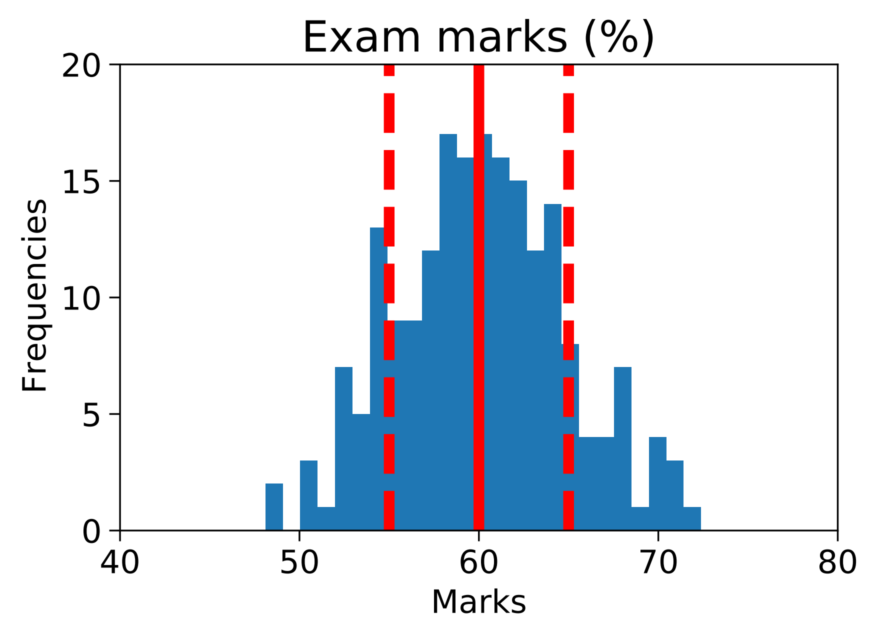
Data Analysis and Machine Learning 4 (DAML)

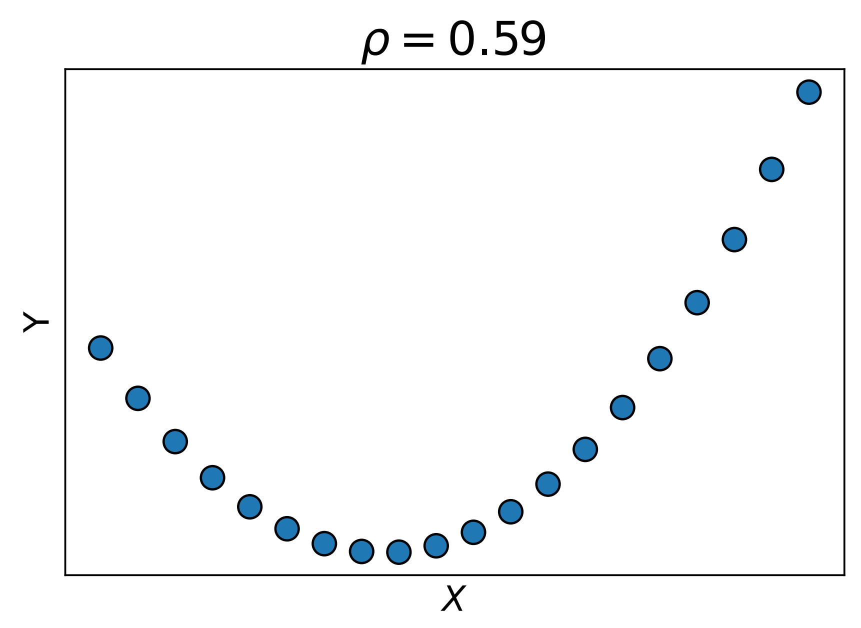
Week 3: Preprocessing, PCA, clustering

Elliot J. Crowley, 29th January 2024

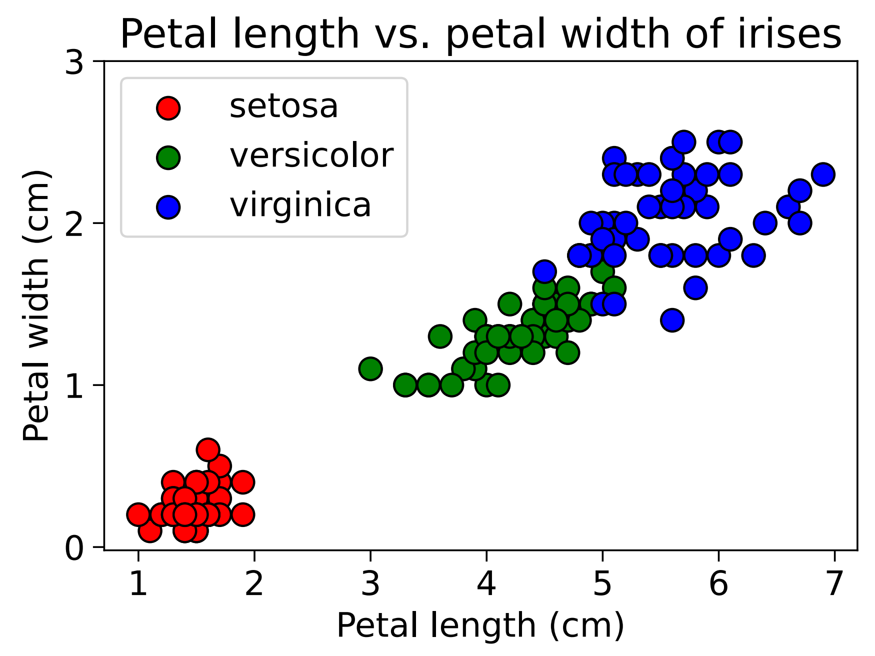
Recap

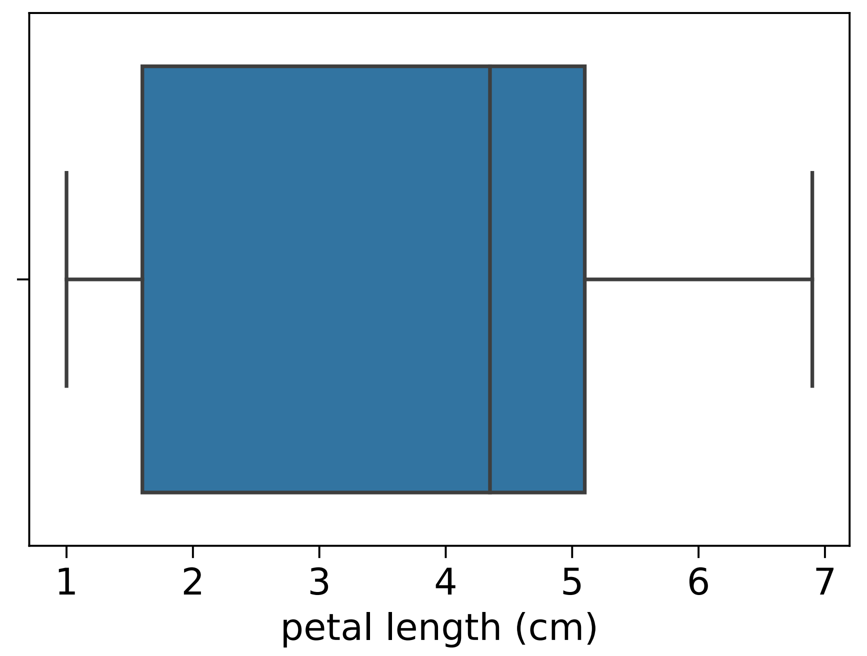
- We reviewed summary statistics for datasets





- We considered different ways to visualise data





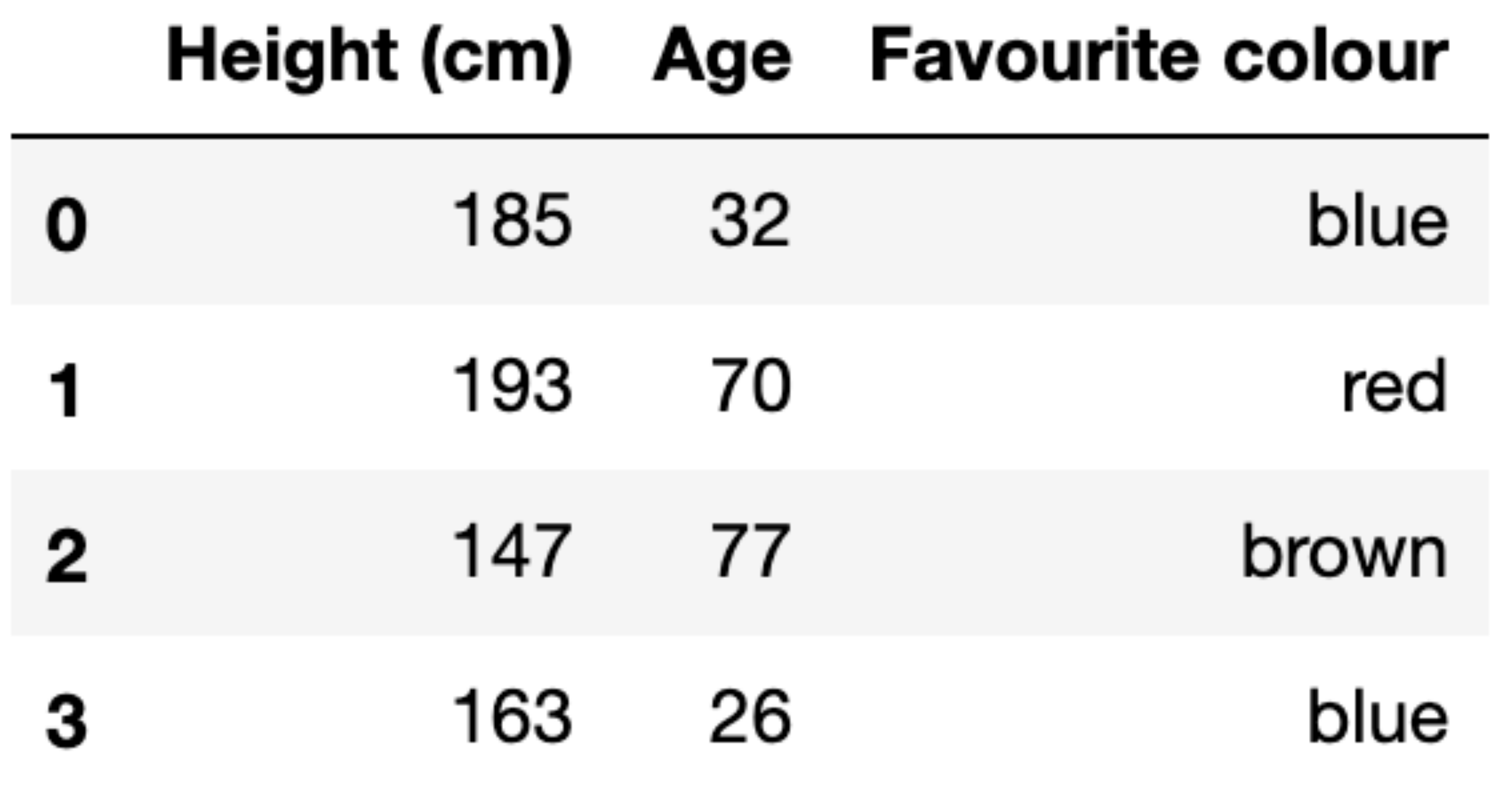
This week

- You will learn how to preprocess data so it can be used for various algorithms
- You will learn about principal component analysis (PCA) and how it can be used for dimensionality reduction
- You will find out how to cluster data using the K-means algorithm

Preprocessing

Matrix inputs

- PCA and many machine learning (ML) methods require a matrix input
- Our dataset must (usually) be represented by a matrix of real numeric values
- Discrete and continuous are both fine; we just pretend everything is continuous
- Given tabular data, we need to convert it into such a matrix

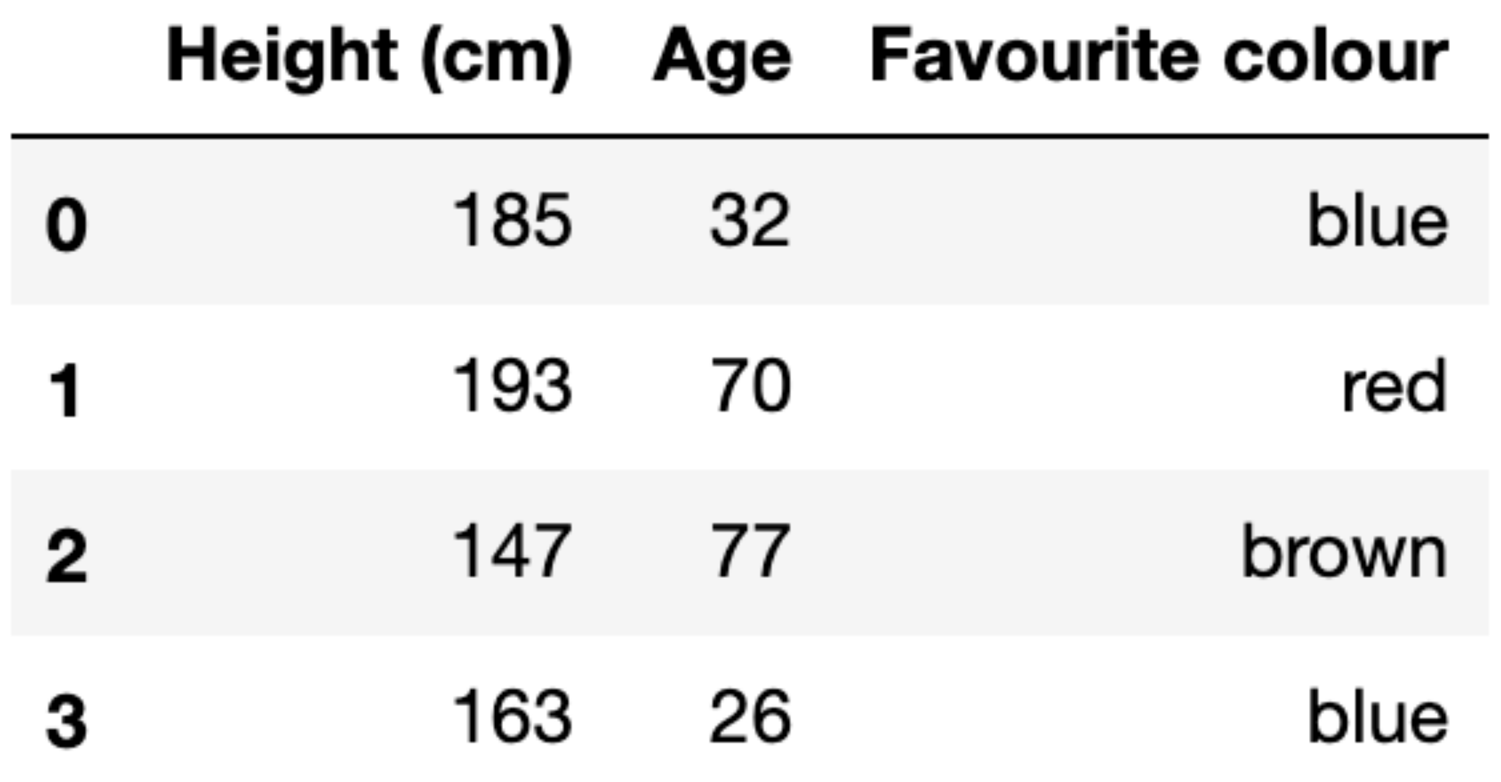


	Height (cm)	Age	Favourite colour
0	185	32	blue
1	193	70	red
2	147	77	brown
3	163	26	blue

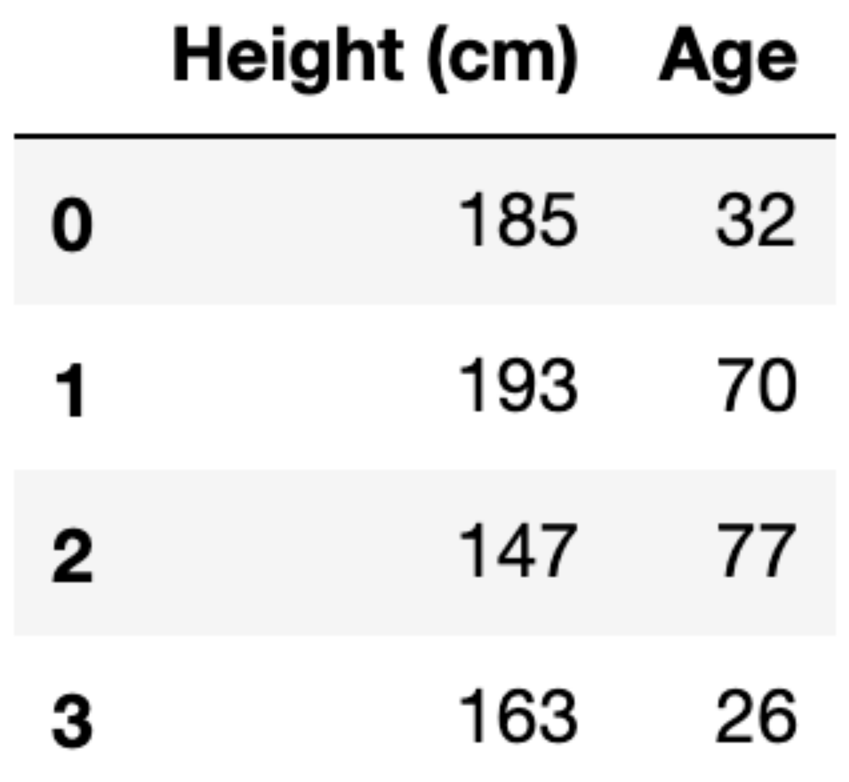
$\Rightarrow$?

