1495 UNIVERSITY OF ABERDEEN | CELEBRATING 525 YEARS 1495 – 2020

ABERDEEN 2040

Data Dimensionality – 1

Data Mining & Visualisation

Lecture 21

2025

Today…

- Data Dimensionality
- Dimensionality Reduction
- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Other Approaches

Data Dimensionality

In describing a dataset, we might refer to its *'dimensionality'.*

The dimensionality of a dataset is simply the number of columns that it has.

This is often denoted as: p

Data Dimensionality

Age
…
…
…

p = 1

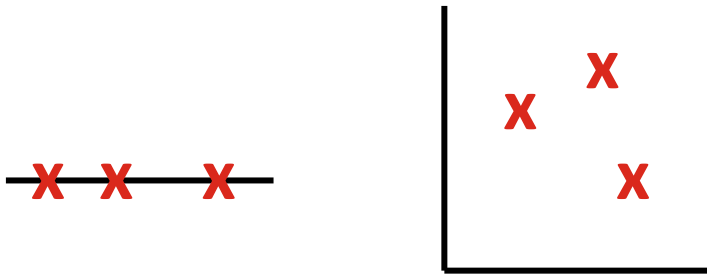
Data Dimensionality

Age
…
…
…

p = 1

Age	Height
…	…
…	…
…	…

p = 2



Data Dimensionality

Age
…
…
…

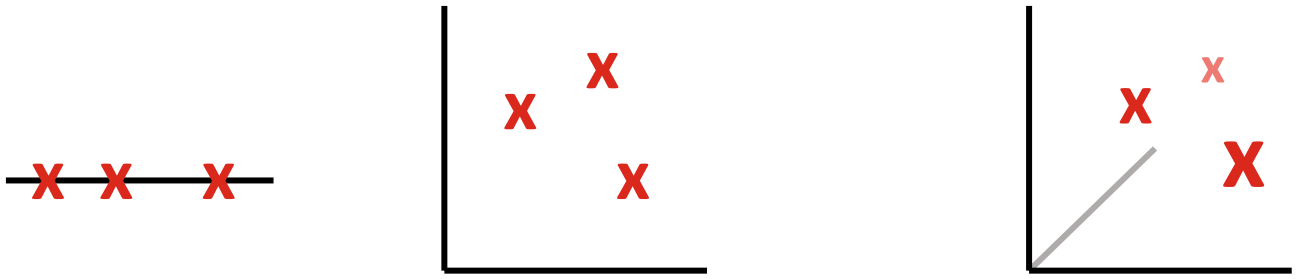
p = 1

Age	Height
…	…
…	…
…	…

p = 2

Age	Height	Weight
…	…	…
…	…	…
…	…	…

p = 3



Data Dimensionality

Age
…
…
…

p = 1

Age	Height
…	…
…	…
…	…

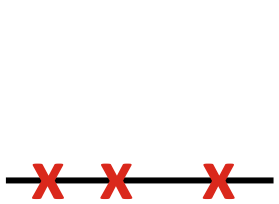
p = 2

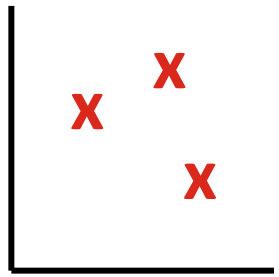
Age	Height	Weight
…	…	…
…	…	…
…	…	…

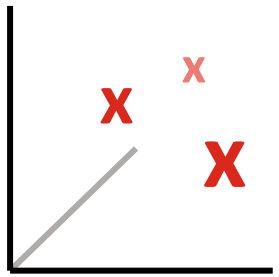
p = 3

Age	Height	Weight	GPA
…	…	…	…
…	…	…	…
…	…	…	…

p = 4







The Curse of Dimensionality

As we can see, the higher the dimensionality of a dataset, the more difficult it is to visualise (and understand) how the variables relate to one another.

This is one example of the issues that come with working on high dimensional data.

Such issues are often referred to as ***'the curse of dimensionality'***.

The Curse of Dimensionality

Another example of an issue associated with ***'the curse of dimensionality'*** is that having more dimensions will often mean that you need more data points.

Data can become sparse, and additional noise and variance can mean that algorithms perform poorly.

The Curse of Dimensionality

Similarly, having higher dimensionality in a dataset can also mean higher computational complexity.

For example, with decision trees, more columns means exponentially more datapoints to consider splitting.

In clustering, distance metrics become more expensive to compute, etc.

Dimensionality Reduction

Dimensionality reduction refers to a set of processes for ***reducing*** the number of dimensions within a dataset, while preserving as much important information as possible.

It is one potential way to counter the curse of dimensionality:

- It can make datasets easier to visualise.
- It can lead to improved computational efficiency.
- It can help reduce overfitting and lead to better generalisation.

Types of Dimensionality Reduction

There are many different approaches for dimensionality reduction used.

We will discuss a few, including how they work.

We will start with Principal Component Analysis (PCA).

Principal Component Analysis

ABERDEEN 2040

Principal Component Analysis (PCA)

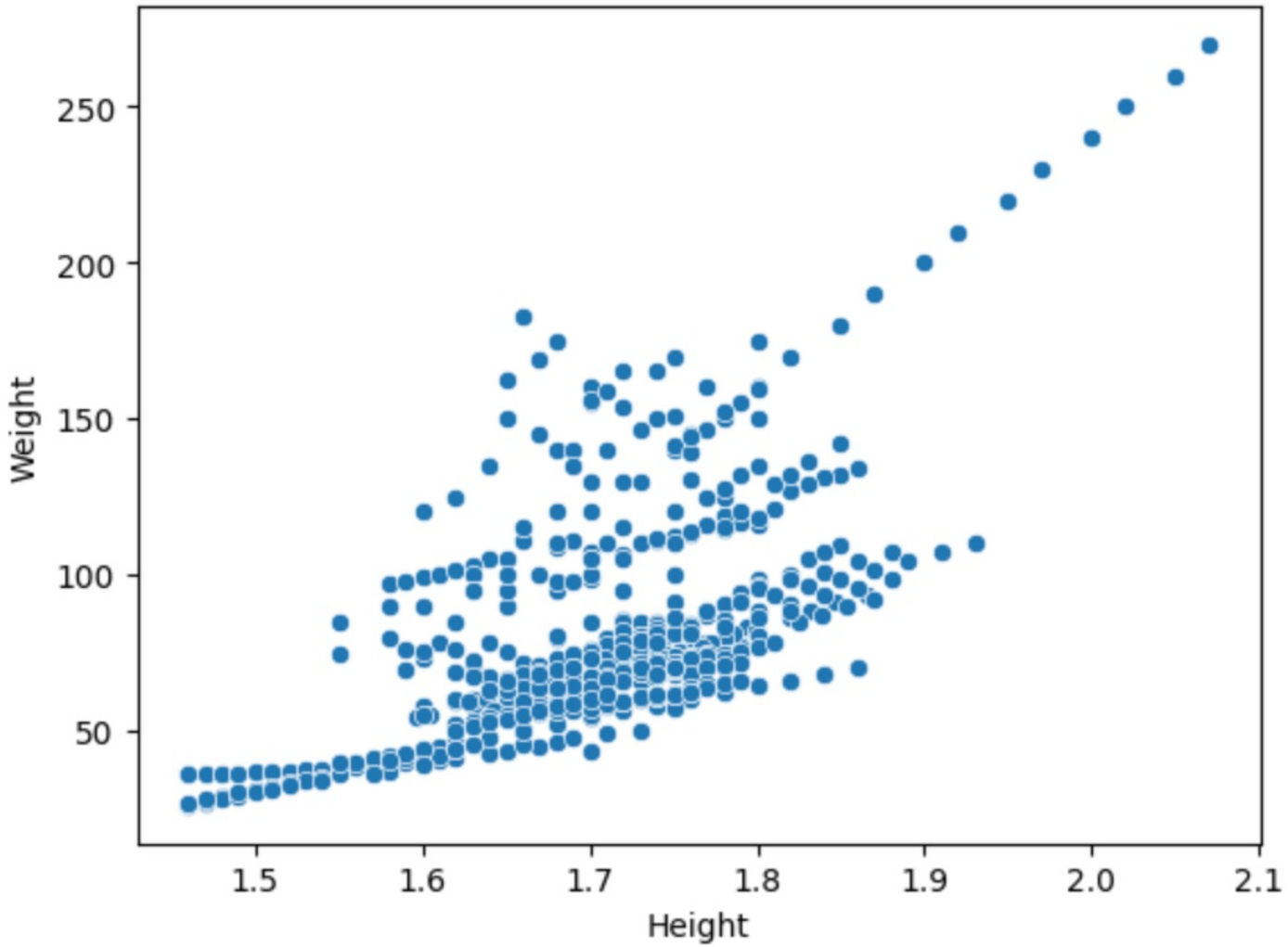
Principal Component Analysis (PCA) aims to transform **high dimensionality data** into **fewer dimensions which maintain the underlying patterns and structure of the data**.

These (new) dimensions are known as ***principal components***.

It is one of the oldest, and most well-established methods of dimensionality reduction, and is still widely used today.

