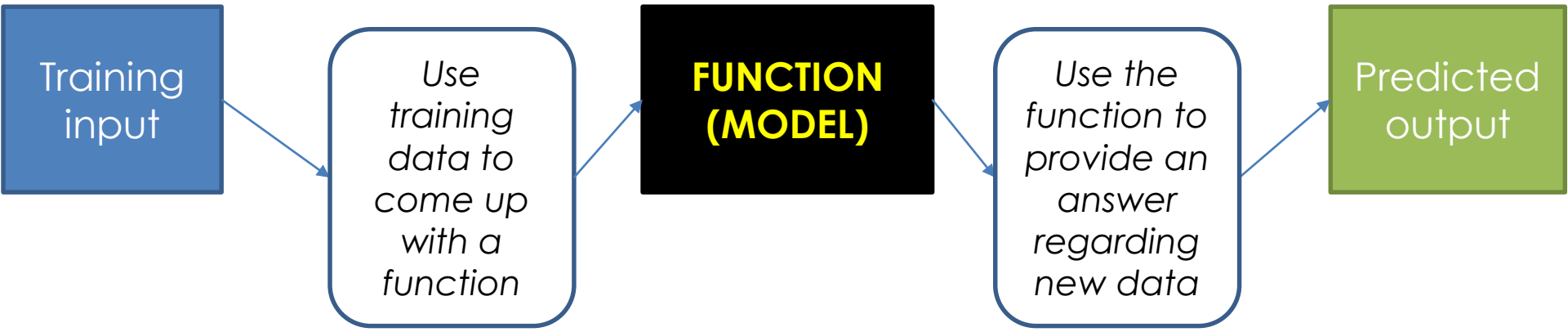
INTRODUCTION TO ARTIFICIAL INTELLIGENCE

4. INTRODUCTION TO **MACHINE LEARNING: REGRESSION AND NEURAL NETWORKS**

The plan for today

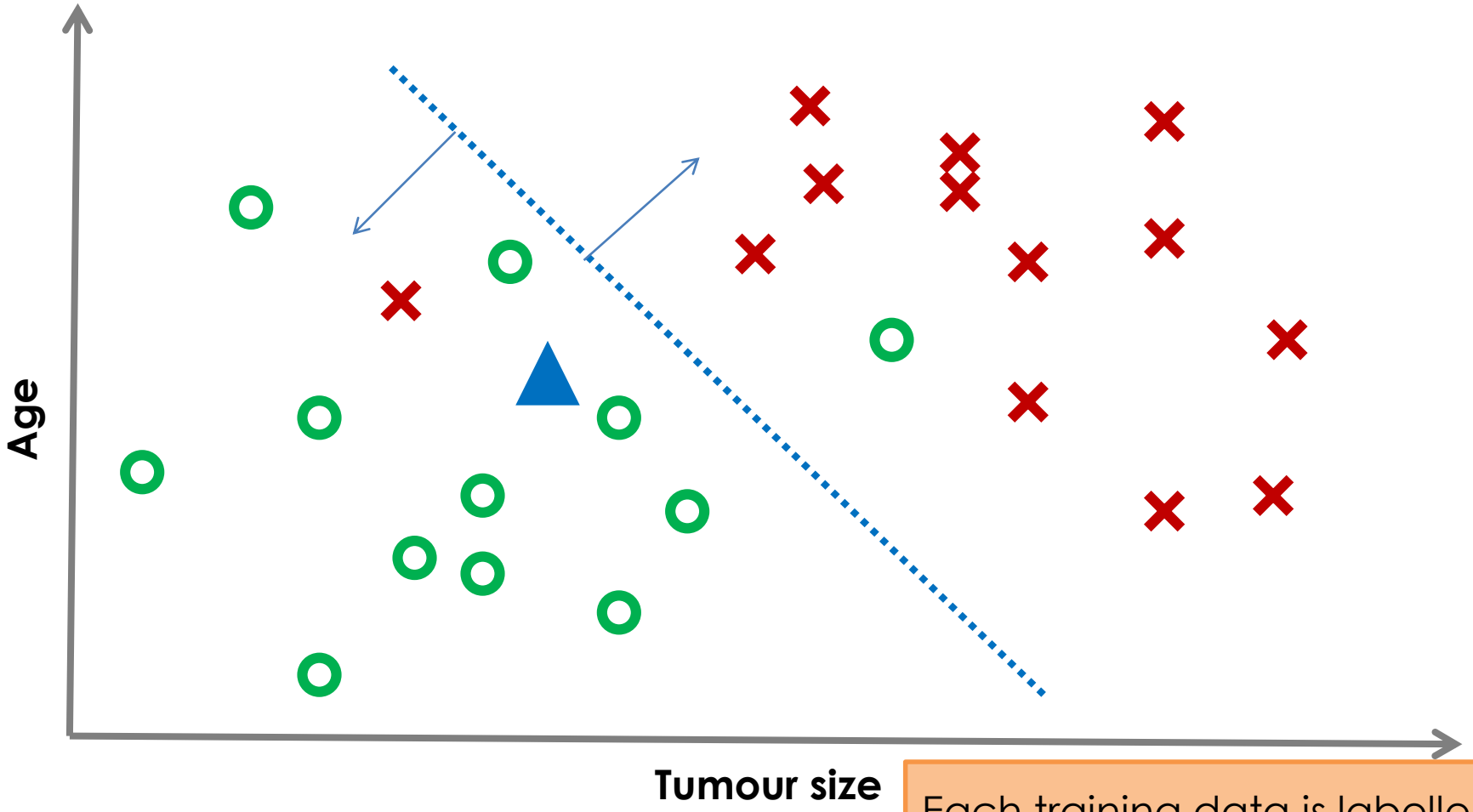
- Introduce **regression** models for supervised learning
  - Learn about statistical relationships
  - Learn about **linear regression**
  - The least squares method to get the 'best-fitting' line
- Introduce the concept of **neural networks**
  - Typical architecture and structure
  - The **perceptron** algorithm for classification

Supervised learning: in a nutshell



- When the prediction is a ***class (category)***, we use **classification**
- When the prediction is a ***number***, we use **regression**

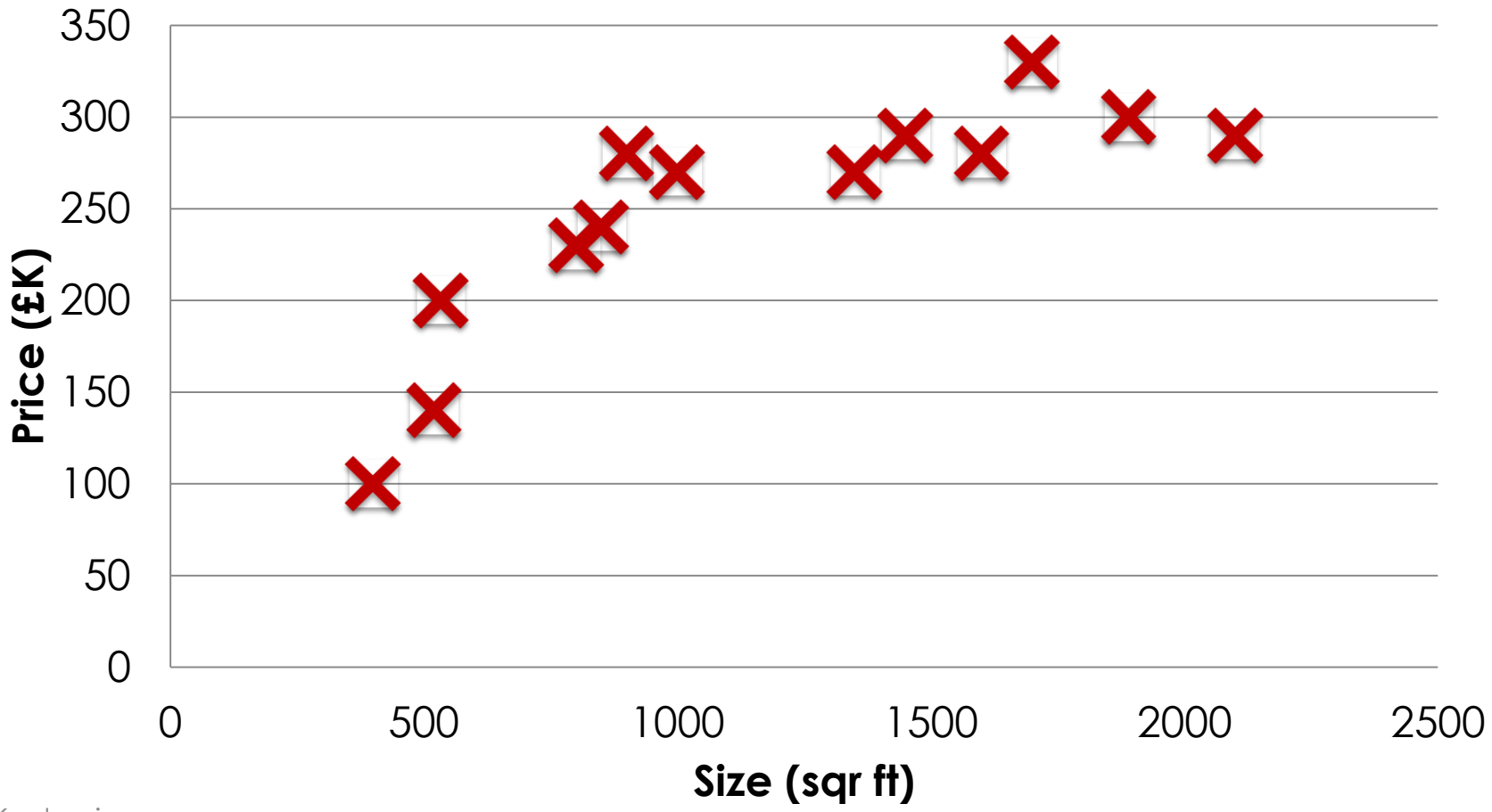
Supervised learning - classification



Each training data is labelled as Cancer (x) or Benign (o).
- What class is '**Δ**'?