Representing a dataset as a matrix

- We have tabular data with N data points (rows) and C features (cols)
- For now, we will drop features that don't correspond to numeric variables
- If there are now D features we can represent the dataset by a $N \times D$ matrix



	Height (cm)	Age	Favourite colour
0	185	32	blue
1	193	70	red
2	147	77	brown
3	163	26	blue



	Height (cm)	Age
0	185	32
1	193	70
2	147	77
3	163	26

$$\mathbf{X} = \begin{bmatrix} 185 & 32 \\ 193 & 70 \\ 147 & 77 \\ 163 & 26 \end{bmatrix}$$

$$\mathbf{X} \in \mathbb{R}^{N \times D}$$

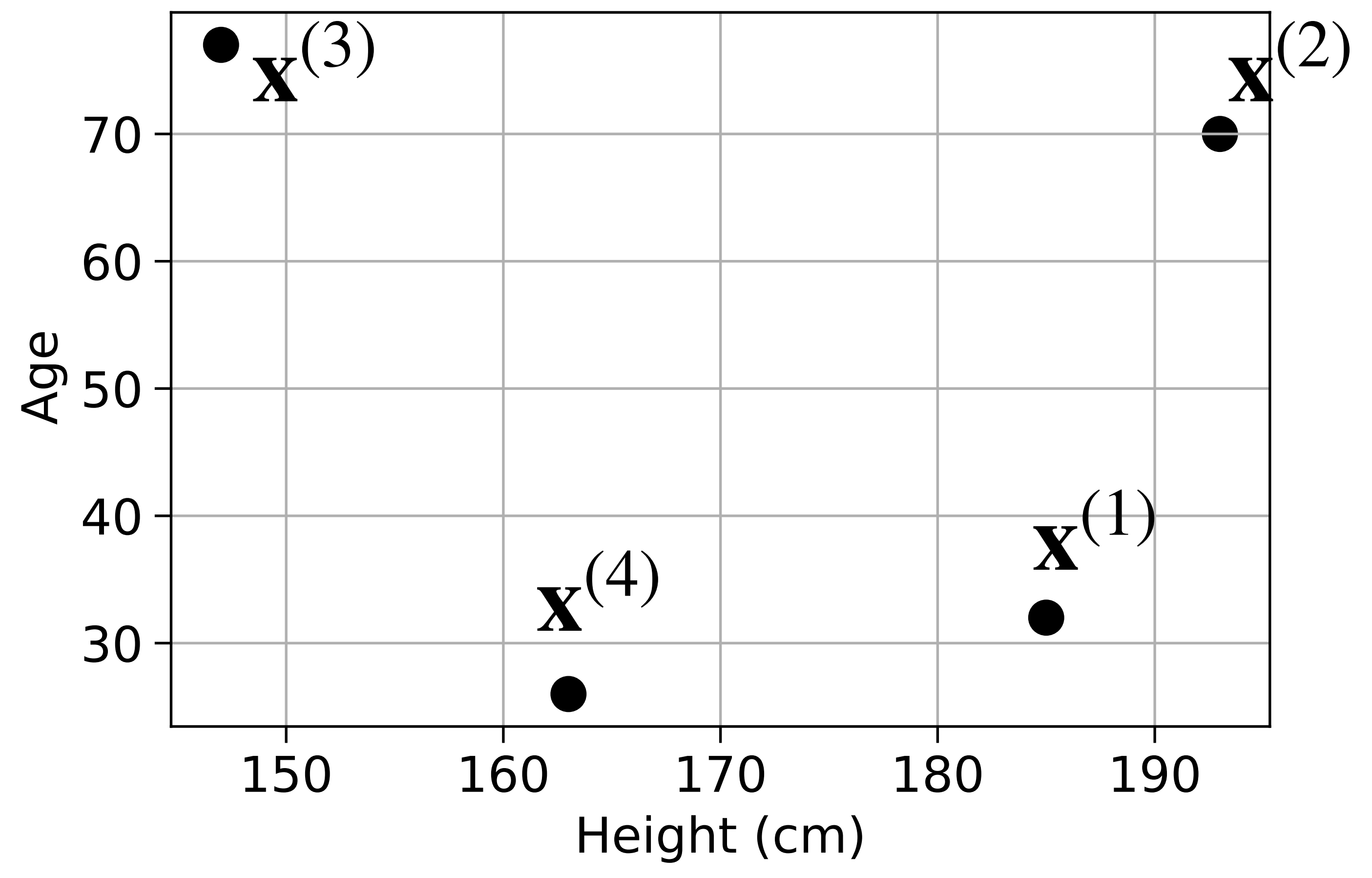
Representing data points as vectors

- We are representing our dataset using a $N \times D$ dataset matrix $\mathbf{X}$
- Each row is a data point that lives in D-dimensional space
- Let's denote these as $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \ldots, \mathbf{x}^{(N)}$ or $\{\mathbf{x}^{(n)}\}_{n=1}^{N}$. They are **vectors**

	Height (cm)	Age
0	185	32
1	193	70
2	147	77
3	163	26

$$\mathbf{X} = \begin{bmatrix} 185 & 32 \\ 193 & 70 \\ 147 & 77 \\ 163 & 26 \end{bmatrix}$$

$$\mathbf{X} \in \mathbb{R}^{N \times D}$$

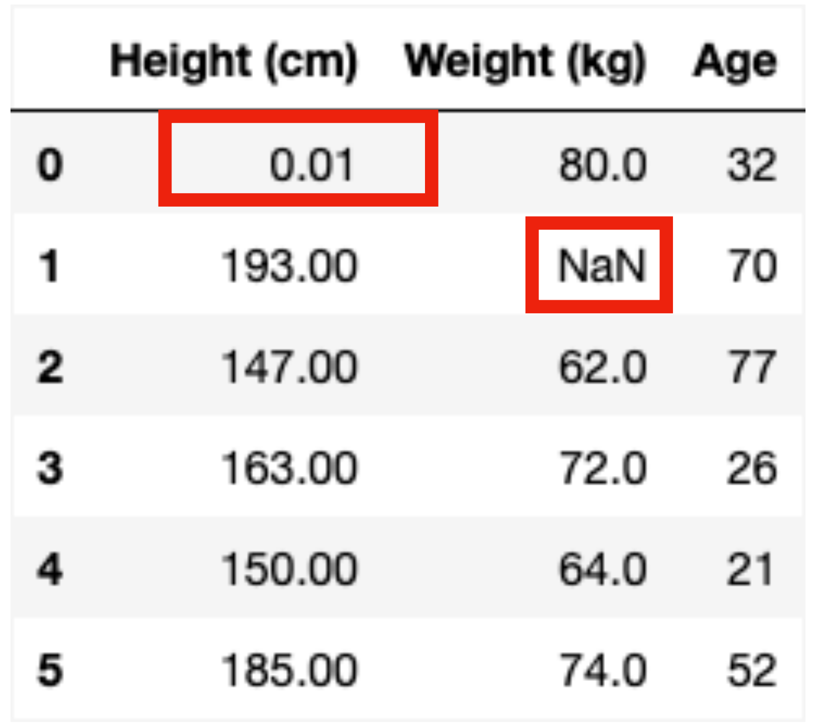


$$\mathbf{x} \in \mathbb{R}^{D}$$

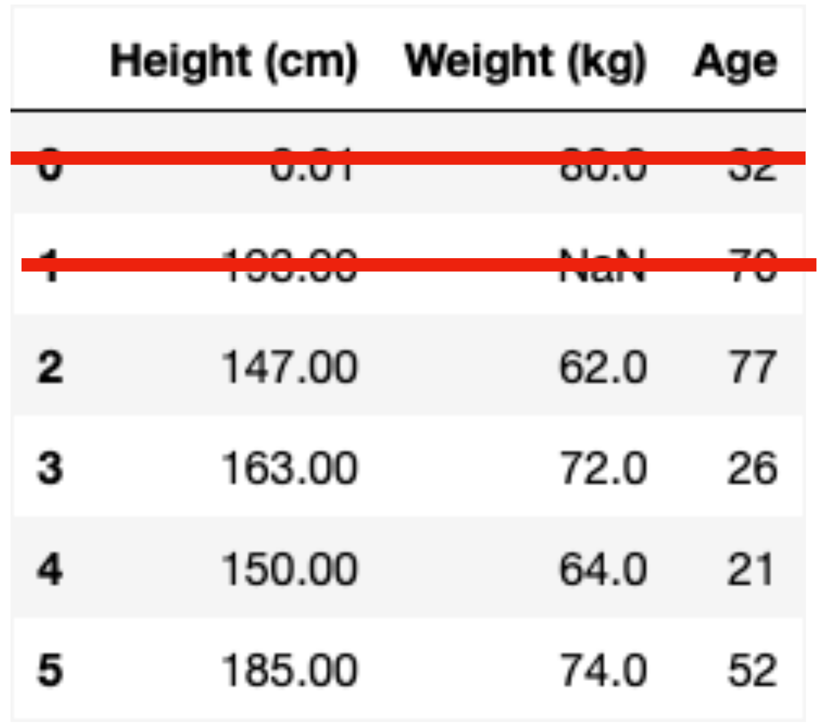
Credit to Iain Murray for the notation

What if we have a missing (or wrong!) value for a feature?

- **Option 1**: Remove the affected data point(/s)



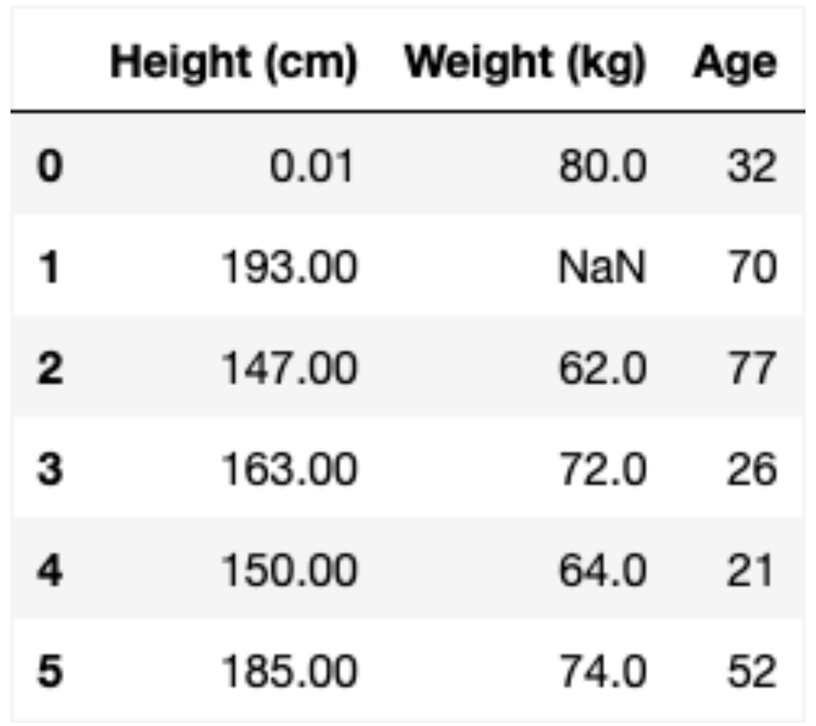
	Height (cm)	Weight (kg)	Age
0	0.01	80.0	32
1	193.00	NaN	70
2	147.00	62.0	77
3	163.00	72.0	26
4	150.00	64.0	21
5	185.00	74.0	52



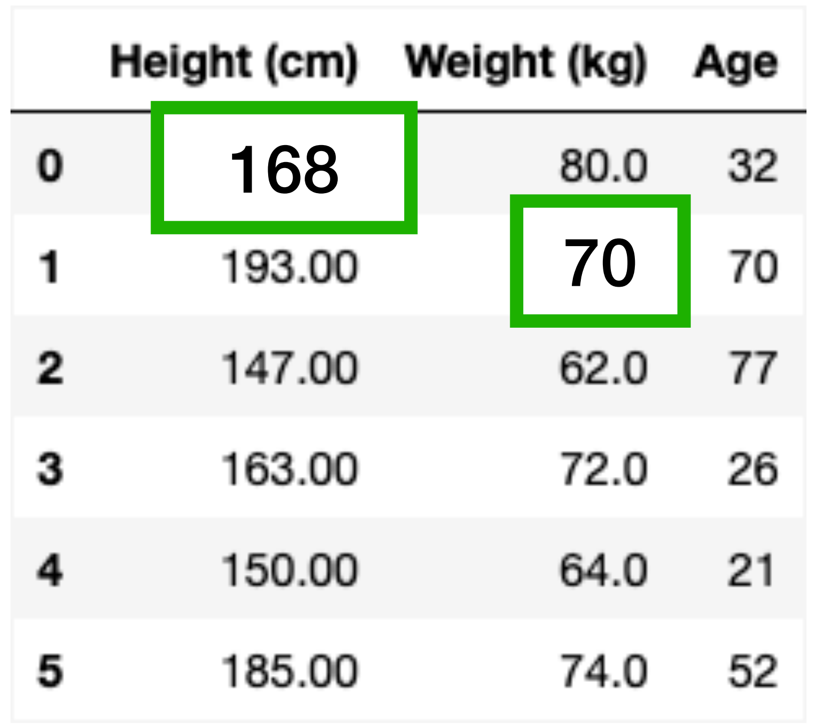
	Height (cm)	Weight (kg)	Age
~~**0**~~	~~0.01~~	~~80.0~~	~~32~~
~~**1**~~	~~193.00~~	~~NaN~~	~~70~~
2	147.00	62.0	77
3	163.00	72.0	26
4	150.00	64.0	21
5	185.00	74.0	52

$$\mathbf{X} = \begin{bmatrix} 147 & 62 & 77 \\ 163 & 72 & 26 \\ 150 & 64 & 21 \\ 185 & 74 & 52 \end{bmatrix}$$

- **Option 2**: Impute a value (e.g. the average for that feature)



	Height (cm)	Weight (kg)	Age
0	0.01	80.0	32
1	193.00	NaN	70
2	147.00	62.0	77
3	163.00	72.0	26
4	150.00	64.0	21
5	185.00	74.0	52

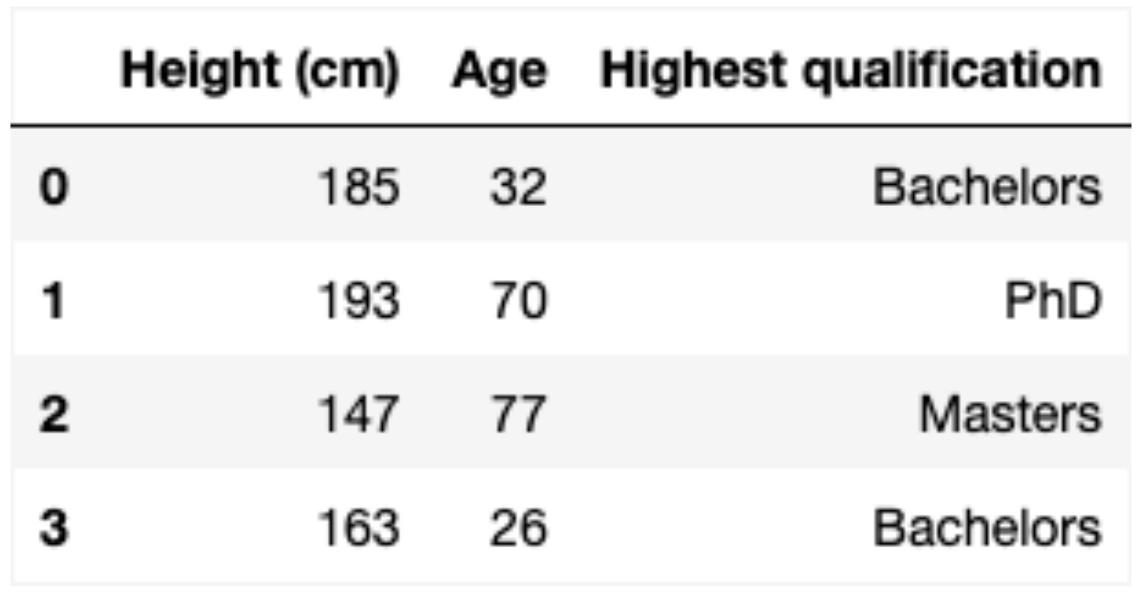


	Height (cm)	Weight (kg)	Age
0	168	80.0	32
1	193.00	70	70
2	147.00	62.0	77
3	163.00	72.0	26
4	150.00	64.0	21
5	185.00	74.0	52

$$\mathbf{X} = \begin{bmatrix} 168 & 80 & 32 \\ 193 & 70 & 60 \\ 147 & 62 & 77 \\ 163 & 72 & 26 \\ 150 & 64 & 21 \\ 185 & 74 & 52 \end{bmatrix}$$

What if we want to include categorical variables?

