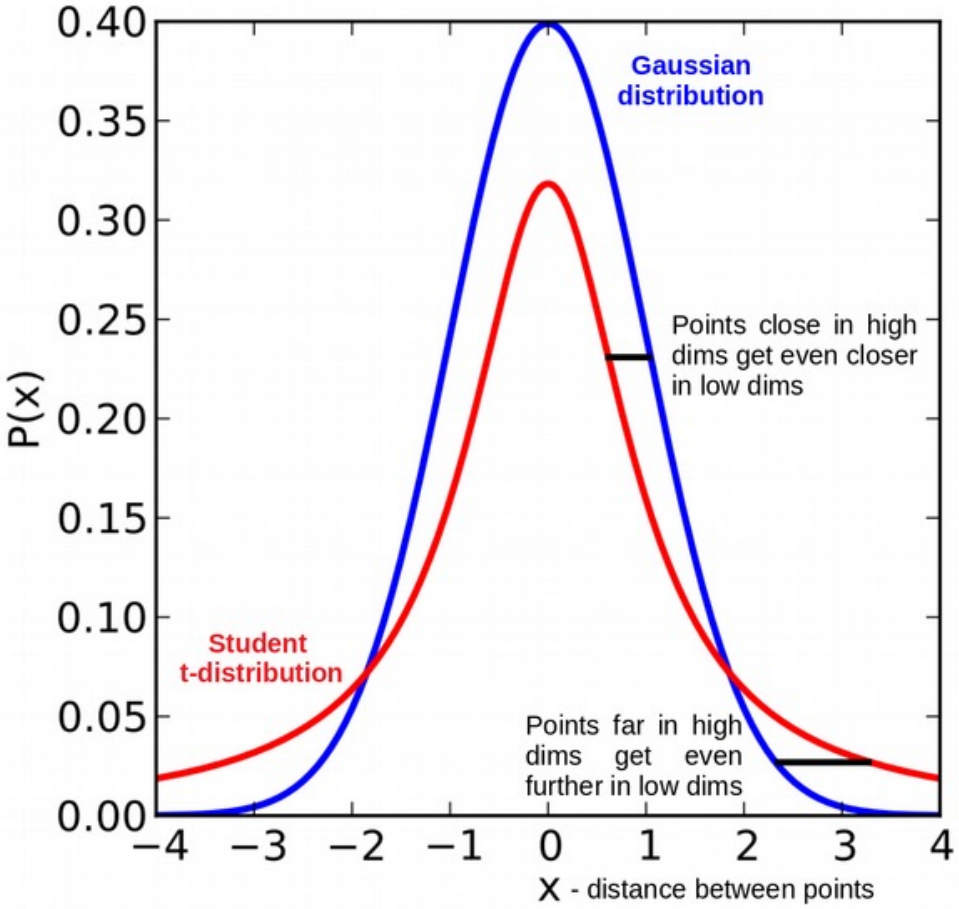
- Now, let’s look at the target low dimensional space…

Theory behind T-SNE – 4 (optional)

- We introduce a distance based on Student-t distribution:

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|y_k - y_l\|^2)^{-1}}$$

Heavy tails, q_{ij} should account for global structure, too…



Theory behind T-SNE – 5 (optional)

- We want to make p_{ij} and q_{ij} as similar as possible, so the low dim space has a similar structure. KL divergence:

$$KL(p||q) = \sum_i \sum_{j \neq i} p_{ij} log \left(\frac{p_{ij}}{q_{ij}} \right)$$

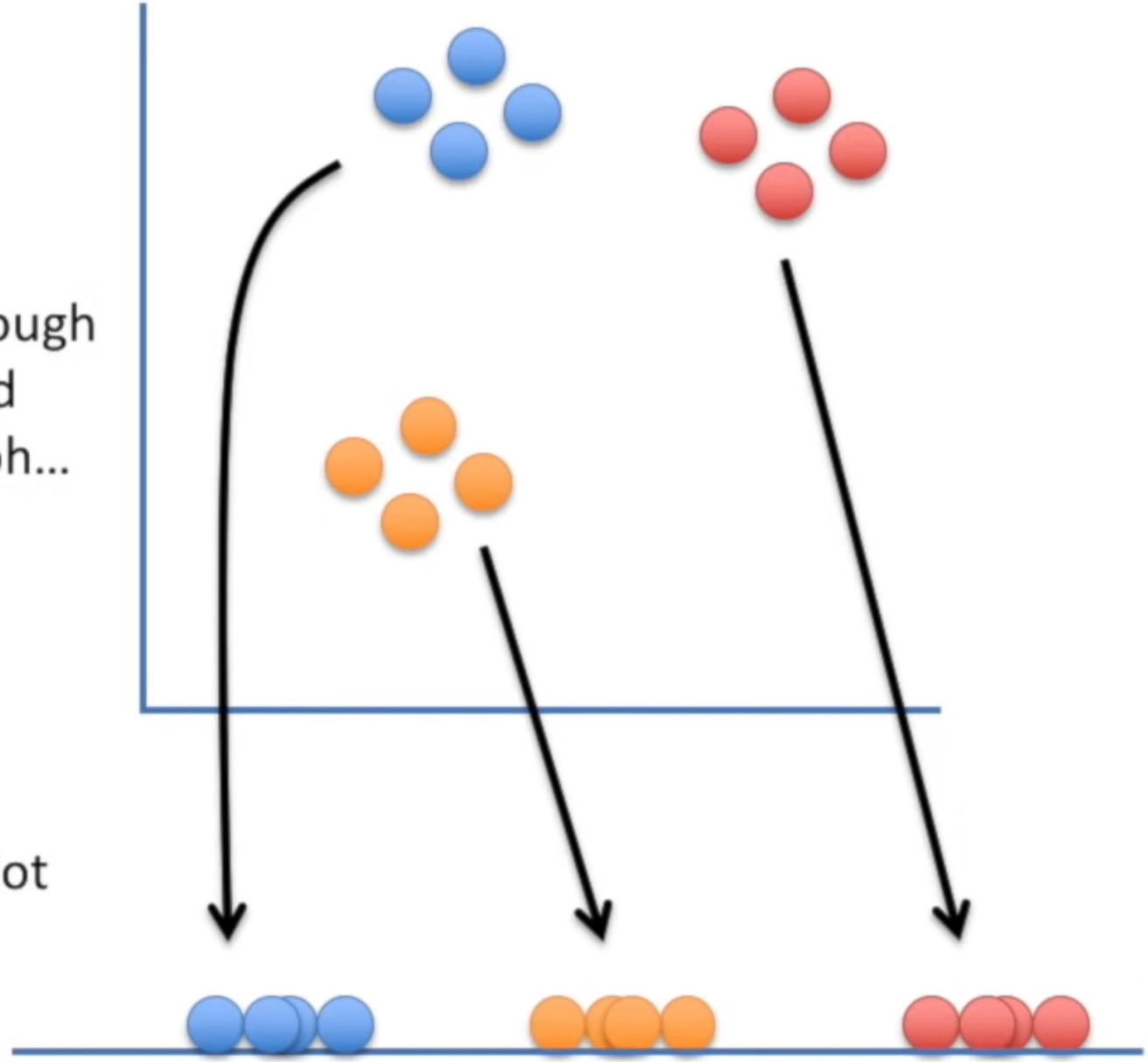
- Intuition:
  - Large p_ij modelled by small q_ij? Big penalty: **close points are mapped to close points**
  - Small p_ij modelled by large q_ij? Small penalty: **far points may end up close in the low dimensional space**

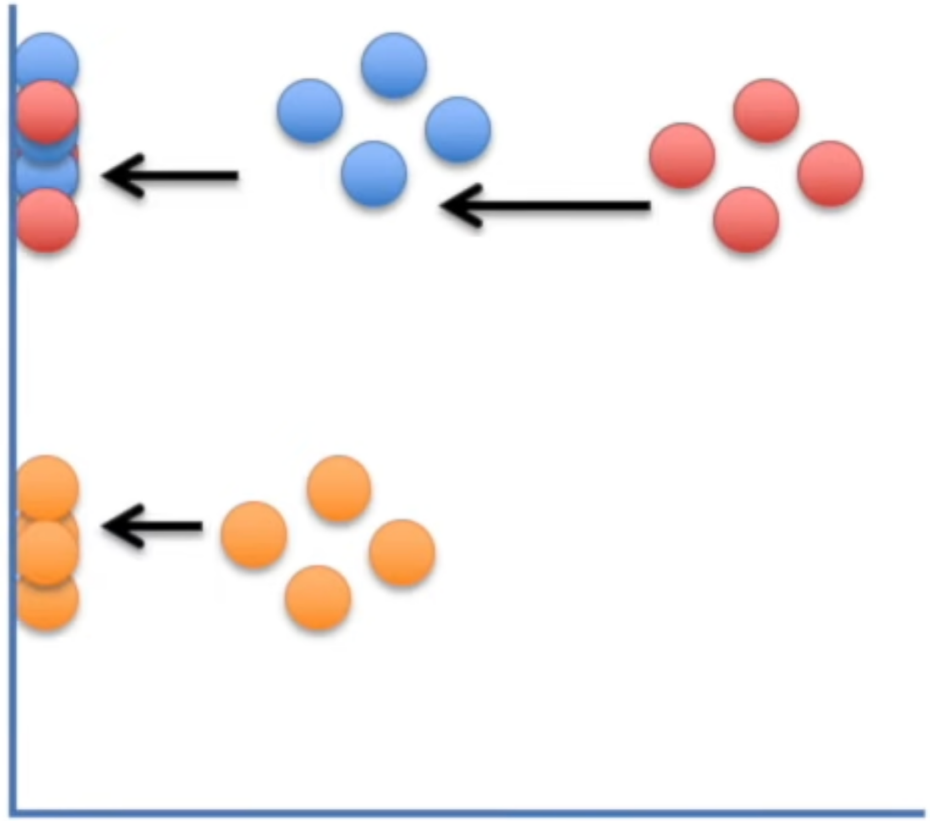
T-SNE mainly preserves local similarity structure!

Here's a basic 2-D scatter plot.

Let's do a walk through of how t-SNE would transform this graph…

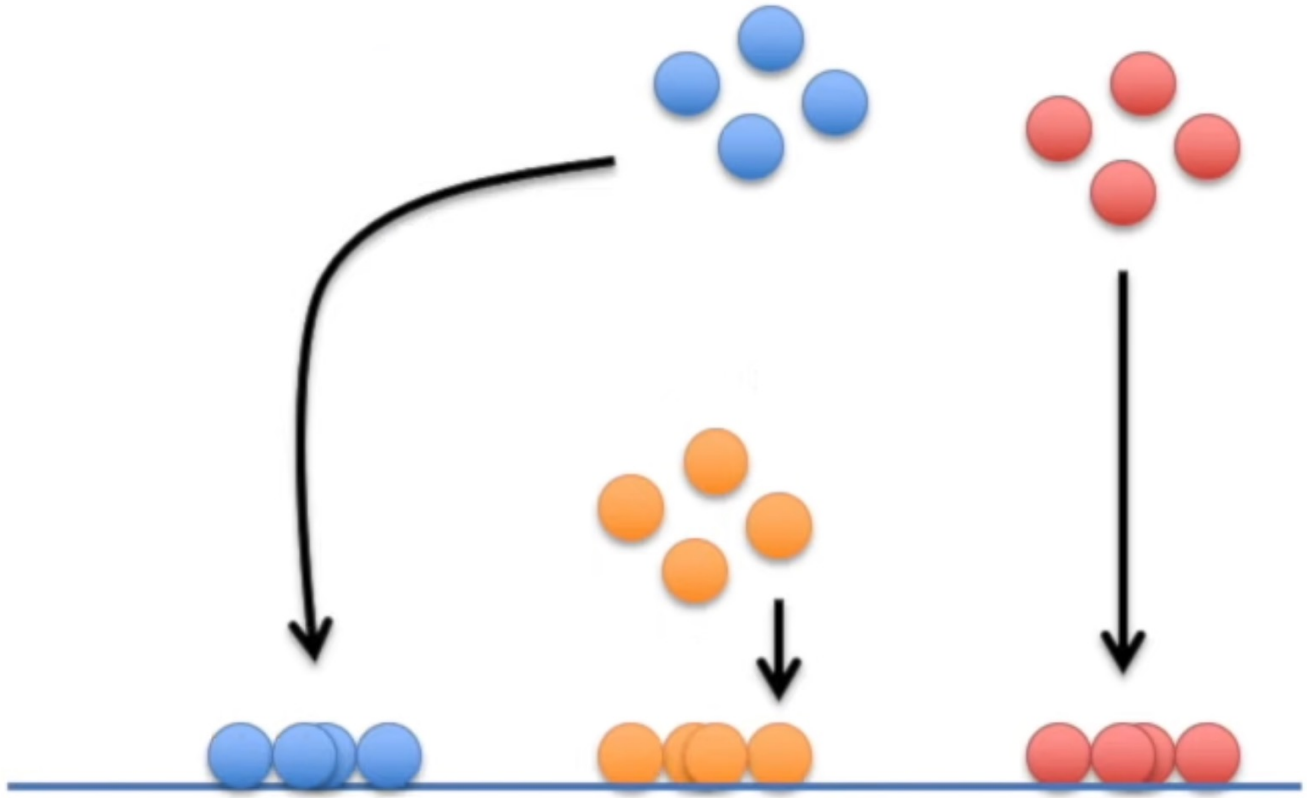
…into a flat, 1-D plot on a number line.





NOTE: If we just projected the data onto one of the axes, we'd just get a big mess that doesn't preserve the original clustering.

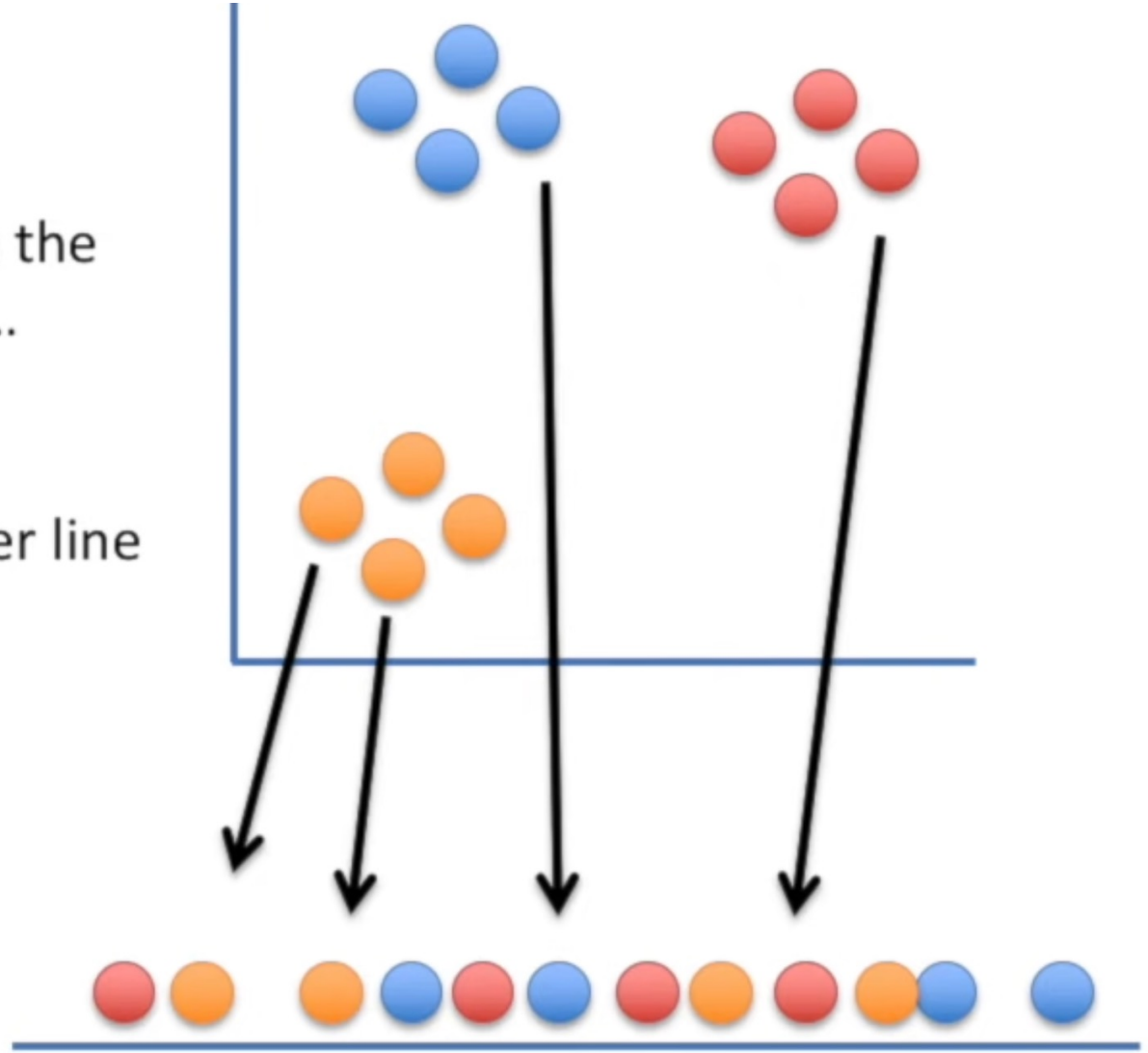
Think of this as a really high-dimensional problem!

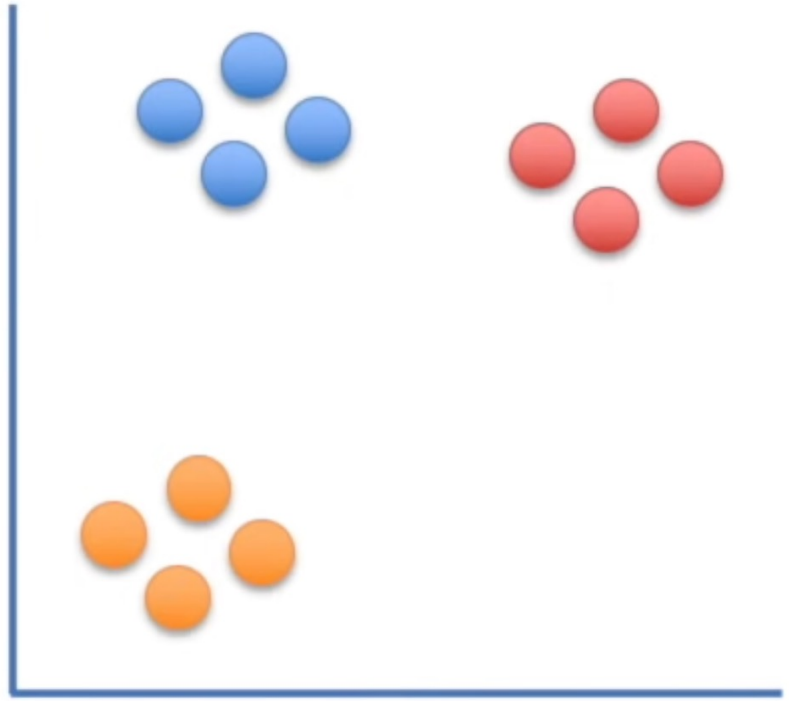


What t-SNE does is find a way to project data into a low dimensional space (in this case, the 1-D number line) so that the clustering in the high dimensional space (in this case, the 2-D scatter plot) is preserved.

We'll start start with the original scatter plot…

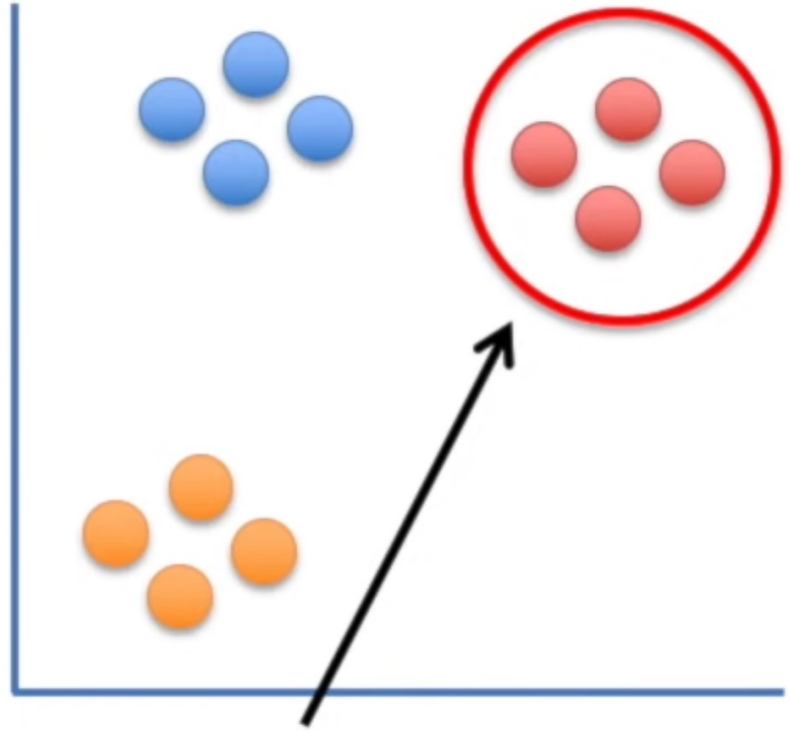
… then we'll put the points on the number line in a random order.