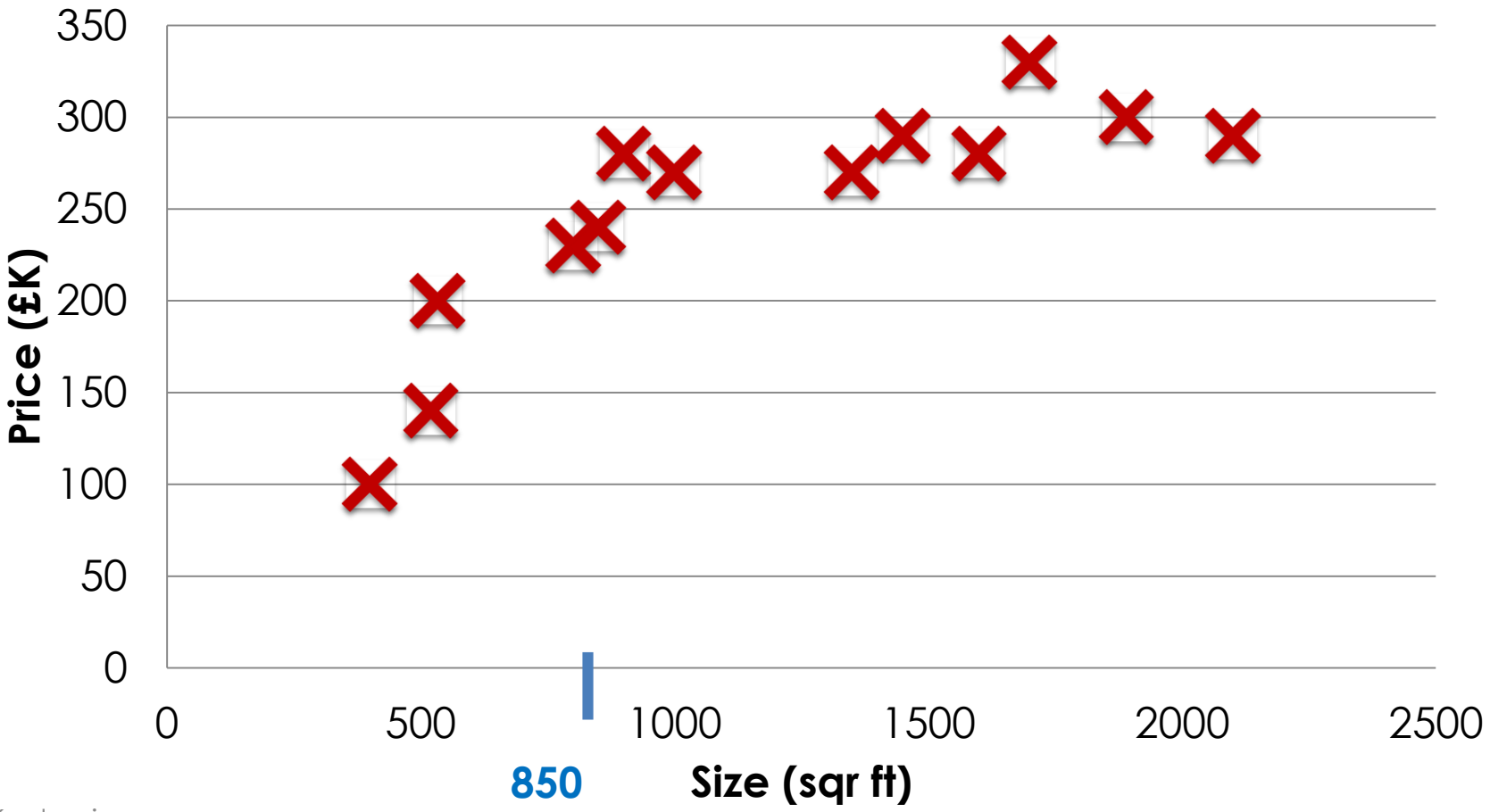
Supervised learning – regression

Housing **Price** Prediction



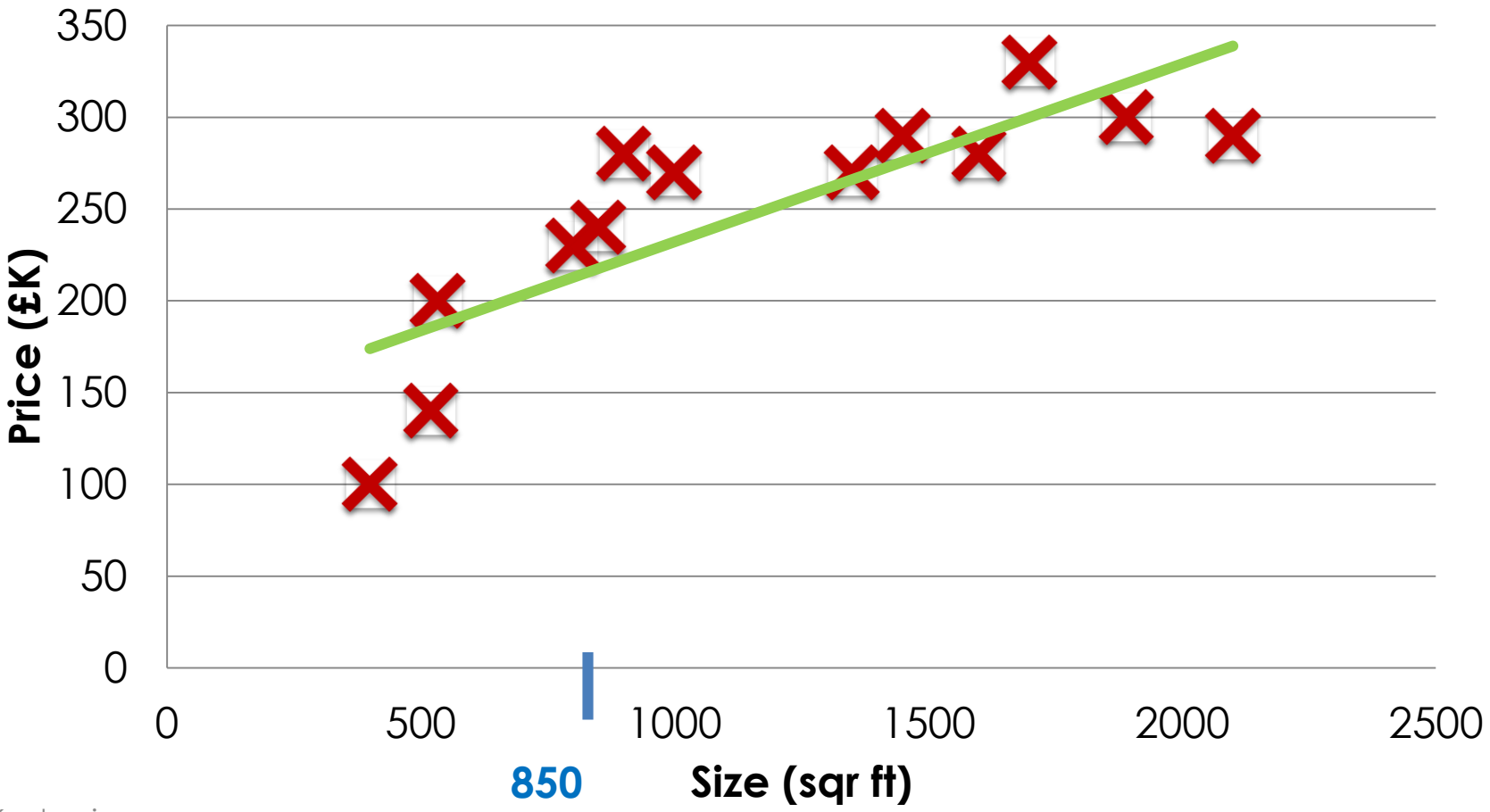
Supervised learning - regression

Housing Price Prediction



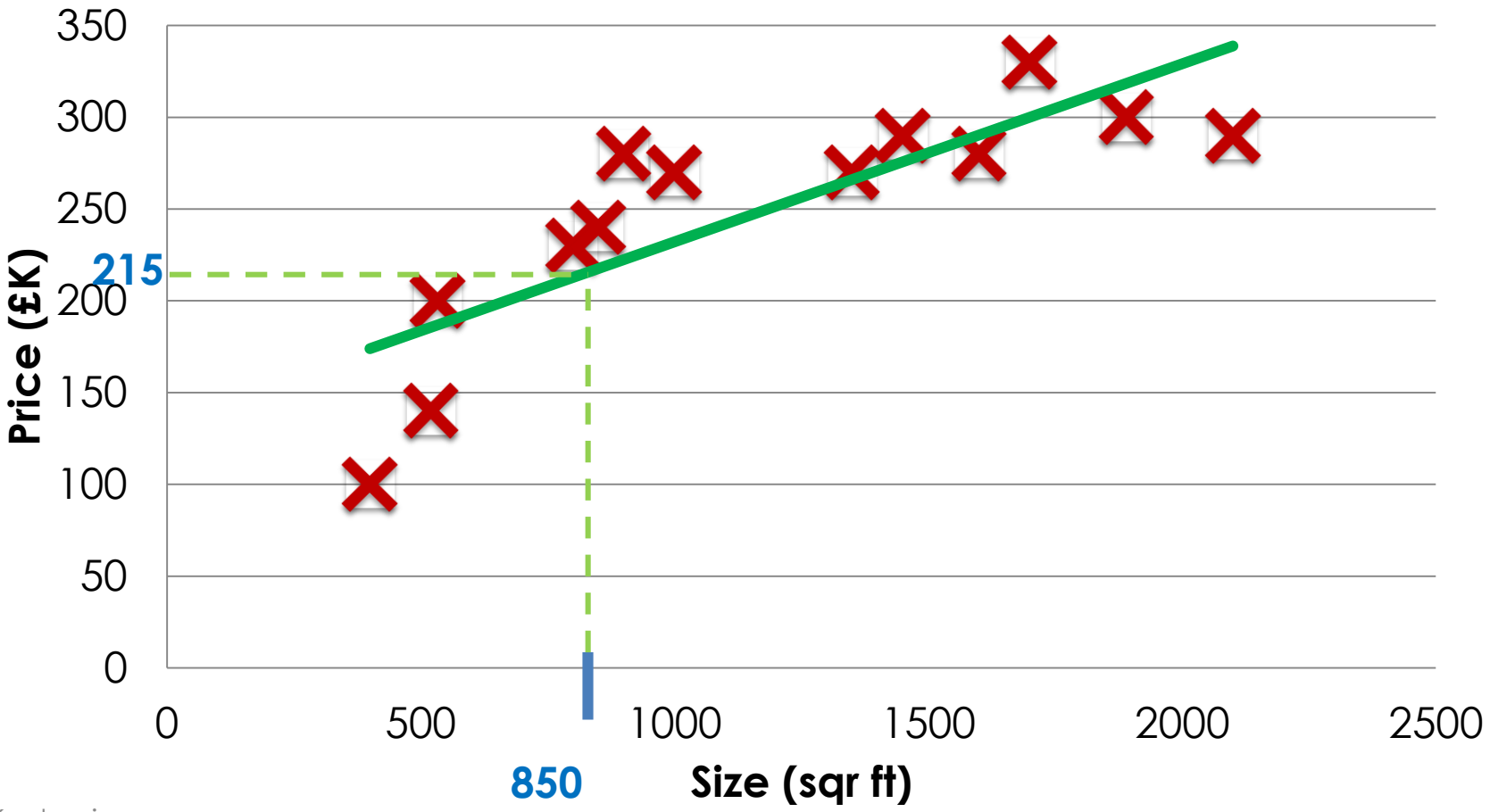
Supervised learning - regression

Housing Price Prediction



Supervised learning - regression

Housing Price Prediction



Supervised learning - regression

Housing Price Prediction

- Supervised learning:
  - We gave the algorithm a dataset in which **examples of the 'right answers'** (size/price) were provided.
  - So the task of the algorithm was to produce more of these right answers.
- This is an example of a '**regression**' problem
- Regression **predicts** a **continuous** valued output (price, in our example)

Quick quiz

- **Problem 1:** You have a large inventory of identical items. You want to predict how many of these items will sell over the next 3 months.
- **Problem 2:** You want to examine individual customer accounts, and for each account, to decide if it has been hacked.