- **If ordinal** we can map to numbers that maintain order

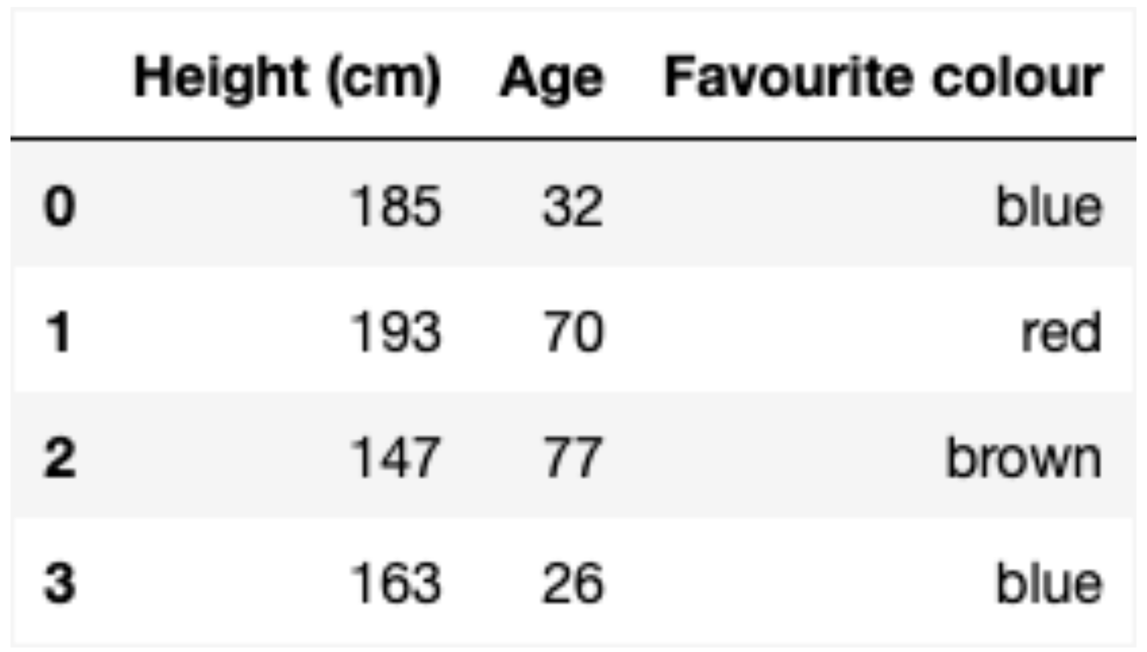


	Height (cm)	Age	Highest qualification
0	185	32	Bachelors
1	193	70	PhD
2	147	77	Masters
3	163	26	Bachelors

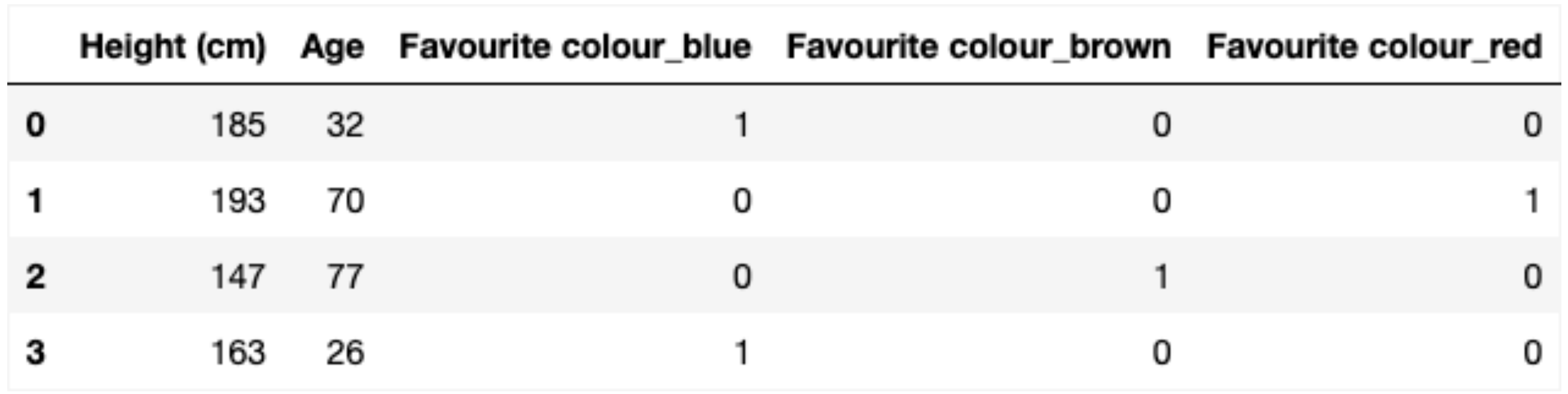
{'Bachelors': 1,
'Masters': 2,
'PhD': 3,}

$$\mathbf{X} = \begin{bmatrix} 185 & 32 & 1 \\ 193 & 70 & 3 \\ 147 & 77 & 2 \\ 163 & 26 & 1 \end{bmatrix}$$

- **If nominal** we can create a **binary feature** for each category



	Height (cm)	Age	Favourite colour
0	185	32	blue
1	193	70	red
2	147	77	brown
3	163	26	blue



	Height (cm)	Age	Favourite colour_blue	Favourite colour_brown	Favourite colour_red
0	185	32	1	0	0
1	193	70	0	0	1
2	147	77	0	1	0
3	163	26	1	0	0

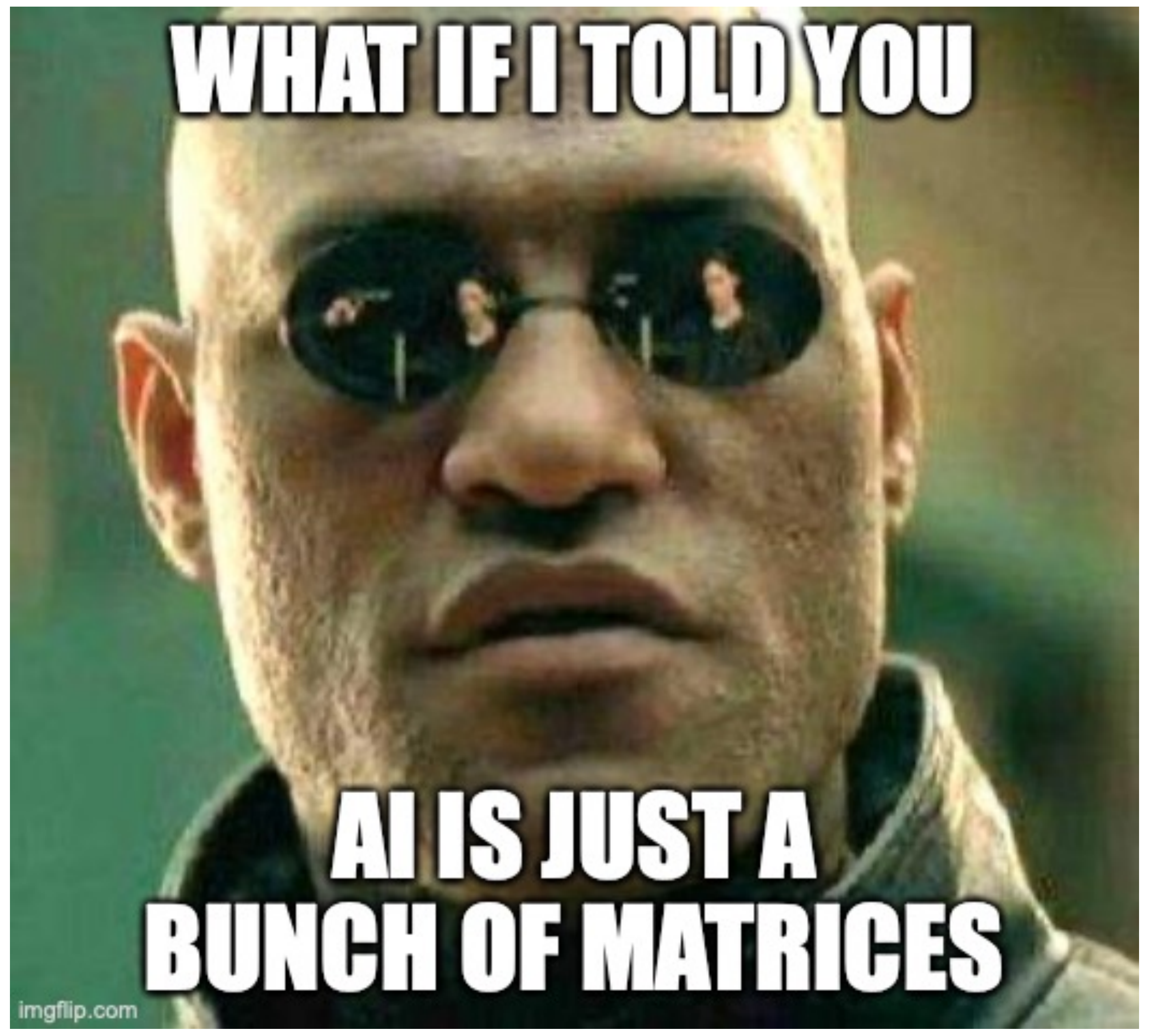
Data points are column vectors

- It is standard with tabular data to have the rows as data points
- But in ML literature it is convention to denote all vectors including data points $\mathbf{x}$ as **column vectors**
- It is also convention to represent a dataset as $\mathbf{X} \in \mathbb{R}^{N \times D}$ (in the same way we just did) where the **rows** are those data points
- **Just be aware of this peculiarity!**

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \qquad \mathbf{x}^{(n)} = \begin{bmatrix} x_1^{(n)} \\ x_2^{(n)} \\ \vdots \\ x_D^{(n)} \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)\top} \\ \mathbf{x}^{(2)\top} \\ \mathbf{x}^{(3)\top} \\ \vdots \\ \mathbf{x}^{(N)\top} \end{bmatrix} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_D^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_D^{(2)} \\ x_1^{(3)} & x_2^{(3)} & \dots & x_D^{(3)} \\ \dots & \dots & \ddots & \vdots \\ x_1^{(N)} & x_2^{(N)} & \dots & x_D^{(N)} \end{bmatrix}$$

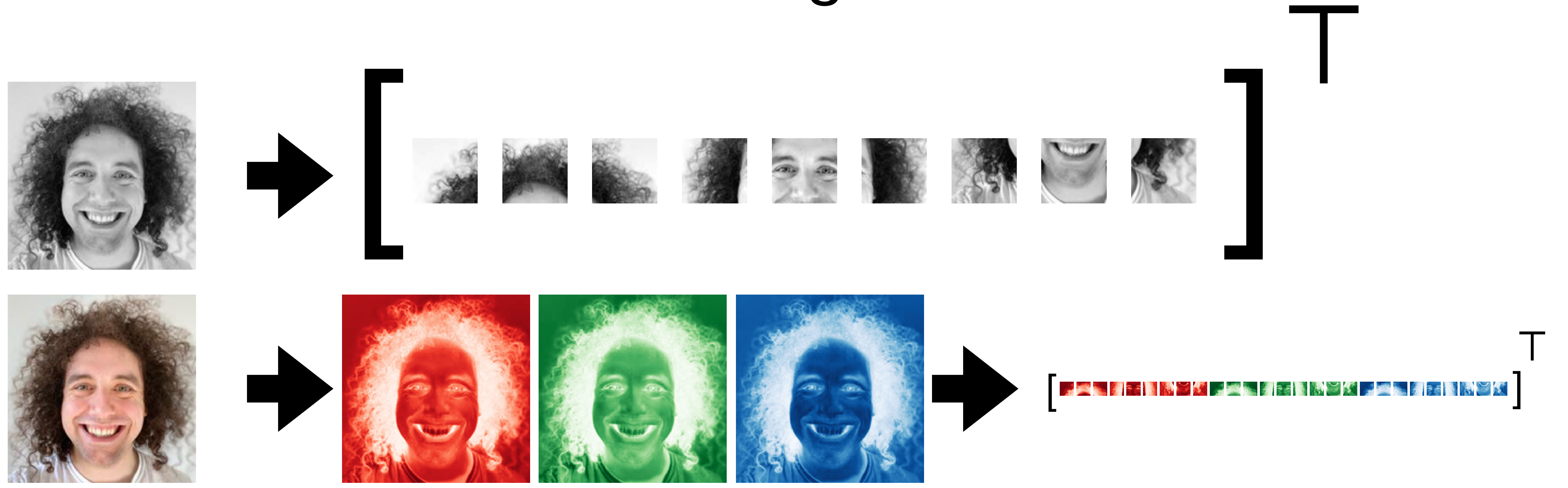
Why vectors?

- We can now use the machinery of linear algebra for PCA and ML
- **Matrices linearly transform vectors**
- Computers are very good at matrix multiplication
- Neural networks consist of multiple matrices (See Week 10!)



Can we represent other types of data as vectors?

- Yes! We can flatten or ***vectorise*** images

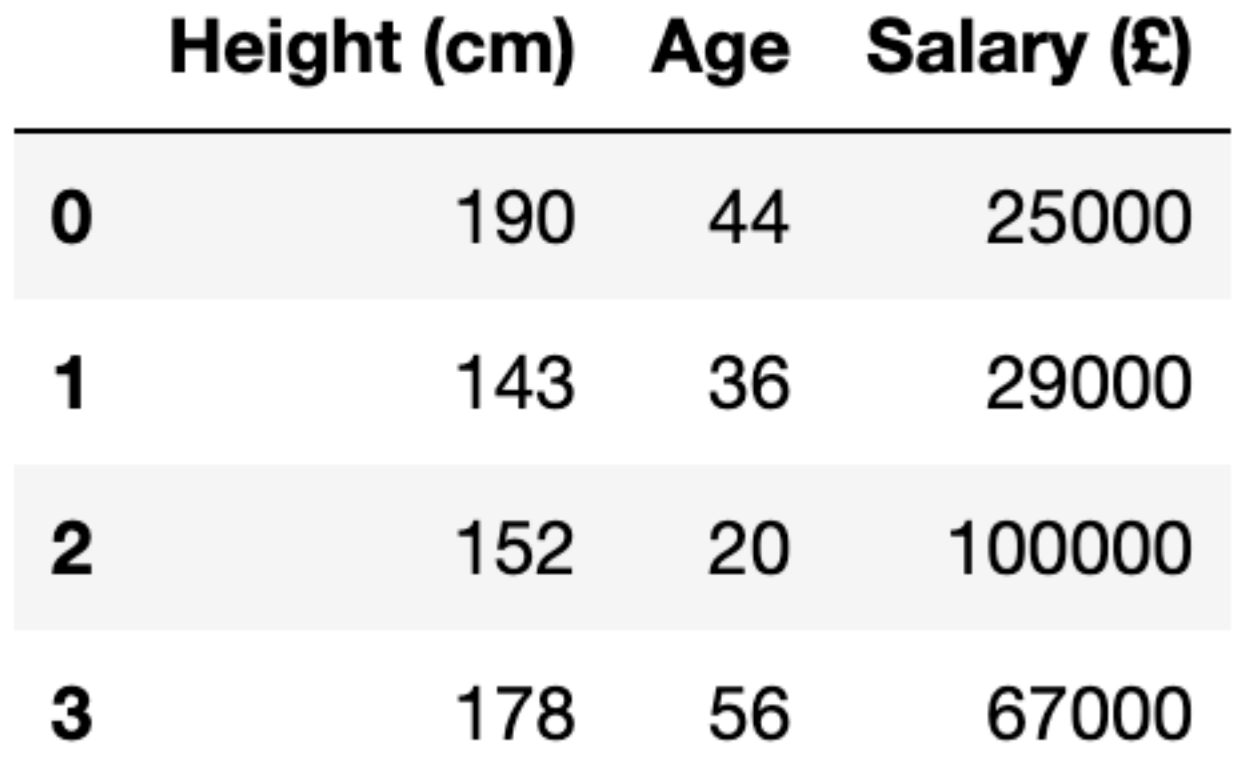


- We can represent text data as a histogram of word counts (a bag of words) e.g. [# “I”, # “like”, # “sausage”, # “hate”]

I like sausage	I hate sausage	sausage sausage
$[1 \quad 1 \quad 1 \quad 0]^{\top}$	$[1 \quad 0 \quad 1 \quad 1]^{\top}$	$[0 \quad 0 \quad 2 \quad 0]^{\top}$

Standardising your data

- Measurements of different features can have vastly different scales
- We want to compare features like-for-like and not let those with naturally large values dominate
- The solution is to **standardise** your data
- We want each column of $\mathbf{X}$ to have a mean of 0 and a SD of 1



	Height (cm)	Age	Salary (£)
0	190	44	25000
1	143	36	29000
2	152	20	100000
3	178	56	67000

$$\Rightarrow \quad \mathbf{X} = \begin{bmatrix} 190 & 44 & 25000 \\ 143 & 36 & 29000 \\ 152 & 20 & 100000 \\ 178 & 56 & 67000 \end{bmatrix} \quad \Rightarrow \quad ?$$

Standardising your data

- We want each column of $\mathbf{X}$ to have a mean of 0 and a SD of 1
- For each column, compute the mean and SD
- Then subtract the mean from each value and divide by SD
- **This is essential for PCA and many ML algorithms**

I will use $\sum_n$ to mean "sum over all n"

$$\mathbf{X}_{old} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_D^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_D^{(2)} \\ \dots & \dots & \ddots & \vdots \\ x_1^{(N)} & x_2^{(N)} & \dots & x_D^{(N)} \end{bmatrix}$$

$$\mu_j = \frac{1}{N} \sum_n \mathbf{x}_j^{(n)}$$

$$\sigma_j^2 = \frac{1}{N} \sum_n (\mathbf{x}_j^{(n)} - \mu_j)^2$$

$$\mathbf{X}_{new} = \begin{bmatrix} \frac{x_1^{(1)} - \mu_1}{\sigma_1} & \frac{x_2^{(1)} - \mu_2}{\sigma_2} & \dots & \frac{x_D^{(1)} - \mu_D}{\sigma_D} \\ \frac{x_1^{(2)} - \mu_1}{\sigma_1} & \frac{x_2^{(2)} - \mu_2}{\sigma_2} & \dots & \frac{x_D^{(2)} - \mu_D}{\sigma_D} \\ \dots & \dots & \ddots & \vdots \\ \frac{x_1^{(N)} - \mu_1}{\sigma_1} & \frac{x_2^{(N)} - \mu_2}{\sigma_2} & \dots & \frac{x_D^{(N)} - \mu_D}{\sigma_D} \end{bmatrix}$$

Normalising vs. standardising

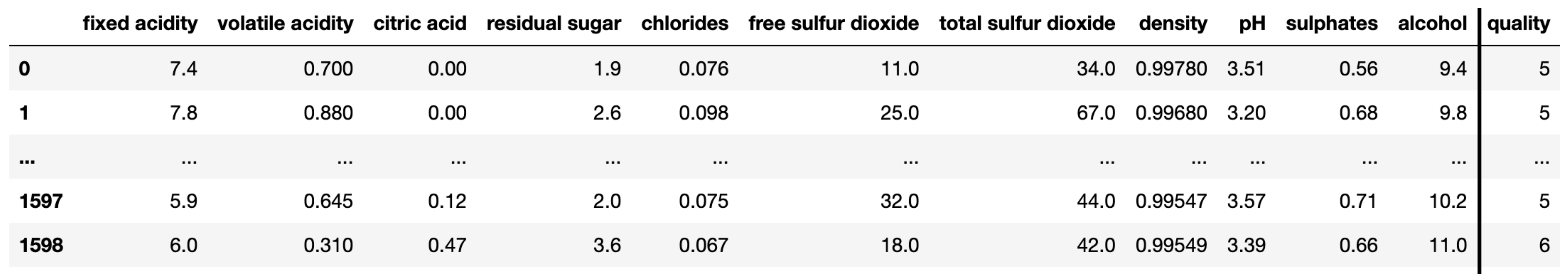
- Nomenclature can vary but in this course **standardising** refers to scaling each variable to zero mean and unit variance
- We can do other forms of scaling e.g. divide each variable by its maximum value
- We will refer to other forms of scaling as **normalising**
- Generally, anything that gets different variables to similar ranges is fine **just make sure you do it!**

If you have a bunch of binary variables you can just leave things alone!