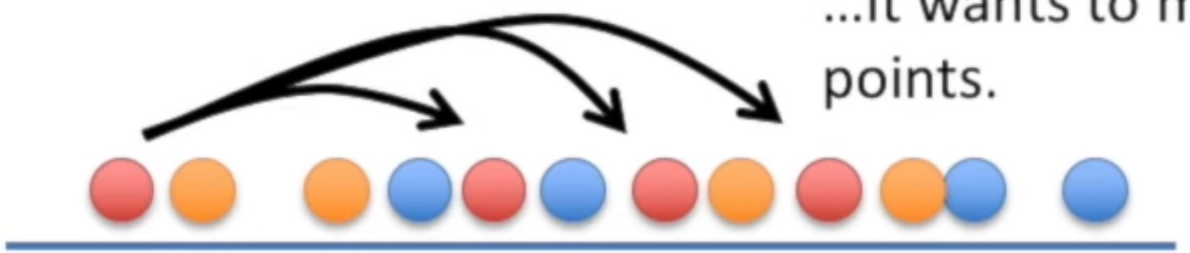
Let’s figure out where to move this first point…

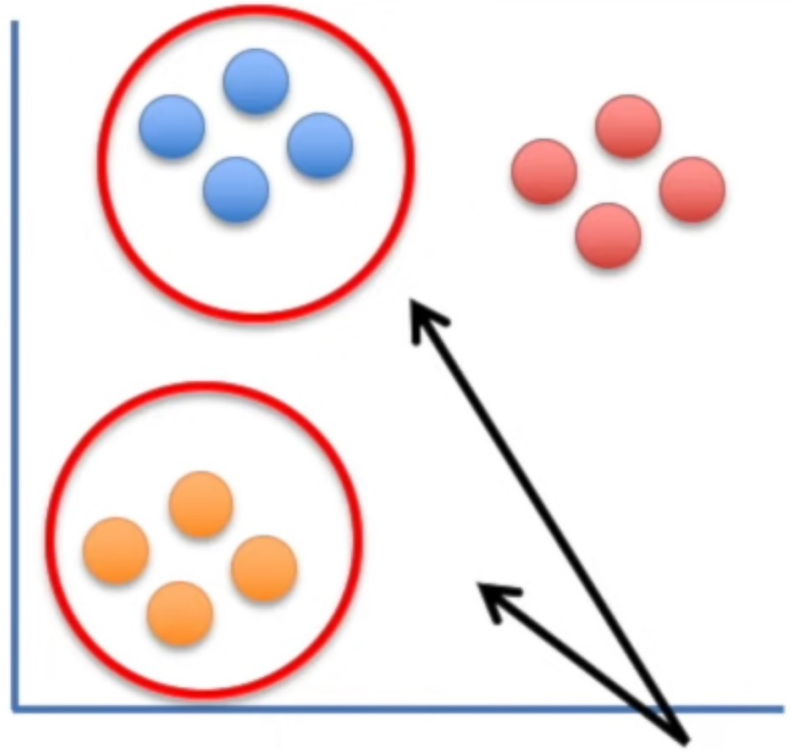
Should it move a little to the left or a little to the right?



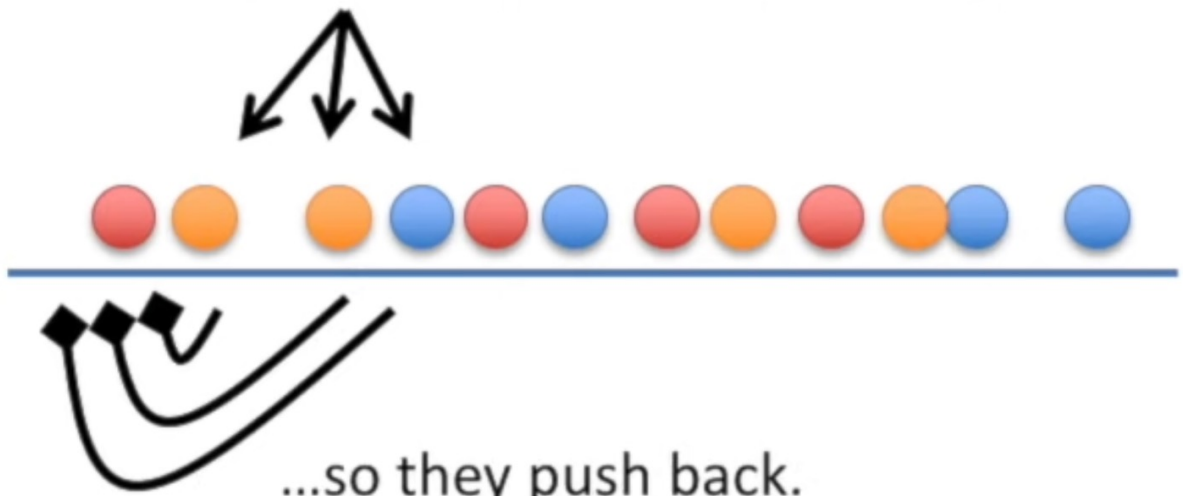
Because it is part of this cluster…

…it wants to move closer to these points.

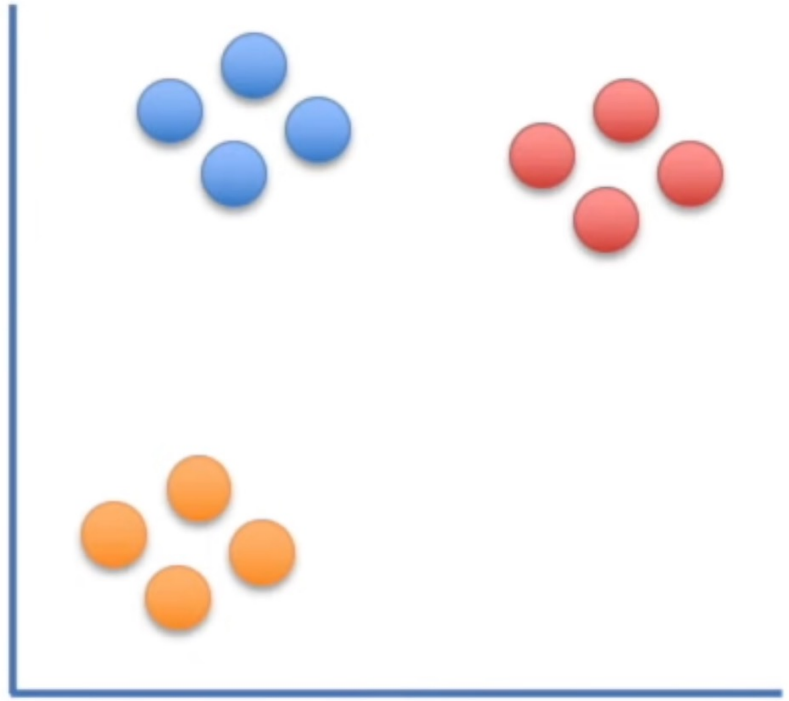




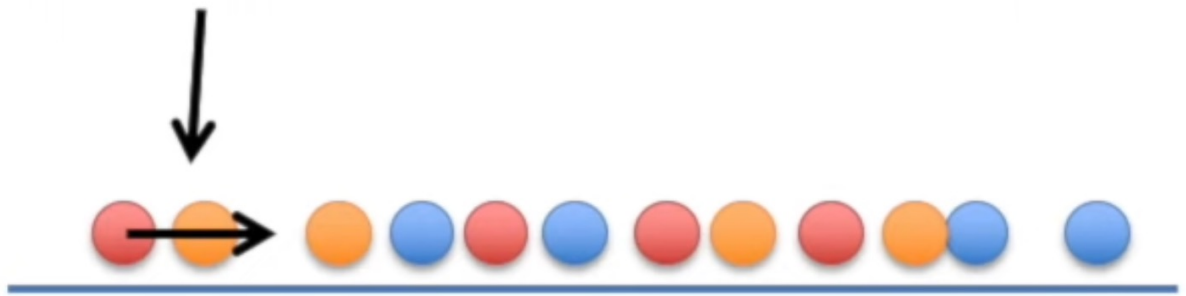
But at the same time, these points… …are far away in the scatter plot…

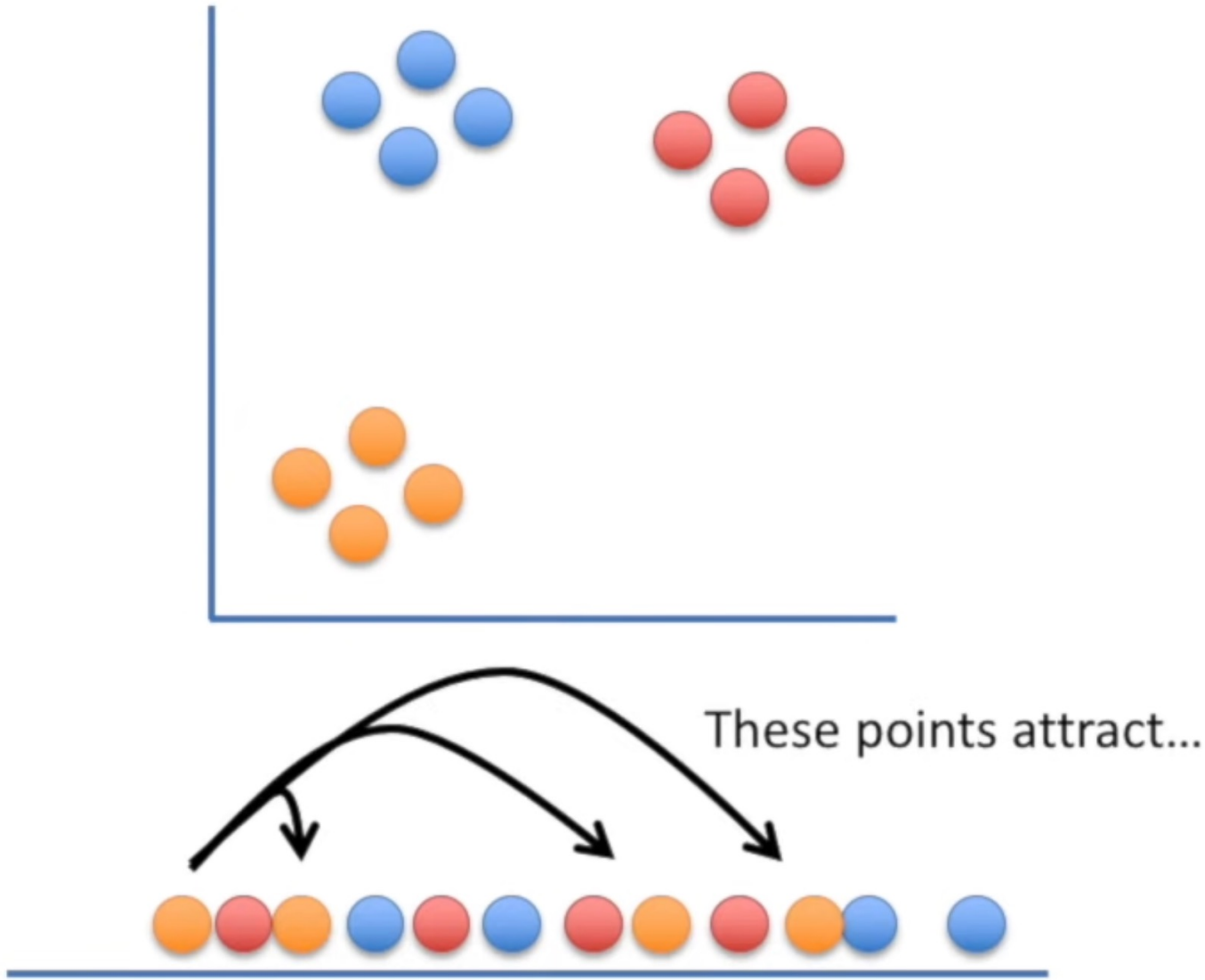


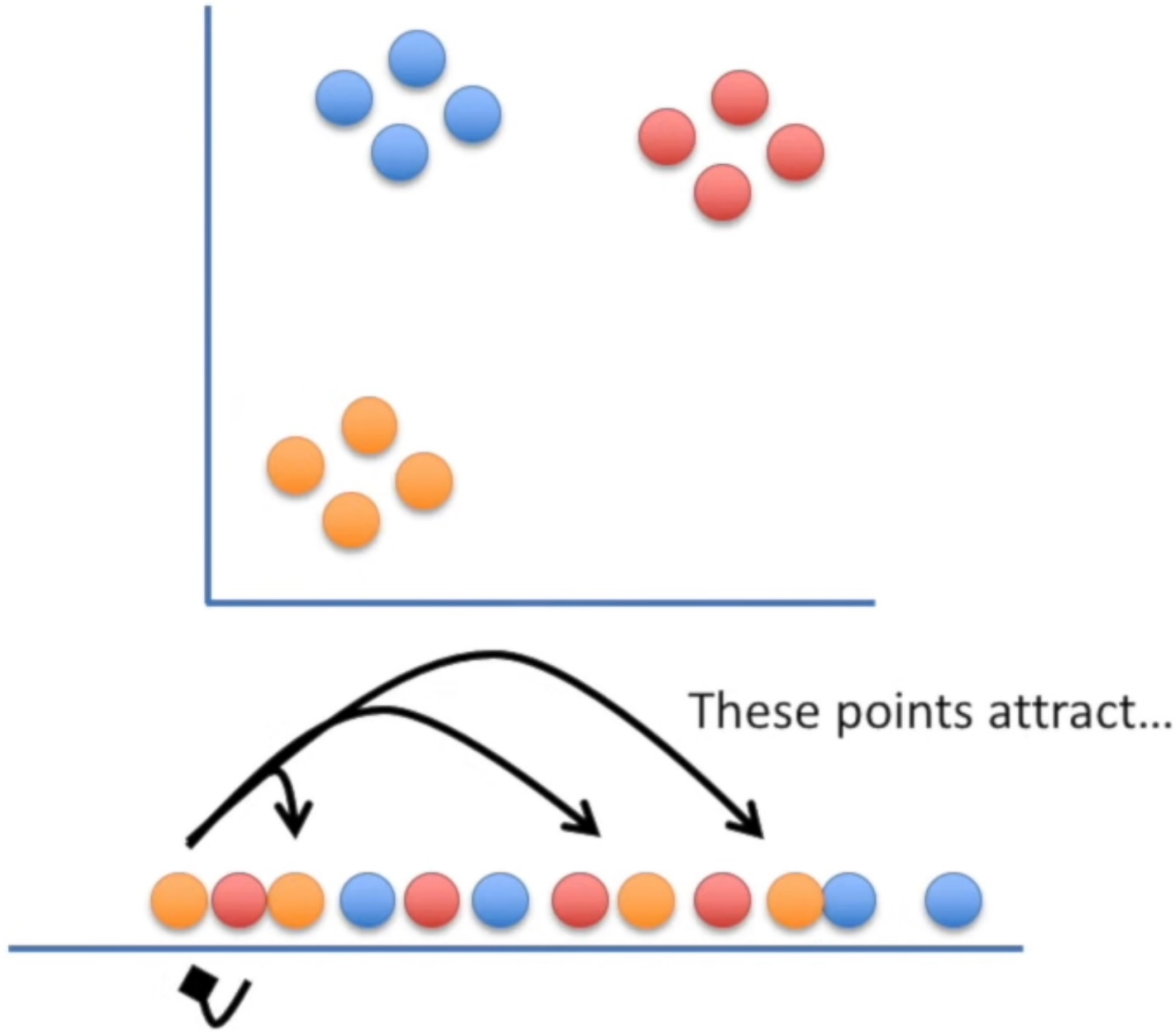
…so they push back.

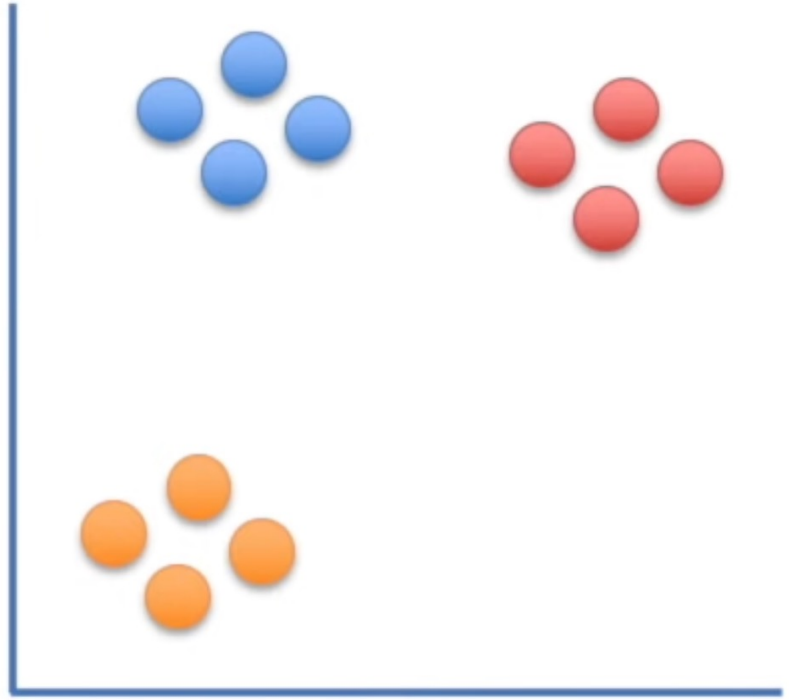


In this case, the attraction is strongest, so the point moves a little to the right.

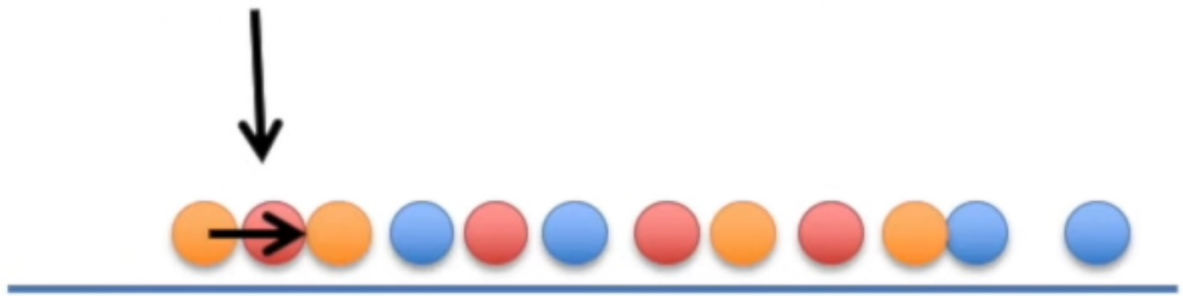


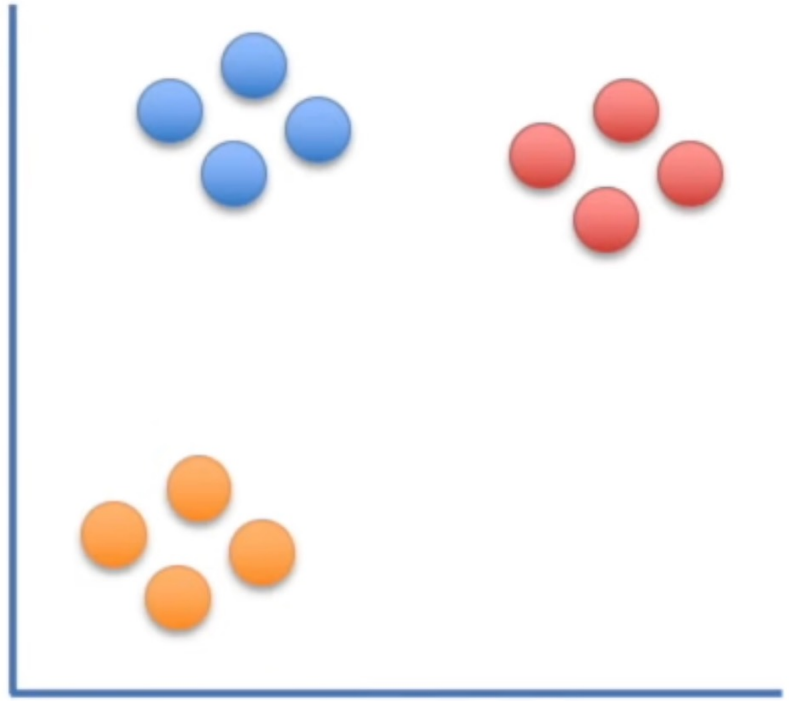

These points attract…


These points attract…
…and this point repels a little bit.