Should you treat these as classification or regression problems?

1. Treat both as classification problems
2. Treat both as classification problems
3. Treat problem 1 as a classification problem and problem 2 as a regression problem
4. Treat problem 1 as a regression problem and problem 2 as a classification problem

Machine learning <3 statistics

- Regression comes from statistics, like many other ML algorithms
- Regression is a model that assumes a **relationship** between the input variable(s) (x) and the single output variable (y).
- More specifically, that y can be calculated (predicted) from the input variable(s) (x).
- Let's take things from the beginning and look at **relationships** between variables

Correlation - Direction

Correlation is the degree of relationship between two variables.

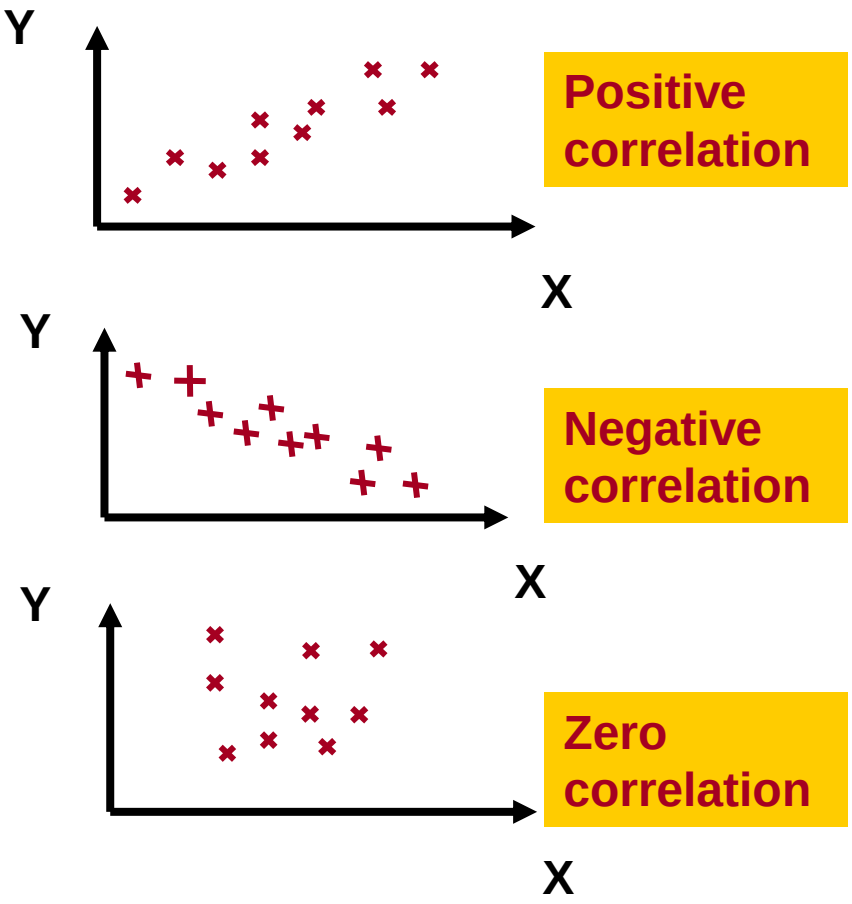
We are interested in two properties of the relationship:

1. Its direction
2. Its strength

Correlation - Direction

Correlation is the degree of relationship between two variables.

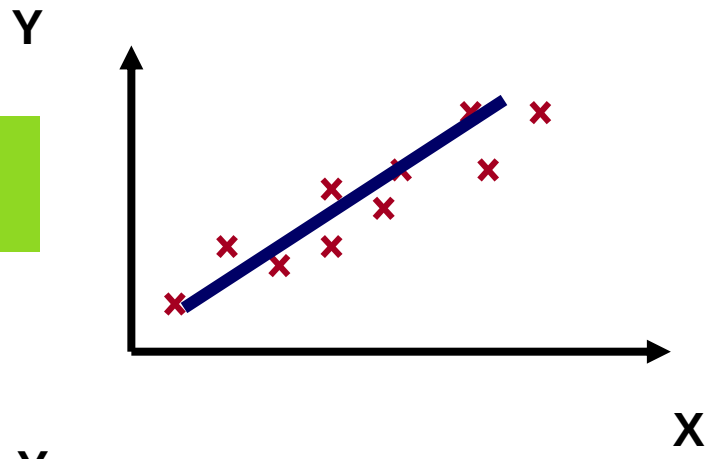
- Direction
  - **Positive** (as the one increases the other variable increases as well)
  - **Negative** (as the one increases the other one decreases)



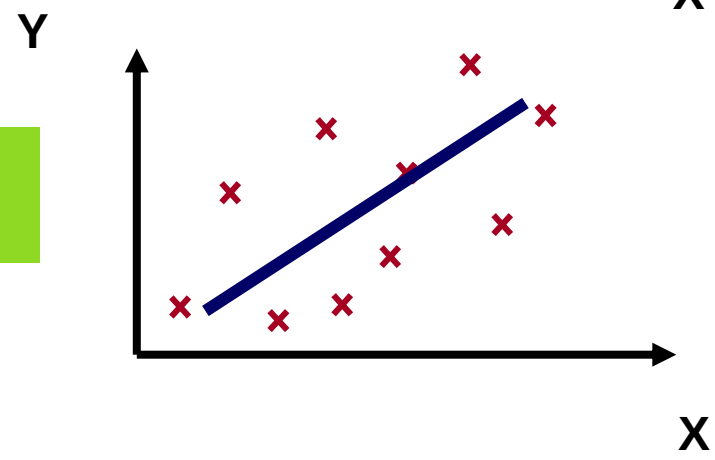
Strength of Correlation

The strength of a correlation is how well the data points fit on a line that represents the relationship between two variables.

"Stronger" correlation



"Weaker" correlation

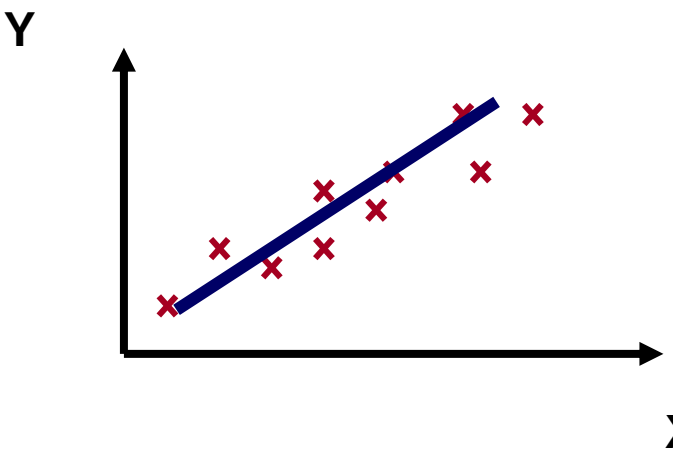


Significance

- Pearson's r is the correlation coefficient we commonly use to see direction and strength.
  - Pearson's r is a value between -1 (a perfect negative correlation) and 1 (a perfect positive correlation).
    - A value close to 0 indicates no correlation.
    - Check here for formula, if you are interested (not assessed).

Correlation and Causality

There is a correlation between ice cream sales (X) and the number of shark attacks on swimmers (Y).



Correlation and Causality

There is a correlation between ice cream sales (X) and the number of shark attacks on swimmers (Y).

Correlation and Causality

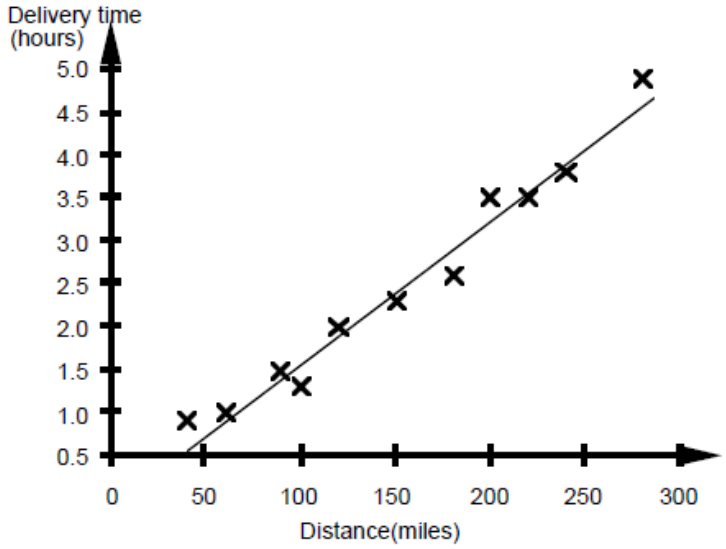
There is a correlation between ice cream sales (X) and the number of shark attacks on swimmers (Y).

This doesn't mean selling ice creams causes shark attacks!

Both X and Y respond to changes in some unobserved variable, probably increase in temperature

Predicting = regression

- Once we have collected some data and observed a significant correlation we can use this relationship to **predict**.
  - E.g. if we know someone's blood pressure (X) can we predict their risk of heart attack (Y)?
  - E.g. If we know the distance a lorry must travel (X) can we predict the delivery time (Y)?