PCA – Intuition

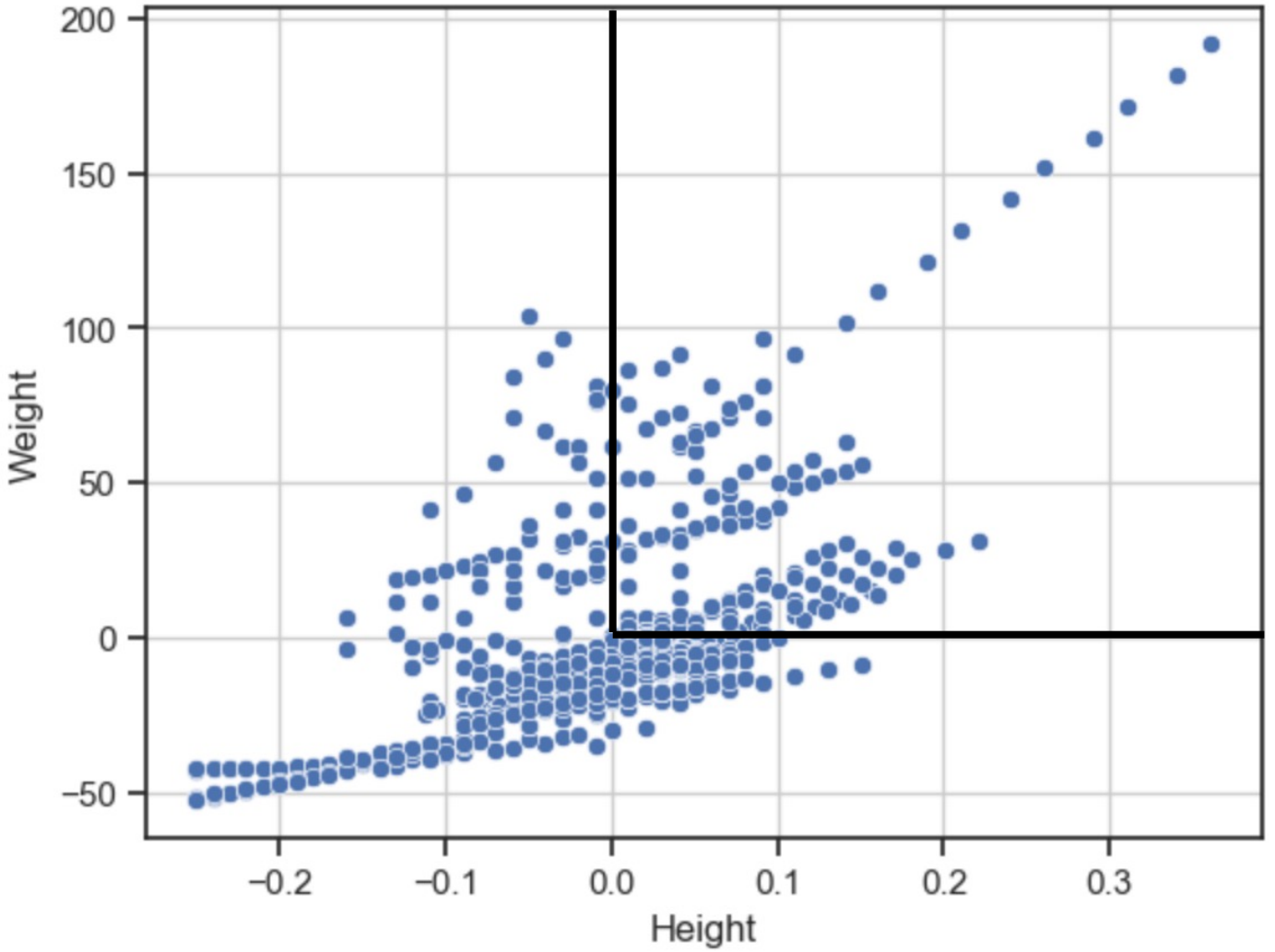
Let’s say we have two variables: Height and Weight



PCA – Intuition

Let’s say we have two variables: Height and Weight

We start by **centering** our data around the origin (0,0)



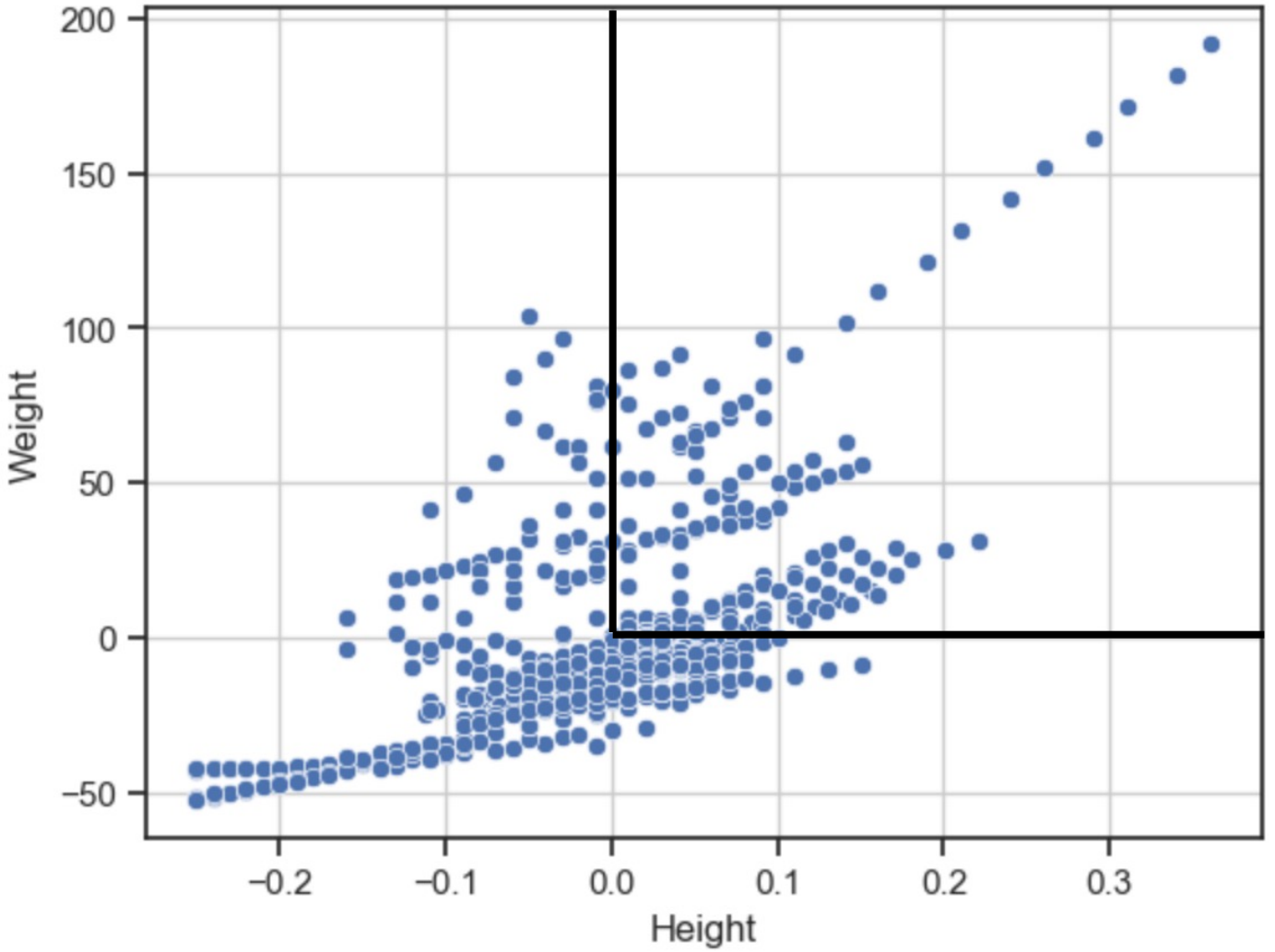
PCA – Intuition

I,e. for each feature (e.g. j), we calculate the mean:

$$\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij}$$

Then, for each datapoint i, we subtract those means:

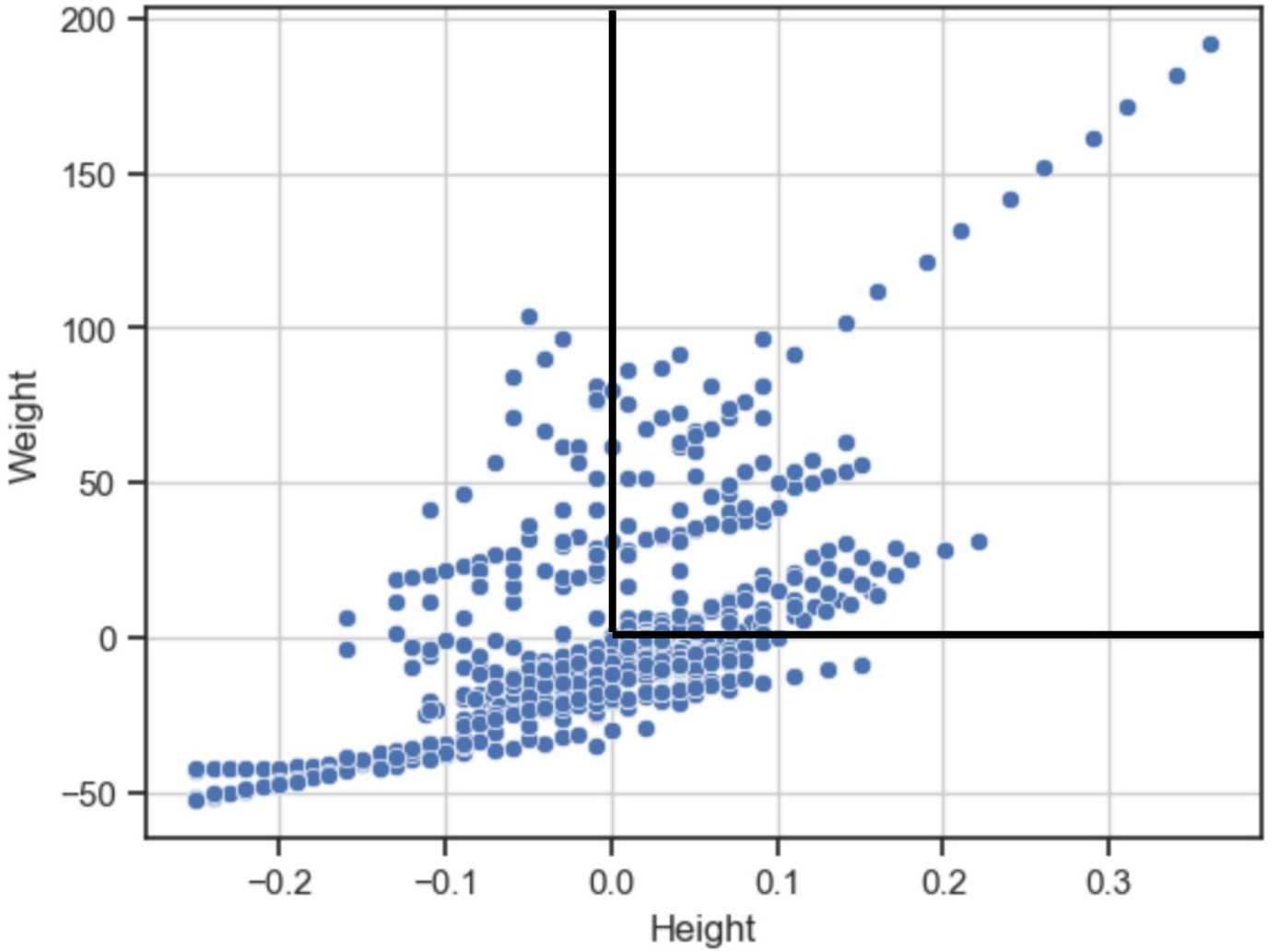
$$x'_{ij} = x_{ij} - \mu_j$$



PCA – Intuition

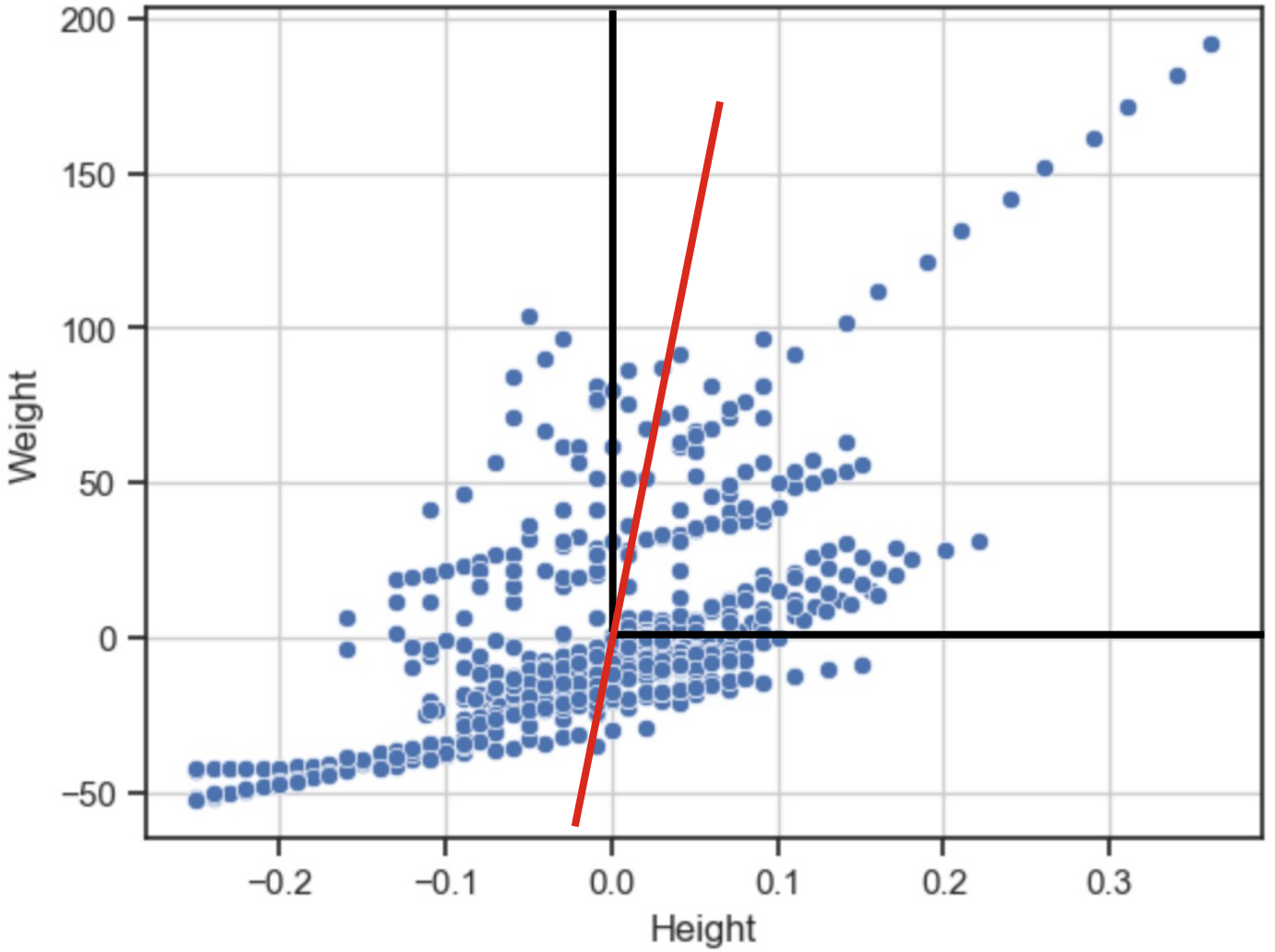
This gives us a new dataset: X'

with a mean height of 0 and a mean weight of 0



PCA – Intuition

If we initialize a random line that intercepts the origin,

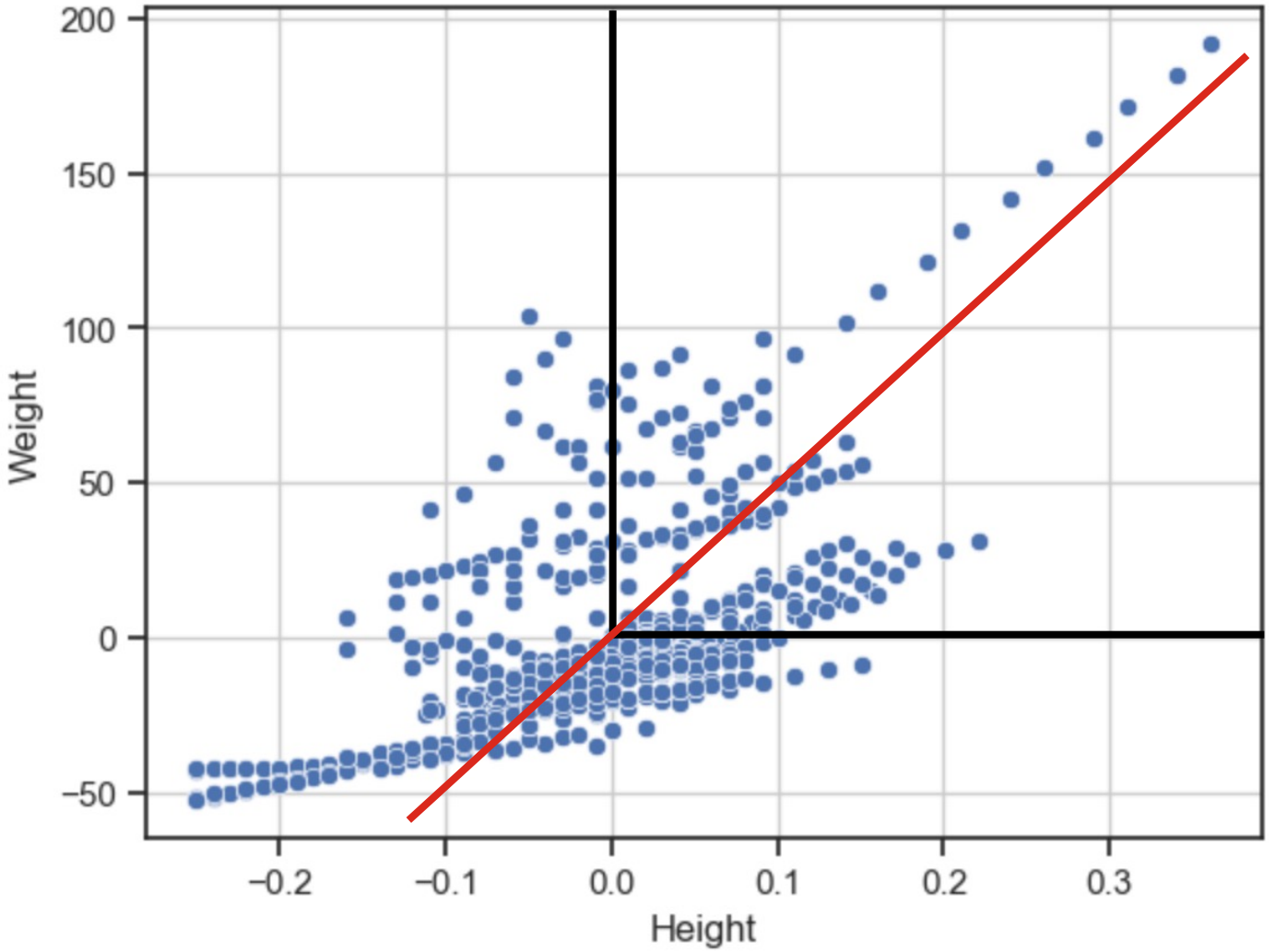


PCA – Intuition

we can then optimize the rotation of that line,

such that we **maximise the variance of the data**

whenever our data points are orthogonally projected onto it.