Principal Component Analysis (PCA)

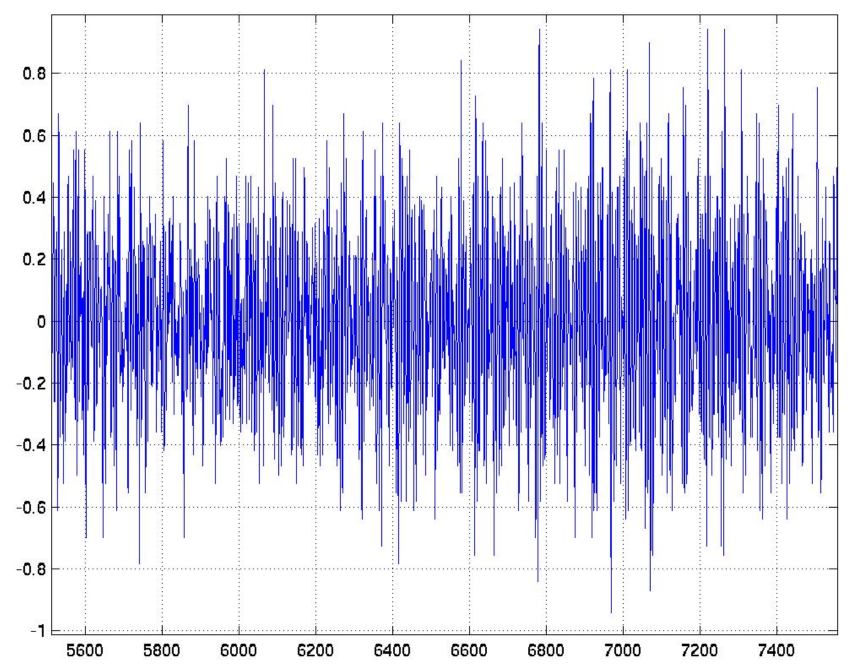
Motivation for PCA

- Most data is high dimensional
- This makes it hard to visualise patterns across a whole dataset



	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

Tables with >3 columns

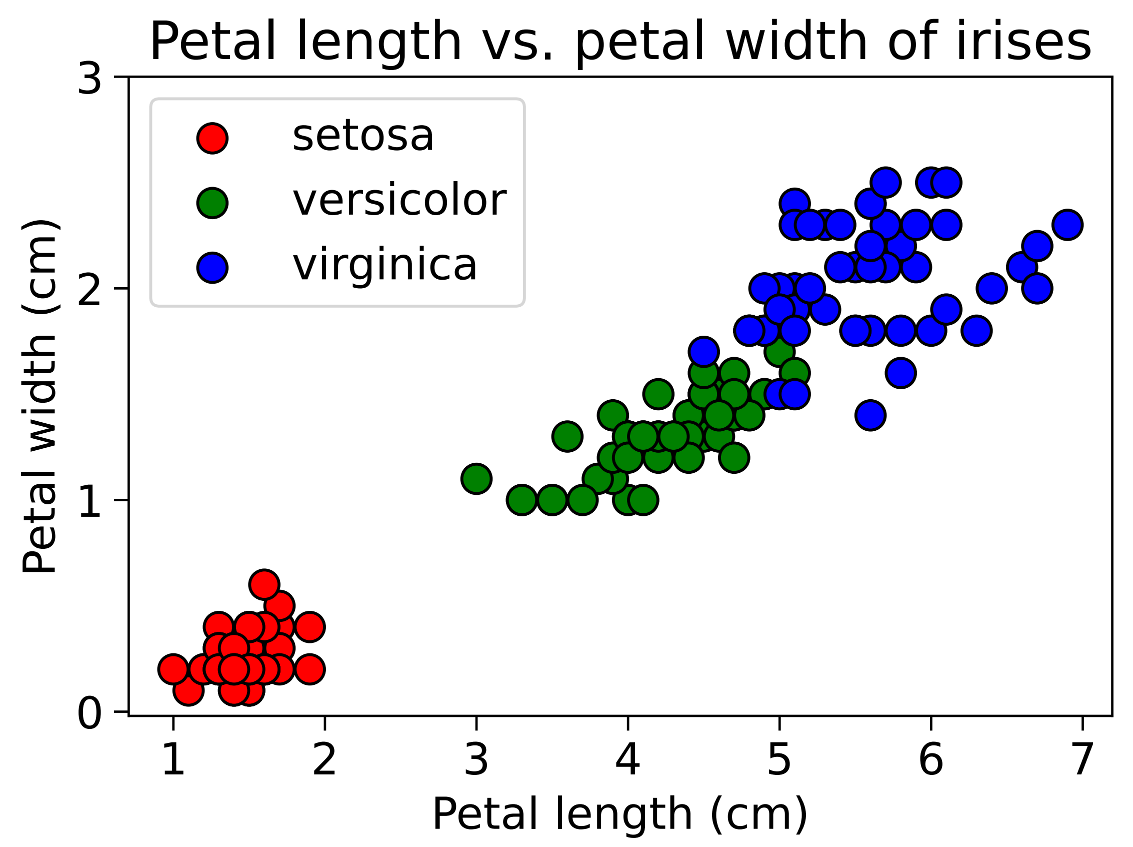


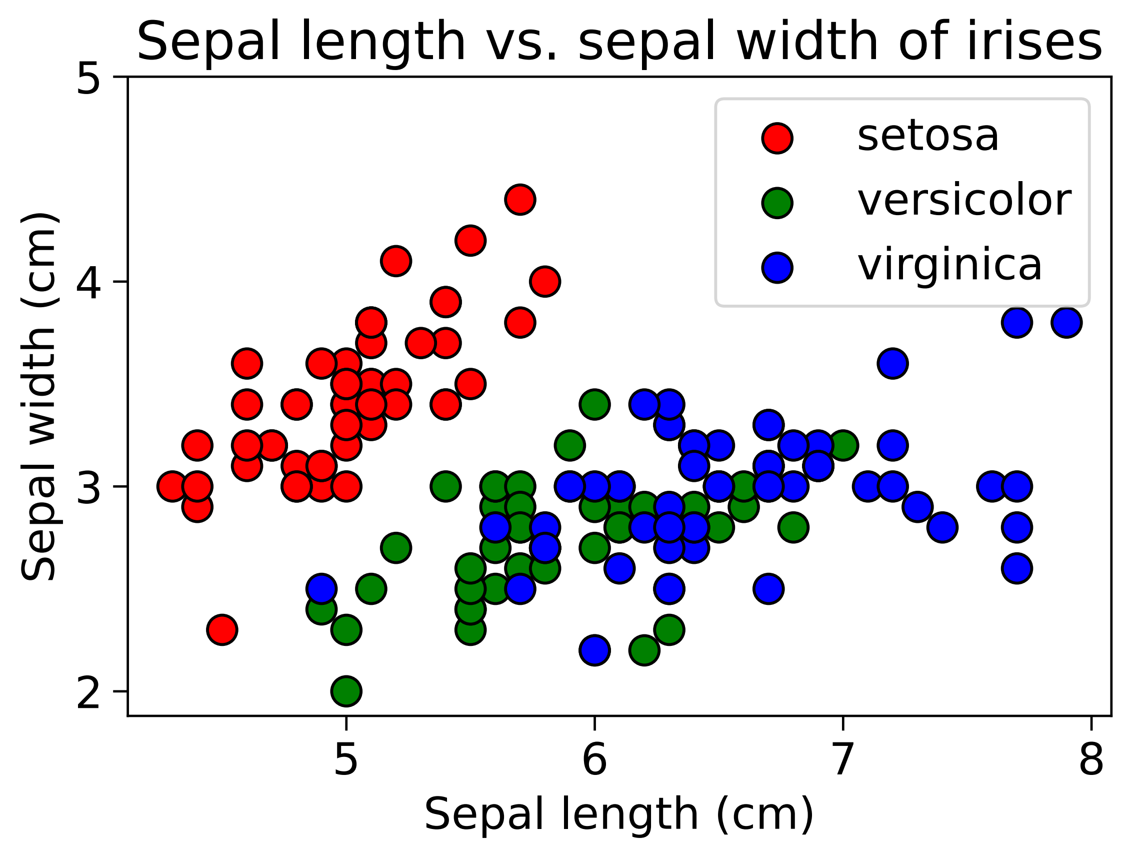
Time series with thousands of points

Images with millions of pixels

Dimensionality reduction

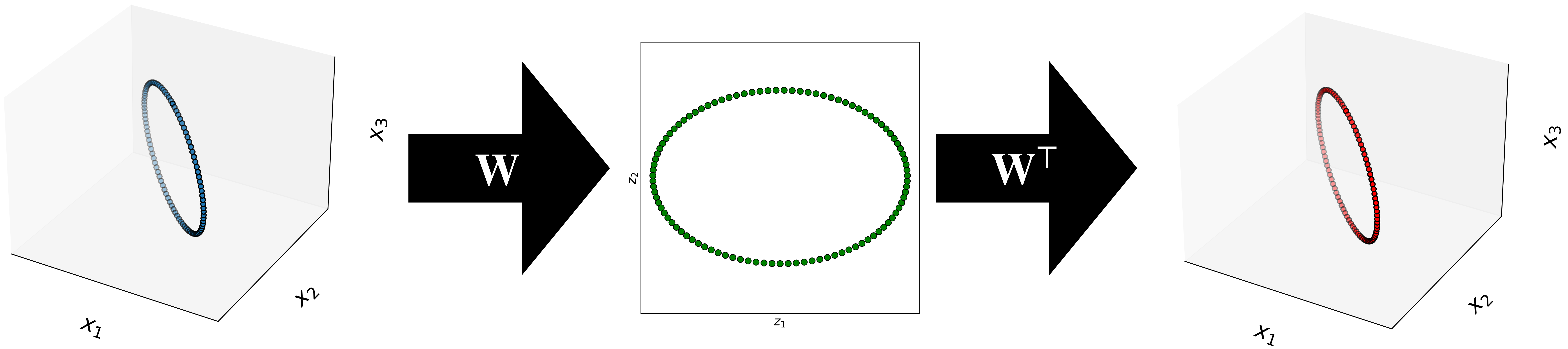
- We could use a linear transform to reduce the dimensionality of our data
- $\mathbf{Z} = \mathbf{X}\mathbf{W}$ with $\mathbf{W} \in \mathbb{R}^{D \times d}$ transforms $\{\mathbf{x}^{(n)}\}_{n=1}^{N}$ (the rows of $\mathbf{X}$) into $\{\mathbf{z}^{(n)}\}_{n=1}^{N}$ (the rows of $\mathbf{Z}$) where $\mathbf{x} \in \mathbb{R}^{D}$ and $\mathbf{z} \in \mathbb{R}^{d}$
- Then we could look at a scatter plot of $\{\mathbf{z}^{(n)}\}_{n=1}^{N}$ (if e.g. $d = 2$) to see patterns
- But how do we know what the best transform is?





Minimising reconstruction loss

- Treat the matrix $\mathbf{W} \in \mathbb{R}^{D \times d}$ as an encoder. We apply it to get to a low dimensional space $\mathbf{Z} = \mathbf{X}\mathbf{W}$ where $\mathbf{Z} \in \mathbb{R}^{N \times d}$
- We can then apply its transpose to *decode* $\hat{\mathbf{X}} = \mathbf{Z}\mathbf{W}^\top$ where $\hat{\mathbf{X}} \in \mathbb{R}^{N \times D}$
- The rows of $\hat{\mathbf{X}}$: $\{\hat{\mathbf{x}}^{(n)}\}_{n=1}^{N}$ are reconstructions of the data points$\{\mathbf{x}^{(n)}\}_{n=1}^{N}$
- We should minimise the distance between points and their reconstructions so that $\mathbf{Z}$ is a faithful low dimensional representation of the dataset



Minimising reconstruction loss

- We should minimise the (average square) distance between points and their reconstructions $\frac{1}{N}\sum_n \|\mathbf{x}^{(n)} - \hat{\mathbf{x}}^{(n)}\|^2 = \frac{1}{N}\sum_n \|\mathbf{x}^{(n)} - \mathbf{W}^\top \mathbf{W}\mathbf{x}^{(n)}\|^2$

- We also want the low dimensional features $z_1, z_2, \ldots$to be **uncorrelated** to minimise redundancy between features. This is achieved when $\mathbf{W}^\top \mathbf{W} = \mathbf{I}$

- Overall we want to solve $\underset{\mathbf{W}}{\text{minimise}} \ \frac{1}{N}\sum_n \|\mathbf{x}^{(n)} - \mathbf{W}^\top \mathbf{W}\mathbf{x}^{(n)}\|^2$ s.t. $\mathbf{W}^\top \mathbf{W} = \mathbf{I}$

- PCA gives us the solution to this

$\mathbf{Z}\mathbf{W}^\top = \mathbf{X}\mathbf{W}\mathbf{W}^\top$ for the dataset matrix means $\mathbf{W}^\top \mathbf{z} = \mathbf{W}\mathbf{W}^\top \mathbf{x}$ for each column vector data point

Principal Component Analysis (PCA)

- For a **<u>standardised</u>** dataset $\mathbf{X} \in \mathbb{R}^{N \times D}$, PCA returns a matrix $\mathbf{W}_{PC} \in \mathbb{R}^{D \times D}$
- The columns of $\mathbf{W}_{PC}$: $\{\mathbf{w}_d\}_{d=1}^{D}$ are the **principal components** of the data
- The matrix that solves $\underset{\mathbf{W}}{\mathrm{minimise}} \; \frac{1}{N} \sum_n \|\mathbf{x}^{(n)} - \mathbf{W}^\top \mathbf{W} \mathbf{x}^{(n)}\|^2$ s.t. $\mathbf{W}^\top \mathbf{W} = \mathbf{I}$ for $\mathbf{W} \in \mathbb{R}^{D \times d}$ is $\mathbf{W} = [\mathbf{w}_1 \quad \mathbf{w}_2 \quad \dots \quad \mathbf{w}_d]$
- i.e. it's the matrix whose columns are the first d principal components

See Murphy 20.1.2 for the proof

$\mathbf{XWW}^\top$ for the dataset matrix translates to $\mathbf{WW}^\top\mathbf{x}$ for each column vector data point

Computing principal components

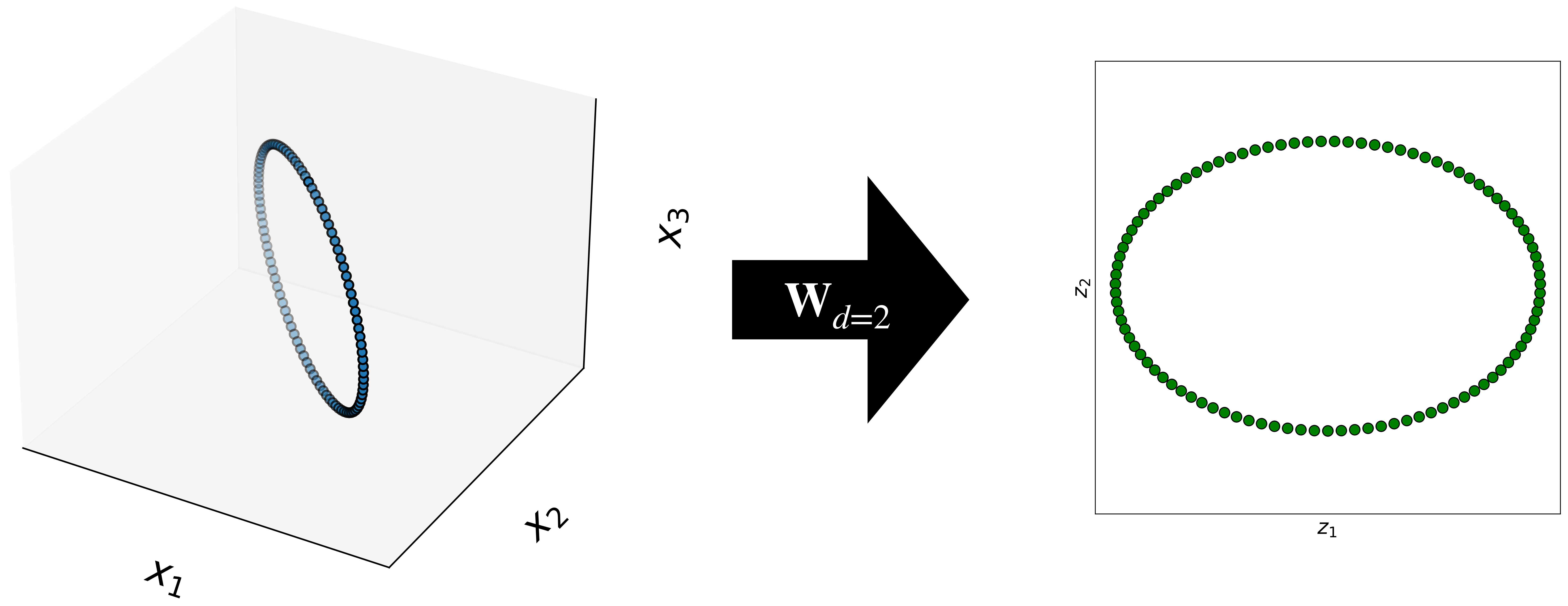
To compute principal components for a **standardised** dataset $\mathbf{X} \in \mathbb{R}^{N \times D}$:

1. Construct the covariance matrix $\mathbf{\Sigma} = \frac{1}{N}\mathbf{X}^\top \mathbf{X}$
2. Eigendecompose $\mathbf{\Sigma}$ to get eigenvalue, eigenvector pairs
3. Sort pairs by decreasing eigenvalue and denote as $\{\lambda_d\}_{d=1}^{D}, \{\mathbf{w}_d\}_{d=1}^{D}$

These vectors are
the principal components

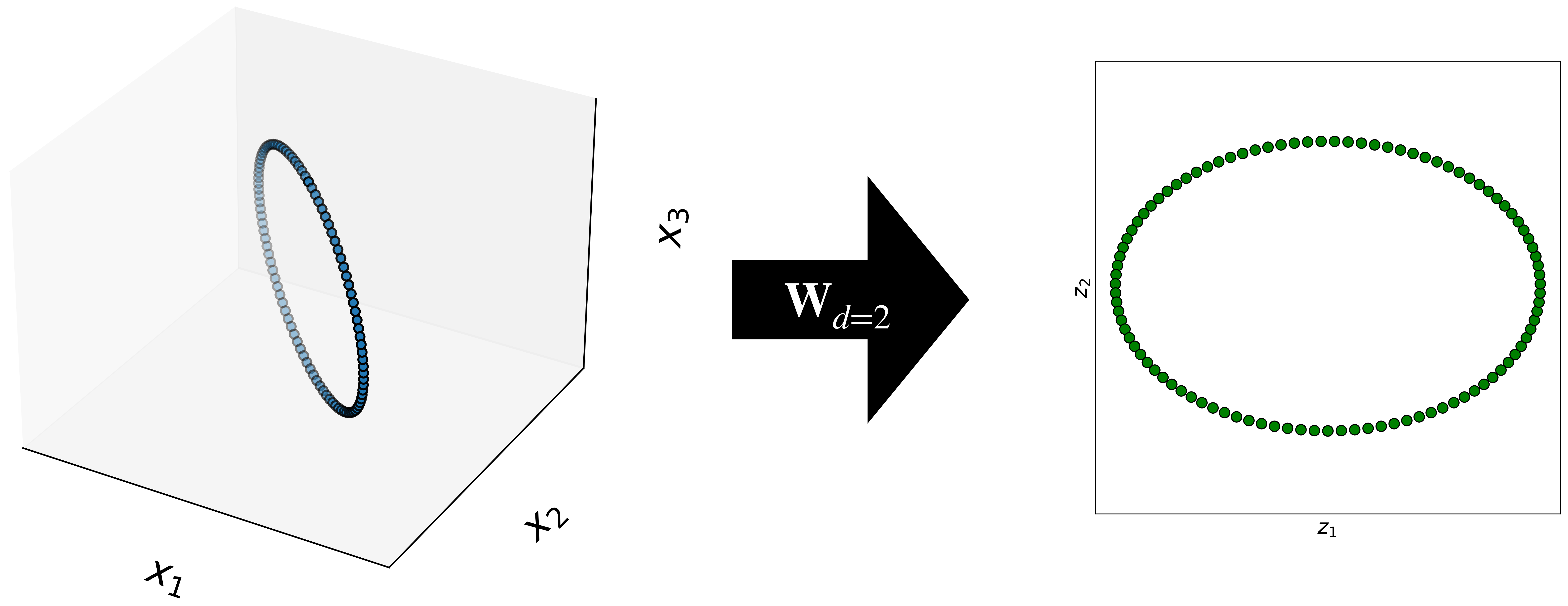
PCA for dimensionality reduction

- PCA gives us $\mathbf{W} \in \mathbb{R}^{D \times D}$ where $\mathbf{W} = [\mathbf{w}_1 \quad \mathbf{w}_2 \quad \cdots \quad \mathbf{w}_D]$
- To reduce to $d < D$ dimensions we can just keep the first d columns
- e.g. $\mathbf{W}_{d=2} = [\mathbf{w}_1 \quad \mathbf{w}_2]$ would take our data to 2D using $\mathbf{Z} = \mathbf{X}\mathbf{W}_{d=2}$



Minimising reconstruction error maximises variance

- PCA gives us the (linear) **direction of maximum variance** in z_1
- It gives us the (orthogonal) next largest direction of maximum variance in z_2
- And so on. This is neat, but to me, less intuitive than reconstruction error



PCA for dimensionality reduction on irises

- The iris dataset contains 150 data points
- Let’s take the numeric columns to form a dataset matrix $\mathbf{X} \in \mathbb{R}^{150 \times 4}$
- Make sure that $\mathbf{X}$ is standardised

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

$$\begin{bmatrix} 5.1 & 3.5 & 1.4 & 0.2 \\ 4.9 & 3.0 & 1.4 & 0.2 \\ 4.7 & 3.2 & 1.3 & 0.2 \\ 4.6 & 3.1 & 1.5 & 0.2 \\ 5.0 & 3.6 & 1.4 & 0.2 \\ \cdots & \cdots & \cdots & \cdots \\ 0.7 & 3.0 & 5.2 & 2.3 \\ 6.3 & 2.5 & 5.0 & 1.9 \\ 6.5 & 3.0 & 5.2 & 2.0 \\ 6.2 & 3.4 & 5.4 & 2.3 \\ 5.9 & 3.0 & 5.1 & 1.8 \end{bmatrix}$$

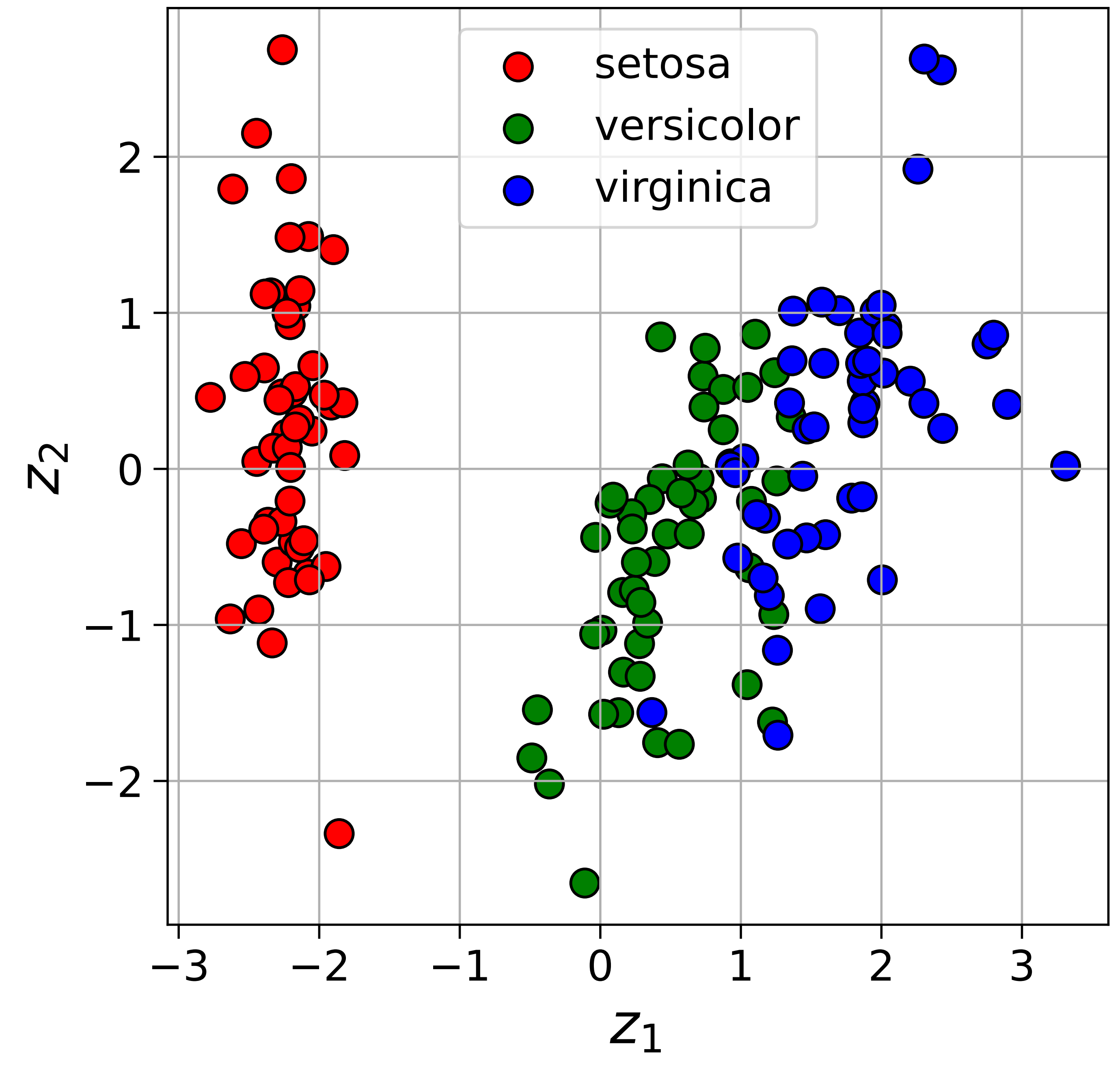
$$\begin{bmatrix} 5.1 & 3.5 & 1.4 & 0.2 \\ 0.9 & 1.0 & -1.3 & -1.3 \\ -1.1 & -0.1 & -1.3 & -1.3 \\ -1.4 & 0.3 & -1.4 & -1.3 \\ -1.5 & 0.1 & -1.3 & -1.3 \\ -1.0 & 1.2 & -1.3 & -1.3 \\ \cdots & \cdots & \cdots & \cdots \\ .0 & -0.1 & 0.8 & 1.4 \\ 0.6 & -1.3 & 0.7 & 0.9 \\ 0.8 & -0.1 & 0.8 & 1.1 \\ 0.4 & 0.8 & 0.9 & 1.4 \\ 0.1 & -0.1 & 0.8 & 0.8 \end{bmatrix}$$

PCA for dimensionality reduction on irises

- Use PCA to form $\mathbf{W}_{PC} \in \mathbb{R}^{4\times4}$
- Now use $\mathbf{Z} = \mathbf{X}\,[\mathbf{w}_1 \quad \mathbf{w}_2]$ to project down to 2D
- Different species are distinguishable just by looking at z_1
- These new dimensions were found automatically

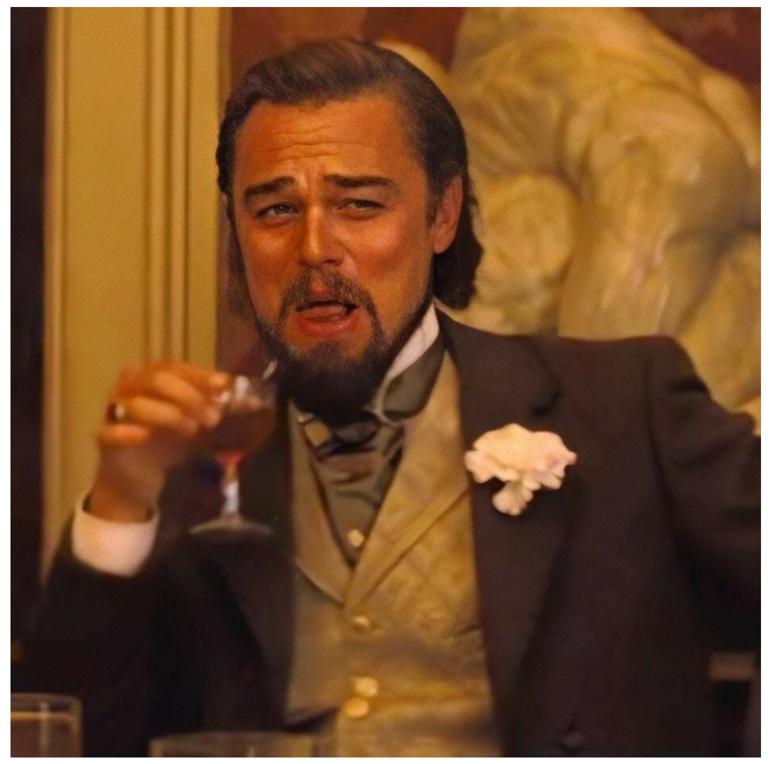
$$z_1 = -0.52x_1 - 0.27x_2 - 0.58x_3 + 0.56x_4$$

$$z_2 = -0.38x_1 + 0.92x_2 + 0.02x_3 + 0.07x_4$$



PCA for dimensionality reduction on wine

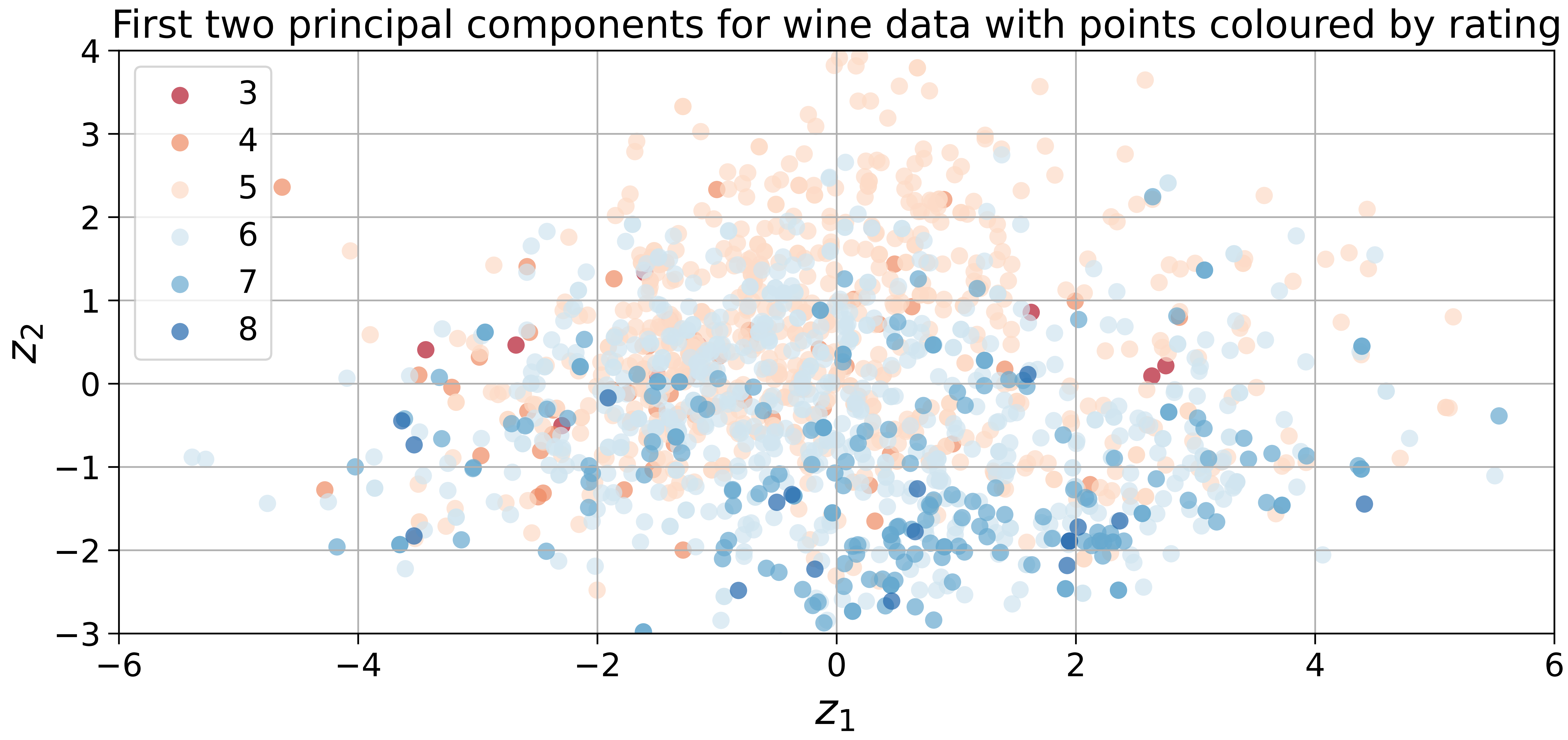
- We have a red wine dataset $\mathbf{X} \in \mathbb{R}^{1599 \times 11}$
- Each wine has also been scored by an expert between 0 and 10
- We can look at a few examples but it's hard to get the full picture