Linear regression

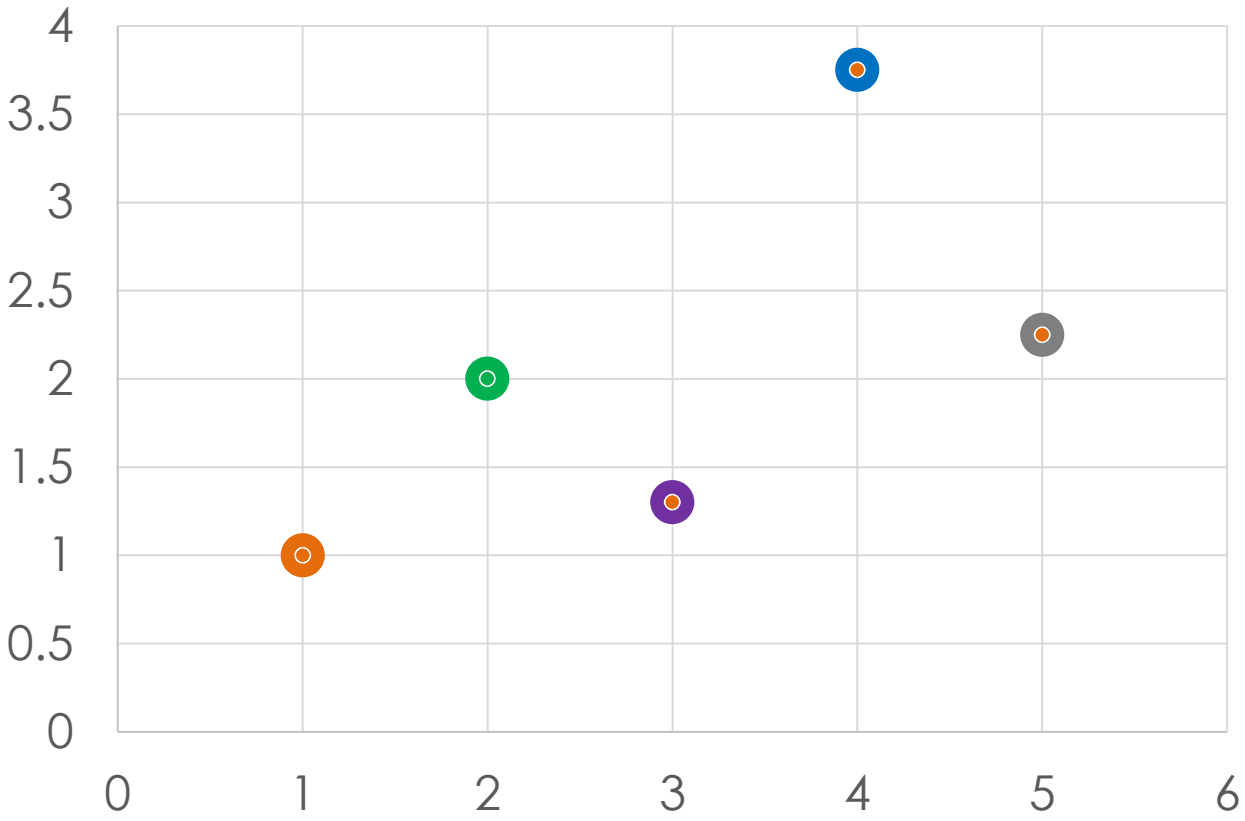
- In **linear** regression, we predict scores on one variable from the scores on another variable(s).
- The variable we are predicting is called:
  - the **output/criterion/dependent** variable and is referred to as **Y**.
- The variable(s) we are basing our predictions on is called:
  - the **input/predictor/independent** variable and is referred to as **X**.
- In simple linear regression, the predictions of Y when plotted as a function of X form a **straight line**.

Simple linear regression

- When there is only **one** predictor variable, the prediction method is called simple linear regression.
- When we have a combination of predictor variables, then the prediction method is called multiple regression.

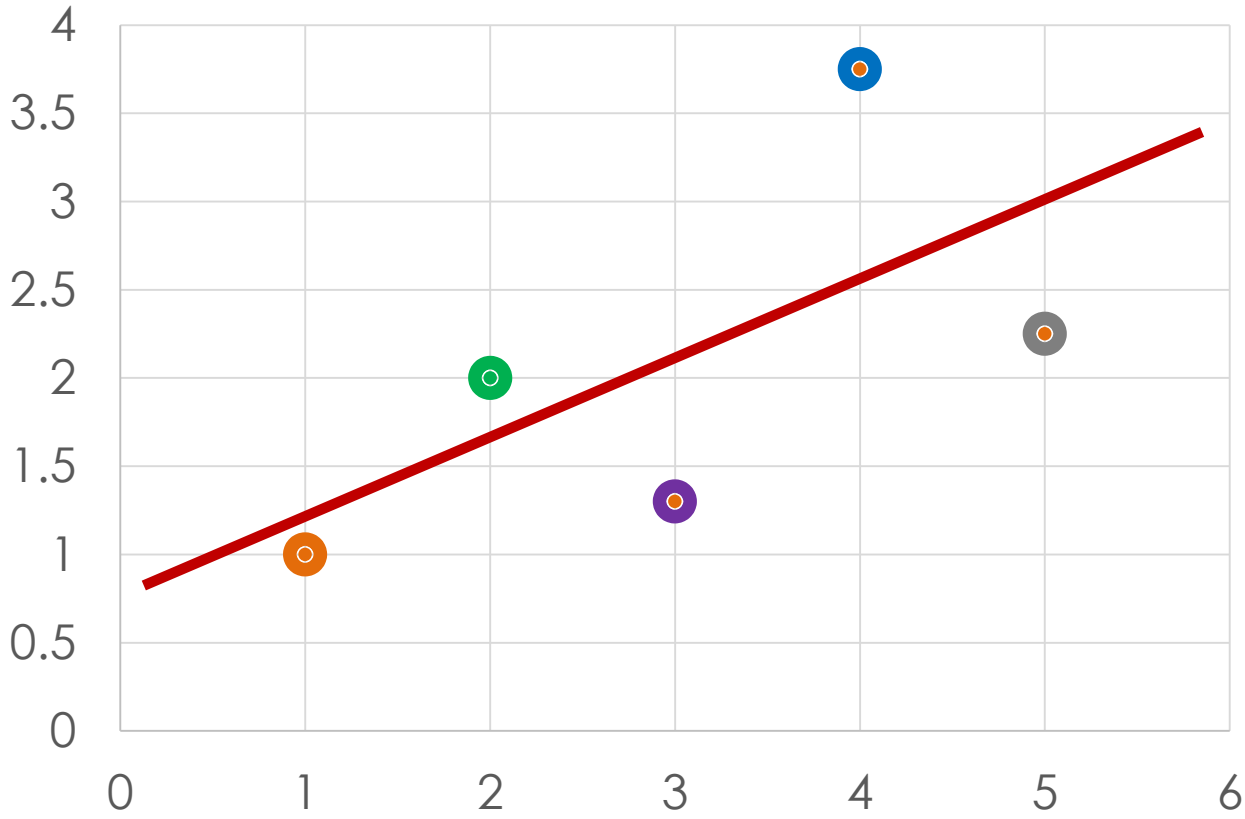
Simple linear regression - example

X	Y
1.00	1.00
2.00	2.00
3.00	1.30
4.00	3.75
5.00	2.25



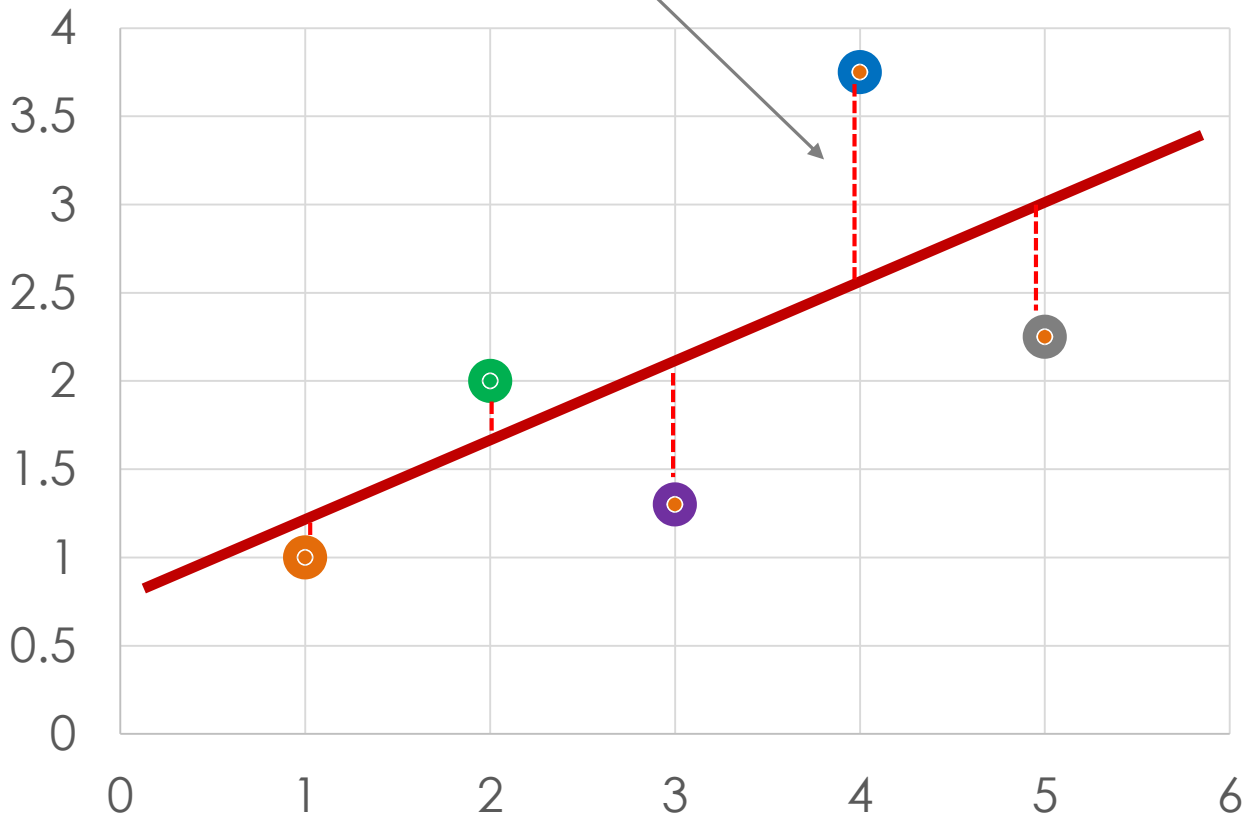
Linear regression

- The 'best fitting' line is called the regression line
- It shows the predicted score on Y for each possible value of X