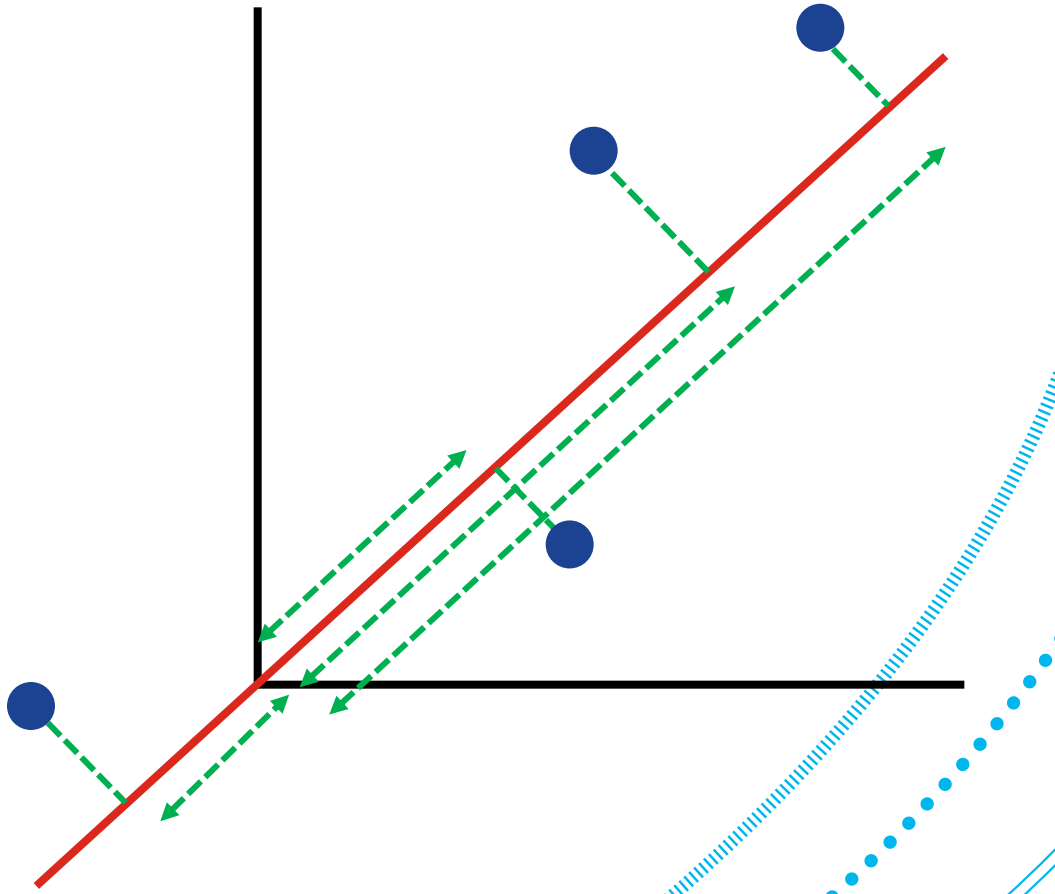


PCA – Intuition

In other words, if we orthogonally project our datapoints onto a line that intercepts the origin,

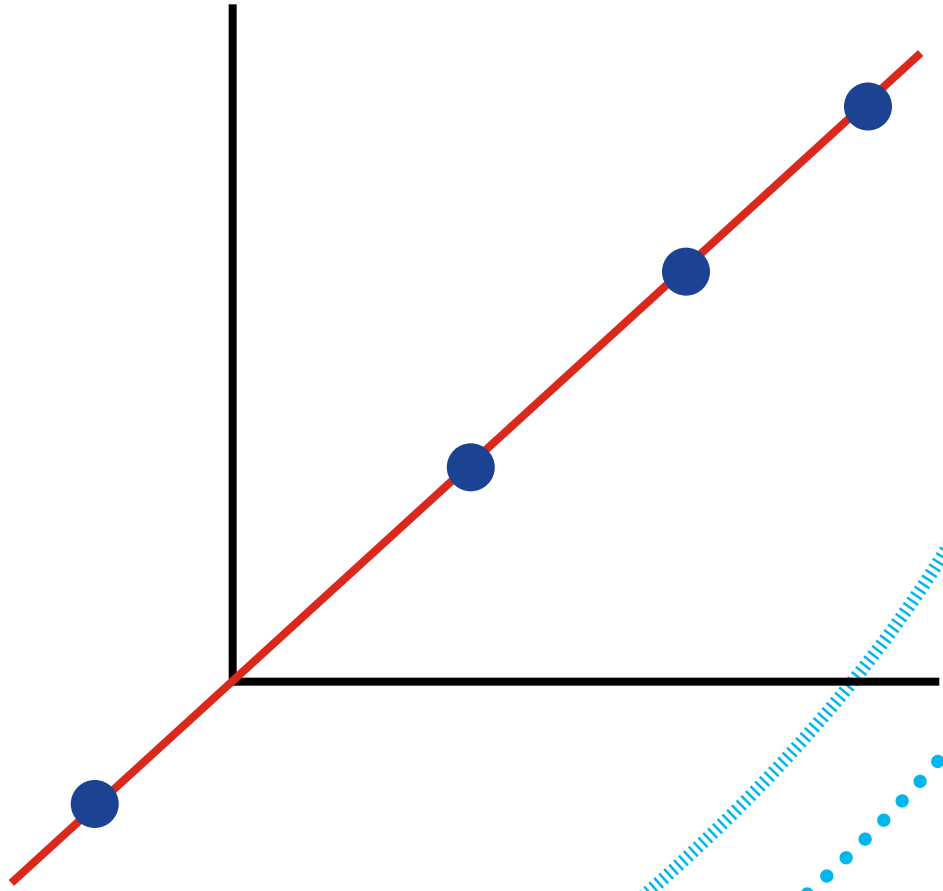
we're looking for the rotation of this line, which **maximises** the Sum of Squared Distances (SS)

between our projected datapoints and the origin (0, 0).



PCA – Intuition

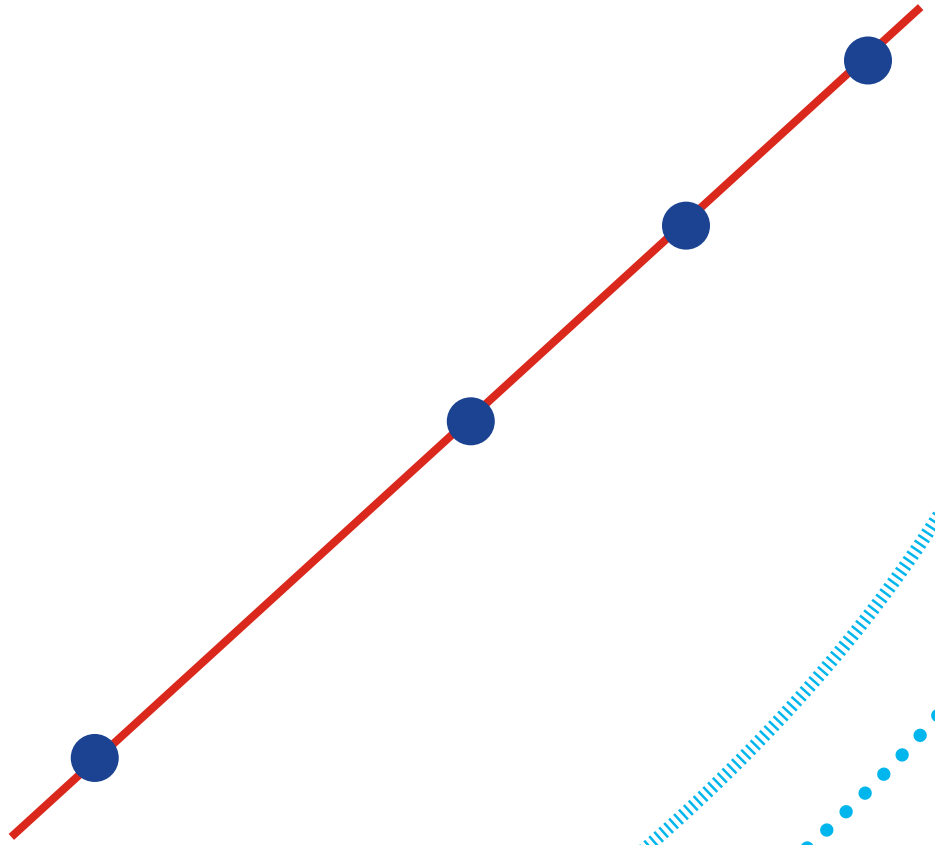
Our projected datapoints then become the values of our new principal component (PC1).



PCA – Intuition

Our projected datapoints then become the values of our new principal component (PC1).