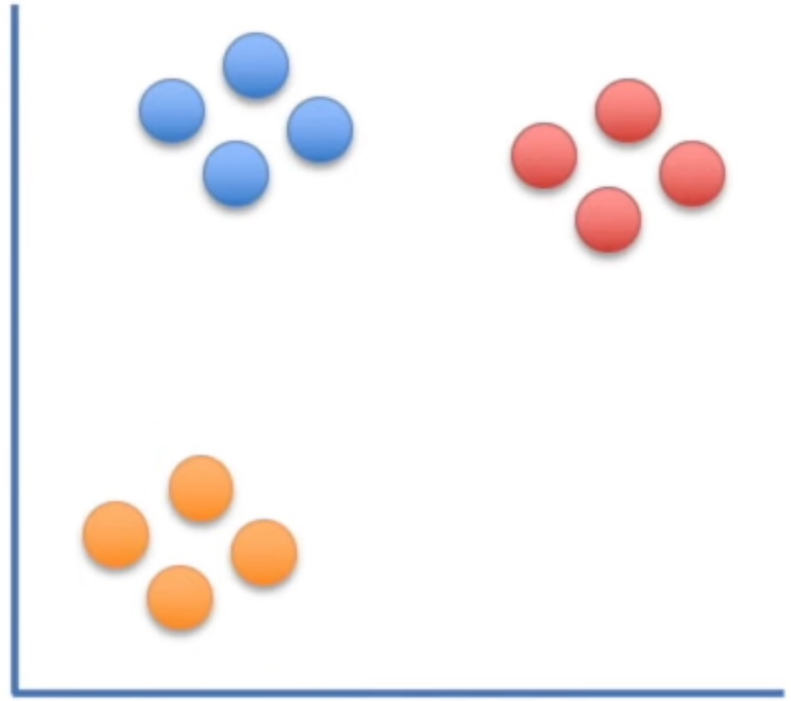


So it moves a little to closer to the other orange points.



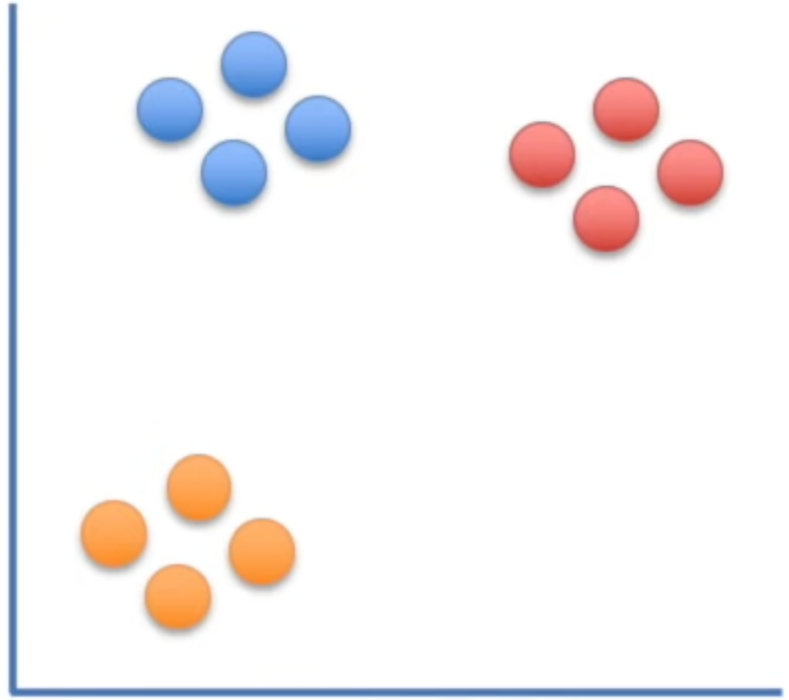


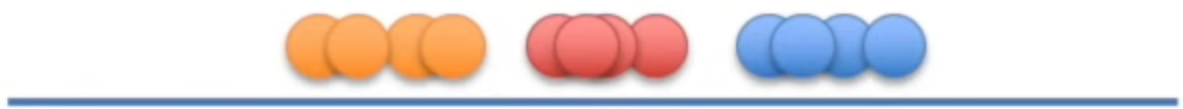
At each step, a point on the line is attracted to points it is near in the scatter plot, and repelled by points it is far from…



At each step, a point on the line is attracted to points it is near in the scatter plot, and repelled by points it is far from…

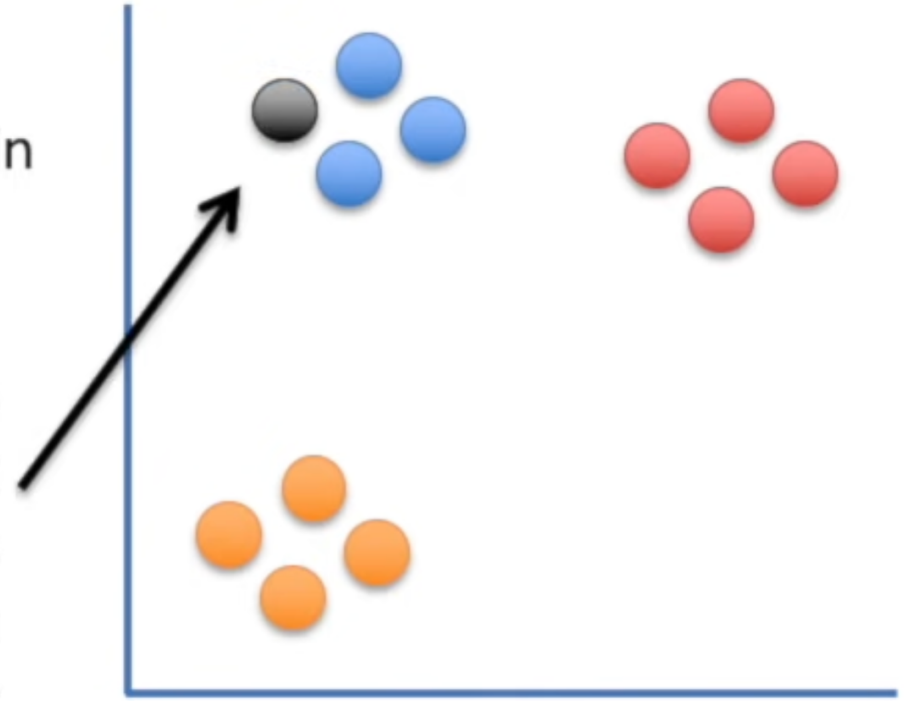
Now that we've seen the what t-SNE tries to do, let's dive into the nitty-gritty details of how it does what it does.



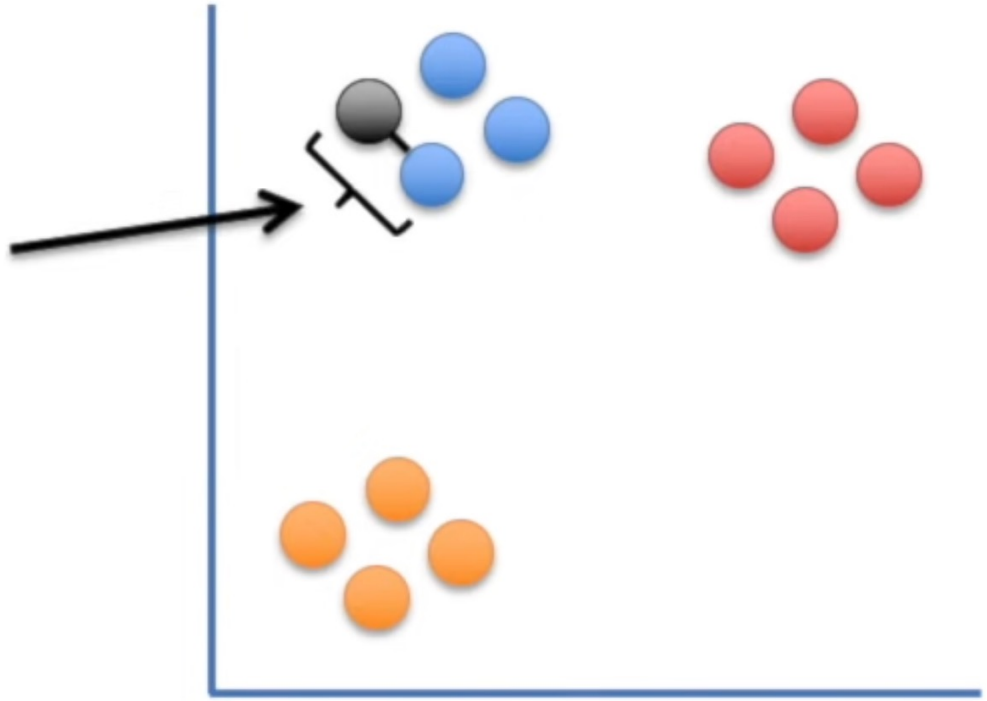


Step 1: Determine the "similarity" of all the points in the scatter plot.

For this example, let's focus on determining the similarities between this point and all of the other points.

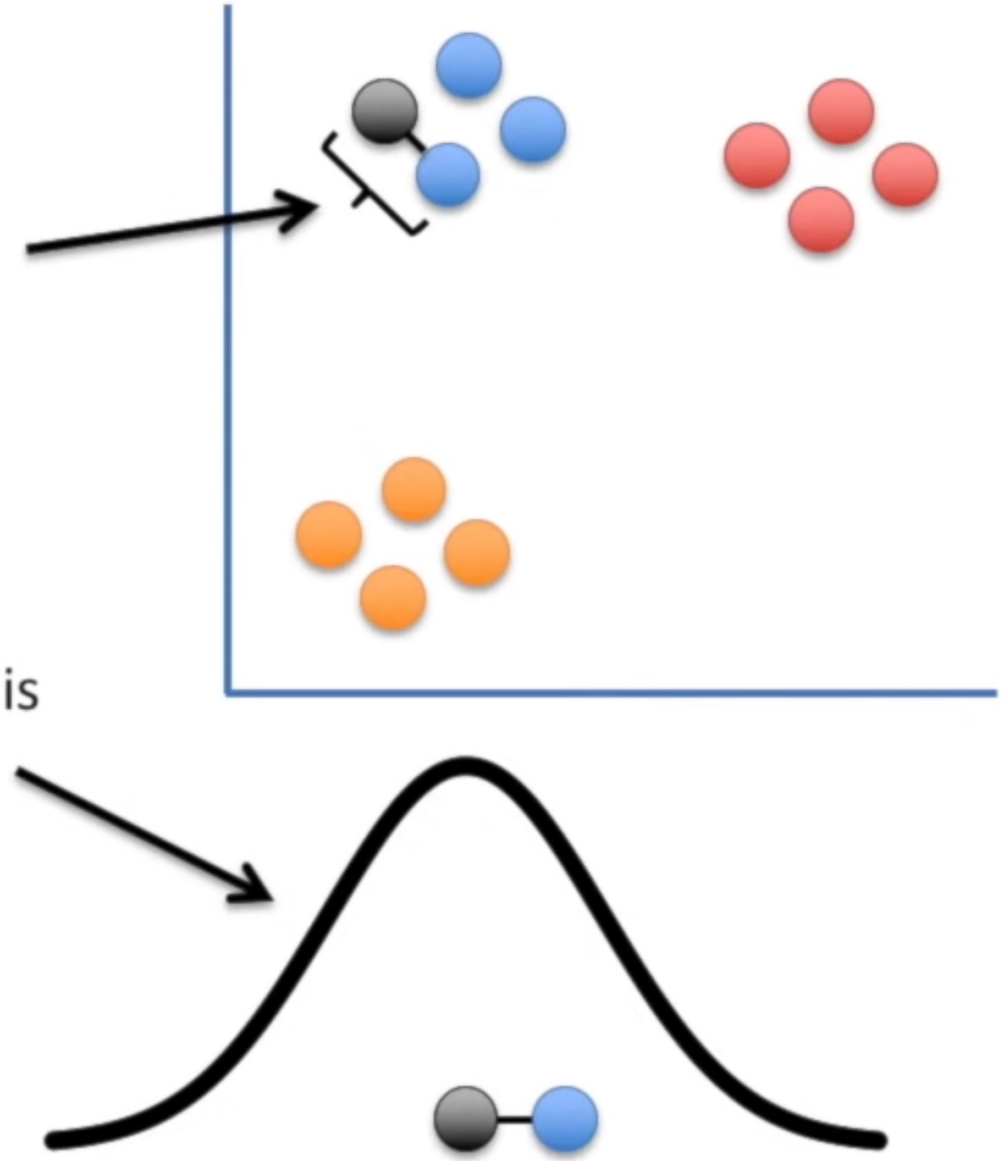


First, measure the distance between two points…

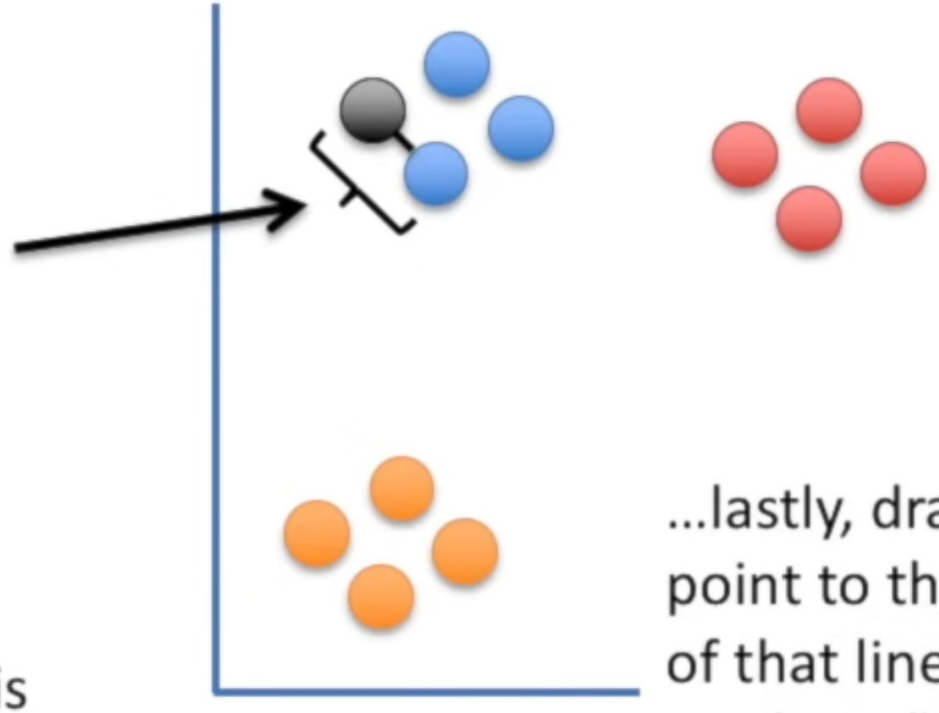


First, measure the distance between two points…

Then plot that distance on a normal curve that is centered on the point of interest…

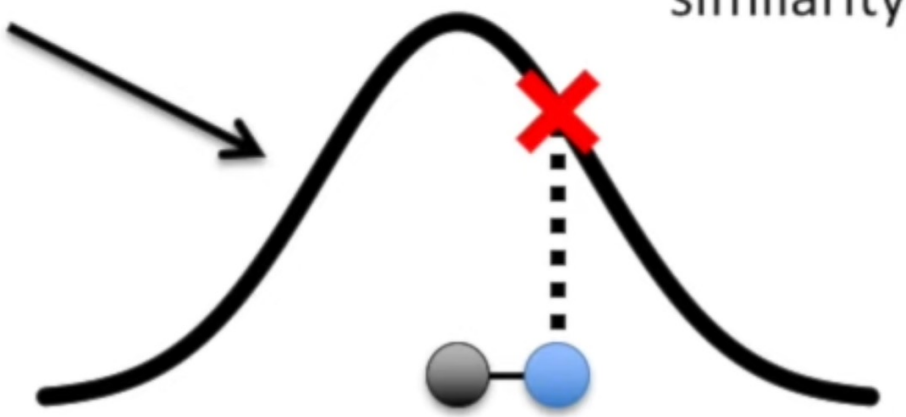


First, measure the distance between two points…

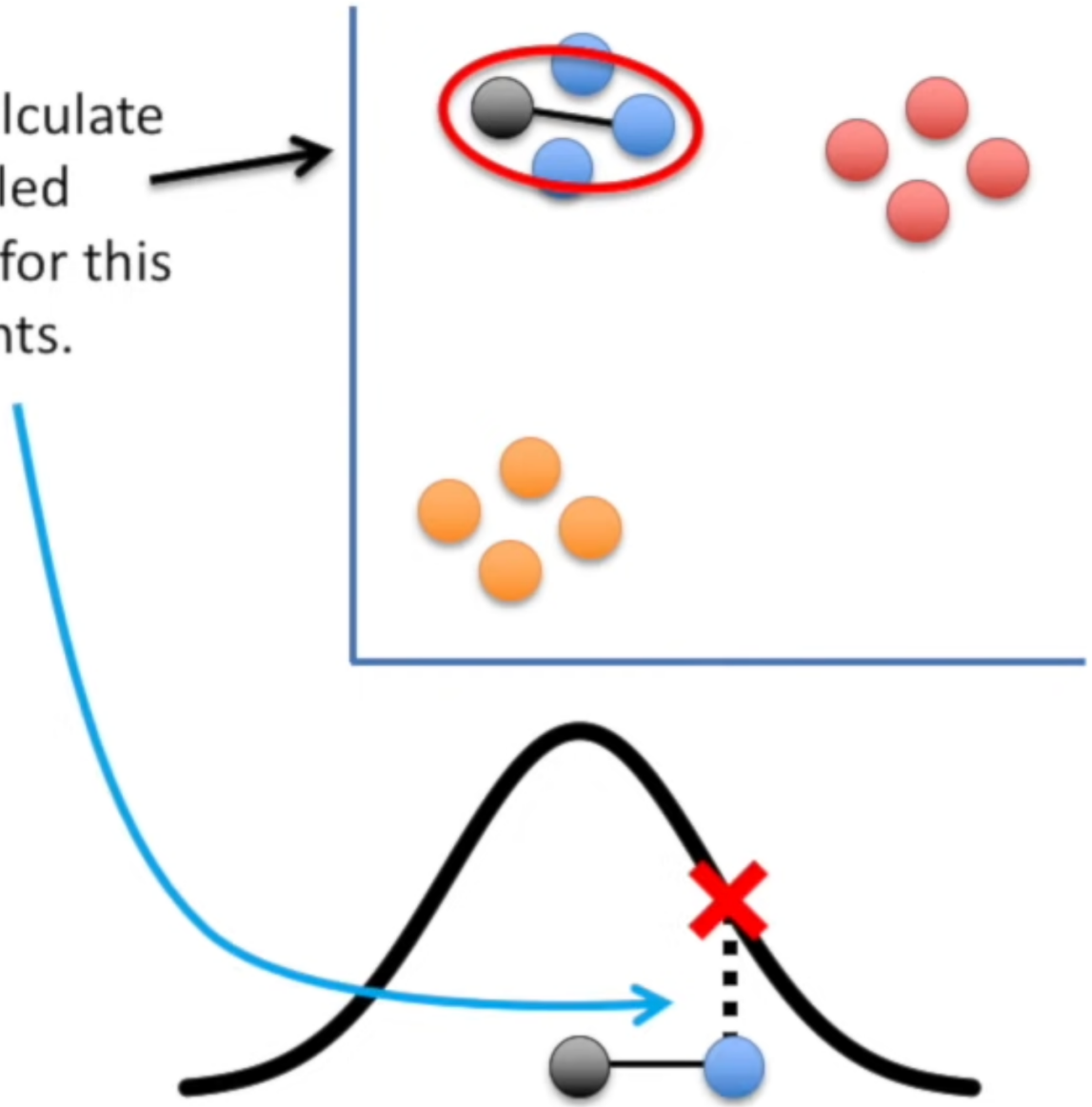


…lastly, draw a line from the point to the curve. The length of that line is the “unscaled similarity”.