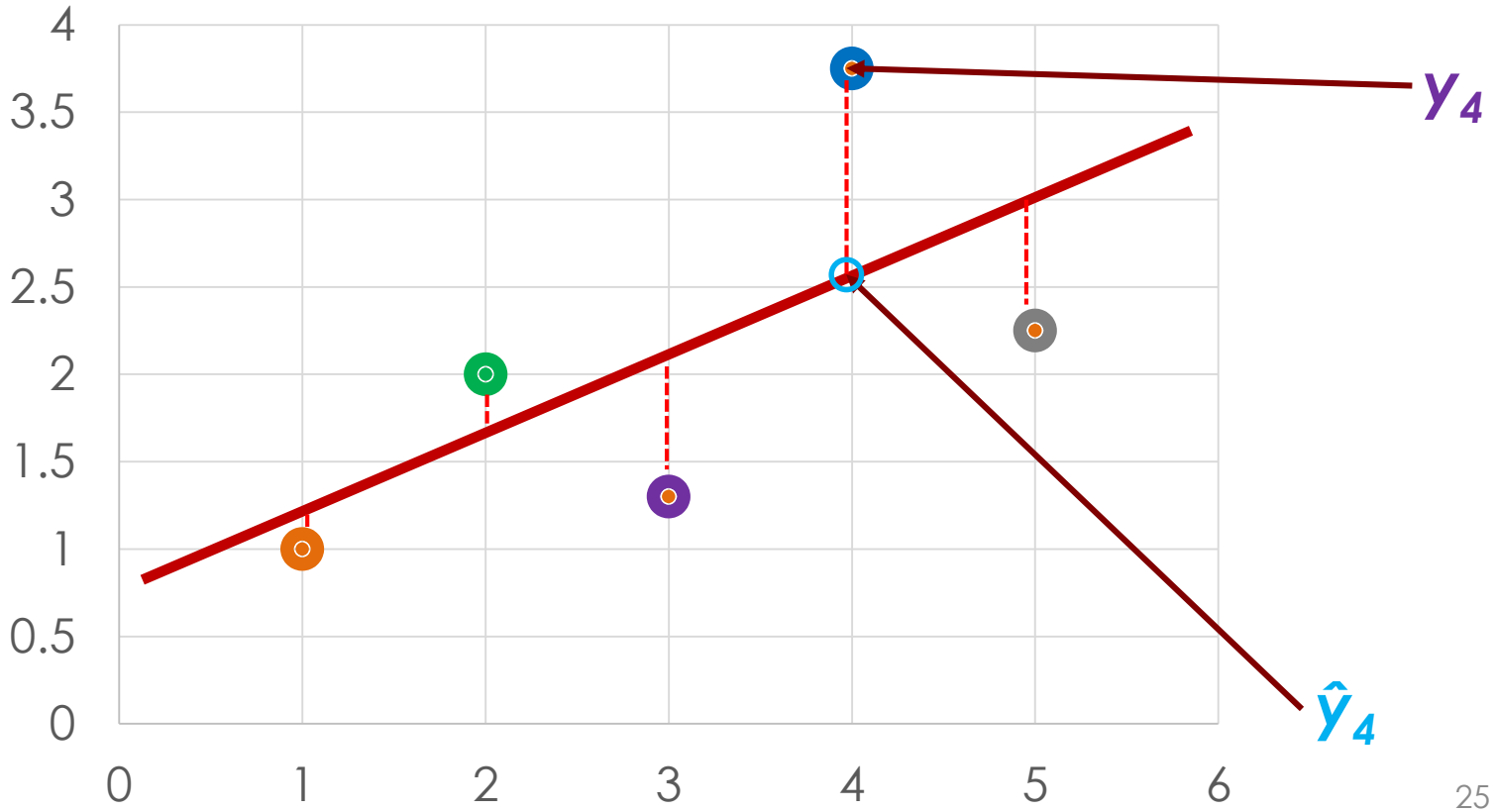
Linear regression

- The **vertical dashed lines** represent the errors of prediction.
  - Prediction error is also called residual
  - The error of prediction of the orange point is small, while it is large for the blue point.



Linear regression

- The error of prediction is the value of the point (**y**) *minus* the predicted value (the value on the line) ($\hat{y}$).
- If the value of y is 1.00 and its $\hat{y}$ is 1.21, what is its prediction error?

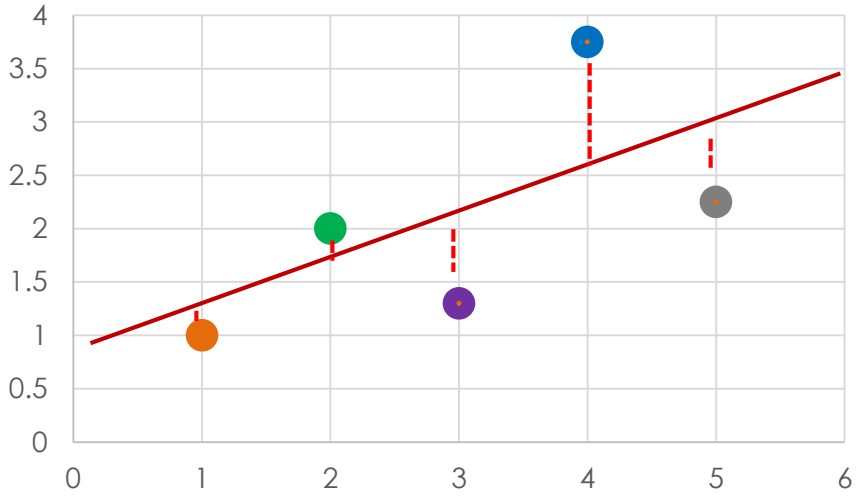


Linear regression

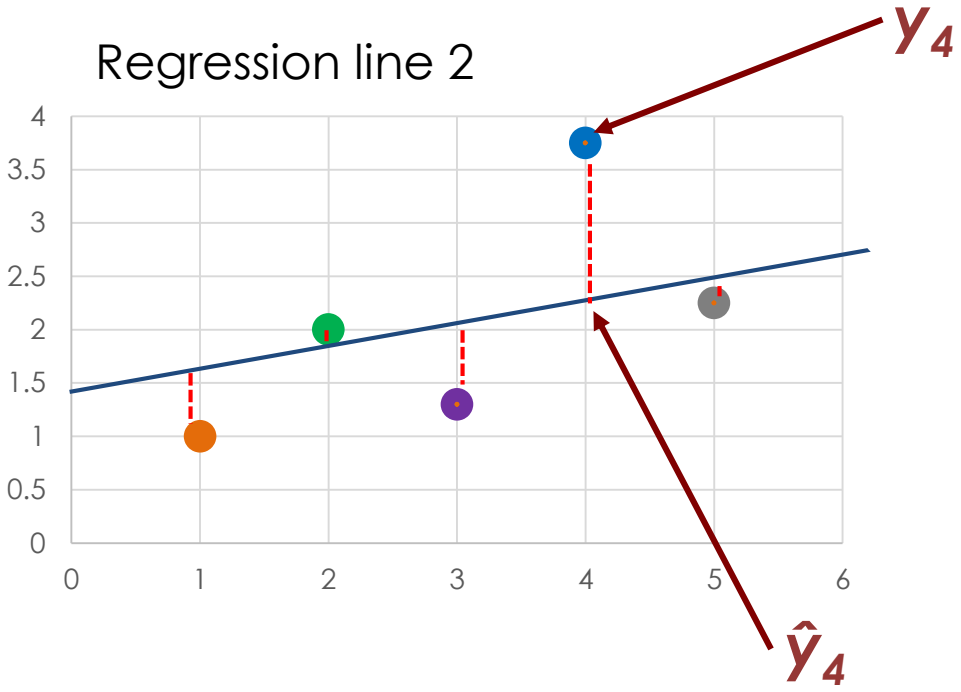
- How do we find the 'best fitting line'?
  - Is it line 1 or line 2?

$$Sum\ of\ \ Squared\ \ Errors = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Regression line 1