PC1 maintains the underlying patterns of our previous two variables…



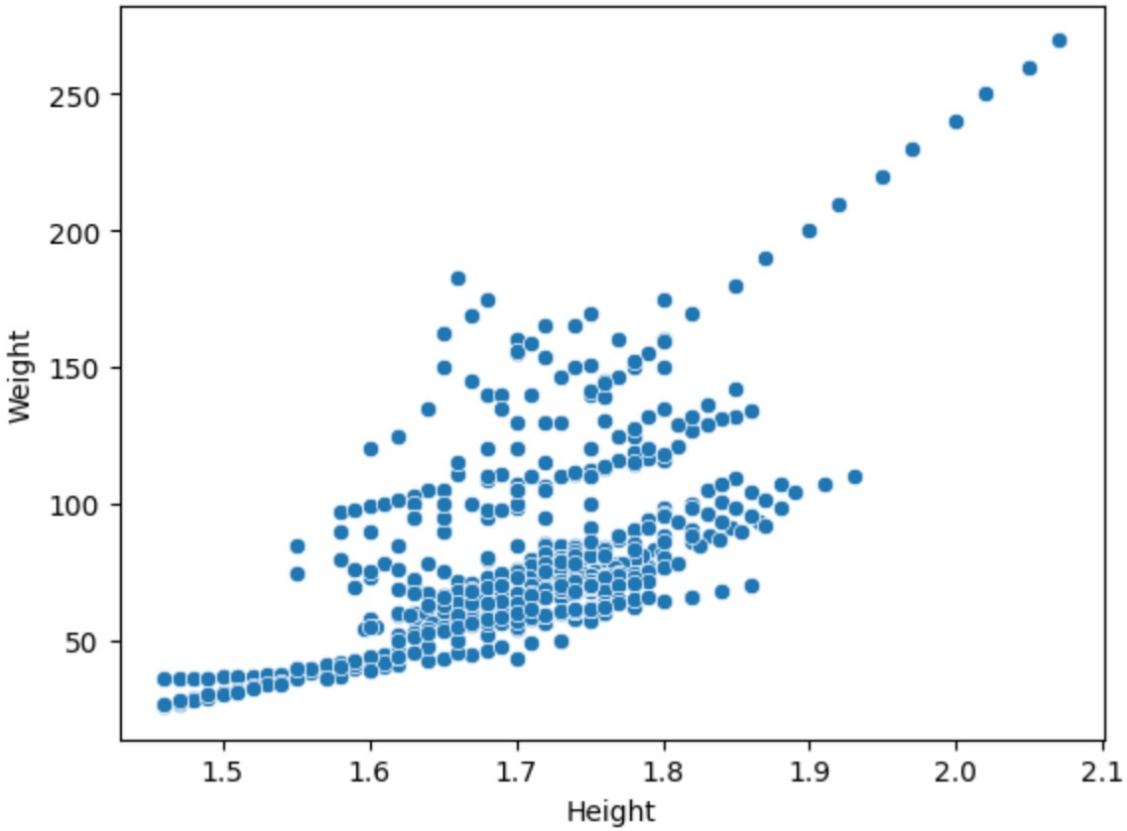
PCA – Intuition

Our projected datapoints then become the values of our new principal component (PC1).

PC1 maintains the underlying patterns of our previous two variables…

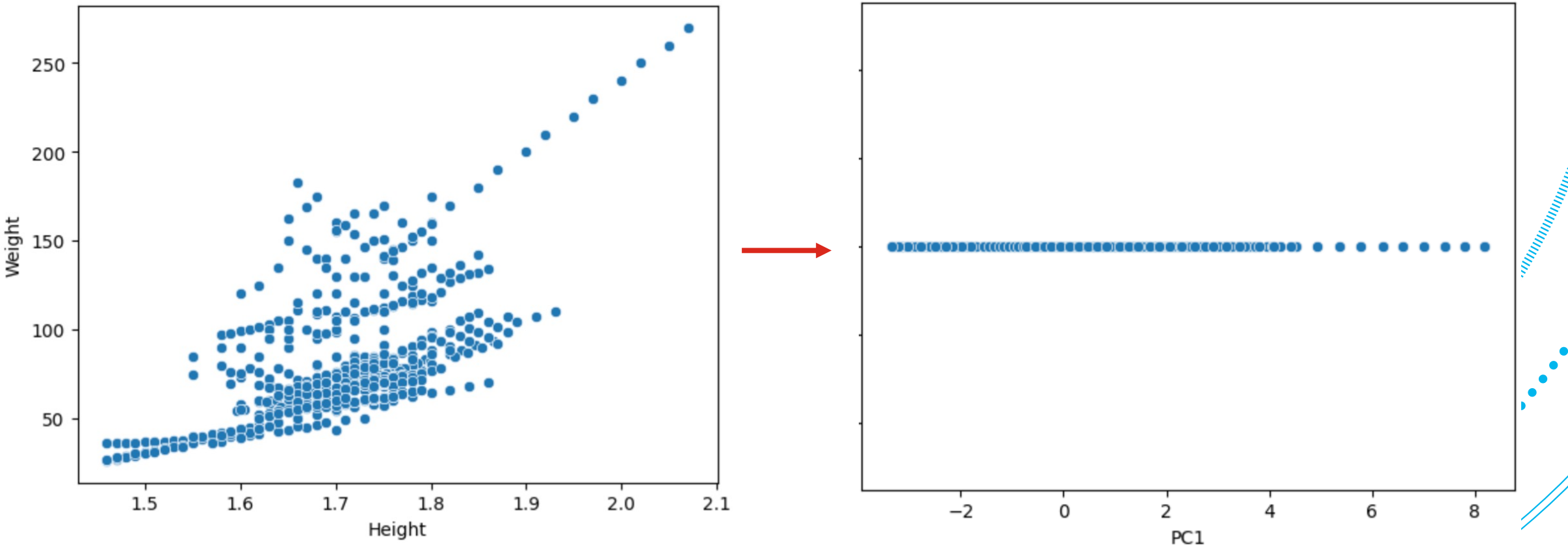
but with only one dimension!

PCA



Going back to our previous data…

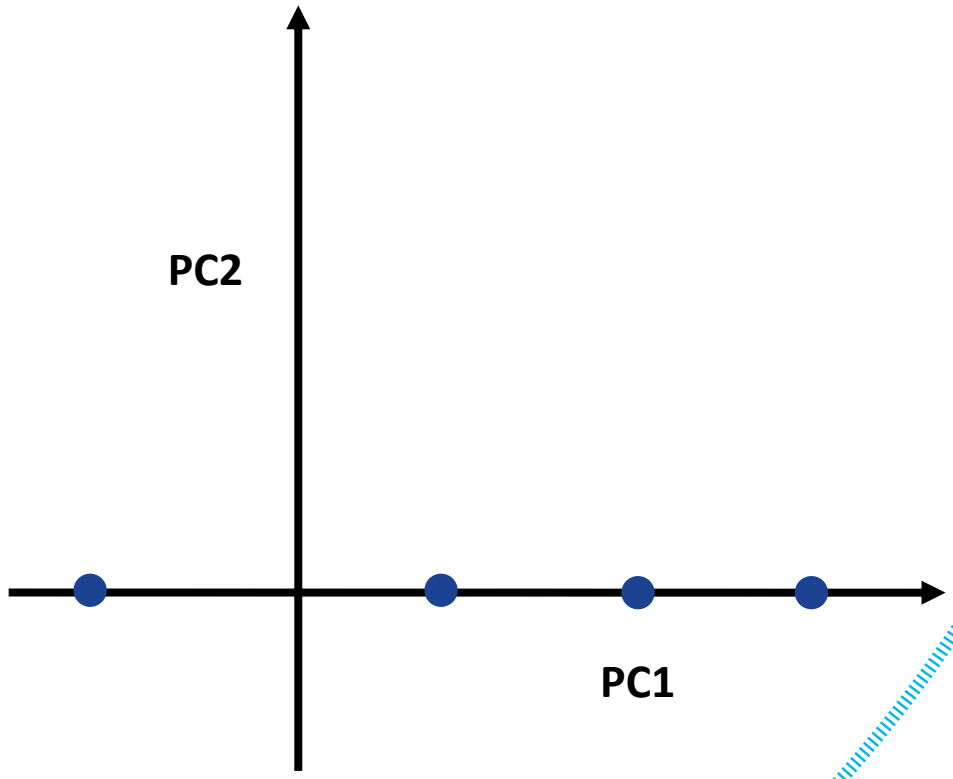
PCA



PCA – Intuition

So PC1 becomes our new X axis.

If we want to find PC2, we then look orthogonally…

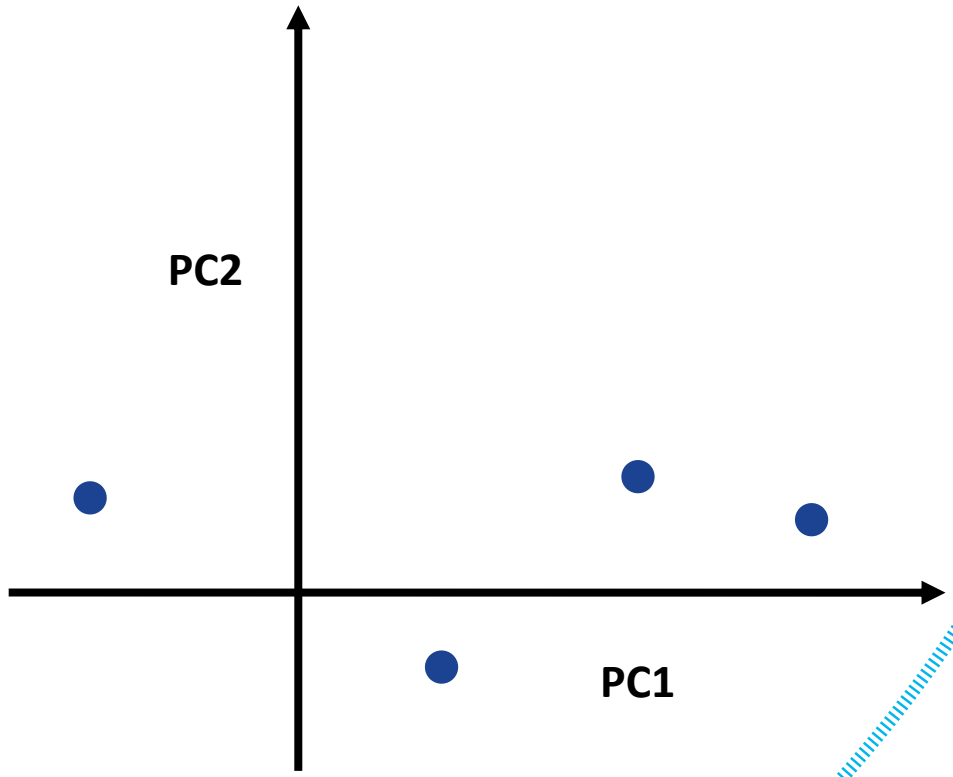


PCA – Intuition

So PC1 becomes our new X axis.

If we want to find PC2, we then look orthogonally…

For 2D data, this becomes our new Y axis.