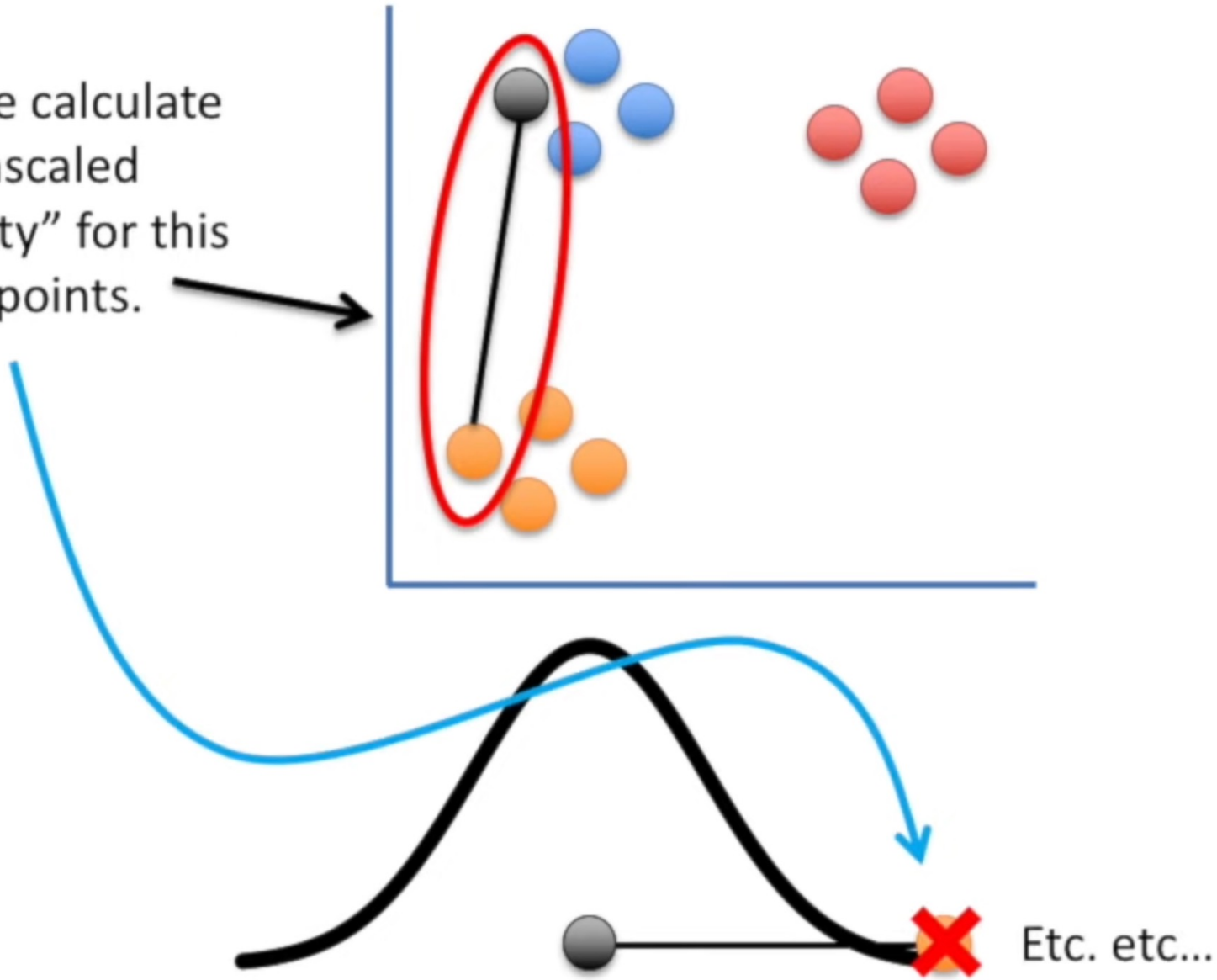
Then plot that distance on a normal curve that is centered on the point of interest…



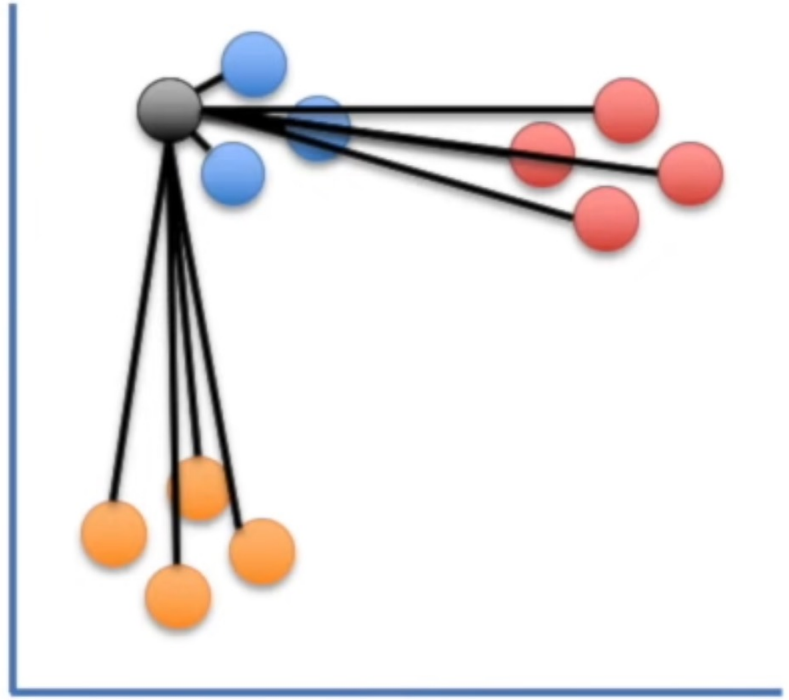
Now we calculate the “unscaled similarity” for this pair of points.



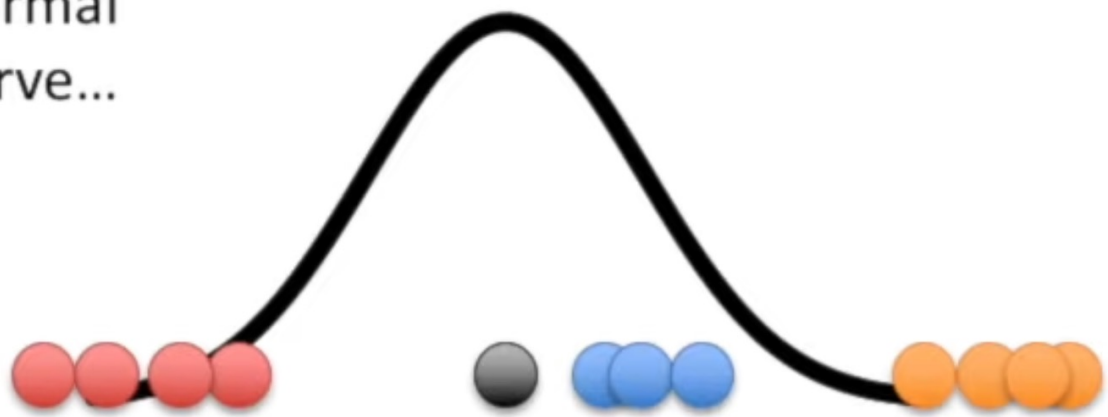
Now we calculate the “unscaled similarity” for this pair of points.



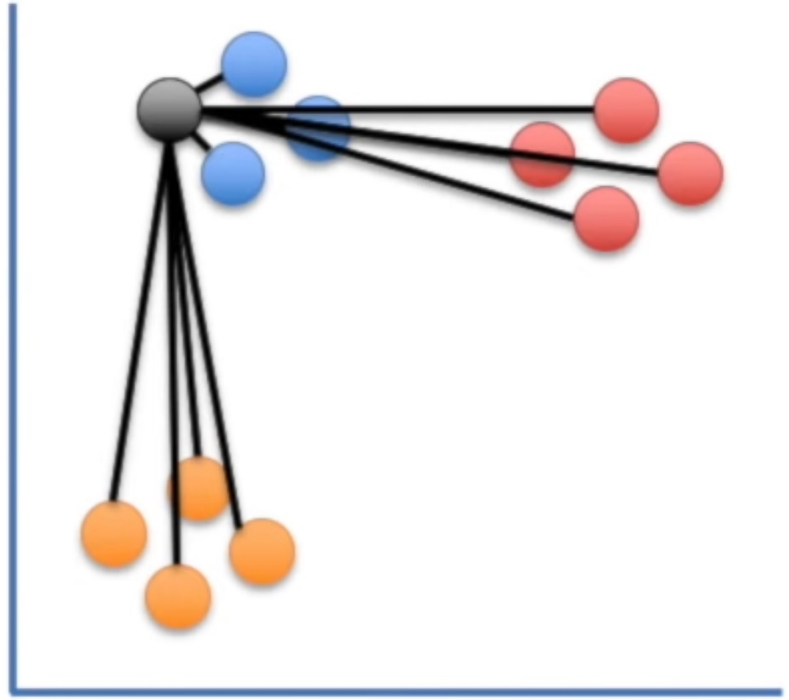
Ultimately, we measure the distances between all of the points and the point of interest…



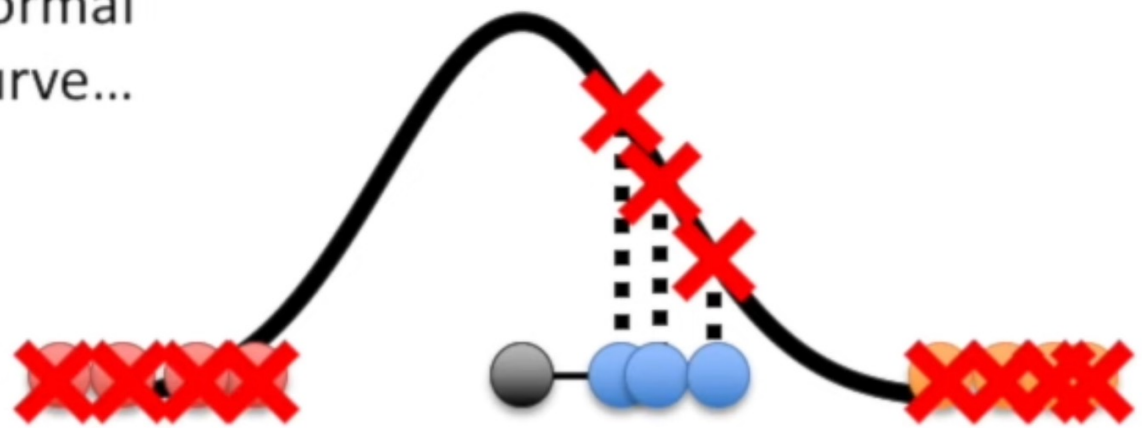
Plot them on the normal curve…



Ultimately, we measure the distances between all of the points and the point of interest…

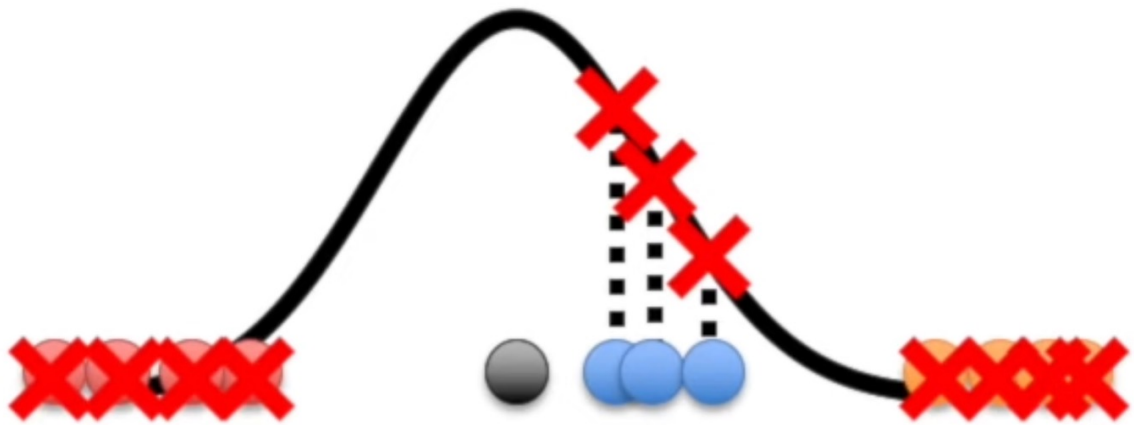


Plot them on the normal curve…

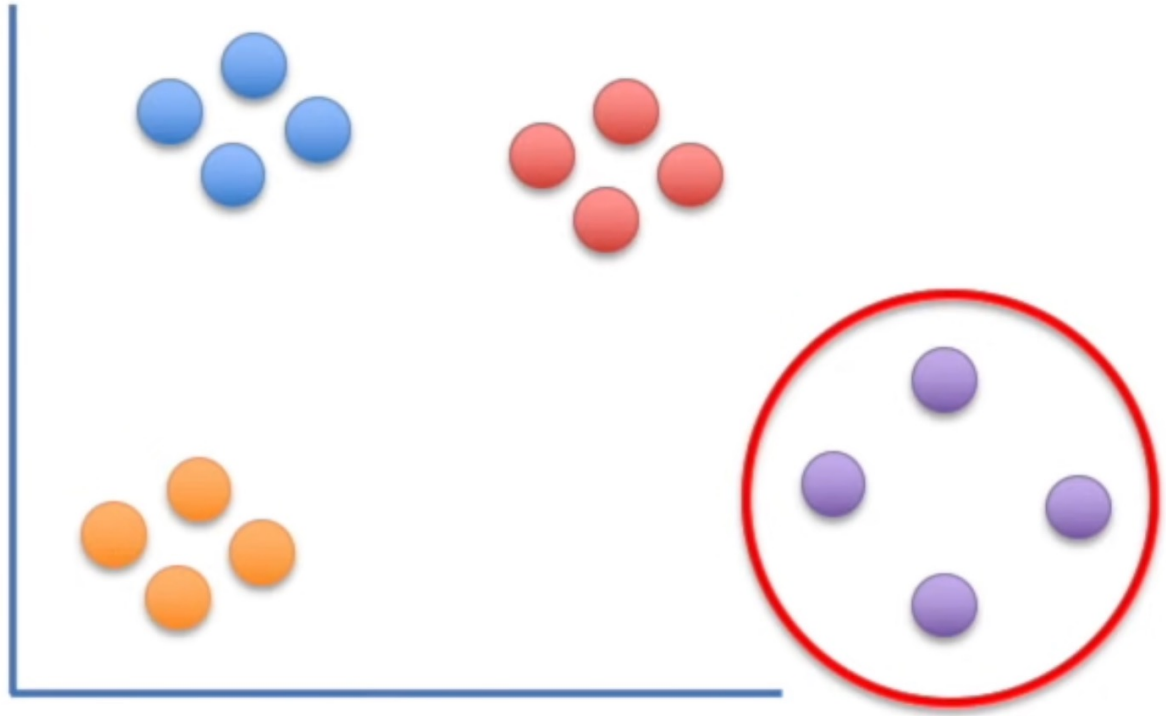


…and then measure the distances from the points to the curve to get the unscaled similarity scores with respect to the point of interest.

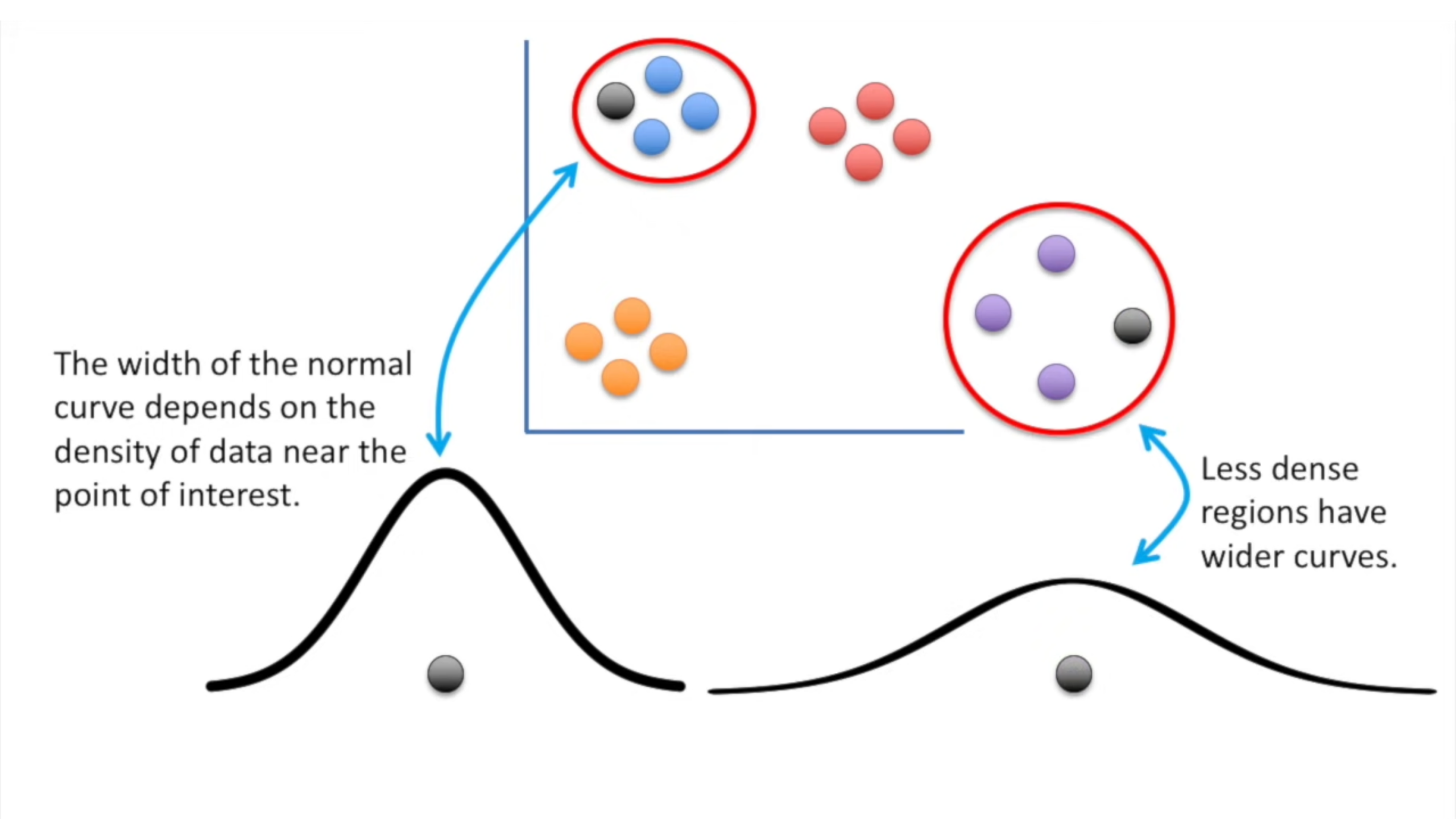
The next step is to scale the unscaled similarities so that they add up to 1.

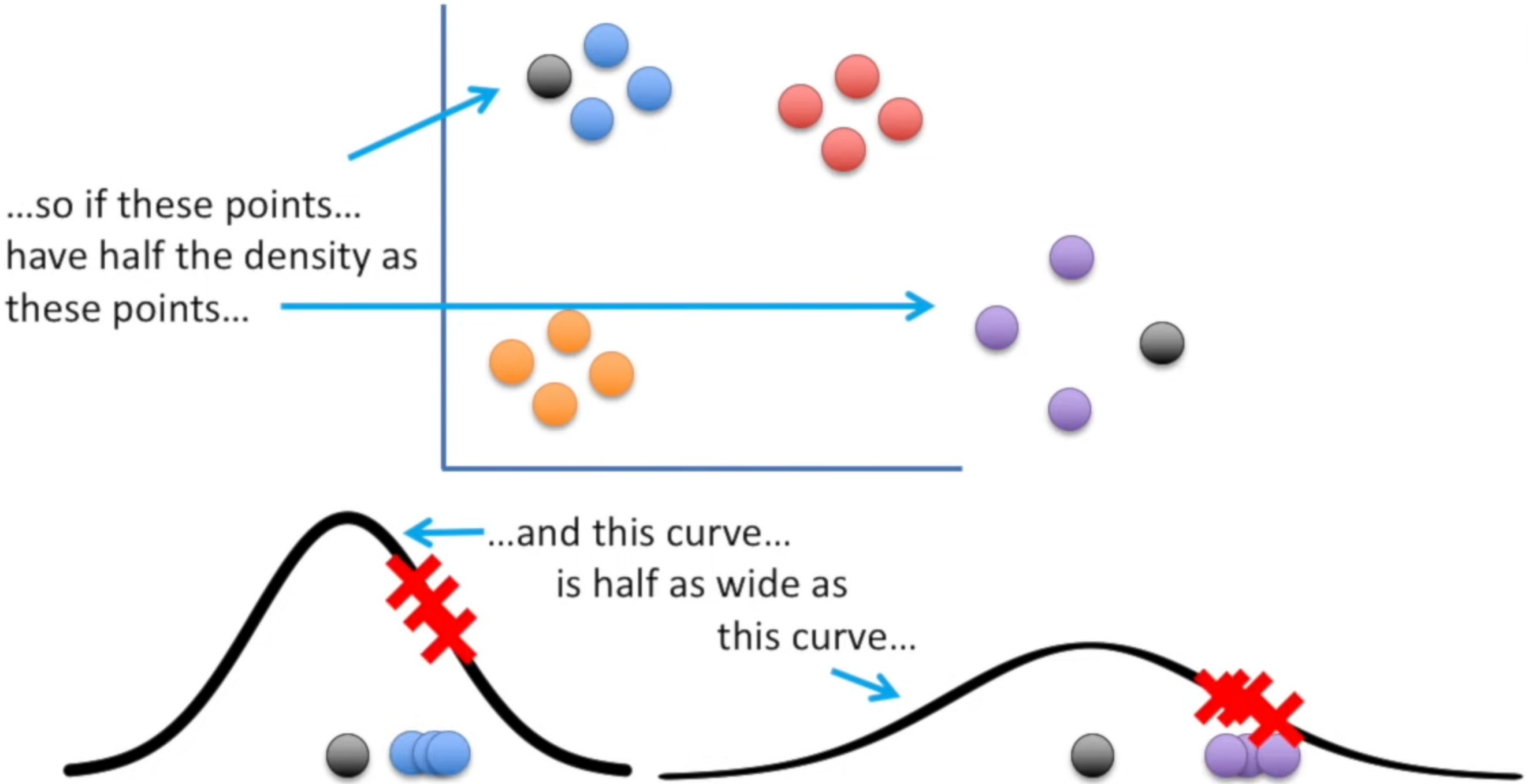


It has to do with something
I didn't tell you earlier...