	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

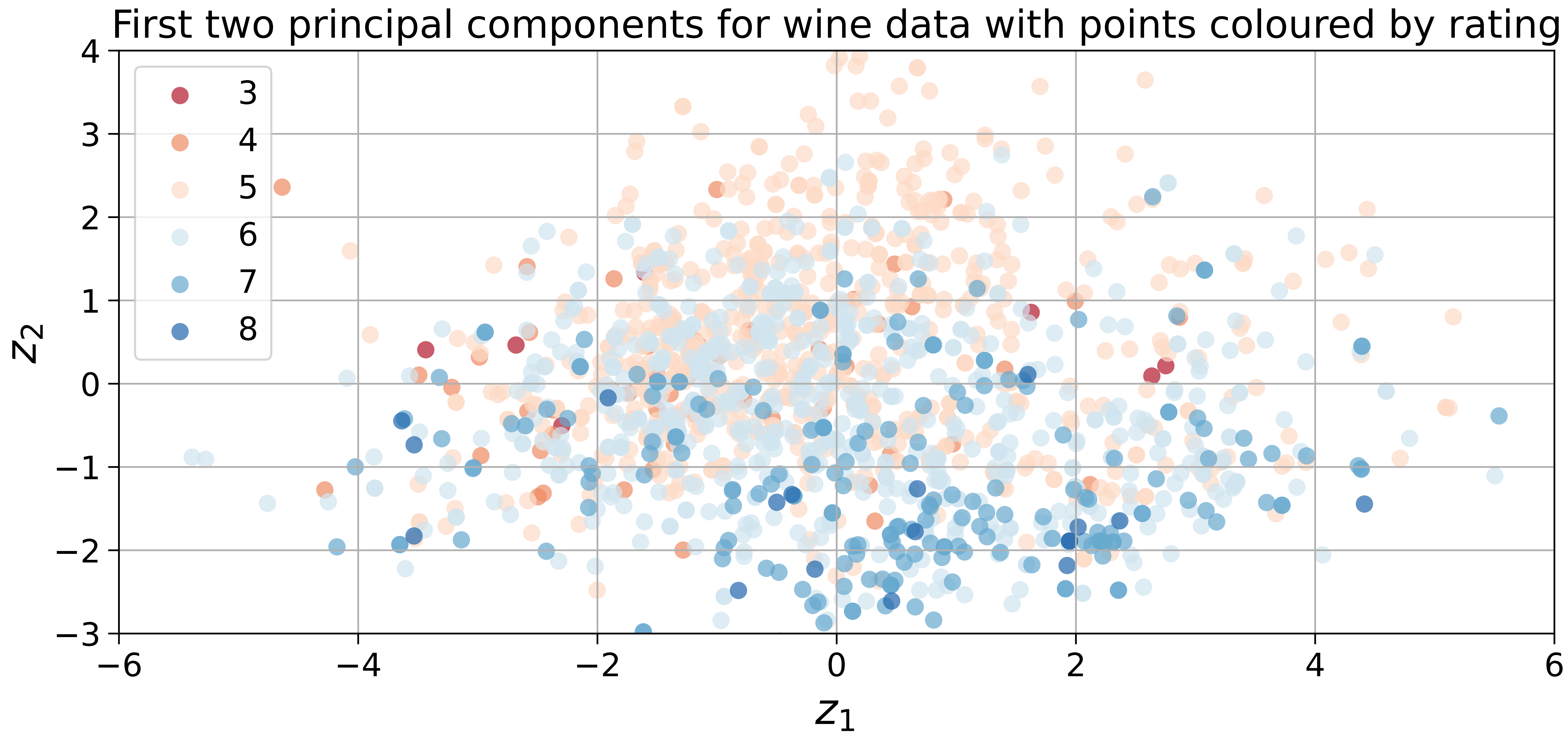
PCA for dimensionality reduction on wine

- Let's standardise our data, and then use PCA to form $\mathbf{W}_{PC} \in \mathbb{R}^{11\times 11}$
- Now use $\mathbf{Z} = \mathbf{X}\,[\mathbf{w}_1 \quad \mathbf{w}_2]$ to project down to 2D



PCA for dimensionality reduction on wine

- We can see in this space that good wines tend to be near the bottom
- What makes a good wine? A negative z_2 of course!



Good wine recipe: make z_2 negative

- The new dimensions are just linear combinations of the original dimensions

$$z_2 = -0.11x_1 + 0.27x_2 - 0.15x_3 + 0.27x_4 + 0.15x_5 + 0.51x_6 + 0.57x_7 + 0.23x_8 + 0.01x_9 - 0.04x_{10} - 0.39x_{11}$$

- In a lot of cases the new dimensions aren’t very intuitive
- PCA is best used for exploratory data analysis

Importance of components

- Performing PCA gives us eigenvalue, eigenvector pairs $\{\lambda_d\}_{d=1}^{D}, \{\mathbf{w}_d\}_{d=1}^{D}$
- The eigenvectors are our principal components
- The eigenvalues are an importance weighting for each component

The first principal component explains $\frac{\lambda_1}{\sum_{d=1}^{D} \lambda_d}$ of the variance of the data

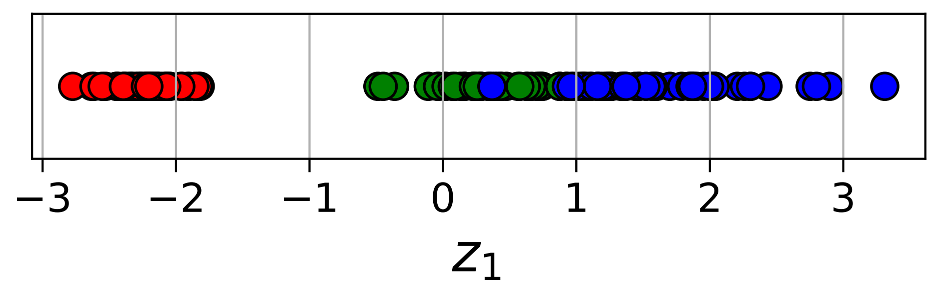
Importance of components

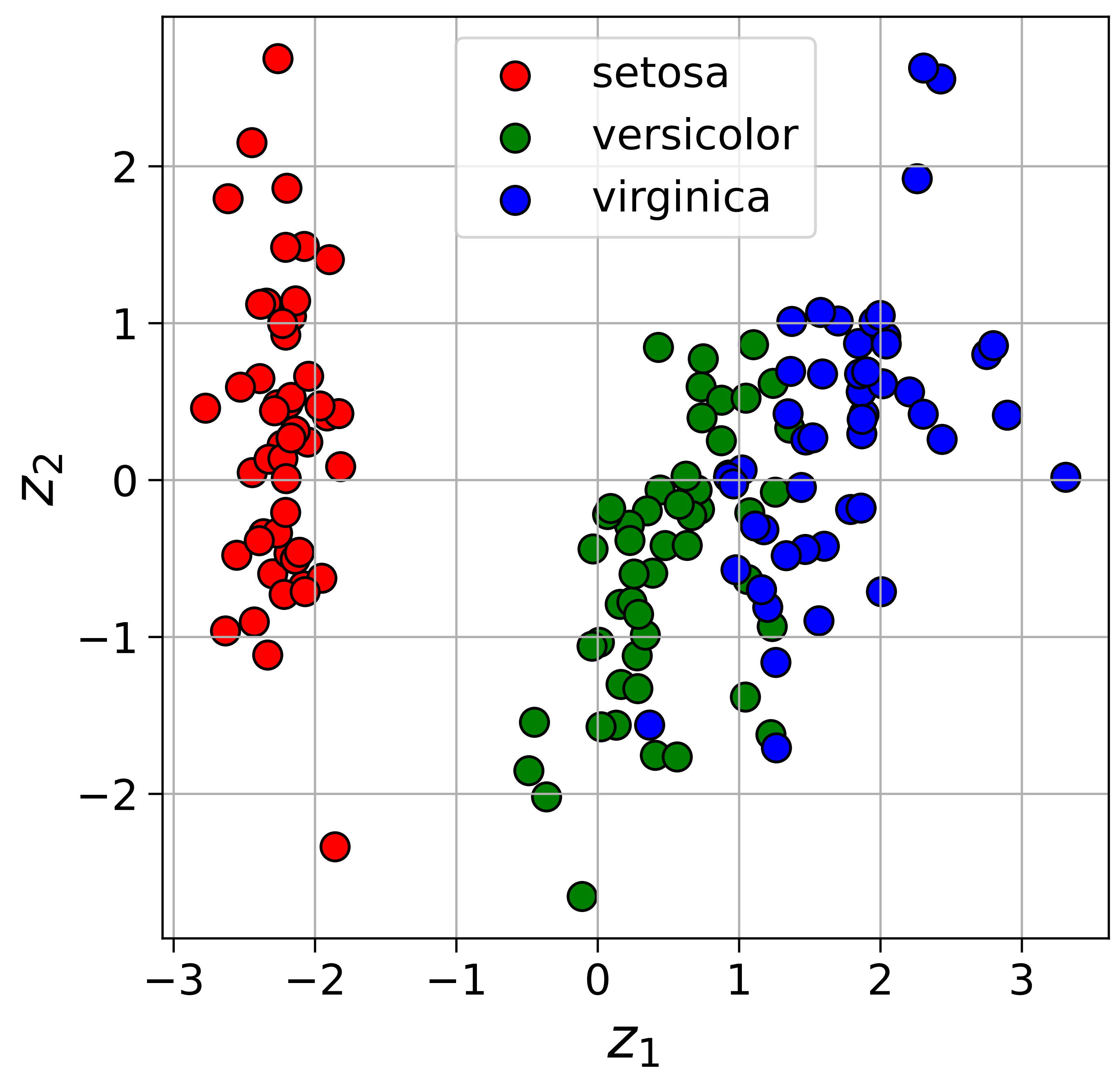
The first principal component explains $\frac{\lambda_1}{\sum_{d=1}^{D} \lambda_d}$ of the variance

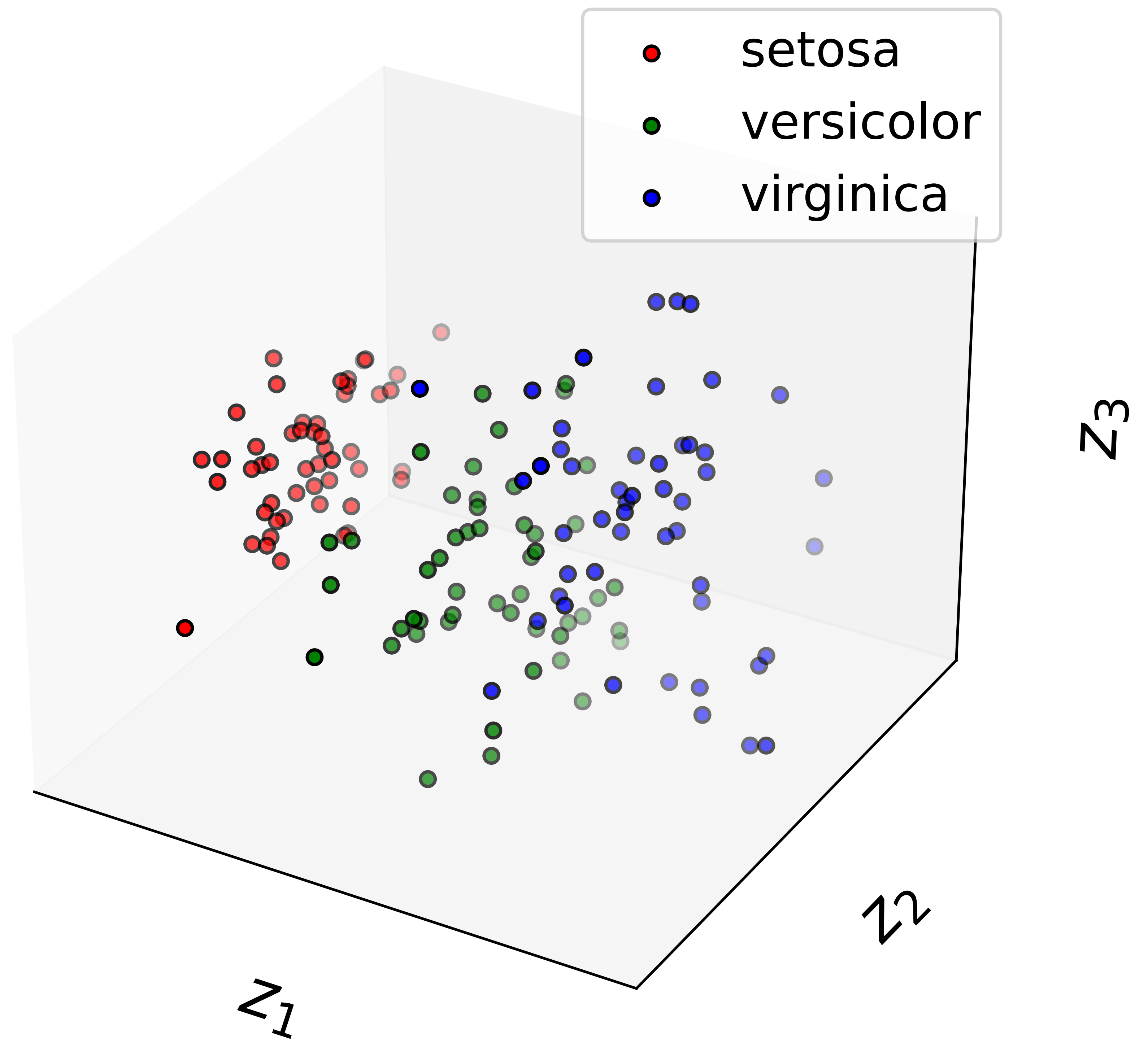
It follows that the first M principal components account for $\frac{\sum_{m=1}^{M} \lambda_m}{\sum_{d=1}^{D} \lambda_d}$