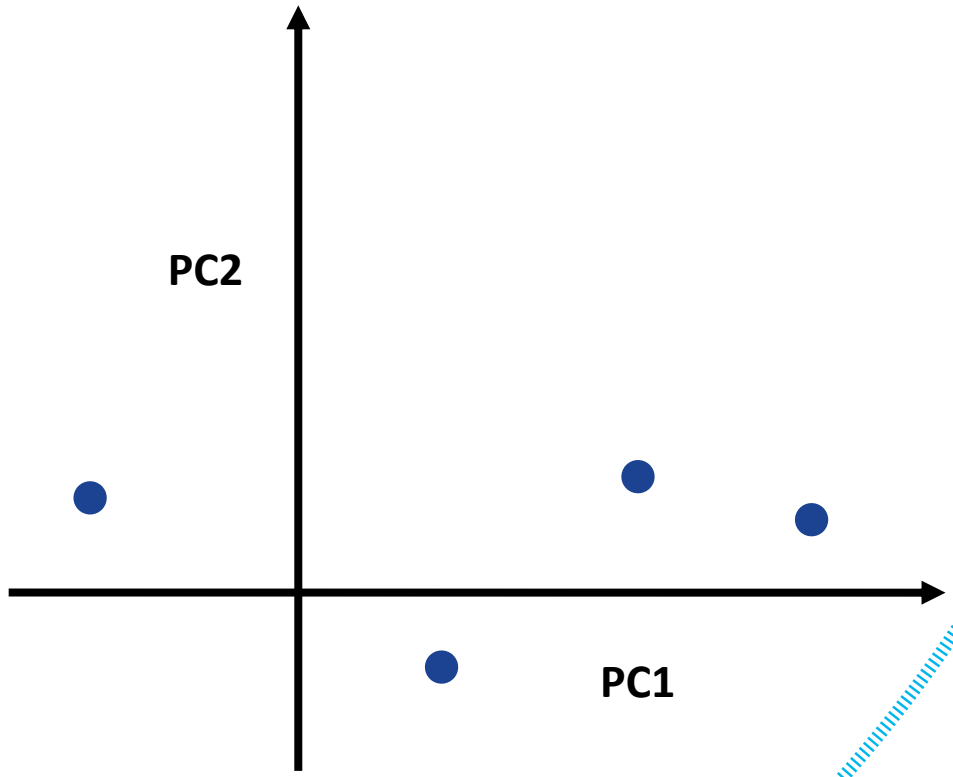


PCA – Intuition

Note that PC1 will always optimize to capture the most variance.

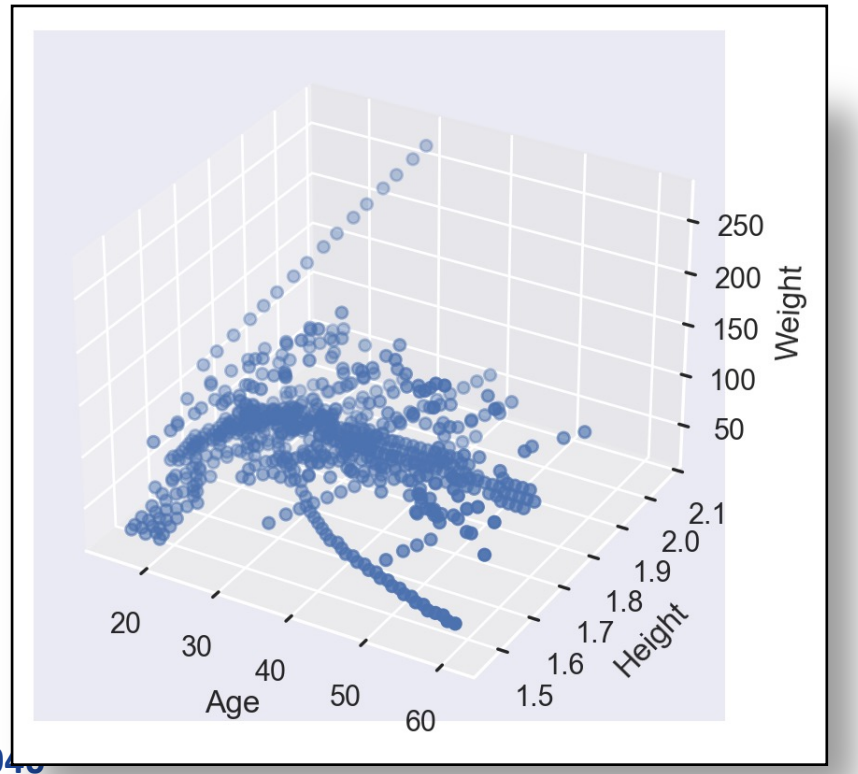
PC1 > PC2 > … > PCN

If you have 3+ dimensions, idea is the same, though it requires a few more steps.



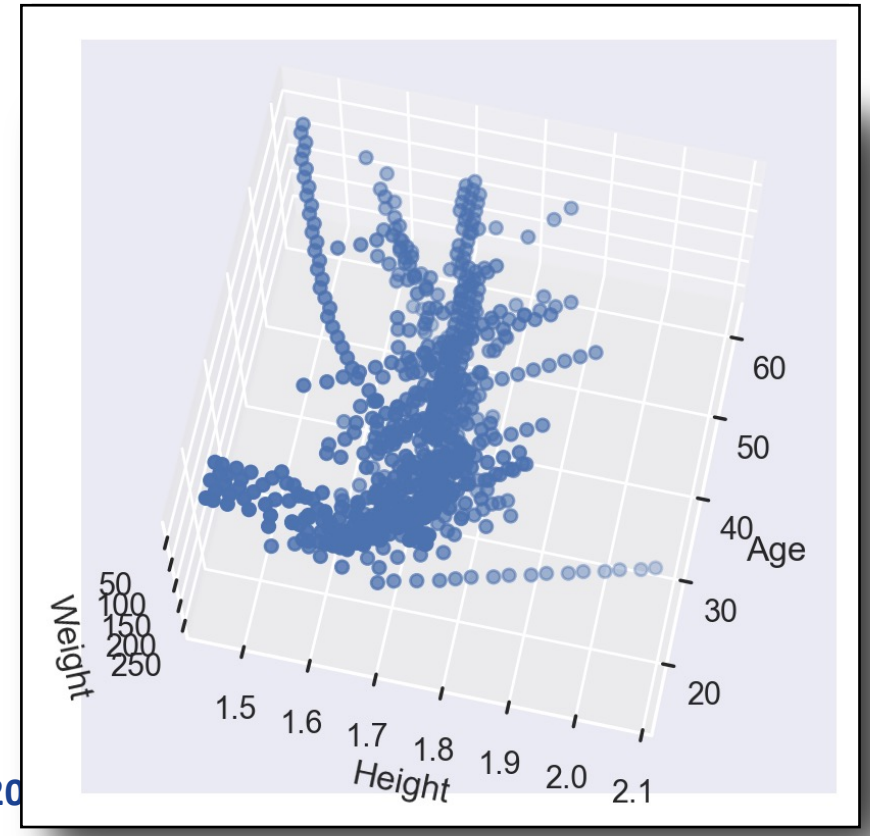
PCA – In Simpler Terms

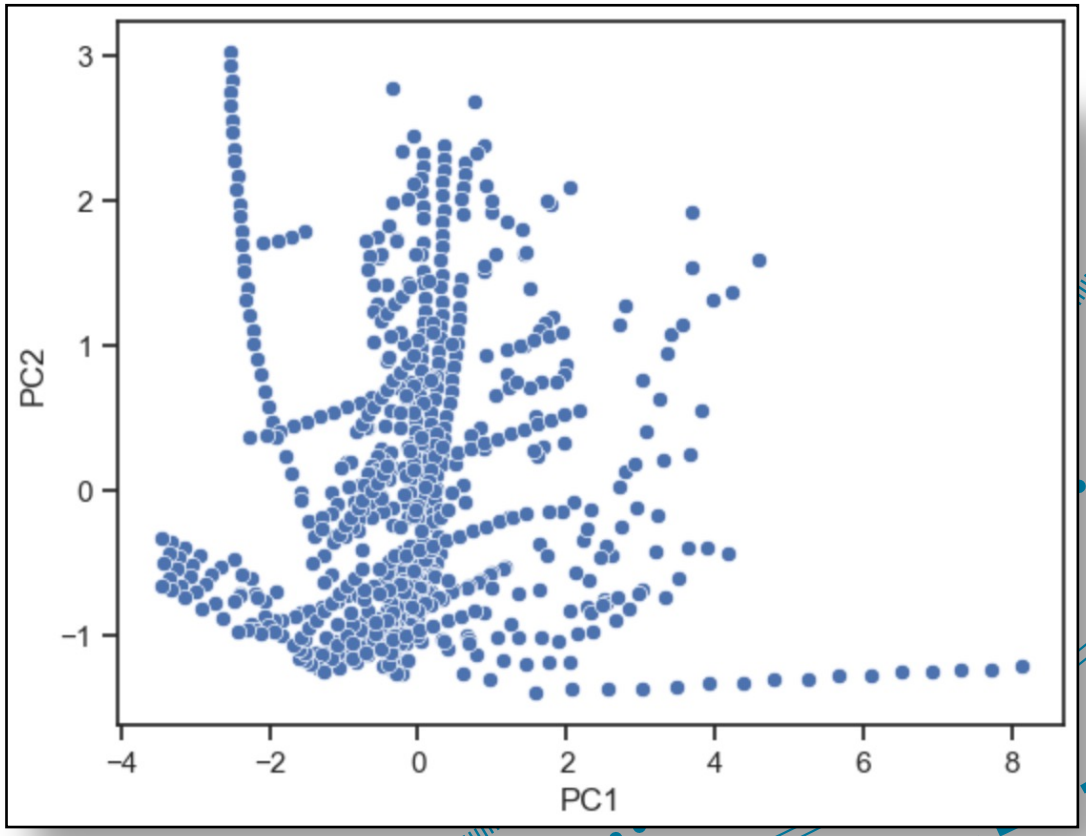
One way to think of what's happening, is imagine we have 3 dimensions of data and we want to find 2 PCs.