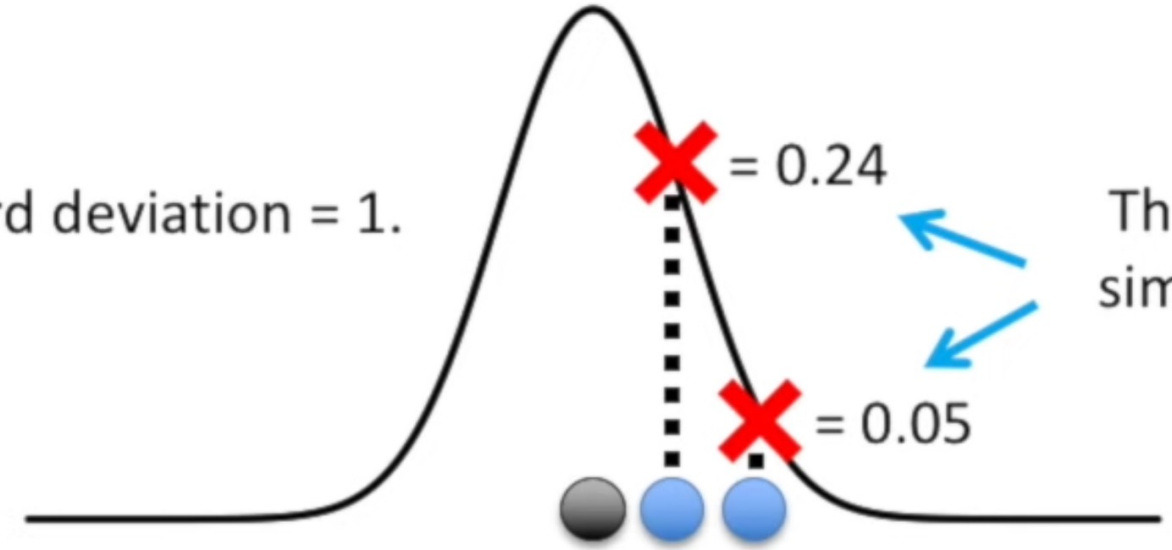
...and to illustrate the concept, I need to add a cluster that is half as dense as the others.


The width of the normal curve depends on the density of data near the point of interest.
Less dense regions have wider curves.


...so if these points...
have half the density as
these points...
...and this curve...
is half as wide as
this curve...
...then scaling the similarity scores will
make them the same for both clusters.

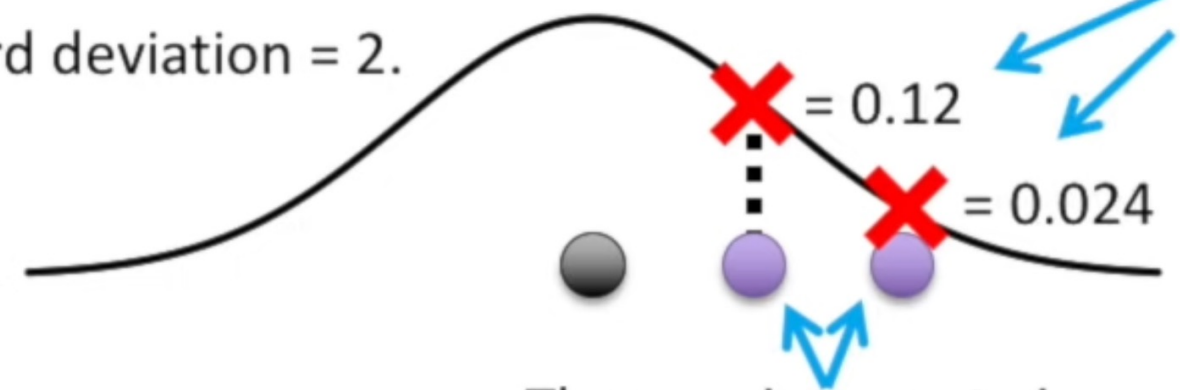
Here's an example…

This curve has a standard deviation = 1.



The "unscaled" similarity values

This curve has a standard deviation = 2.

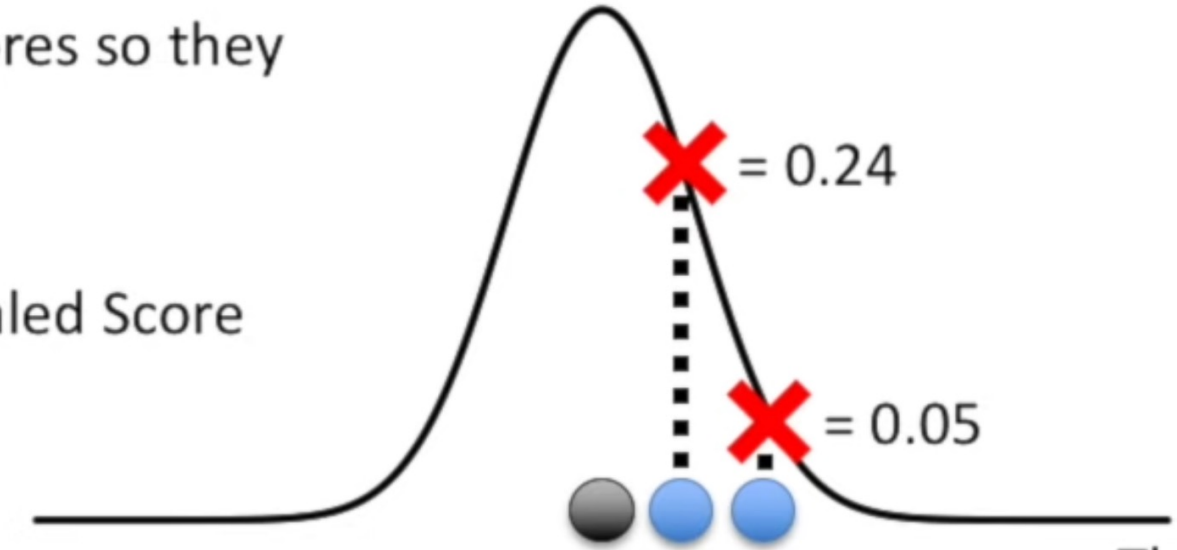


The "unscaled" similarity values are half of the other ones.

These points are twice as far from the middle.

To scale the similarity scores so they sum to 1:

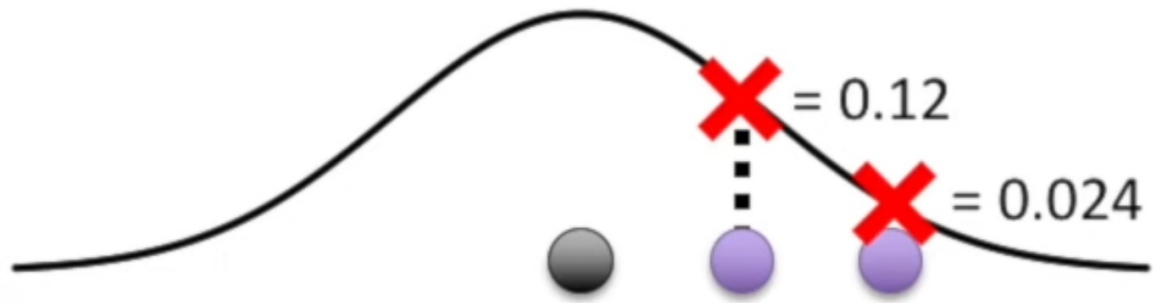
$$\frac{\text{Score}}{\text{Sum of all scores}} = \text{Scaled Score}$$



$$\frac{0.24}{0.24 + 0.5} = 0.82$$

$$\frac{0.05}{0.24 + 0.5} = 0.18$$

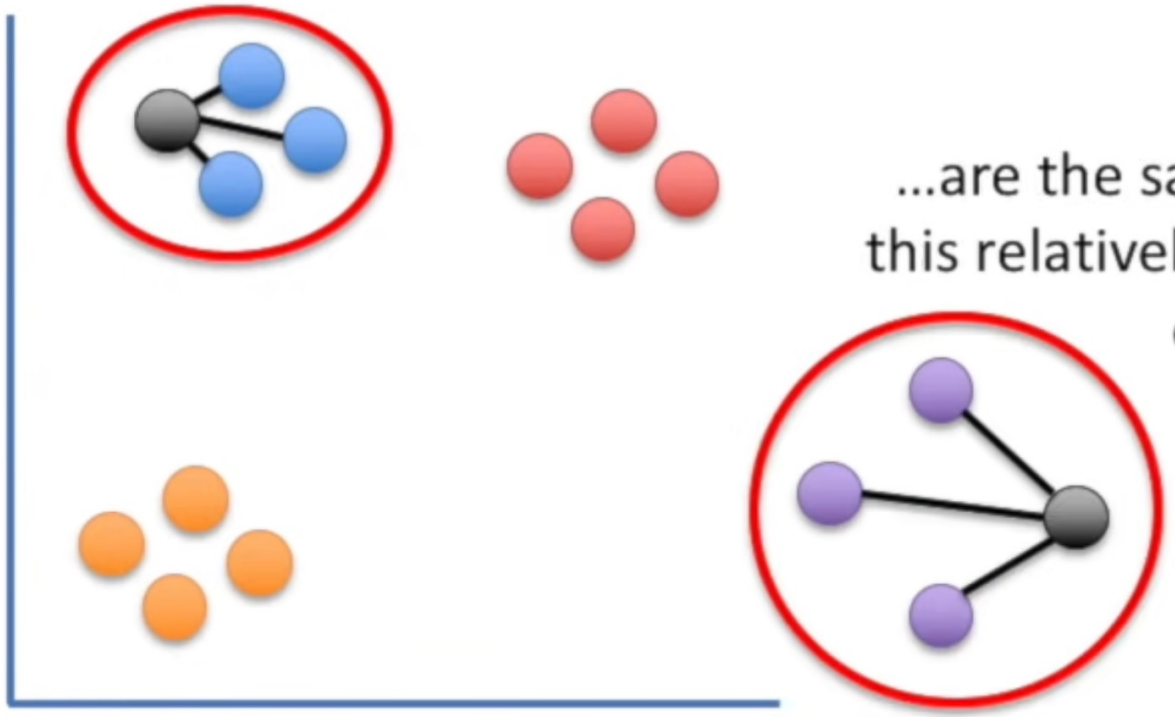
These are the same as these!



$$\frac{0.12}{0.12 + 0.024} = 0.82$$

$$\frac{0.024}{0.12 + 0.024} = 0.18$$

That implies that the scaled similarity scores for this relatively tight cluster…



…are the same for this relatively loose cluster!

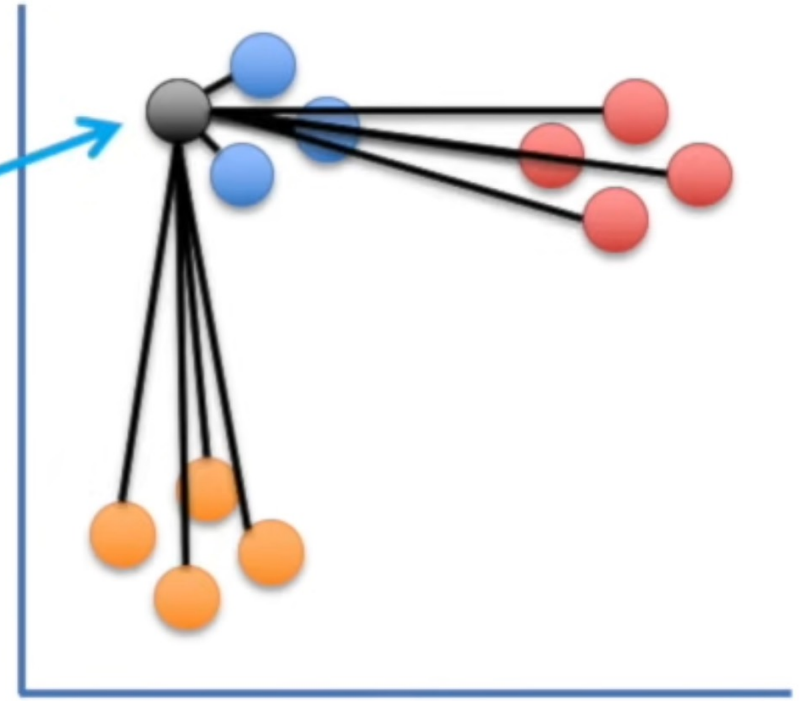
$$\frac{0.24}{0.24 + 0.5} = 0.82$$

$$\frac{0.05}{0.24 + 0.5} = 0.18$$

$$\frac{0.12}{0.12 + 0.024} = 0.82$$

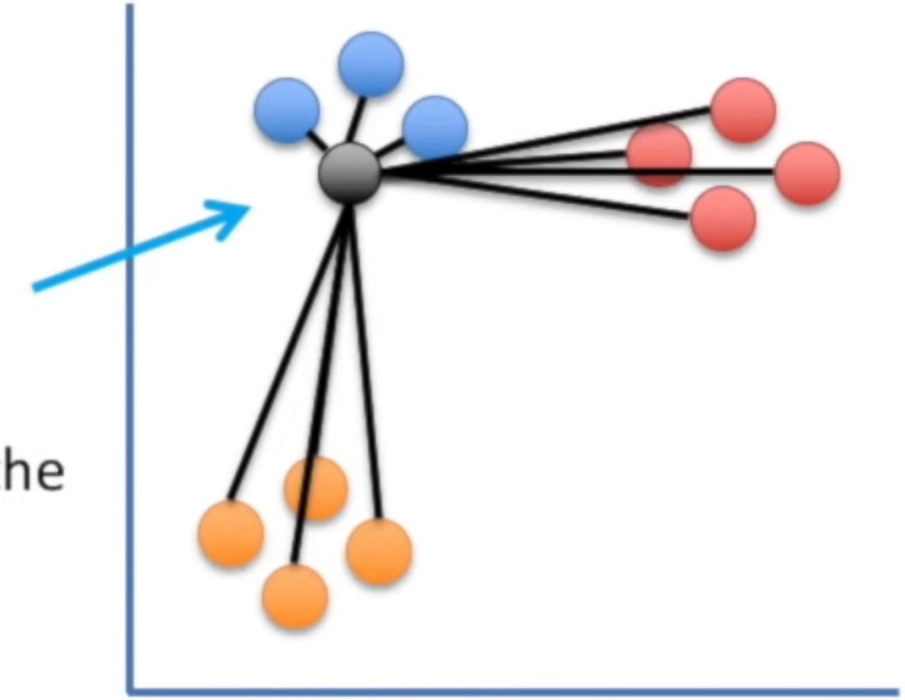
$$\frac{0.024}{0.12 + 0.024} = 0.18$$

We've calculated similarity scores for this point.

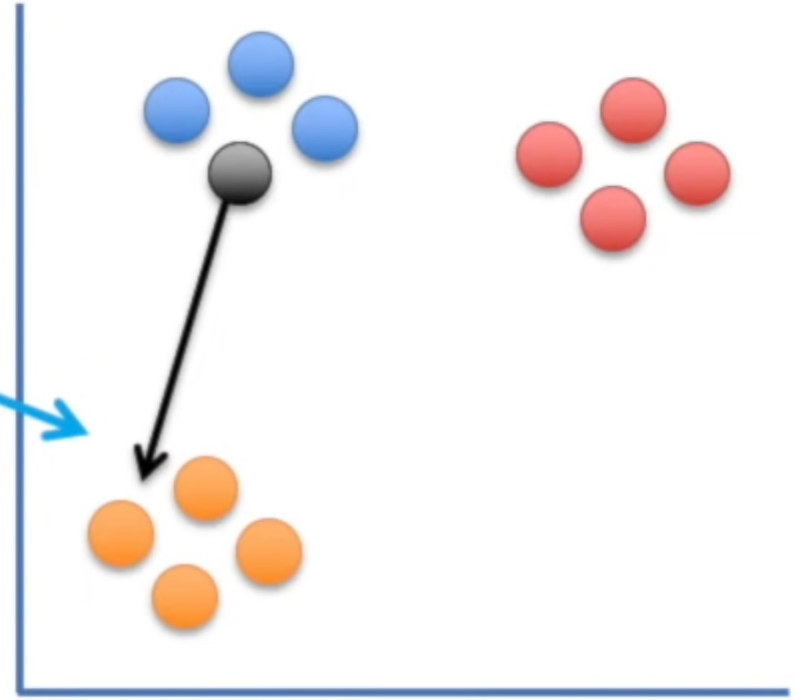


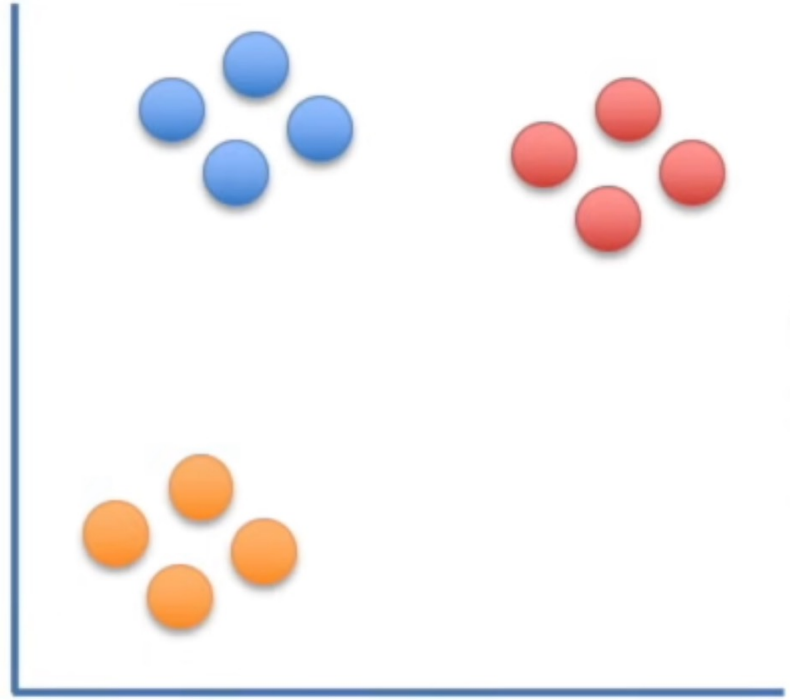
Now we do it for this point…

…and we do it for all the points.

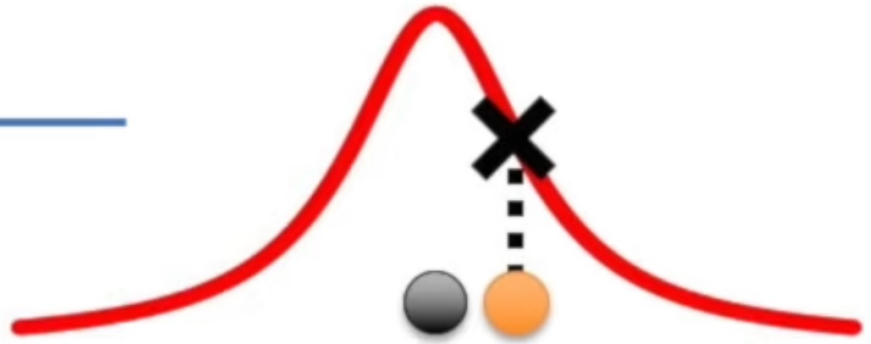


Because the width of the distribution is based on the density of the surrounding data points, the similarity score to this node…



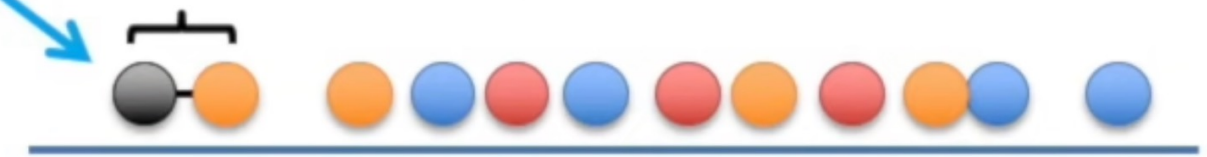


…and lastly, drawing a line from the point to a curve. However, this time we're using a "t-distribution".