Be careful throwing away dimensions if not enough variance is explained

Explaining variance of irises

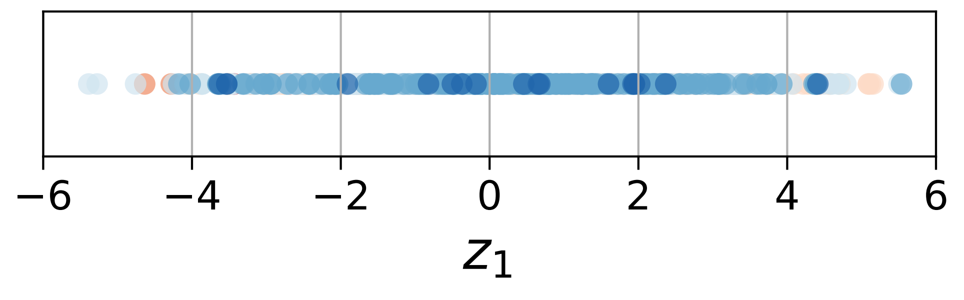


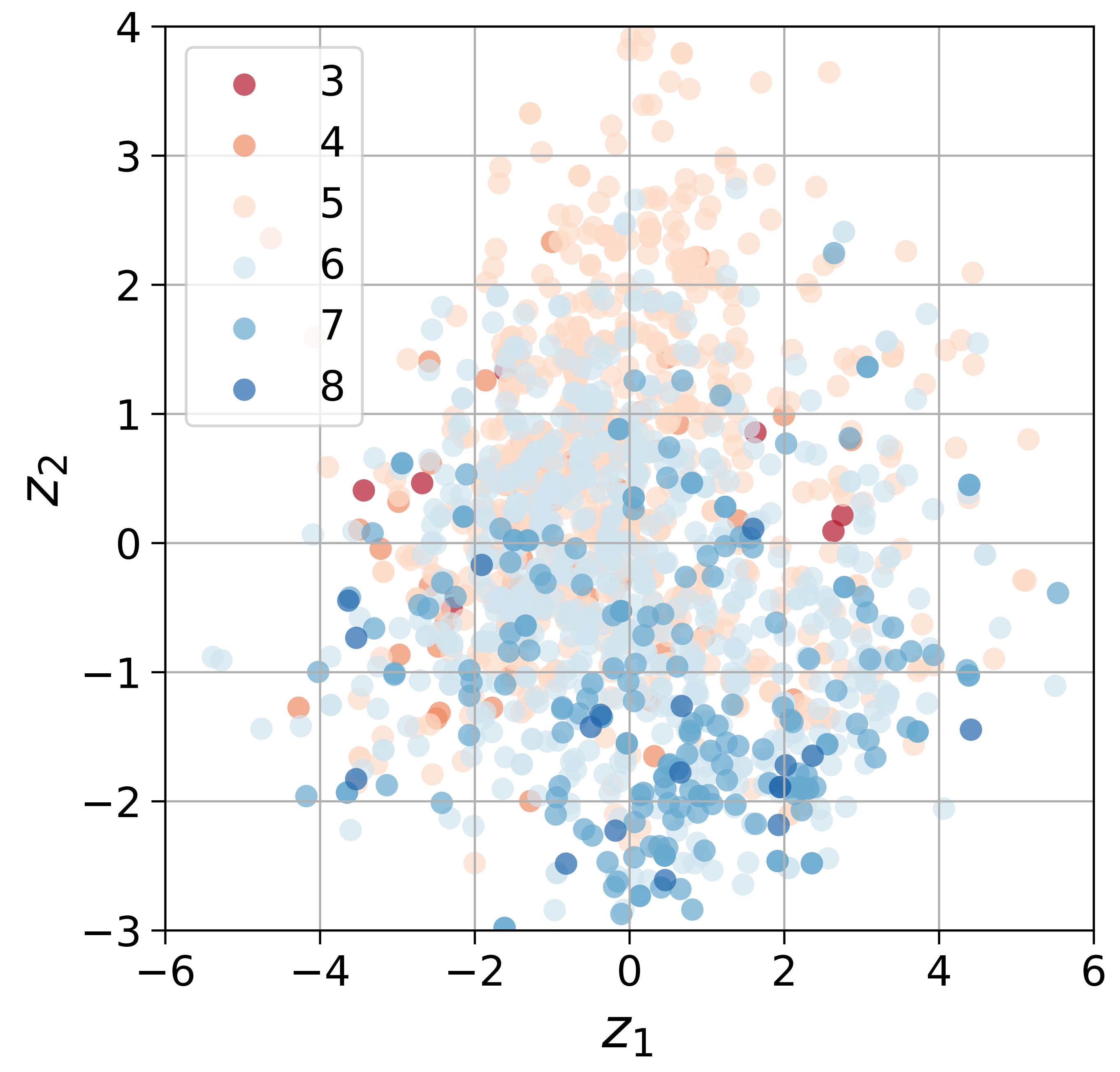


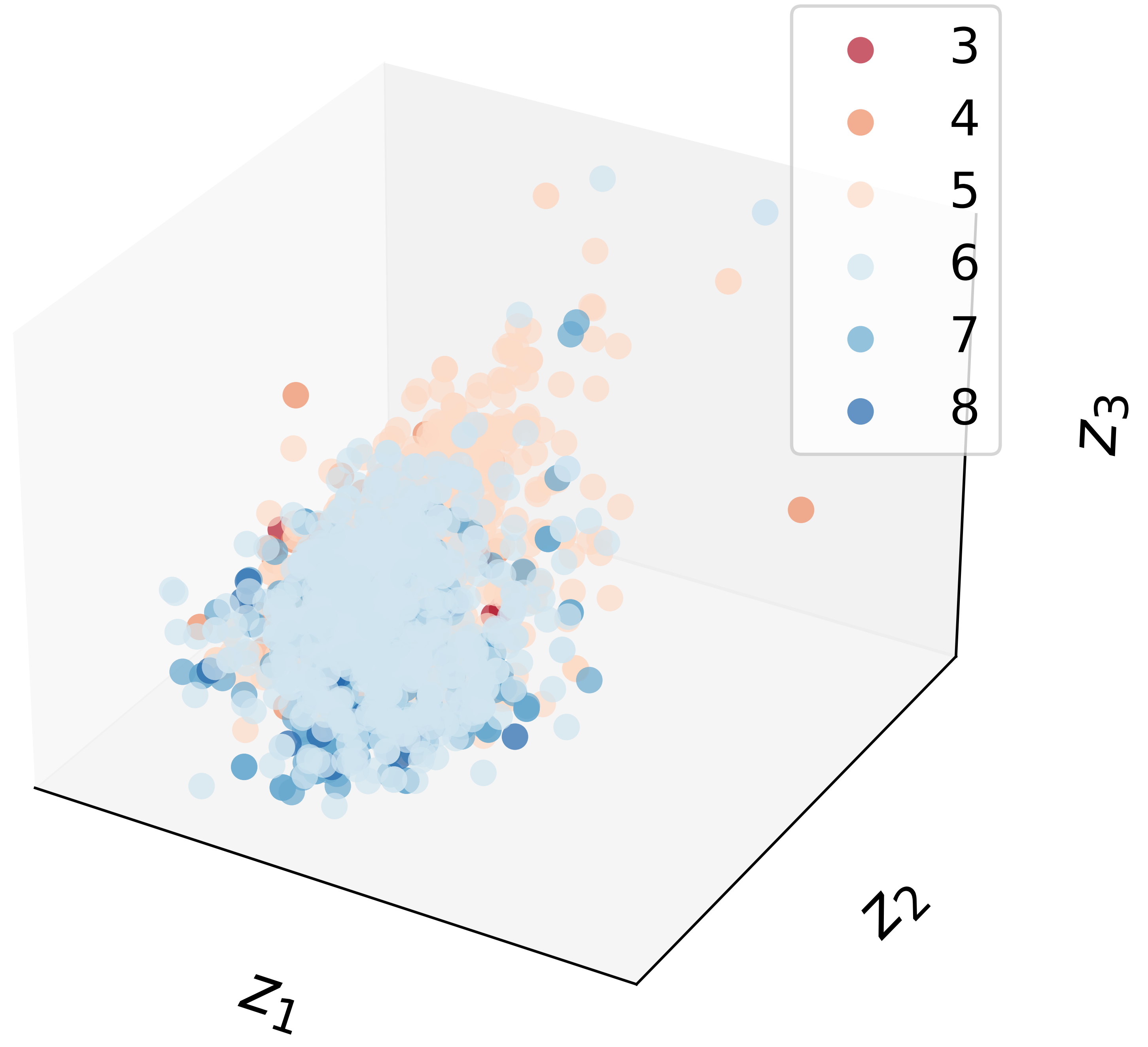


1D: 73% 2D: 96% 3D: 99%

Explaining variance of wine







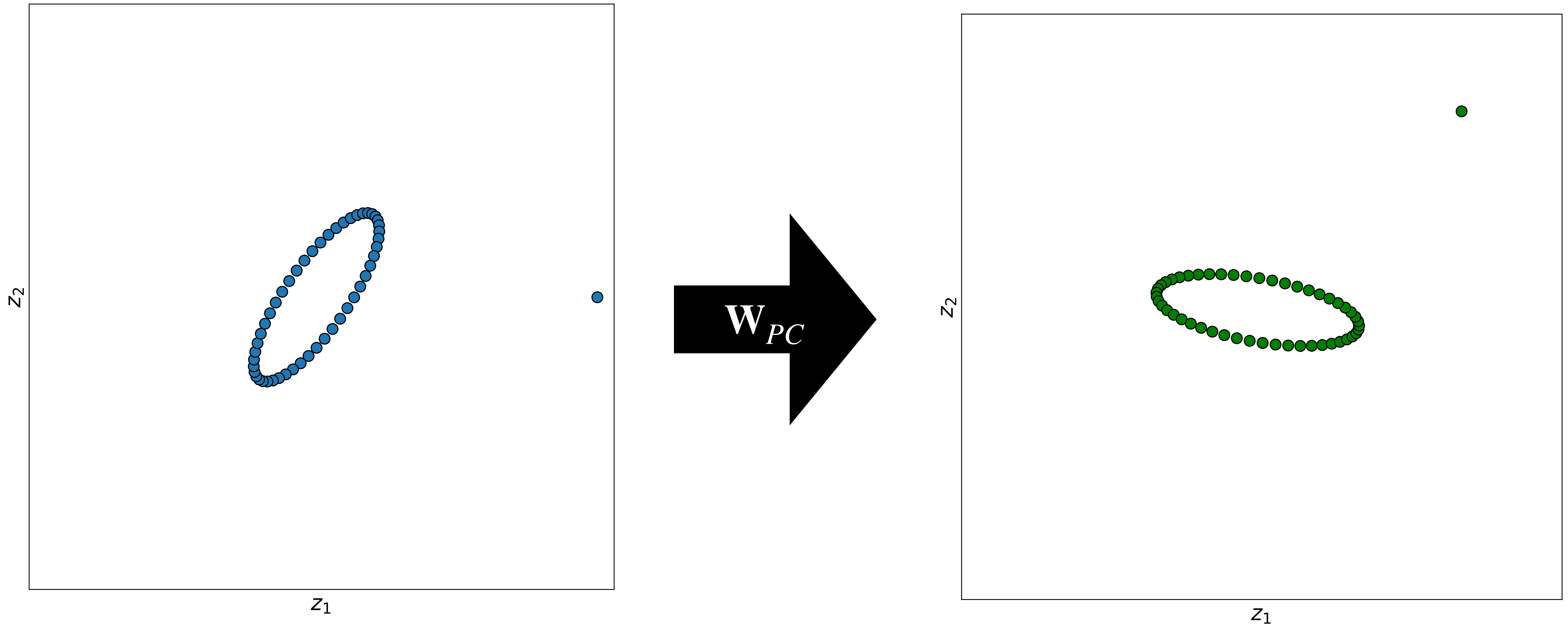
1D: 28%

2D: 45%

3D: 60%

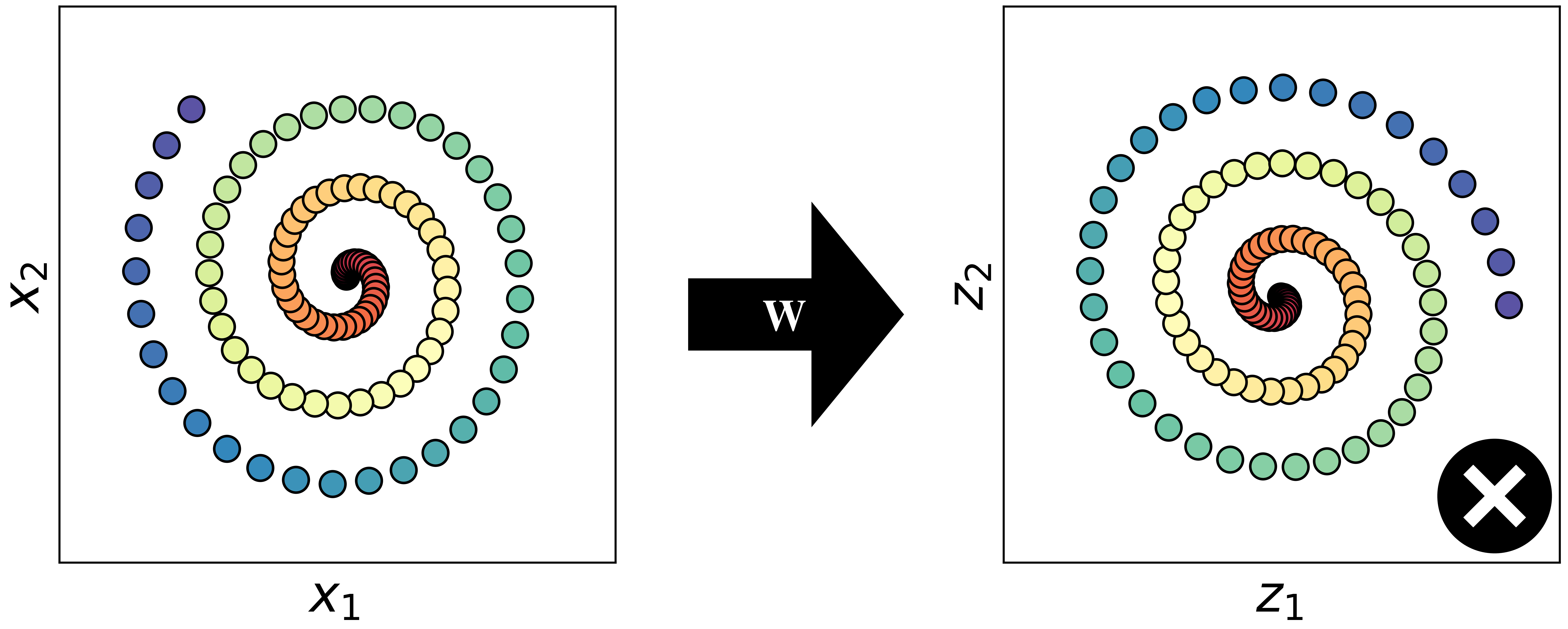
Limitations: PCA is susceptible to outliers

Outliers can change the direction of maximum variance



Limitations: PCA is linear

If the true direction of maximum variance isn't a line, PCA can't find it



Clustering with K-means

Motivation

You have a dataset that you want to split into groups

- people with low, medium, high income for marketing
- grouping shoppers to recommend products
- identifying personality types for a dating website

PUBLIX COUPON EXPIRES 5/31/10

Redeem at Publix.

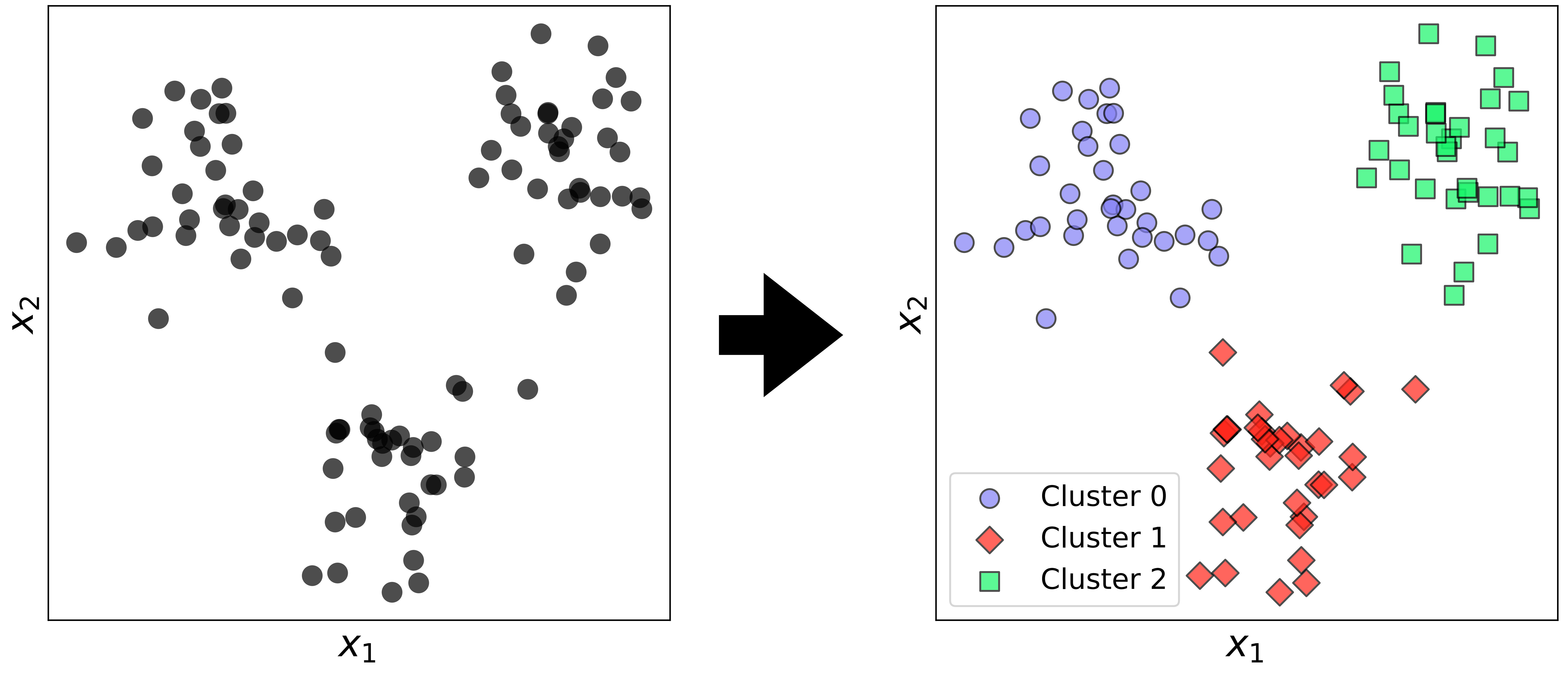
L.U. # 5793

SAVE $1.50

on any Publix® Pasta with the purchase of any TWO (2) Ragú® Pasta Sauce products

K-means

- We can use K-means to automatically split our dataset in groups
- Other clustering algorithms are available!