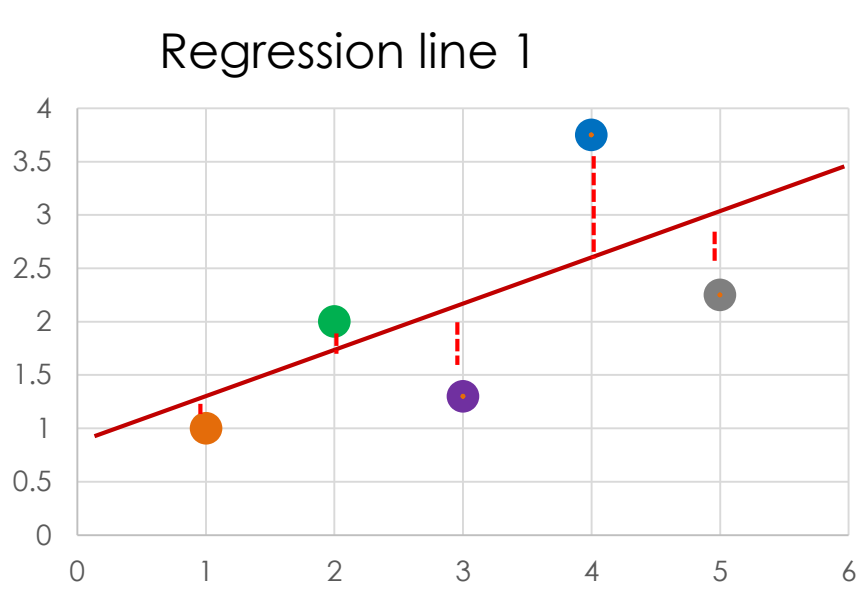


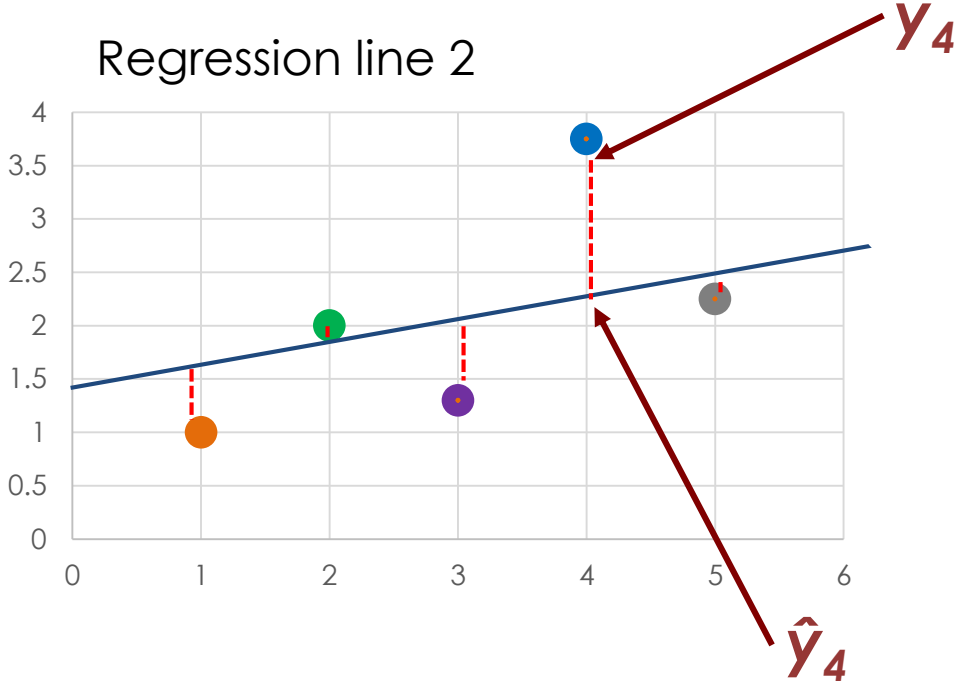
Regression line 2



Least Squares Method

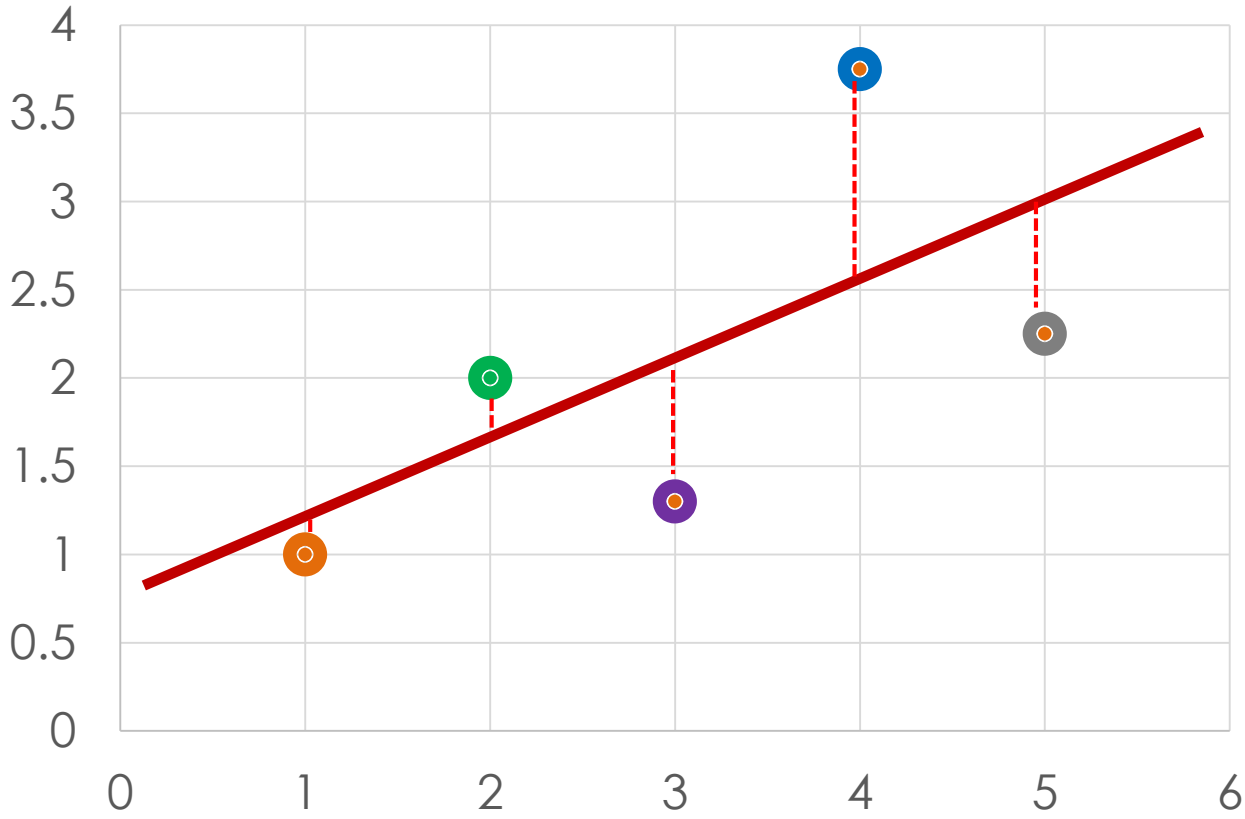
- Look for the line with the smallest SSE
  - The line that minimises the sum of squared errors





Least Squares Method

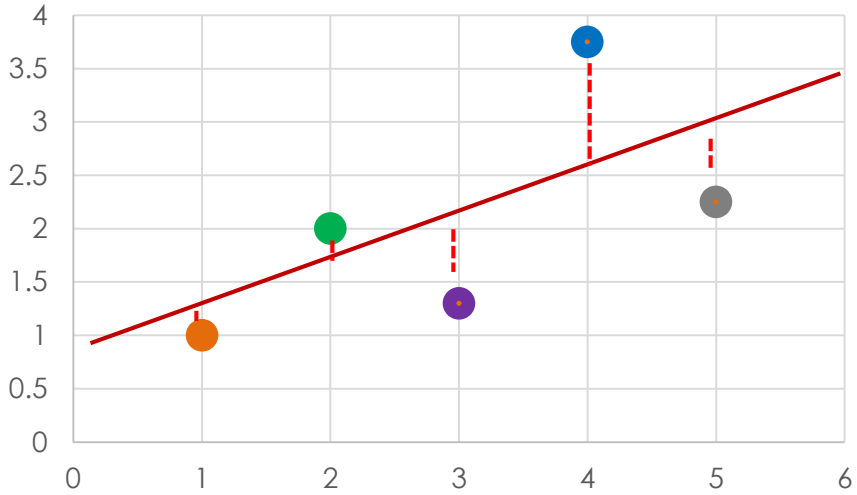
- This is called the least squares method.
- Variety of algorithms to find "best" fit but LSM is by far the most common.
- Seeks to minimise the sum of the squares of the prediction errors.



Least Squares Method Example

- The sum of the squared errors of prediction for this regression line is lower than it would be for any other line.

$$Sum\ of\ \ Squared\ \ Errors = \sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2 = 2.791$$



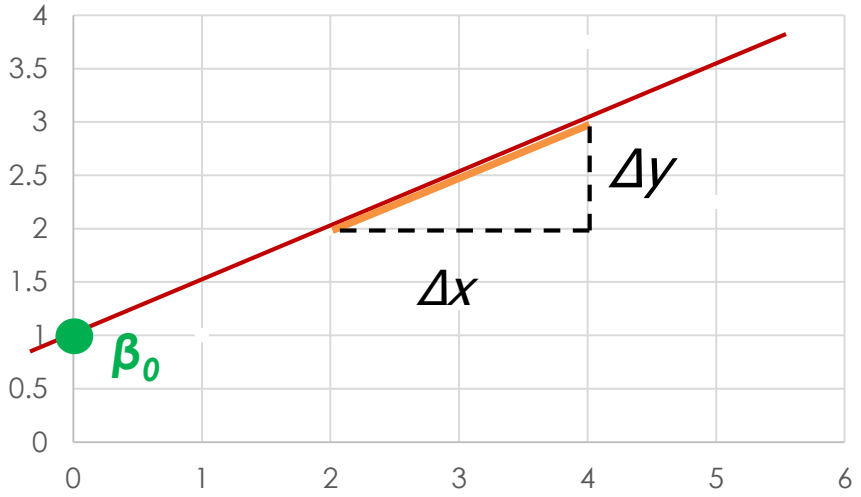
X	Y	$\hat{Y}$	$Y-\hat{Y}$	$(Y-\hat{Y})^2$
1.00	1.00	1.210	-0.210	0.044
2.00	2.00	1.635	0.365	0.133
3.00	1.30	2.060	-0.760	0.578
4.00	3.75	2.485	1.265	1.600
5.00	2.25	2.910	-0.660	0.436

Linear regression

- Now that we found the best fitting line, we can come up with the function (predictive model)
- The formula for a regression line should be familiar…

$\hat{y} = \beta_0 + \beta_1 x$

- β_0 is the *y* intercept
- β_1 is the slope of the line



$$\beta_1 = \frac{\Delta y}{\Delta x}$$

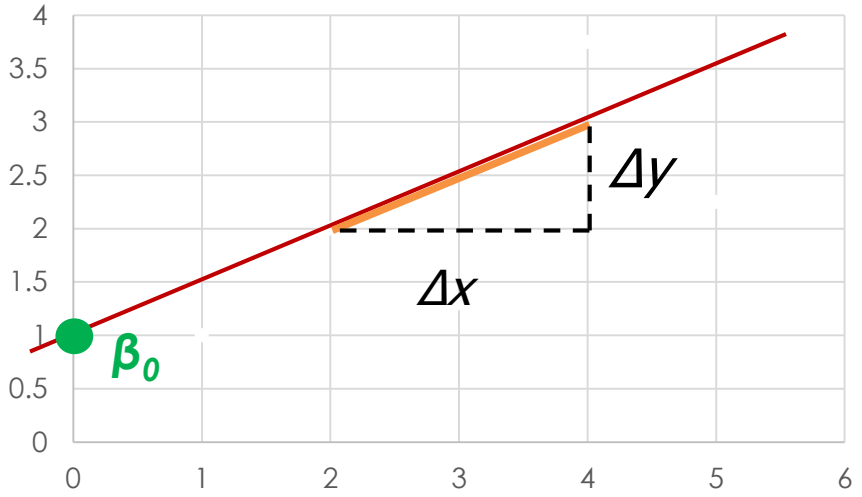
So what is the regression function?

Linear regression

- Now that we found the best fitting line, we can come up with the function (predictive model)
- The formula for a regression line should be familiar…

$\hat{y} = \beta_0 + \beta_1 x$

- β_0 is the *y* intercept
- β_1 is the slope of the line



$$\beta_1 = \frac{\Delta y}{\Delta x}$$

The function is

$\hat{y} = 1 + 0.5x$

$\beta_1 = 1$

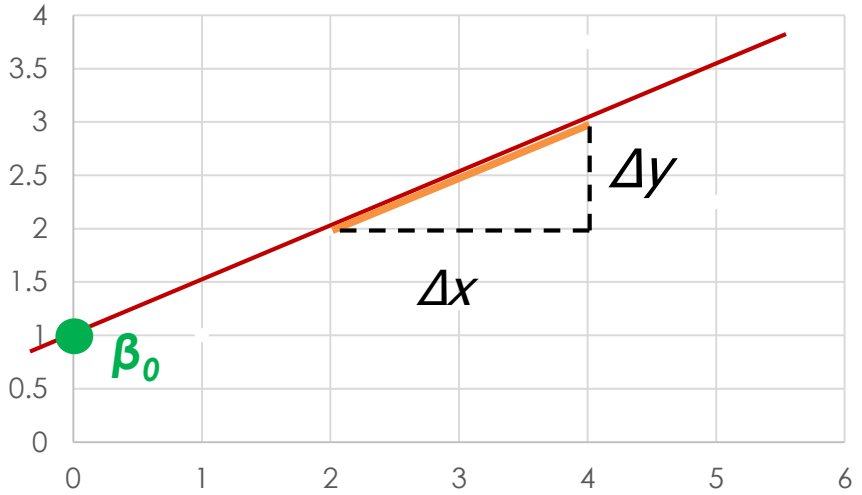
$\beta_0 = (3-2)/(4-2) = 0.5$

Linear regression

- Now that we found the best fitting line, we can come up with the function (predictive model)
- The formula for a regression line should be familiar…

$\hat{y} = \beta_0 + \beta_1 x$

- β_0 is the *y* intercept
- β_1 is the slope of the line



$$\beta_1 = \frac{\Delta y}{\Delta x}$$

So the 'predictive model' is:

$\hat{y} = 1 + 0.5x$

Can you predict ŷ for x = 5?