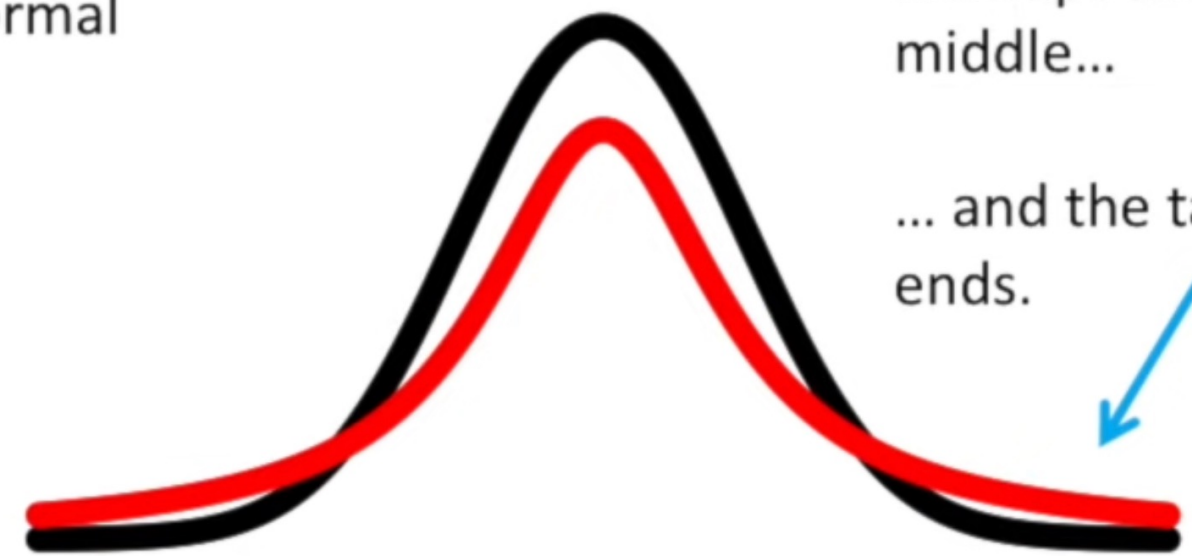
Just like before, that means picking a point…

…measuring a distance…



A “t-distribution”…

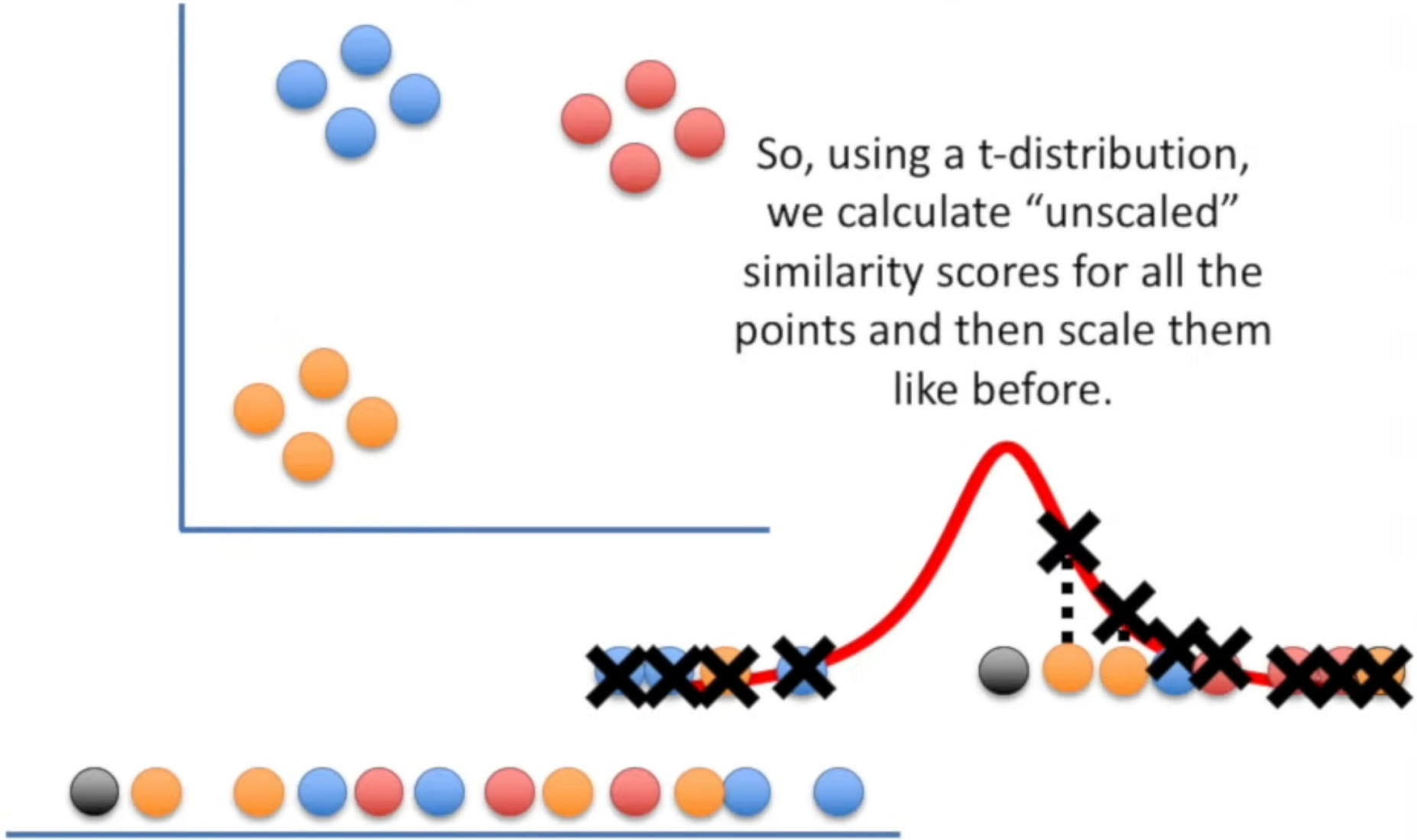
…is a lot like a normal distribution…

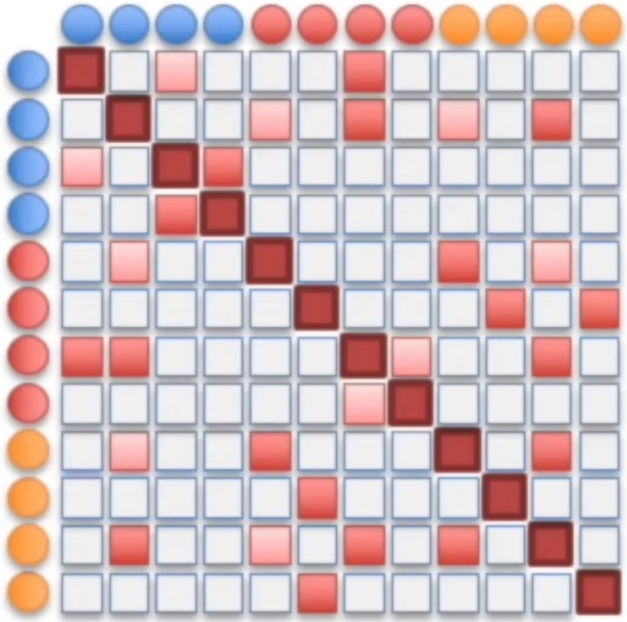


…except the “t” isn’t as tall in the middle…

… and the tails are taller on the ends.

The “t-distribution” is the “t” in t-SNE.

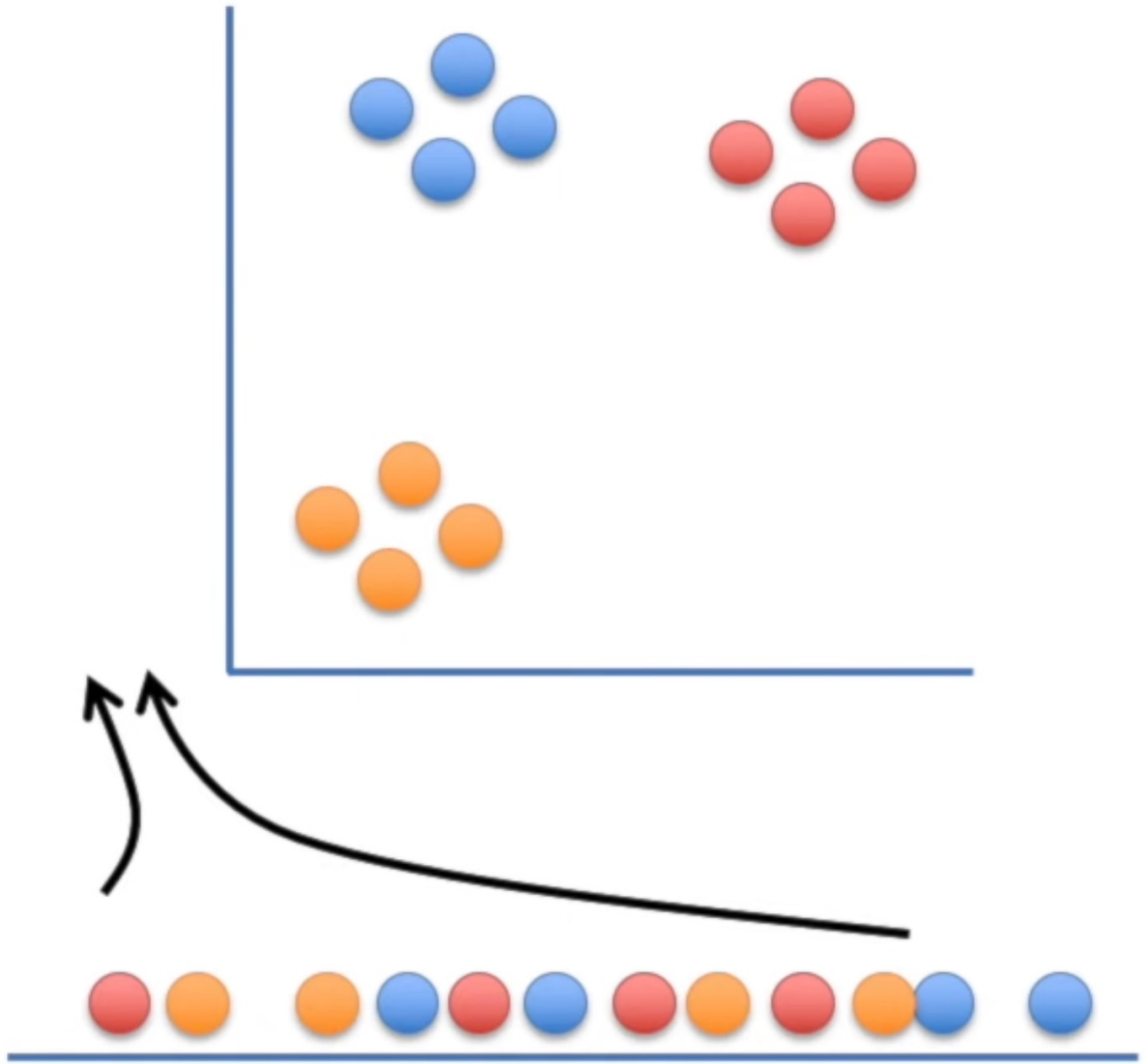

So, using a t-distribution, we calculate “unscaled” similarity scores for all the points and then scale them like before.

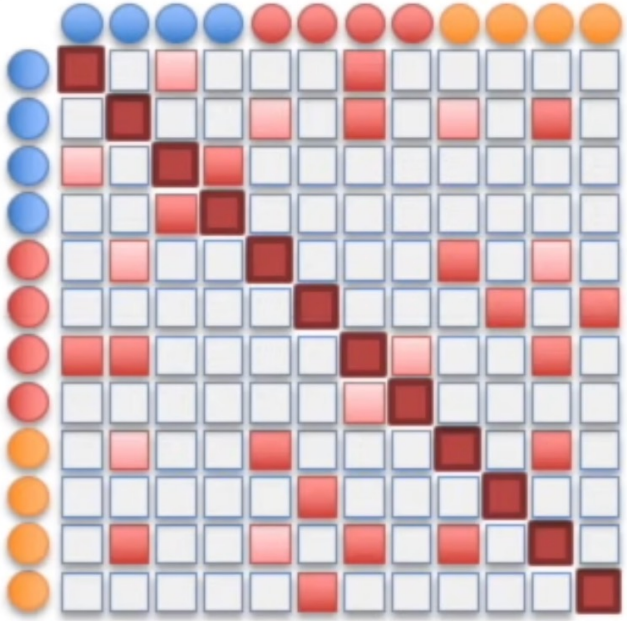


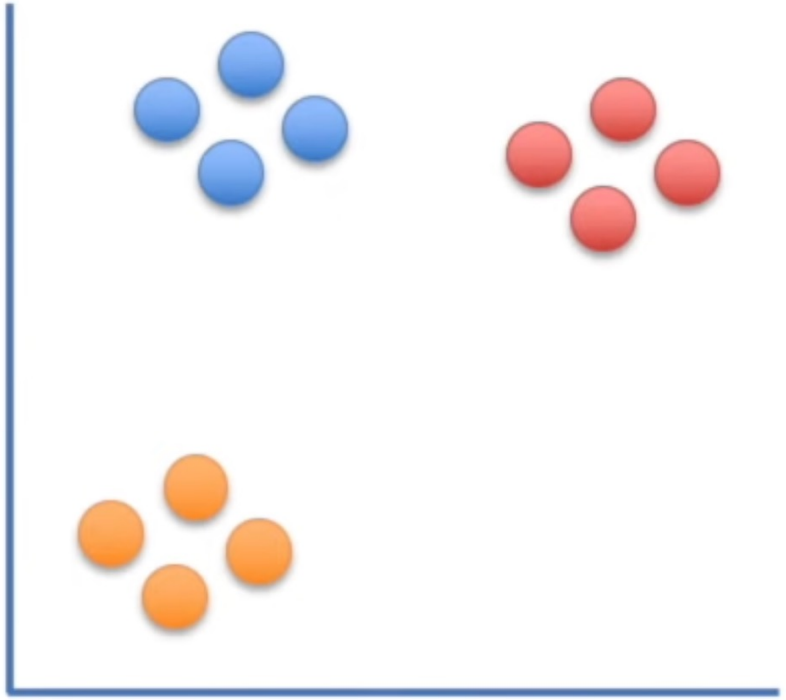
= High similarity

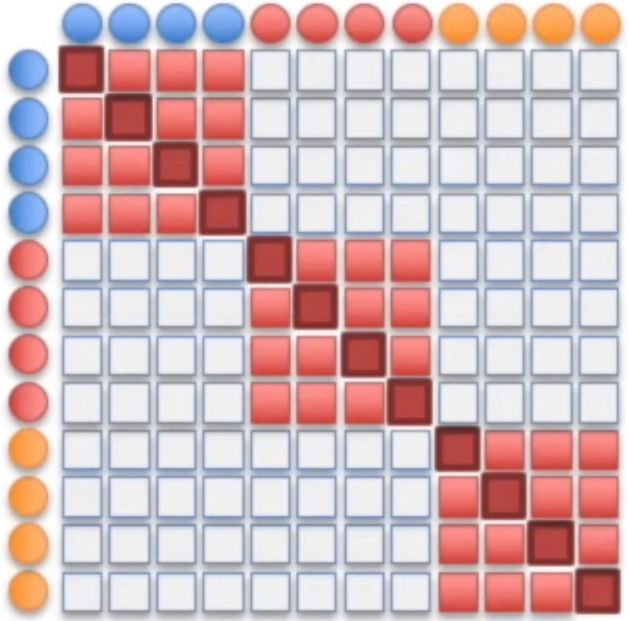
= Low similarity

Like before, we end up with a matrix of similarity scores, but this matrix is a mess…

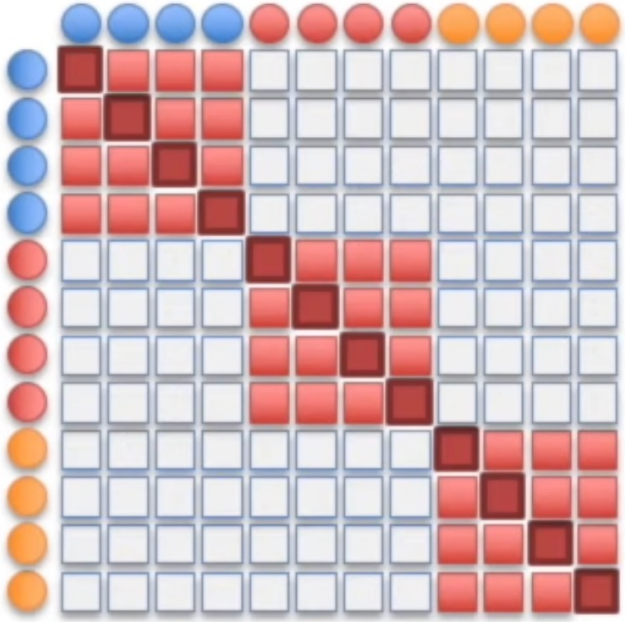


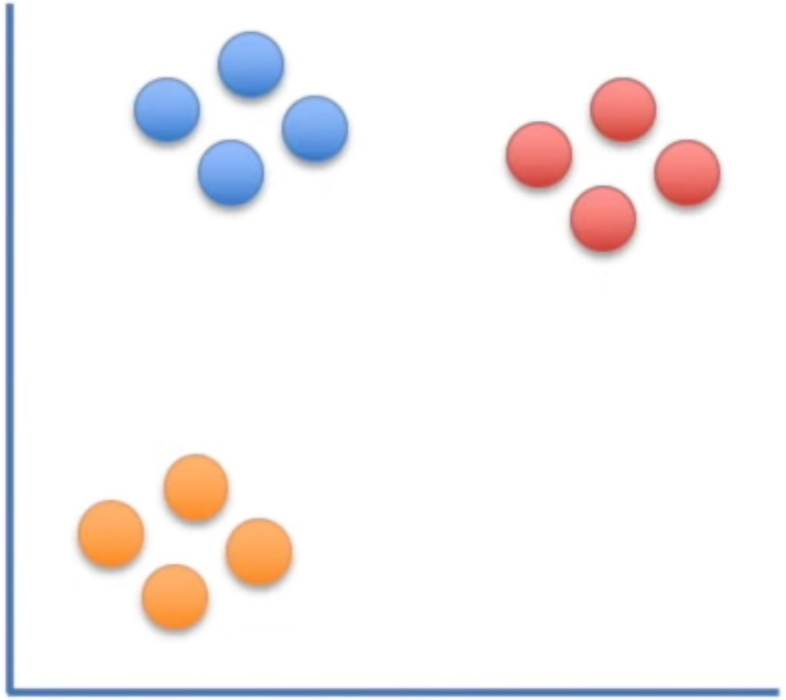


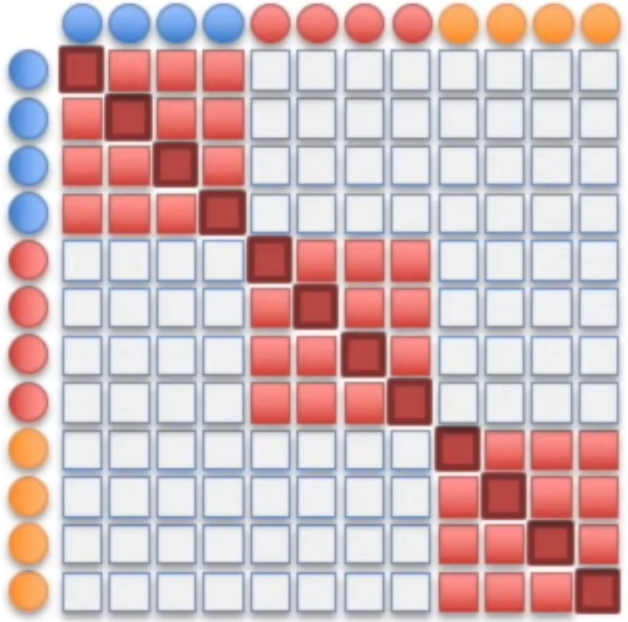




t-SNE moves the points a little bit at a time, and each step it chooses a direction that makes the matrix on the left more like the matrix on the right.