PCA – In Simpler Terms

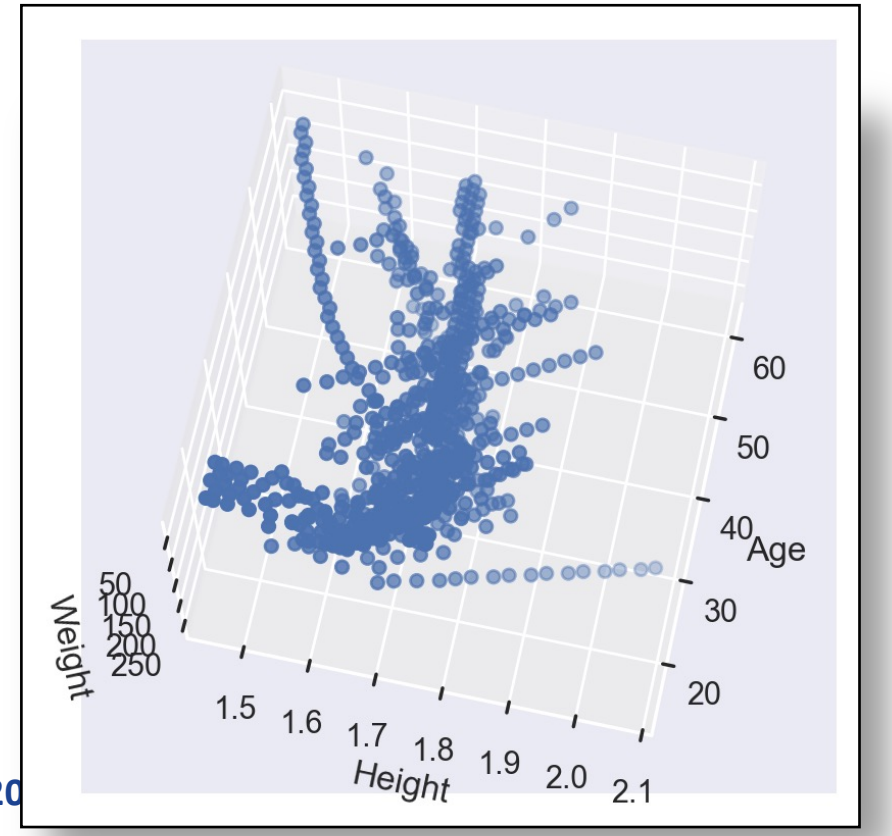
We're effectively 'rotating' our 3D data to map onto 2 axes…

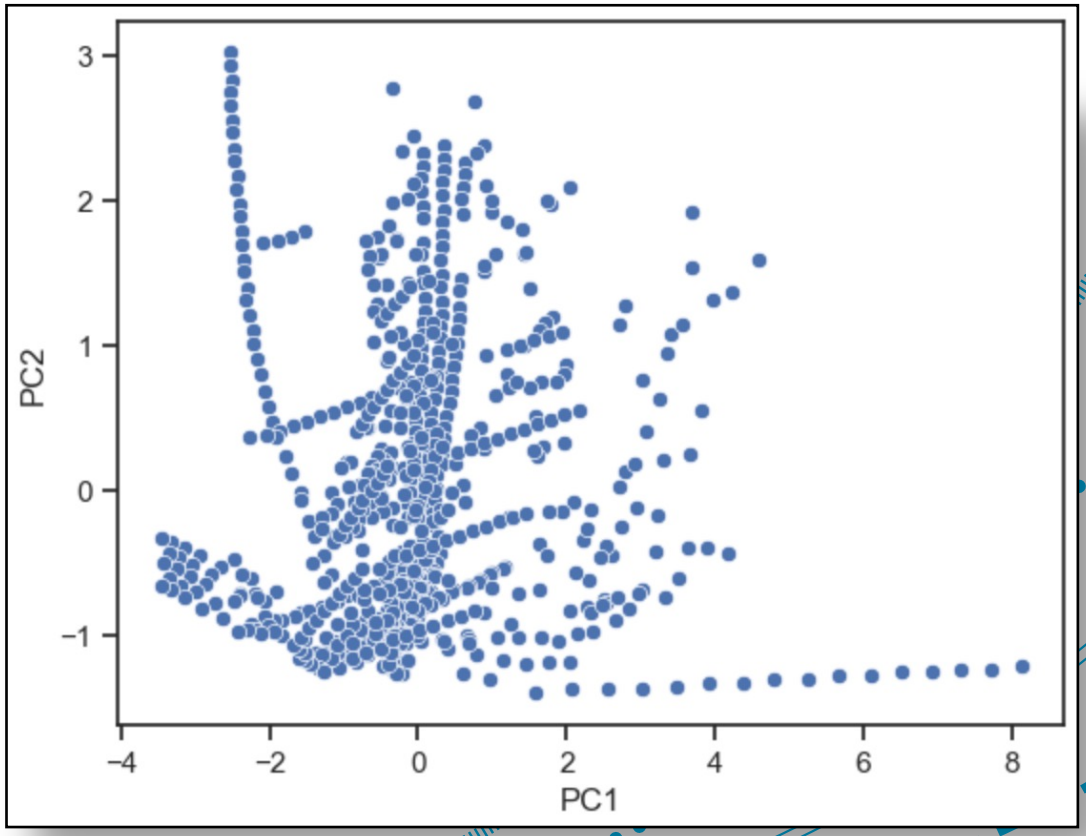




PCA – In Simpler Terms

…to find the best perspective with which to view the most variance within our data!





Implementing PCA

If we want to be a little bit more accurate about how this is typically implemented in practice:

Step 1: We center the data around the origin.

Step 2: We compute the covariance matrix.

Step 3: We compute the *eigenvectors* (direction) and *eigenvalues* (variance) from this matrix.

Step 4: We sort the eigenvectors by their eigenvalues (descending).

Step 5: We multiply our (centered) data by our desired number (k) of principal components.

Advantages of PCA

Compared to other dimensionality reduction techniques, PCA is **relatively fast and computationally efficient**, which makes it scalable.

It also **preserves the global structure** – meaning that it maintains relative distances and directions between the original datapoints.

As a result, PCA offers **better interpretability than other dimensionality reduction approaches**, given that its components are linear combinations of the original data.

Disadvantages of PCA

However, PCA **can only capture linear relationships** in the data, and does not work well with non-linear data.

Furthermore, even though it is *more interpretable* than other dimensionality reduction techniques, the very nature of these principal components means that **they may still be hard to explain naturally**.

Linear Discriminant Analysis

Linear Discriminant Analysis (LDA)

While PCA is perhaps the most well-known form of dimensionality reduction, another commonly used method is:

Linear Discriminant Analysis (LDA).

LDA

Linear Discriminant Analysis (LDA) is very similar to PCA, but instead of reducing the dimensions to **maximise variability**…

We reduce the dimensions to **maximise** the **separability** of some *known** categorical variable.

* Therefore, LDA is technically an example of supervised learning

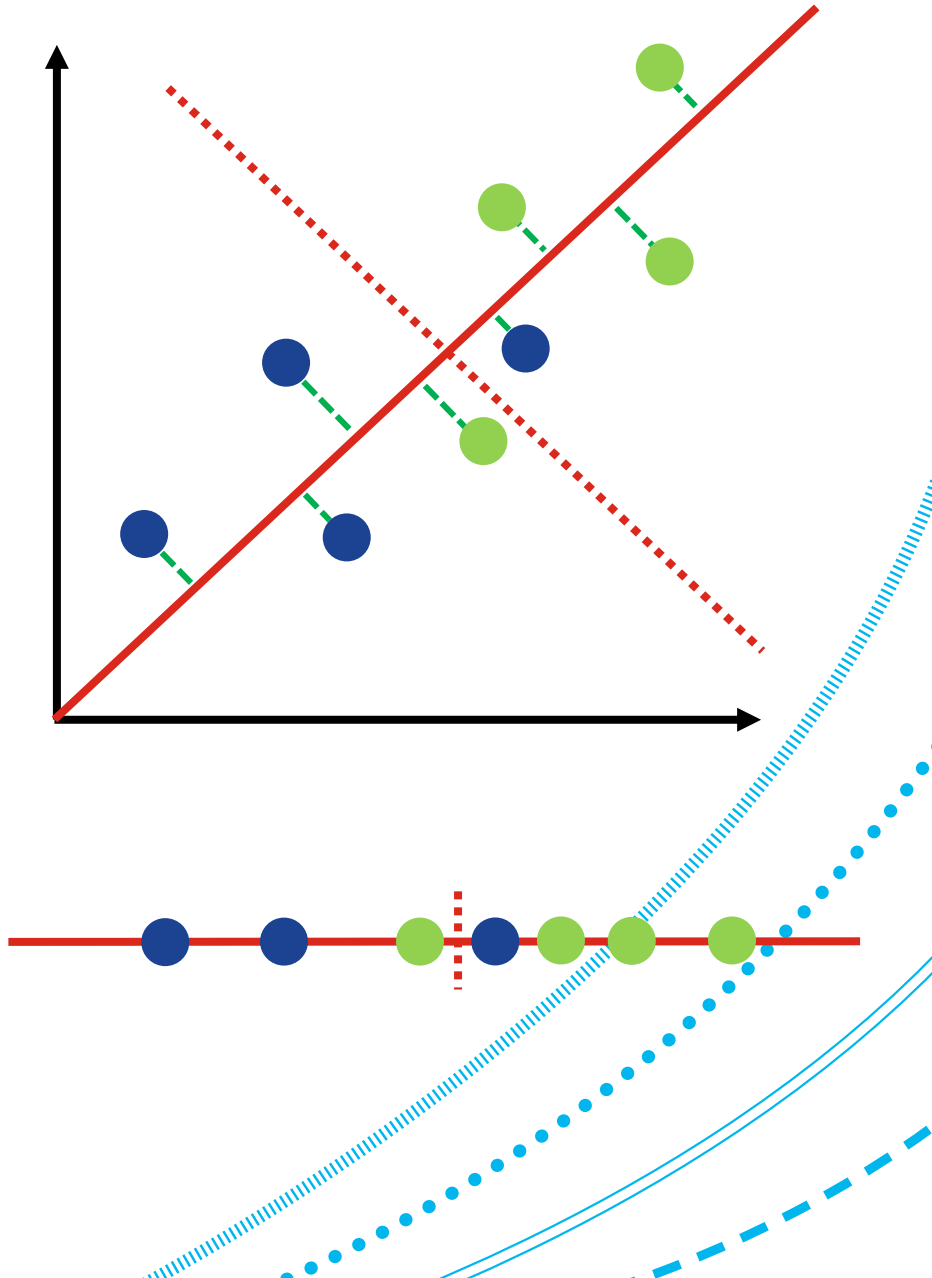
In *some* ways, there are *parallels* between LDA and SVMs (which we talked about during week 2).