Computation of best fitting line

- Obviously it is not efficient to compute the sum of the squared prediction errors, for an infinite number of possible lines!
- So, in reality, we use these two formulas for the intercept $\boldsymbol{\beta_0}$ and the slope $\boldsymbol{\beta_1}$, in order to find the equation of the line that minimises the SSE:

$$b_0 = \bar{y} - b_1 \bar{x}$$

$$b_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$\bar{x}$ is the mean of x_i

$\bar{y}$ is the mean of y_i

Regression Models with more than one predictors (multiple linear regression)

$$\hat{y} = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n$$

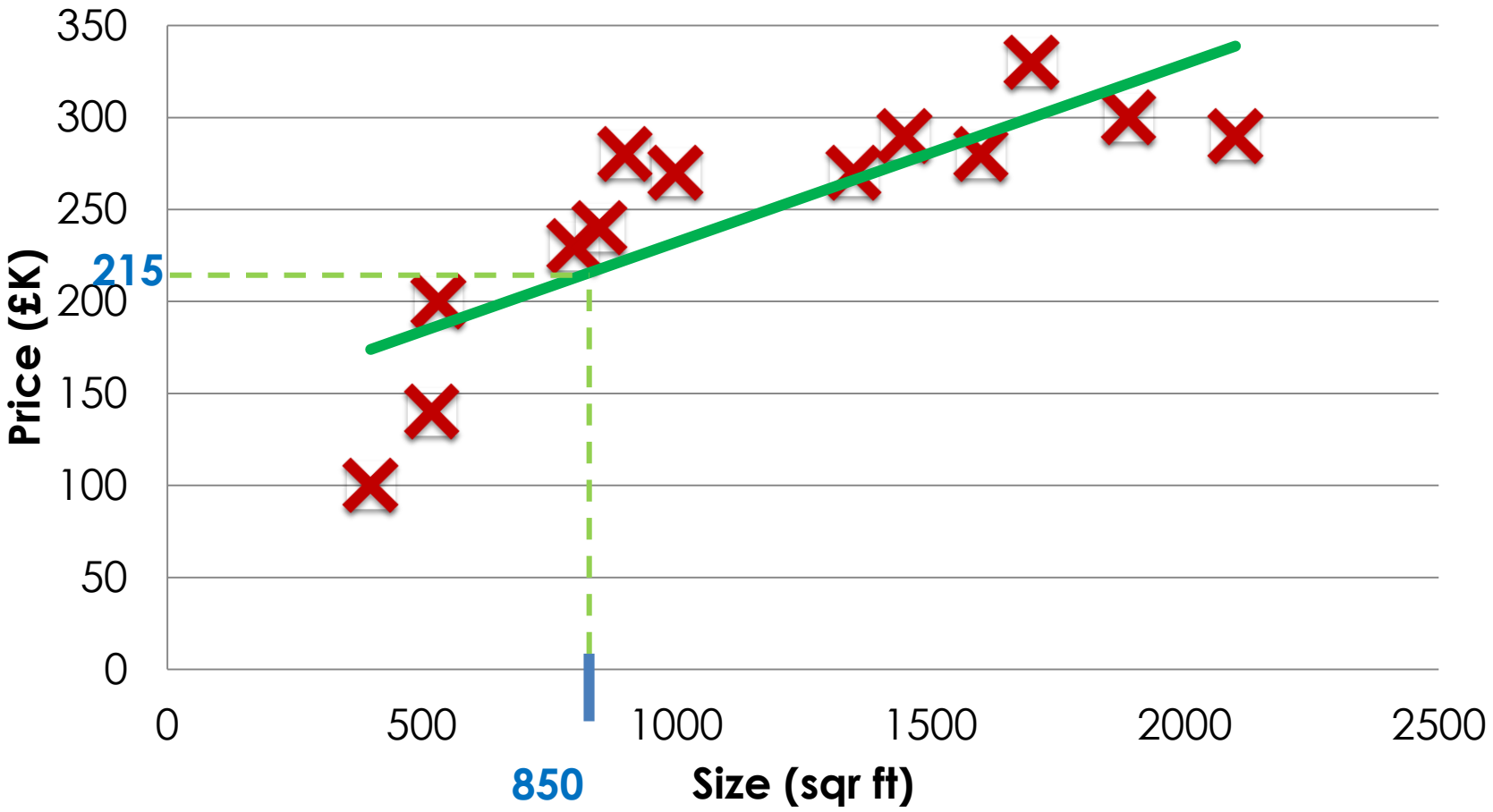
$\hat{y}$	the dependent/output variable *y* being predicted.
β_0	the intercept.
β_1	the slope coefficient
x_i	the independent/input variable (*n* independent variables in a multiple regression model)

Preparing and knowing your data before regression

- **Linear Assumption**. Linear regression assumes that the relationship between your input and output is linear.
  - It does not support anything else. This may be obvious, but it is good to remember when you have a lot of attributes.
  - May be more complex relationships e.g. curvilinear which requires more sophisticated modelling techniques (second, third order polynomial, etc.).

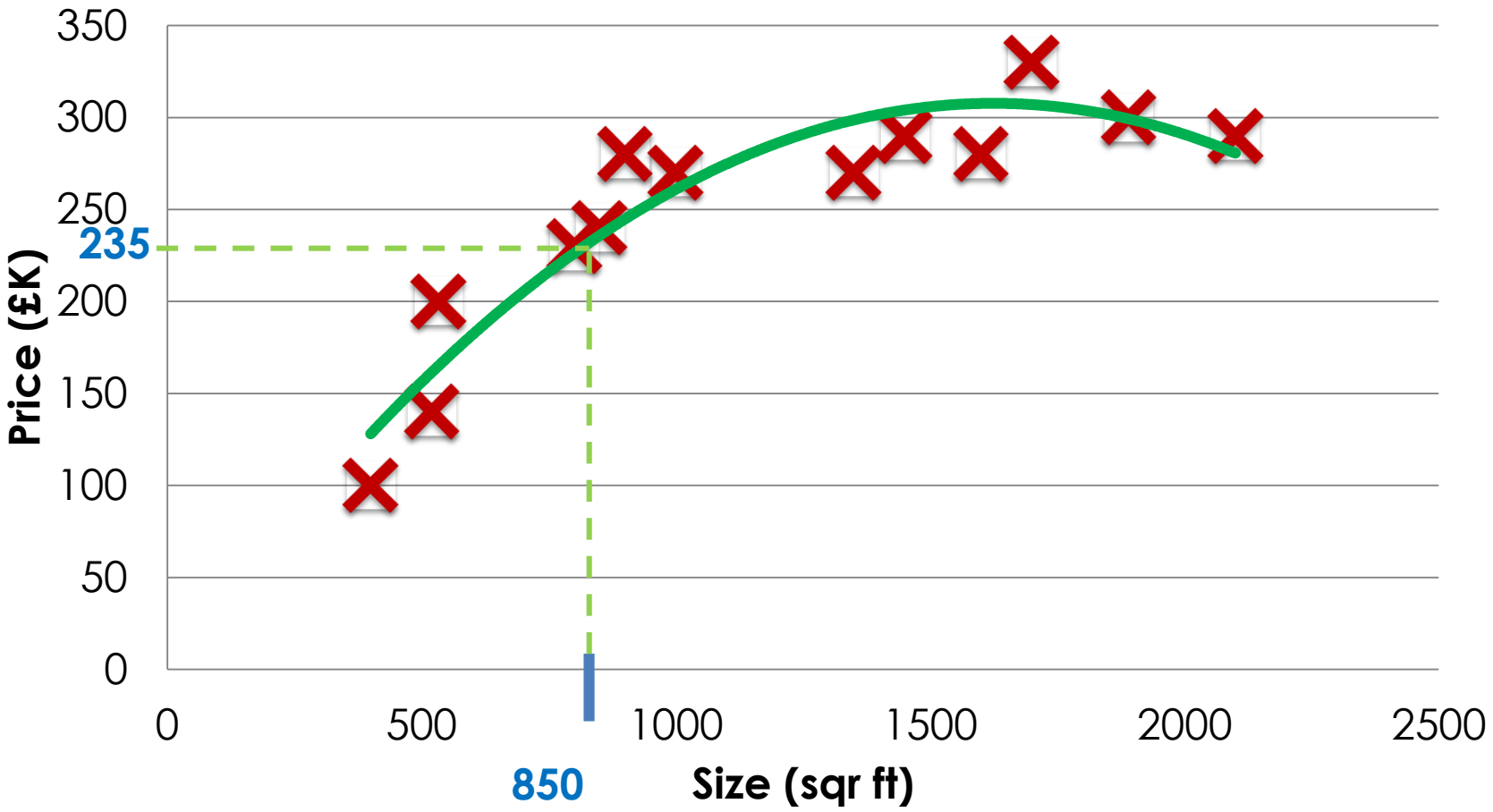
Supervised ML Example

Housing Price Prediction



Supervised ML Example

Housing Price Prediction



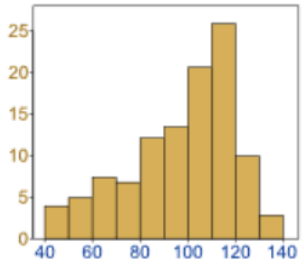
Preparing and knowing your data before regression

- **Remove Outliers**.
  - Linear regression assumes that your input and output variables are not noisy. This is most important for the output variable and you want to remove outliers in the output variable (y) if possible.
- **Remove co-linearity in multiple regression**. Linear regression will over-fit your data when you have highly correlated input variables.
  - Consider calculating pairwise correlations for your input data and removing the most correlated.

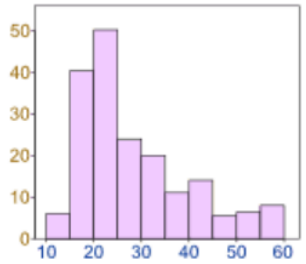
Preparing and knowing your data before regression

- **Normal Distributions**.
  - Linear regression will make more reliable predictions if your input and output variables have a normal (Gaussian, bell-shaped) distribution.