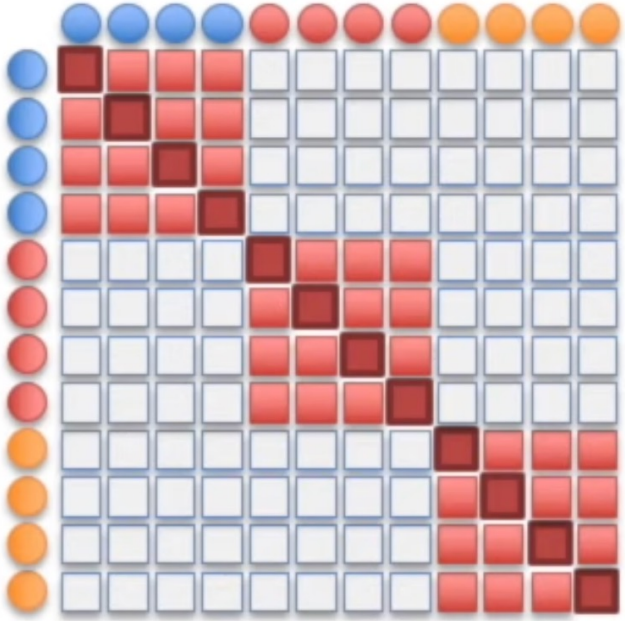


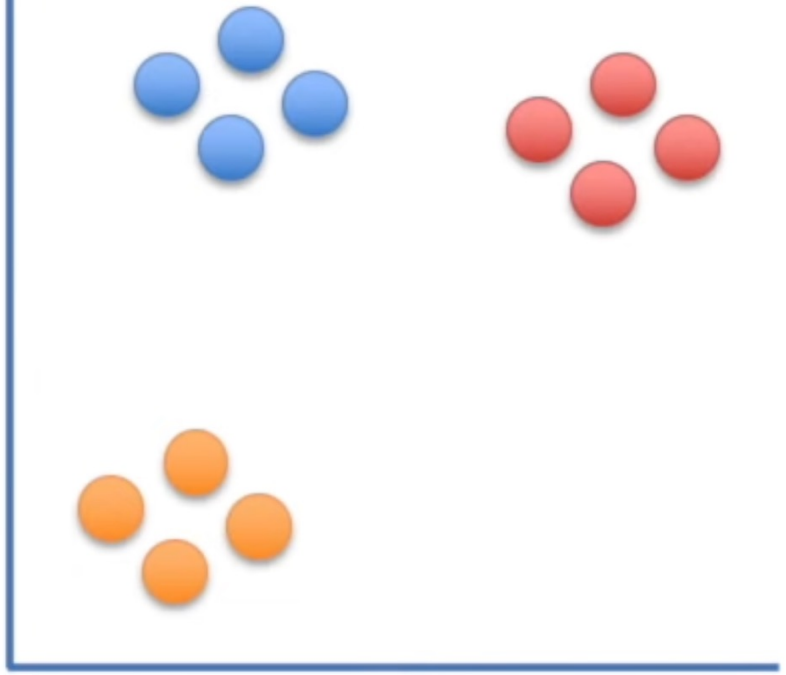


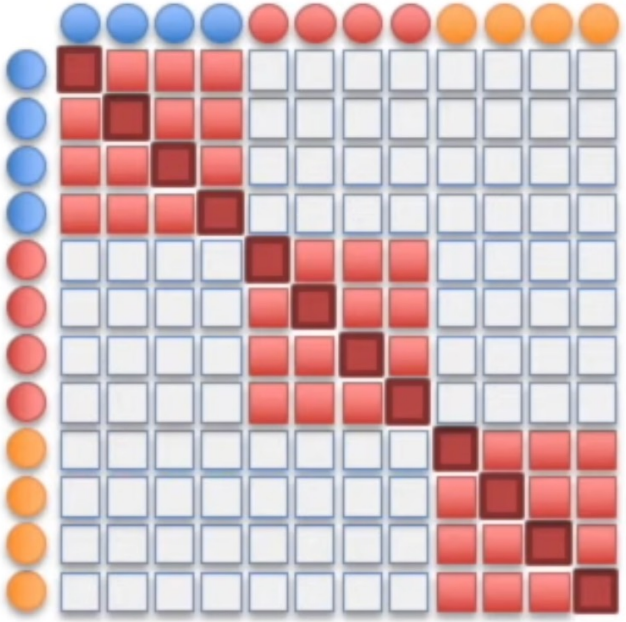


t-SNE moves the points a little bit at a time, and each step it chooses a direction that makes the matrix on the left more like the matrix on the right.

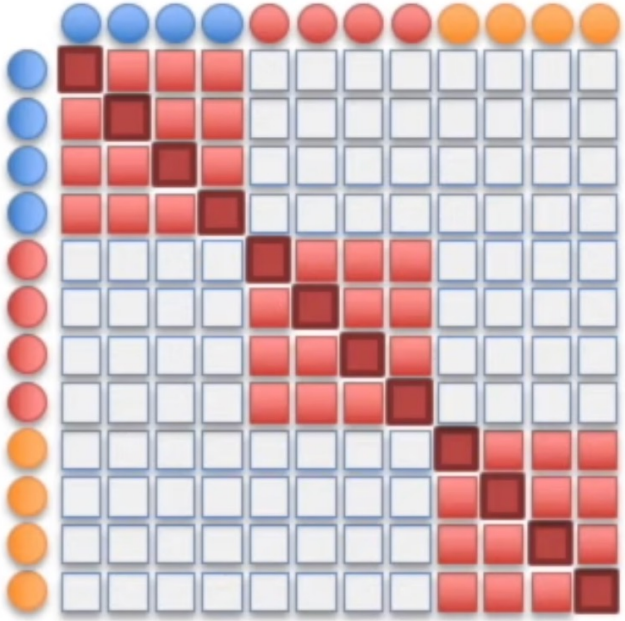
It uses small steps, because it's a little bit like a chess game and can't be solved all at once. Instead, it goes one move at at time.

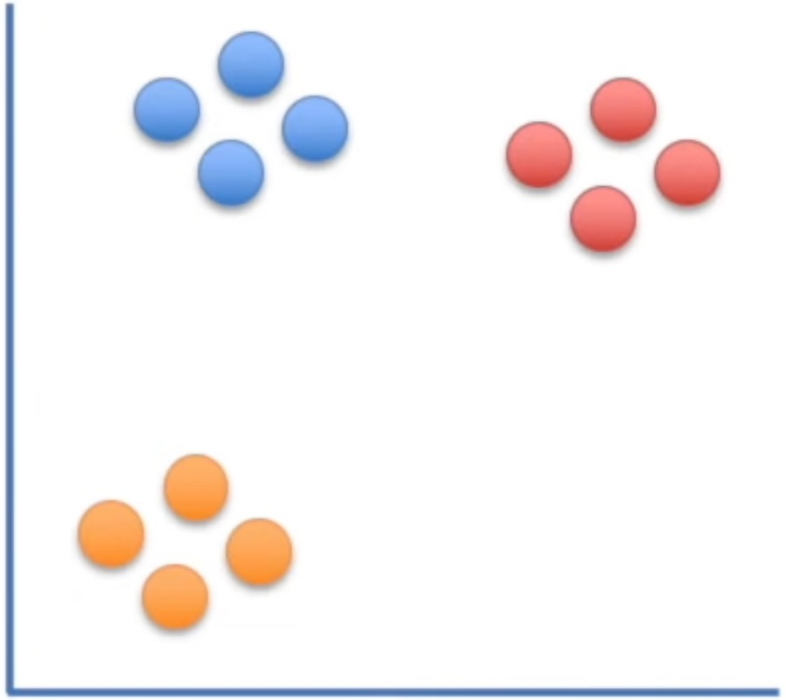


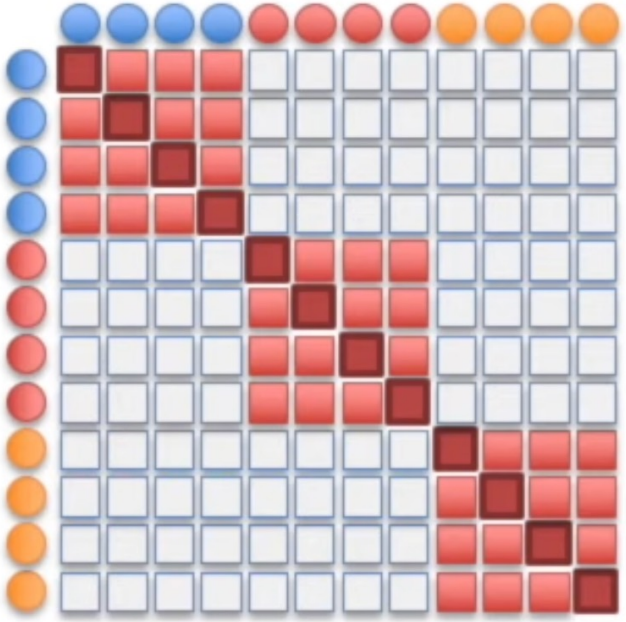




...without it the clusters would all clump up in the middle and be harder to see.







…without it the clusters would all clump up in the middle and be harder to see.

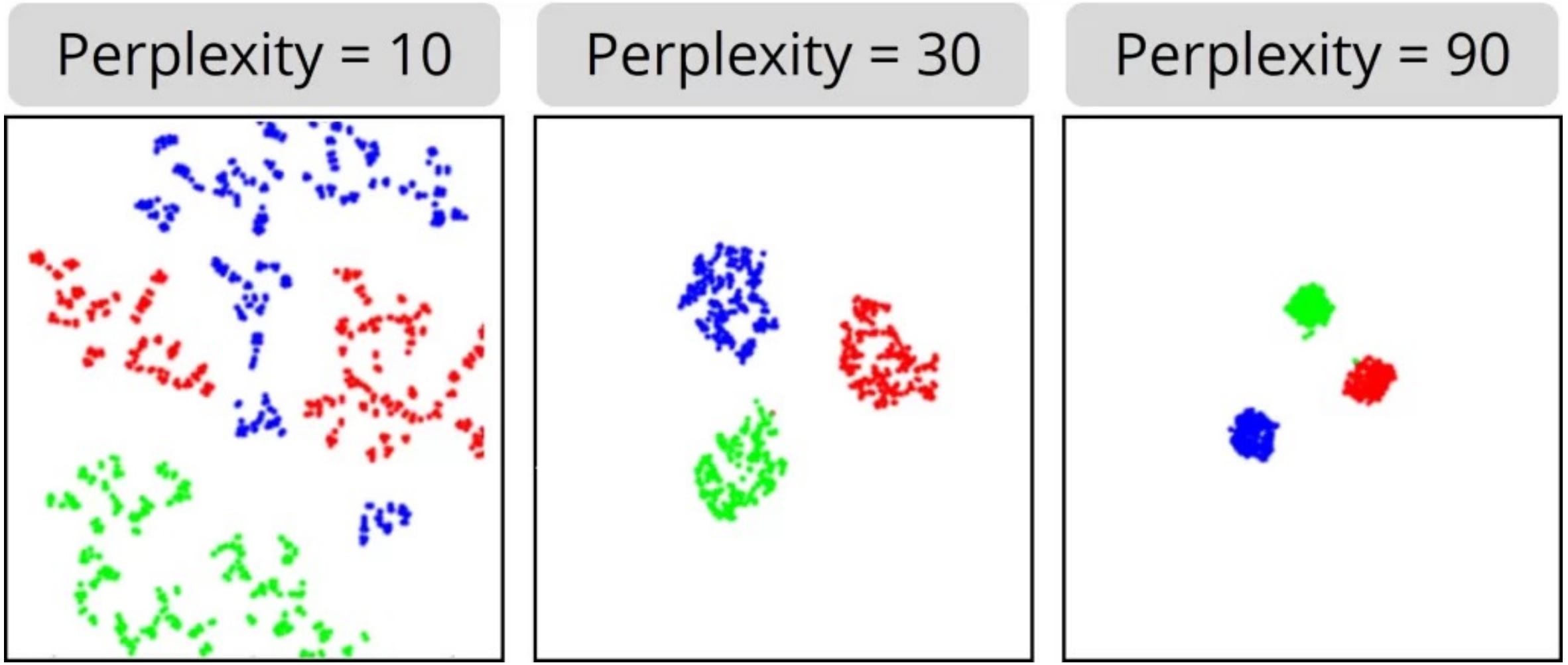
t-SNE Perplexity

- There is an hyperparameter in the method, called ‘Perplexity’
- A hyperparameter in machine learning is a setting that you choose before training a model. It controls how the learning process works but is not learned from the data.
- Perplexity influences how the algorithm balances local and global structure in the lower-dimensional representation.
- Perplexity can be thought of as a smooth measure of the effective number of neighbors each point considers when constructing the low-dimensional embedding. It controls how much attention t-SNE pays to nearby versus distant points.

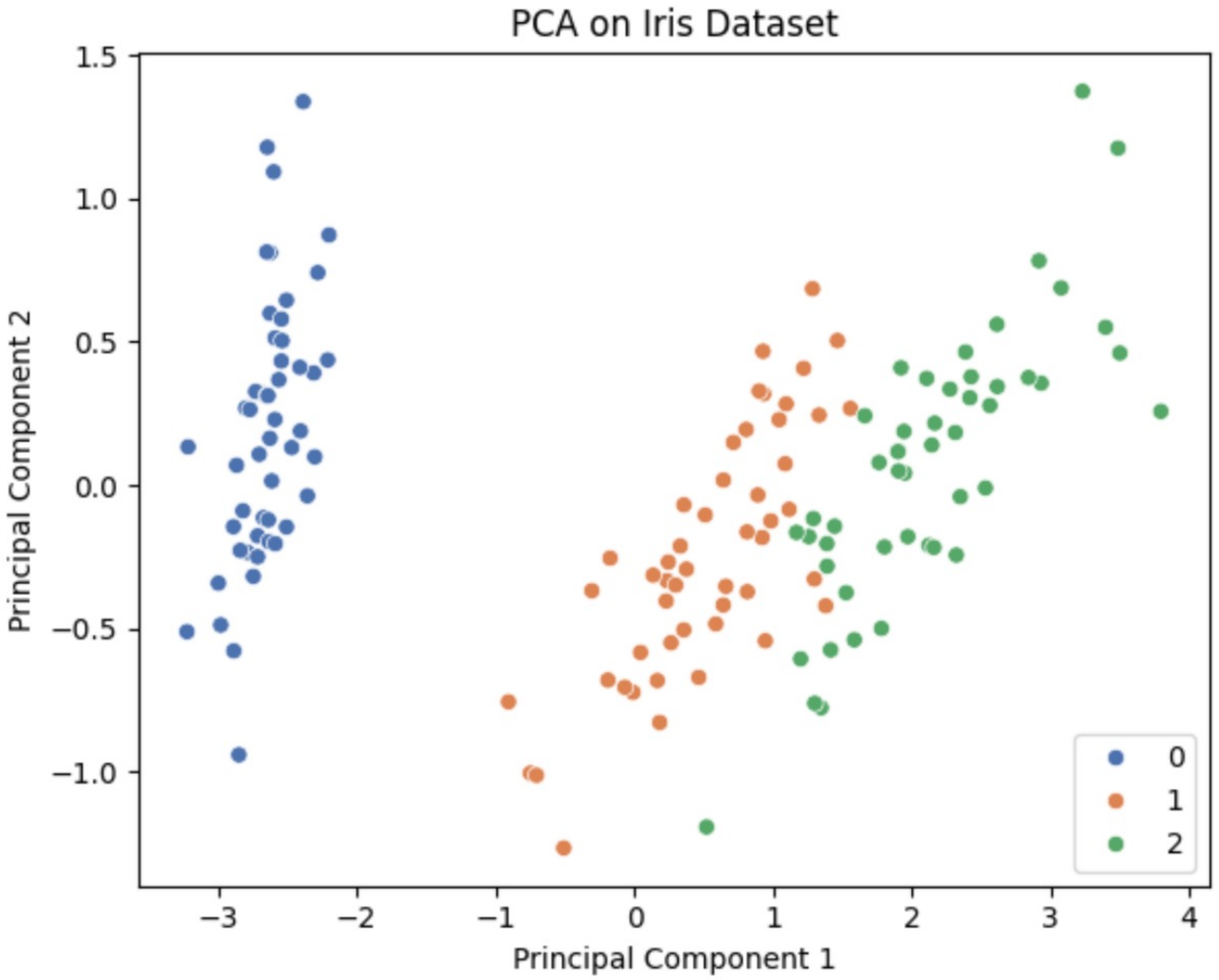
t-SNE Perplexity

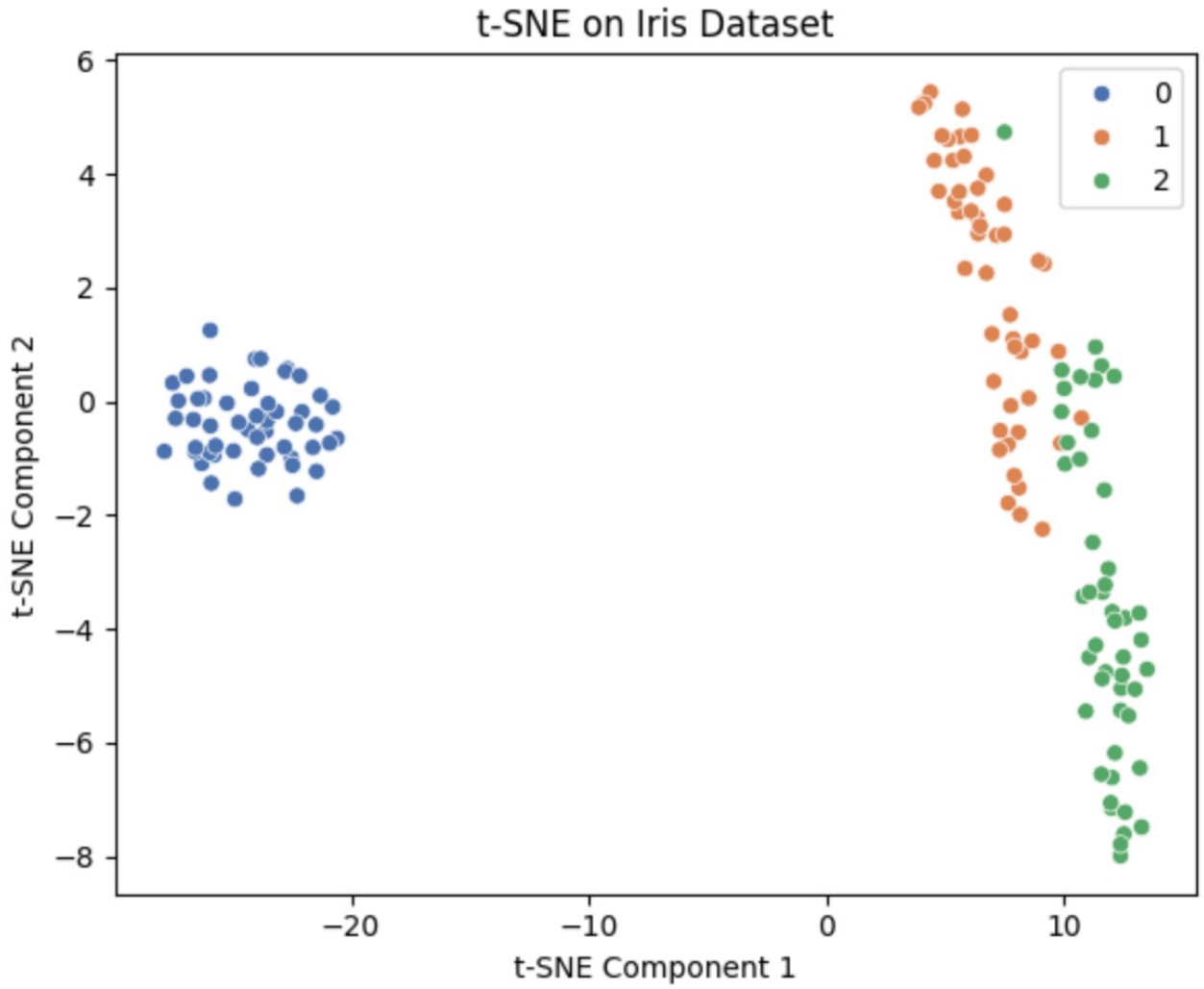
- There is an hyperparameter in the method, called ‘Perplexity’
- A hyperparameter in machine learning is a setting that you choose before training a model. It controls how the learning process works but is not learned from the data.
- Perplexity influences how the algorithm balances local and global structure in the lower-dimensional representation.
- Perplexity can be thought of as a smooth measure of the effective number of neighbors each point considers when constructing the low-dimensional embedding. It controls how much attention t-SNE pays to nearby versus distant points.