LDA – Intuition

Imagine we have 2-dimensional data:

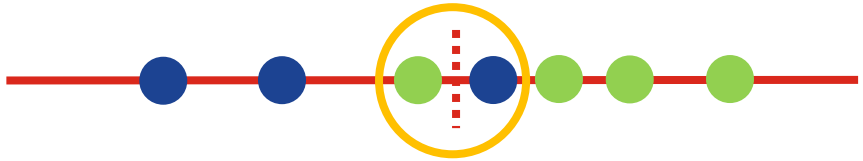
In simple terms, what we want to do is create a new X axis (LD1)…

…which maximises our ability to linearly separate our categories



LDA – Intuition

Note that, unlike (some forms of) SVMs, LDA does not create hard margins.

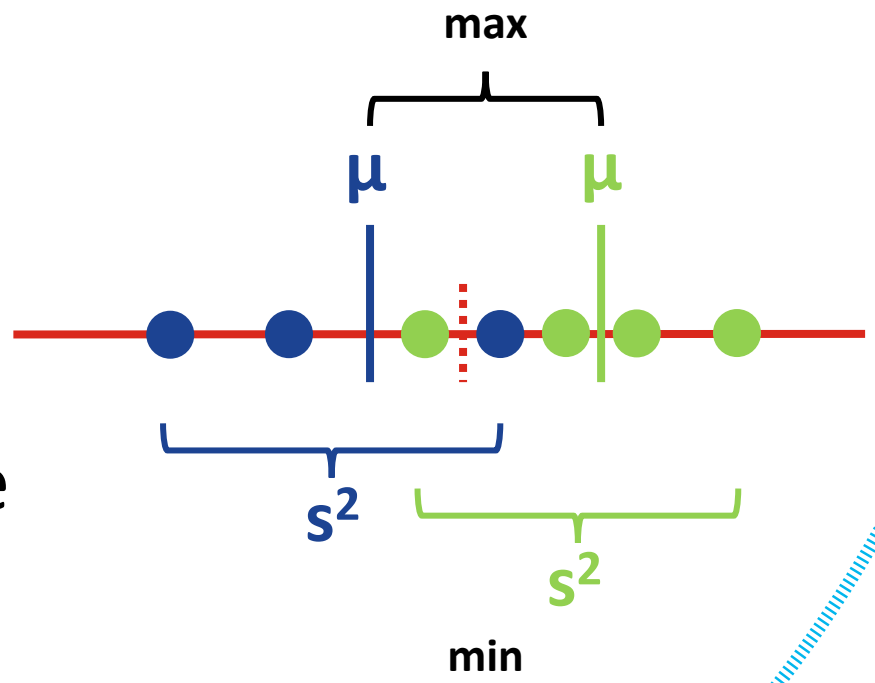


LDA – Intuition

Note that, unlike (some forms of) SVMs, LDA does not create hard margins.

Instead, it looks to **maximise** the distance between the means of each category…

…while **minimising** the dispersal, or scatter, within each category.



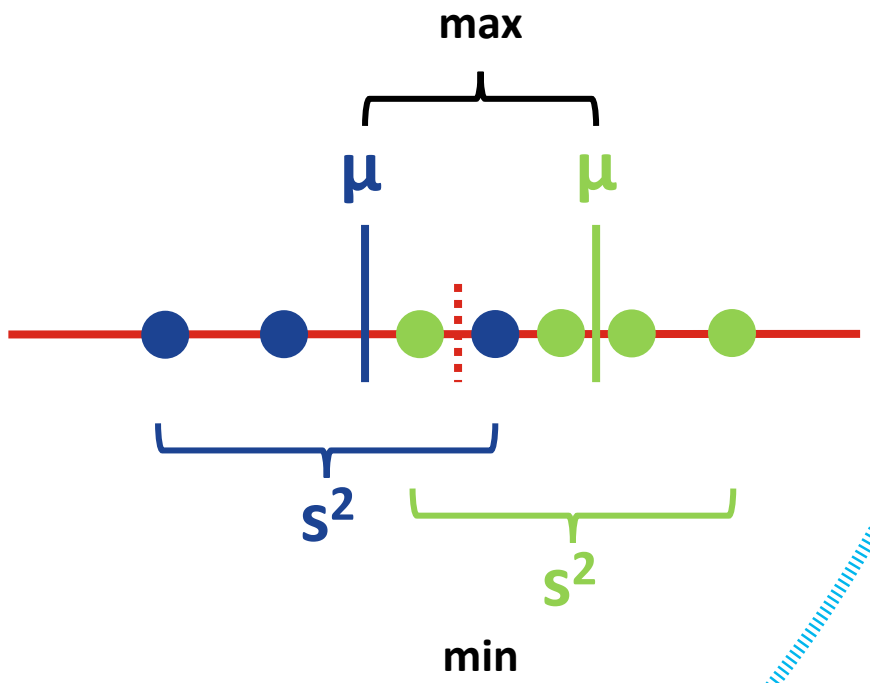
LDA – Intuition

In other words, we look to find the direction of the line that **maximises the ratio** of *between-class scatter* to *within-class scatter*:

$$\frac{(\mu - \mu)^2}{s^2 + s^2}$$

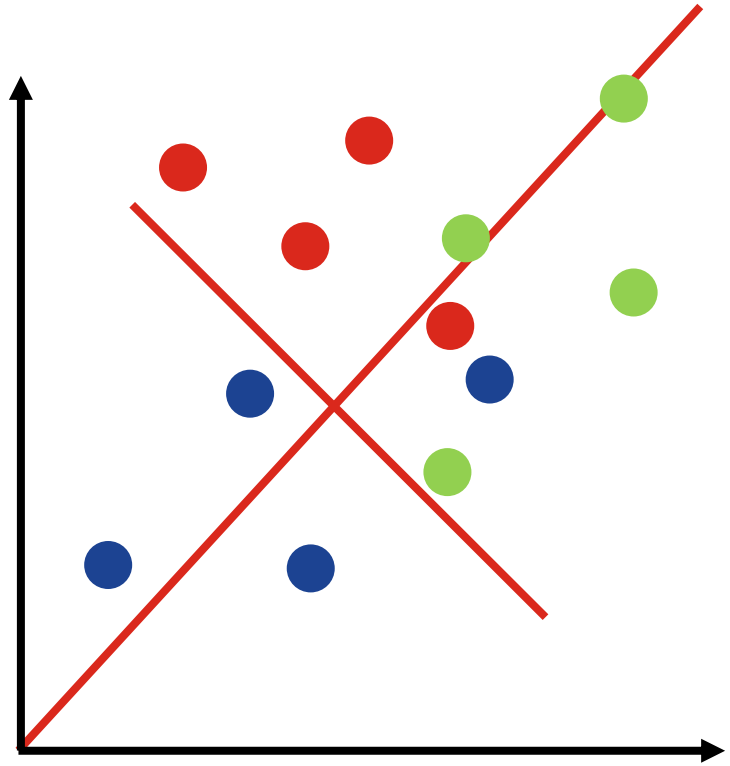
(between class scatter)

(within class scatter)



LDA – Intuition

If we have more than two categories, the process is similar



$$\frac{(\mu - \mu)^2 + (\mu - \mu)^2 + (\mu - \mu)^2}{s^2 + s^2 + s^2}$$

Except we create *two* new orthogonal axes (LD1, LD2).

Implementing PCA

Again, a slightly more practical implementation of LDA involves:

Step 1: We compute class means and the overall mean across all data.
Step 2: We compute a within-class scatter matrix and a between-class scatter matrix.
Step 3: We compute the optimal *eigenvectors* (direction) and *eigenvalues* (variance).
Step 4: We sort the eigenvectors by their eigenvalues (descending).
Step 5: We multiply our (original) data by our desired number of linear discriminants (k; up to a maximum number of C-1).

LDA

In all, like with PCA, LDA also works to minimize large numbers of dimensions (variables) into a smaller number of linear discriminants (LD1, LD2, etc.).

However, unlike PCA, it does so with a focus on **linearly separating the data** based on class labels.

Advantages of LDA

Unlike PCA and many other forms of dimensionality reduction, LDA can **account for class labels**, making it a supervised learning approach.

It is also **relatively fast and computationally efficient**, like PCA is.

And, like PCA, it is **more interpretable** than some (non-linear) dimensionality approaches.

Disadvantages of LDA

However, again, LDA **can only capture linear relationships** in the data, and does not work well with non-linear data.

It can also **distort the global structure** of the data, meaning that it may not preserve the overall geometric distances and characteristics of the data if doing so improves the separation of the classes.

It can also be **sensitive to outliers or noisy data**.

Other Forms of Dimensionality Reduction

Non-Linear Dimensionality Reduction

There also exist approaches for *non-linear* dimensionality reduction, which we'll discuss in a bit more detail in the next lecture.

These include **t-Distributed Stochastic Neighbor Embedding** (**t-SNE**), and **Uniform Manifold Approximation and Projection** (**UMAP**).

Both of these can handle complex, nonlinear relationships within data, though they have different approaches and characteristics for doing so.