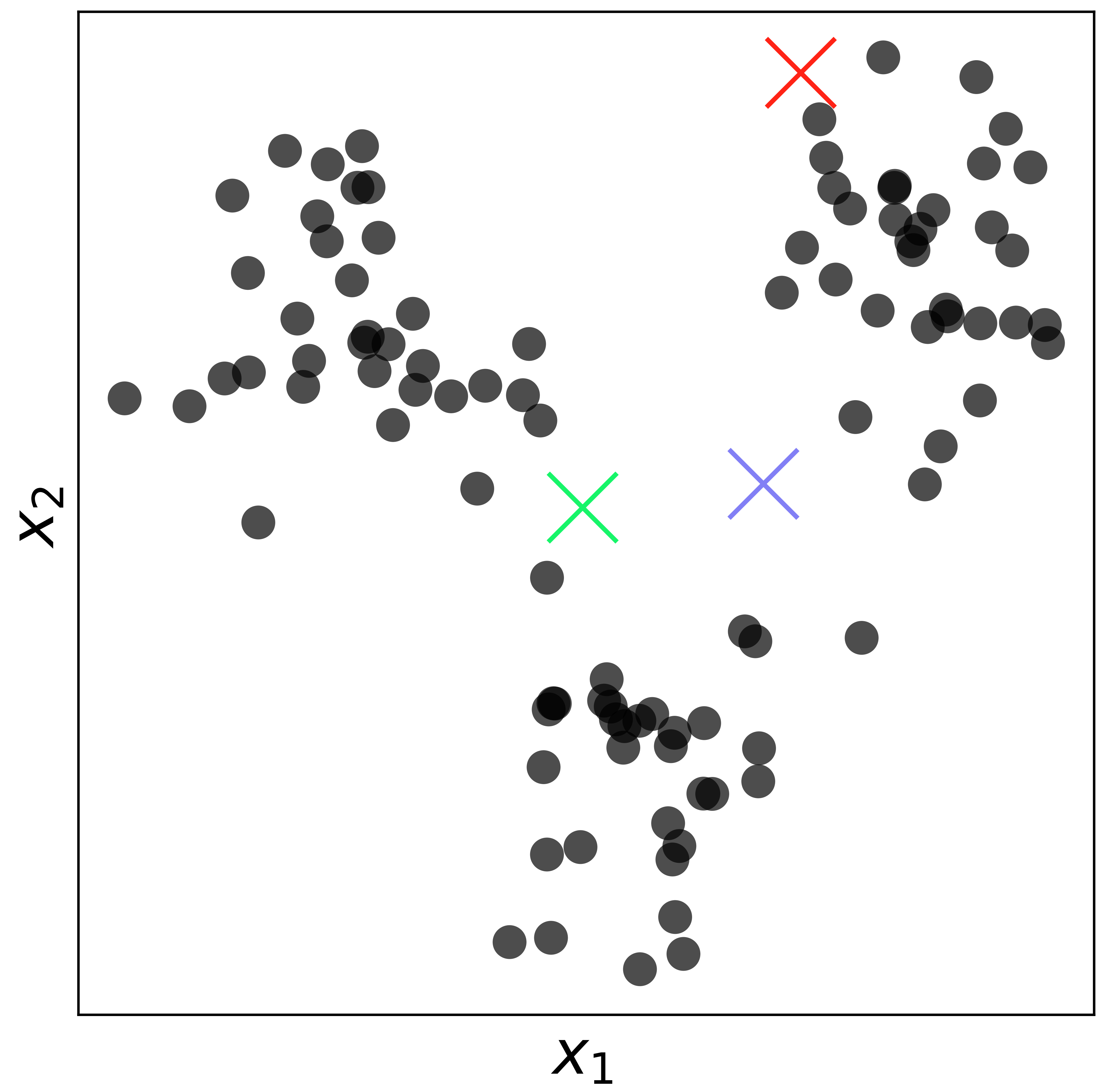
K-means algorithm

- Select the number of clusters K
- Initialise the cluster centres $\{\mathbf{c}_k\}_{k=1}^{K}$ at random
- Repeat:
    1. Assign each (ideally standardised) data point to its nearest cluster centre
    2. Update cluster centres as mean of their assigned points
- Until no change

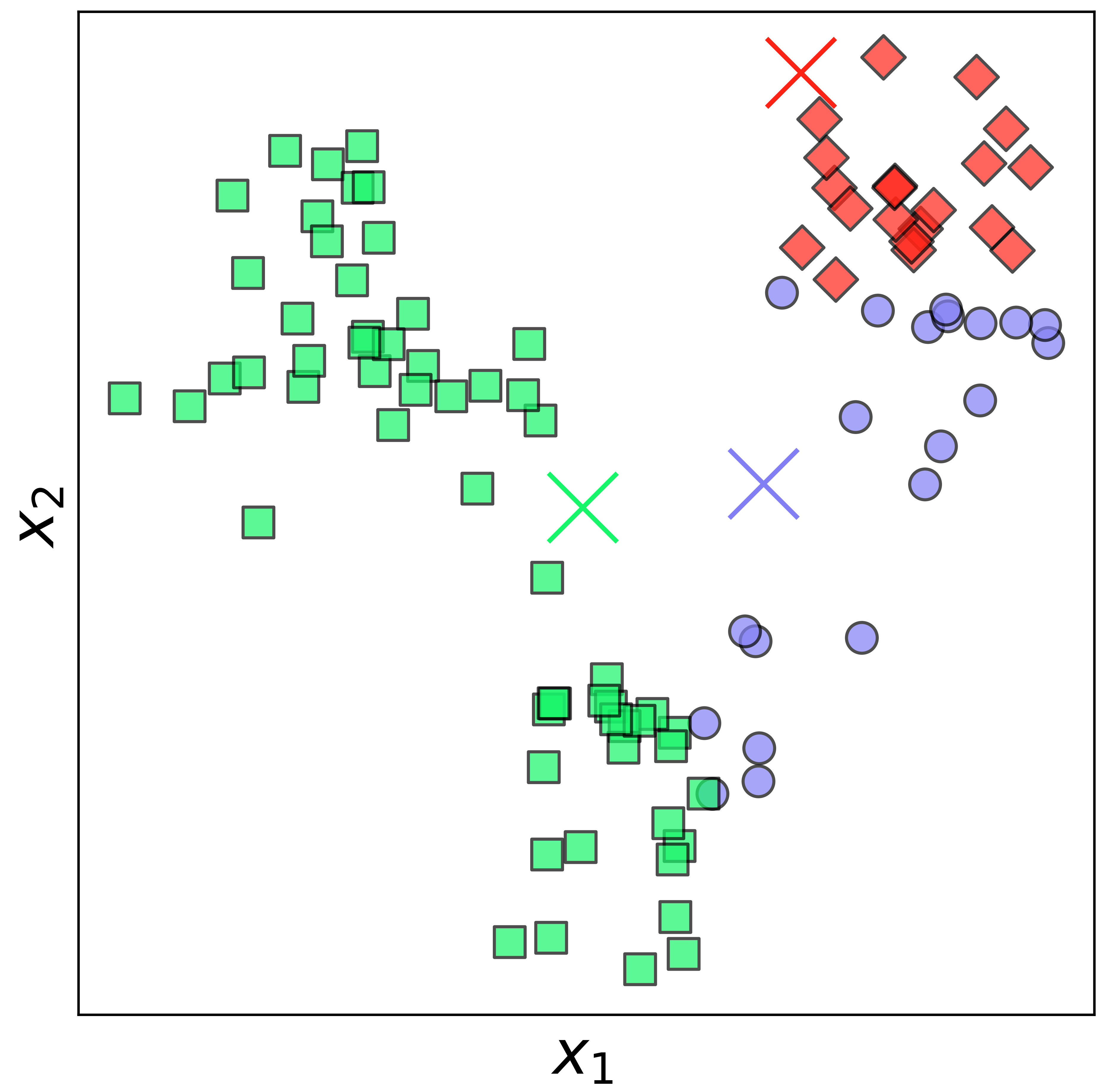
Credit: Andrew Zisserman (for the slide idea, not the algorithm)

K-means walkthrough with $K = 3$

Initialise the cluster centres $\{\mathbf{c}_k\}_{k=1}^{K}$ at random

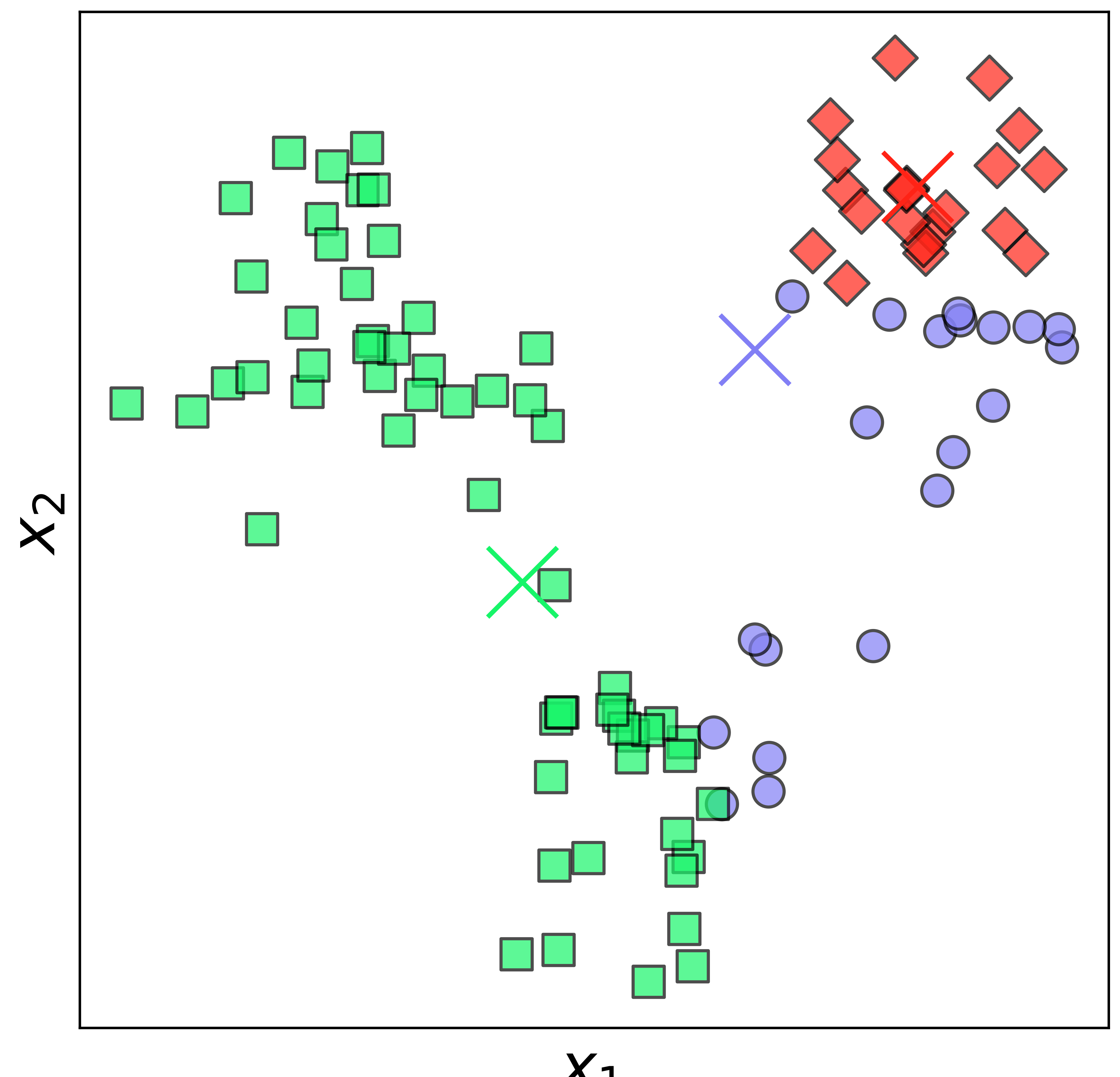


Assign each data point to its nearest cluster centre

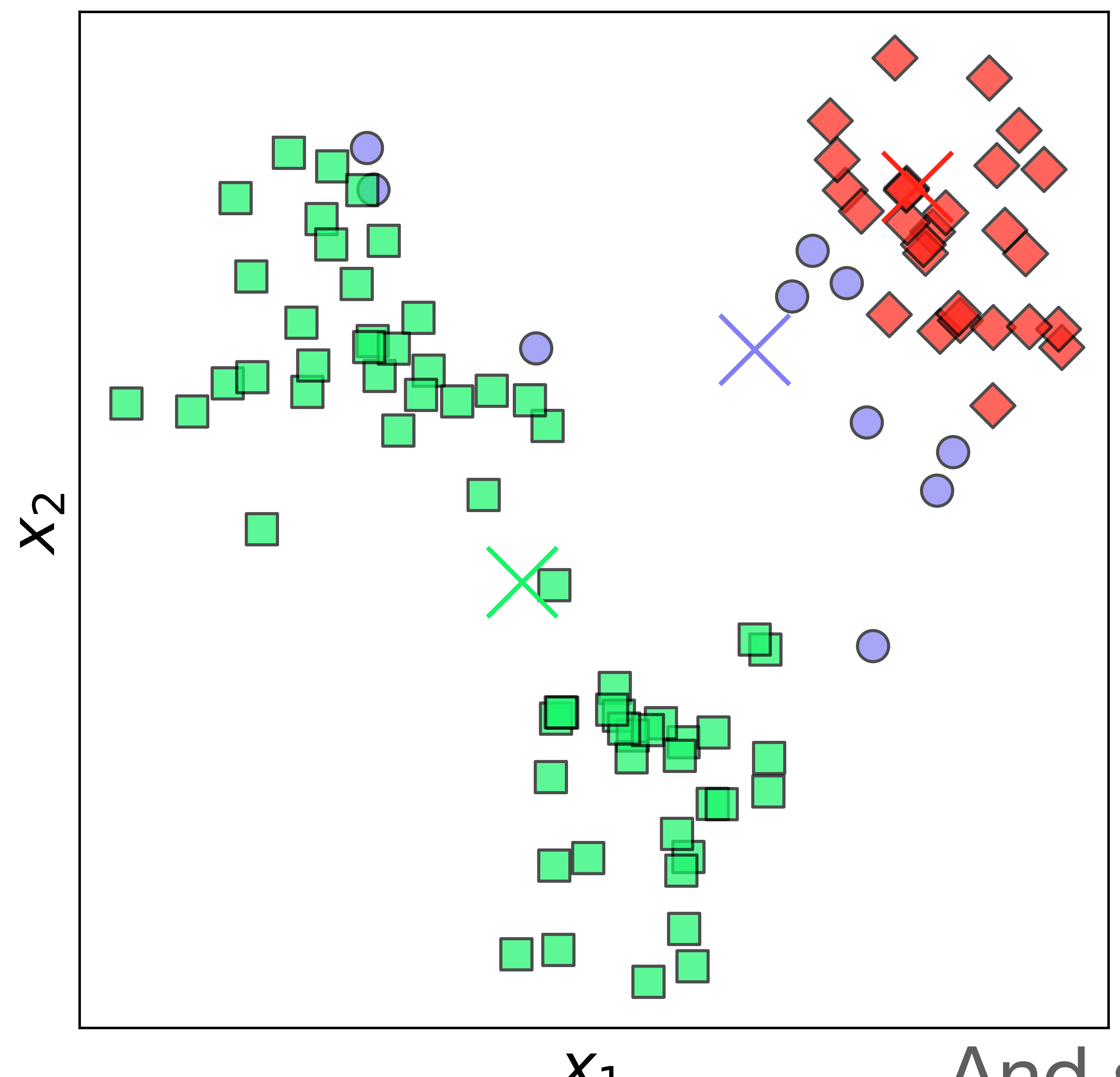


K-means walkthrough with $K = 3$

Update cluster centres as mean of their assigned points



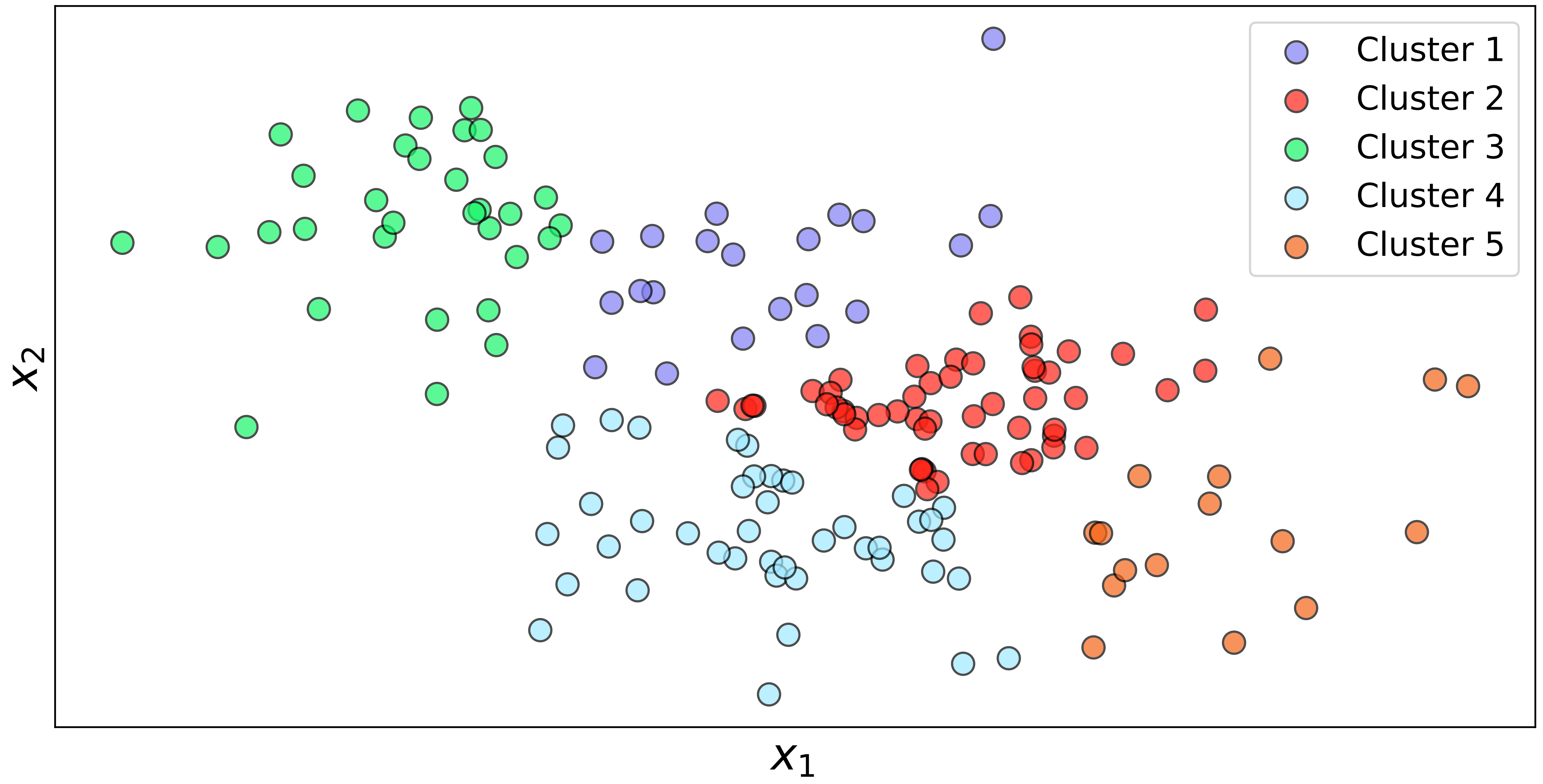
Assign each data point to its nearest cluster centre



And so on!

Warning!

- K-means is very sensitive to where the initial cluster centres are placed
- The number of clusters is user defined
- The clusters **might not be meaningful**



This data is just noise!

Summary

- We have learnt how to preprocess data
- We have seen how PCA can be used for dimensionality reduction
- We have been introduced to K-means and how it can cluster data