t-SNE Perplexity

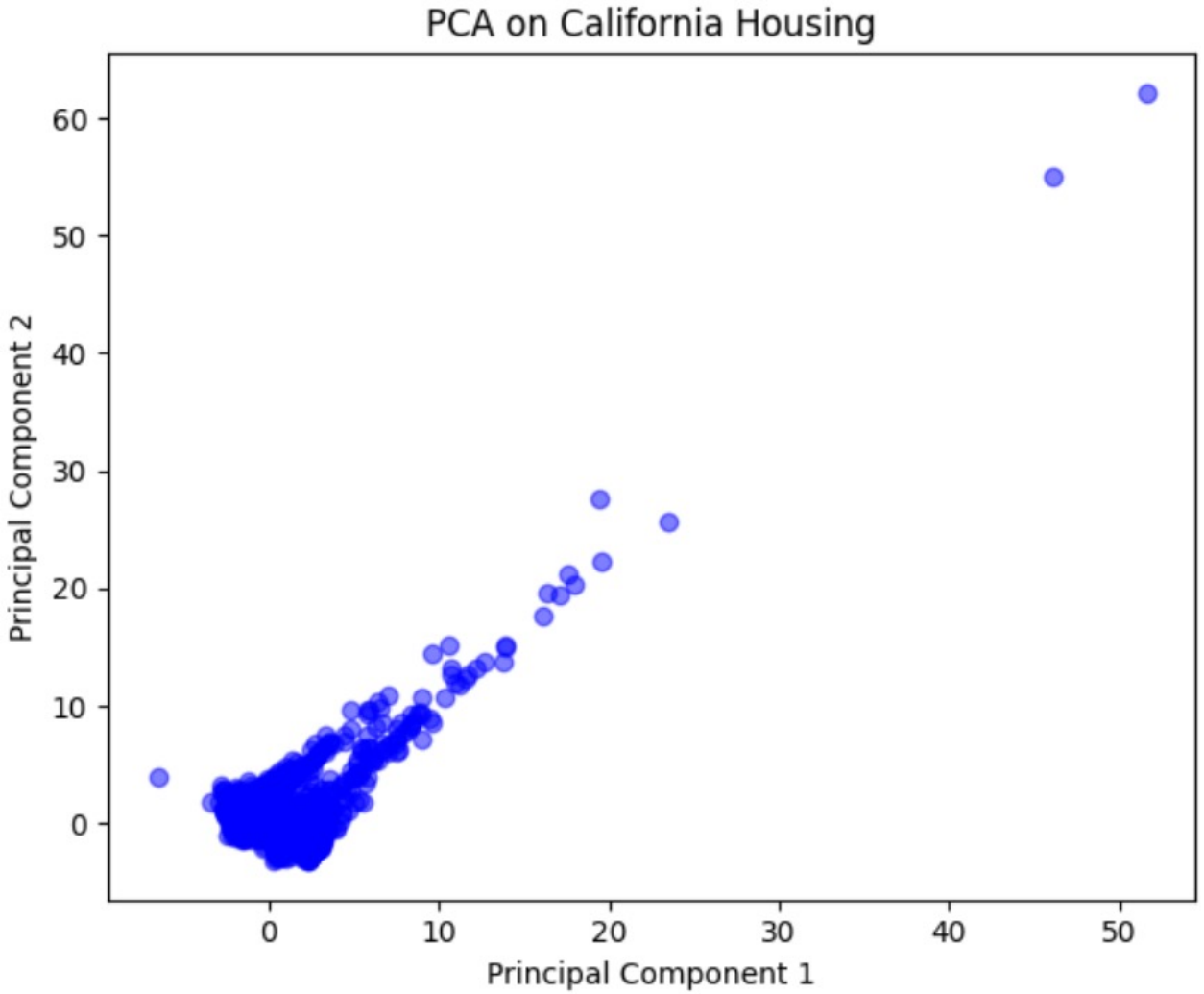


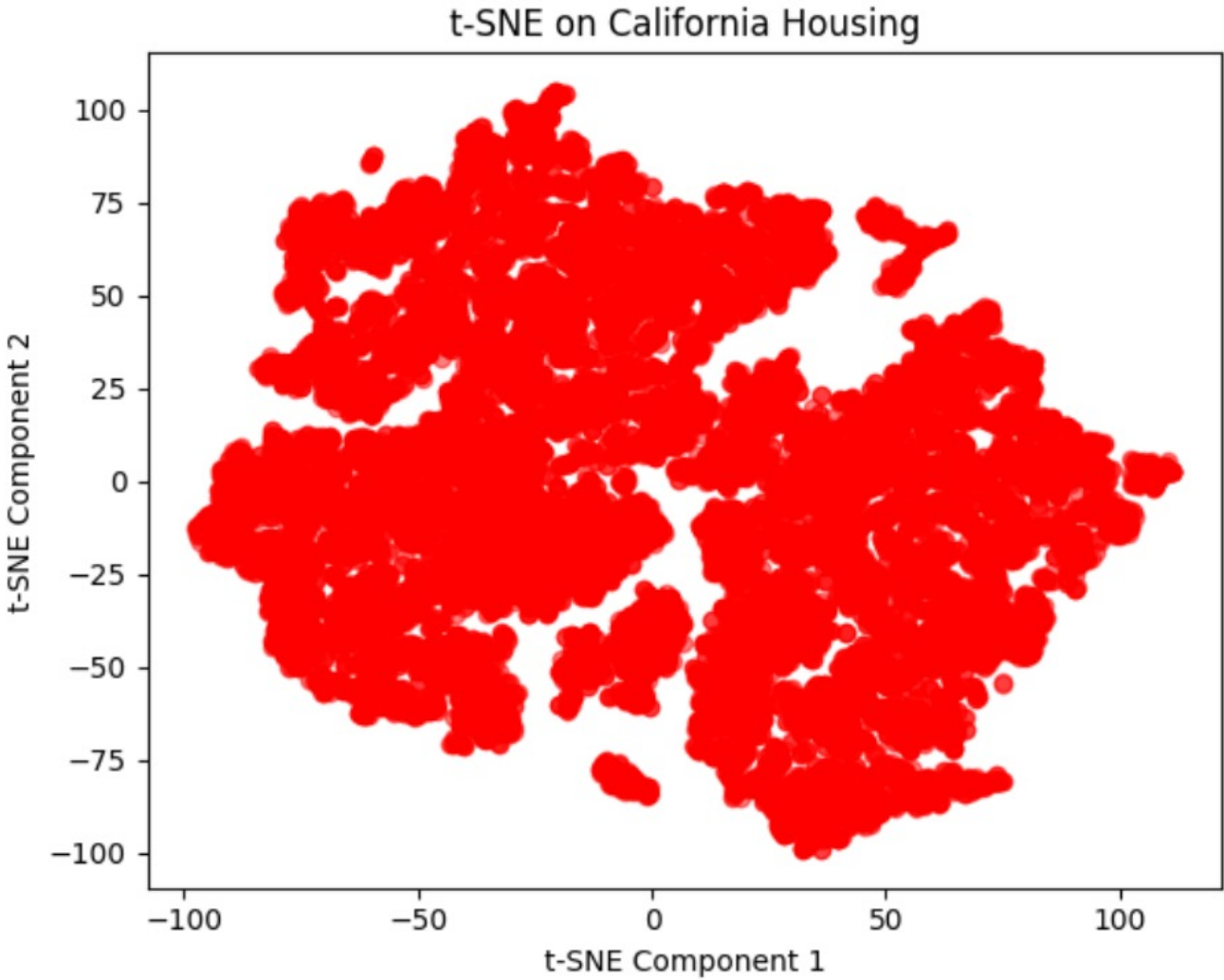
On the Iris dataset





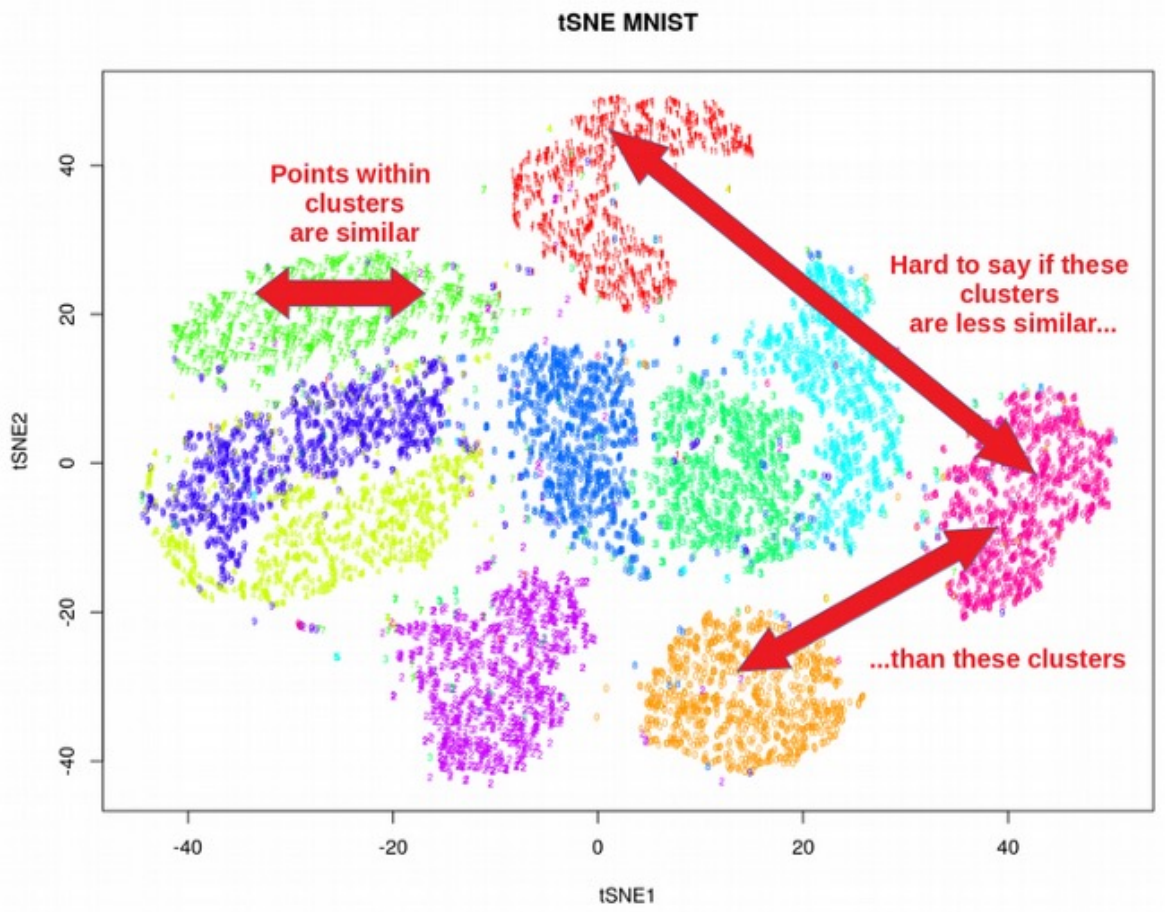
On the California Housing dataset





t-SNE Shortcomings

- Choice of perplexity is tricky!
- Be careful in interpreting t-SNE!!! https://distill.pub/2016/misread-tsne/
- t-SNE does not scale well
- t-SNE does not preserve global data structure
- It's ok for data viz, not for feature extraction!



UMAP

Uniform Manifold Approximation and Projection

UMAP in a nutshell

- Uniform Manifold Approximation and Projection for Dimension Reduction (2018)
- Much **faster** than t-SNE
- **Preserves global structure better** than t-SNE

	t-SNE	UMAP
COIL20	20 seconds	7 seconds
MNIST	22 minutes	98 seconds
Fashion MNIST	15 minutes	78 seconds
GoogleNews	4.5 hours	14 minutes

https://github.com/lmcinnes/umap

https://www.youtube.com/watch?v=nq6iPZVUxZU&ab_channel=Enthought

Theory behind UMAP (quick overview)

- Based on algebraic topology and Riemannian geometry
- UMAP constructs a high dimensional, fuzzy graph representation of the data, then optimizes a low-dimensional graph to be as structurally similar as possible.

McInnes et al., (2018). UMAP: Uniform Manifold Approximation and Projection. Journal of Open Source Software, 3(29), 861. https://doi.org/10.21105/joss.00861

UMAP vs t-SNE

With UMAP, each category is clustered (local structure), while similar categories tend to colocate (global structure)

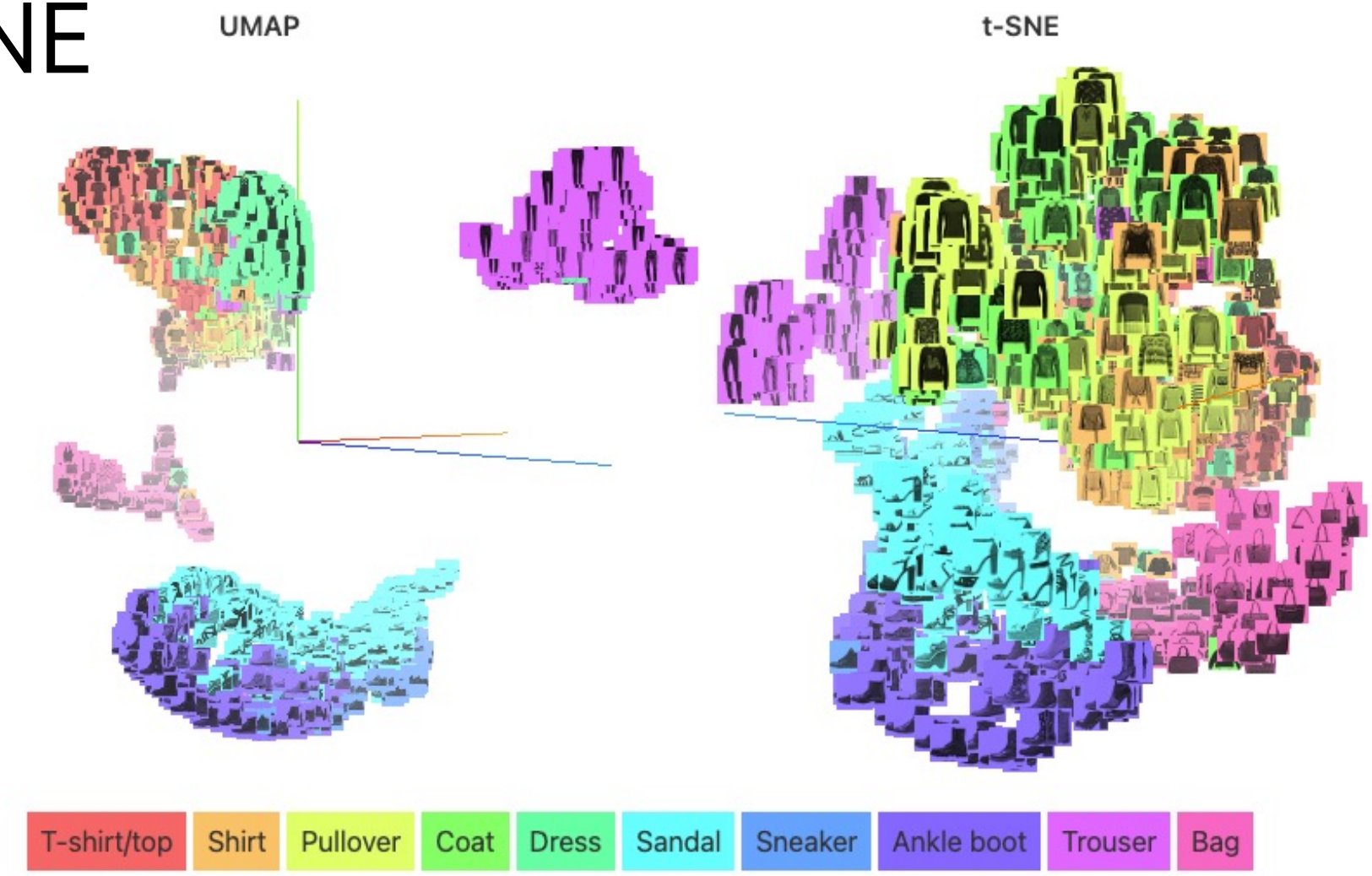


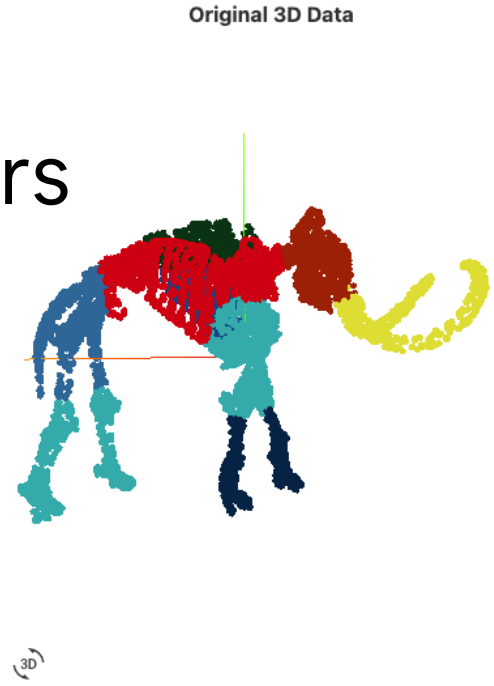
Figure 2: *Dimensionality reduction applied to the Fashion MNIST dataset. 28x28 images of clothing items in 10 categories are encoded as 784-dimensional vectors and then projected to 3 using UMAP and t-SNE.*

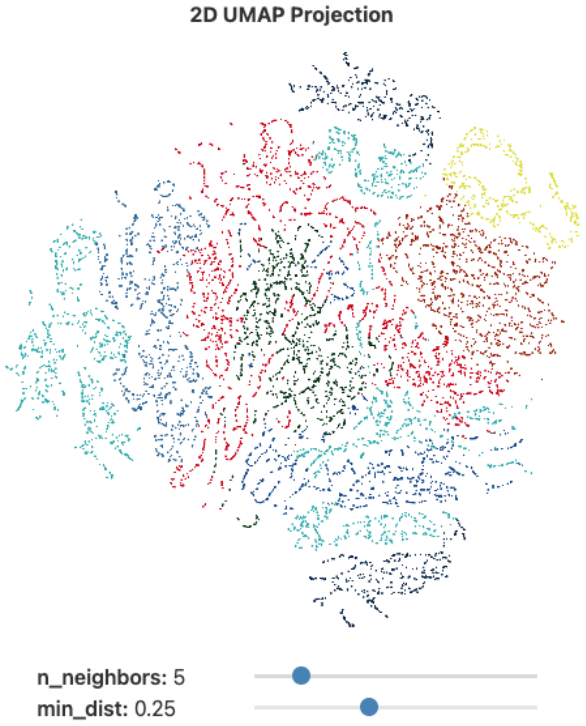
https://pair-code.github.io/understanding-umap/

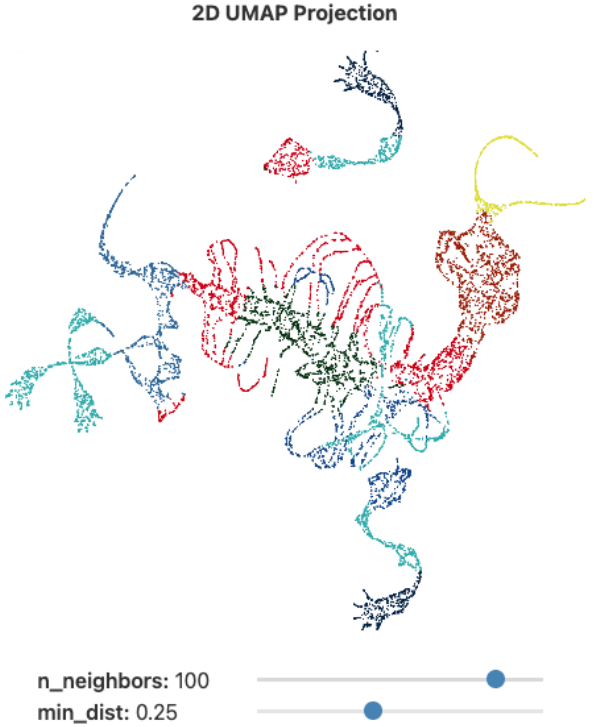
Input parameters

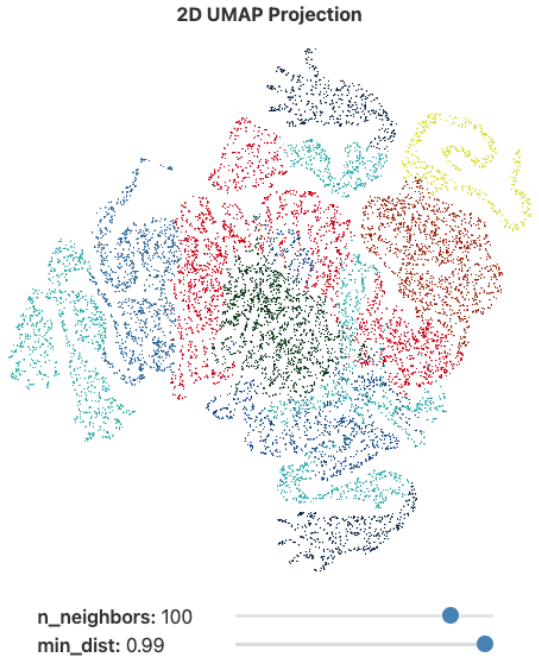
- **n_neighbours**: trade-off between preservation of local (low values) and global (high values) structure
- **min_dist**: minimum distance between points in low-dim space. Low values lead to more tightly packed embeddings

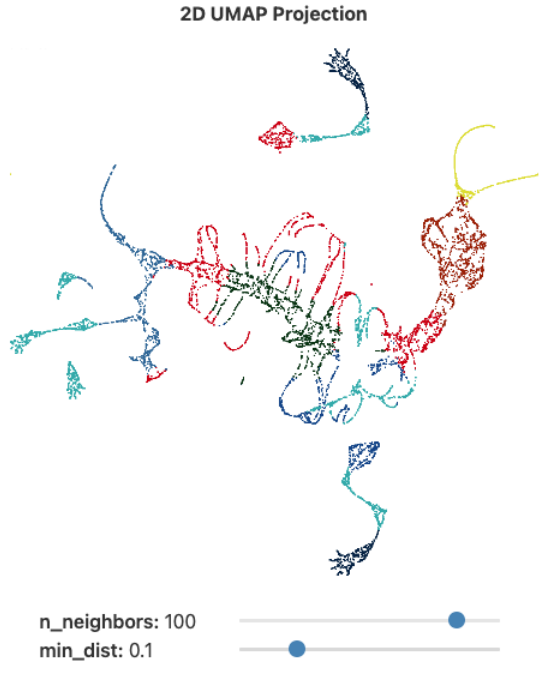
https://pair-code.github.io/understanding-umap/











UMAP Pros & Cons

Cons:

- Hyperparameters choice is crucial
- Cluster sizes are not significant
- Distance between clusters might not mean anything
- Different results in different runs

Pros:

- **Faster** than tSNE
- More focus on **global structure**
- Viable tool for **feature engineering**

Thank you!

Gian Antonio Susto