It can be spread out more on the left

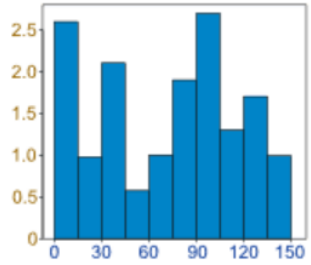


Or more on the right



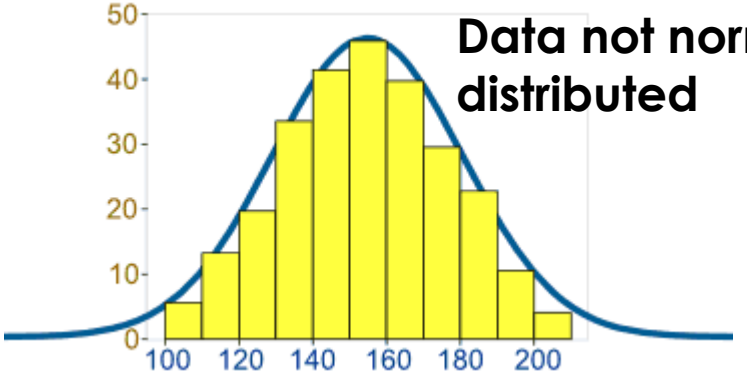
Data not normally distributed

Or it can be all jumbled up



Normal distribution

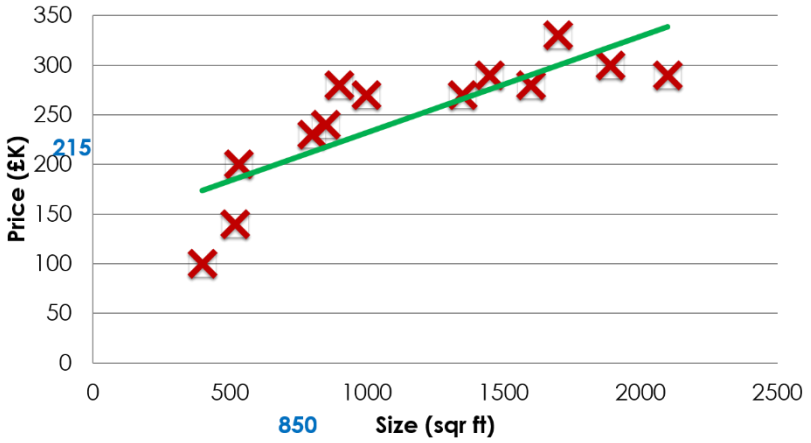
- When data is around a central value (the mean) with no bias left or right, we say that the data follows a normal distribution:



- Height, weight, IQ and other physical measures follow a **normal distribution** of values
- For a large enough population sample, many observations will result in a normal distribution
  - Simple explanation here: https://www.mathsisfun.com/data/standard-normal-distribution.html
- Many statistical tests assume that data within a variable is distributed normally

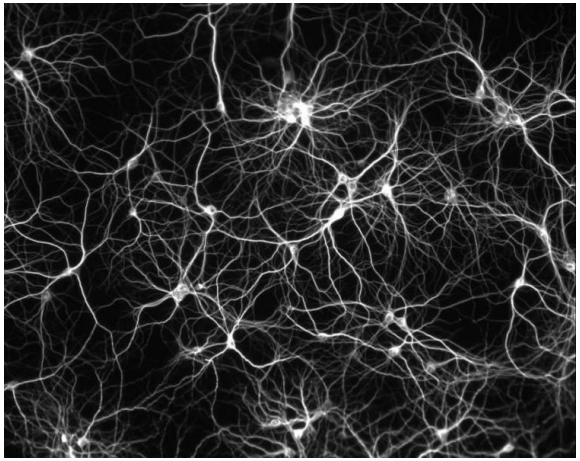
Preparing and knowing your data before regression

- **Rescale Inputs**:
  - Linear regression will often make more reliable predictions if you rescale input variables using standardisation or normalisation ($X_{standardized} = (X - \mu)/\sigma$)
- **Beware of extrapolation:**
  - When you use a regression model, be cautious when you use values for the independent variable that are outside the range of values used to create the model. That is called **extrapolation**, and it can produce unreasonable predictions.



Neural Networks (NNs)

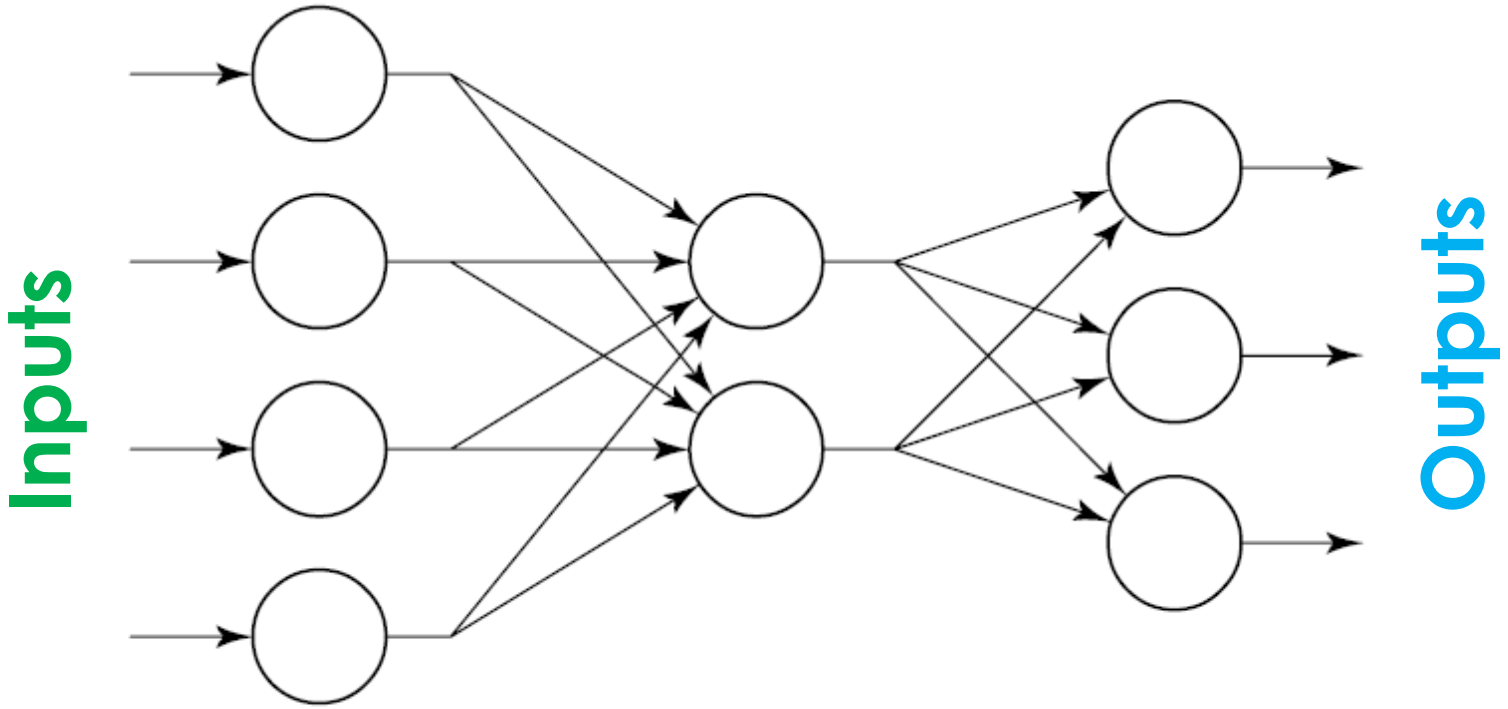
- Inspiration from Biology
- Biological neurons:
  - Have many interconnections
  - Inputs and outputs
  - Make use of simple thresholds



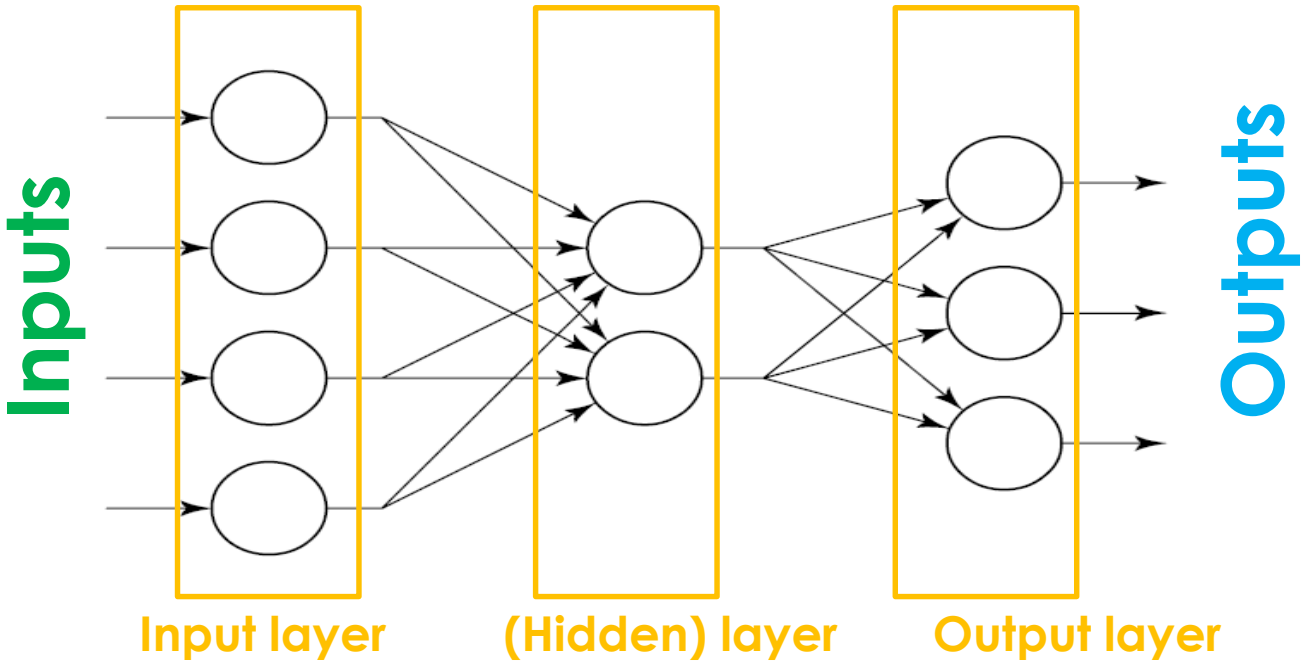
Neural Networks

- Consists of a number of simple **nodes** (neurons)
- The nodes are connected by **weighted links**
- Each node receives a number of **inputs**
- Each node produces one **output** which can be sent through one or more links
- A neural network learns through repeated **adjustments** of these weights.

Architecture of a typical NN

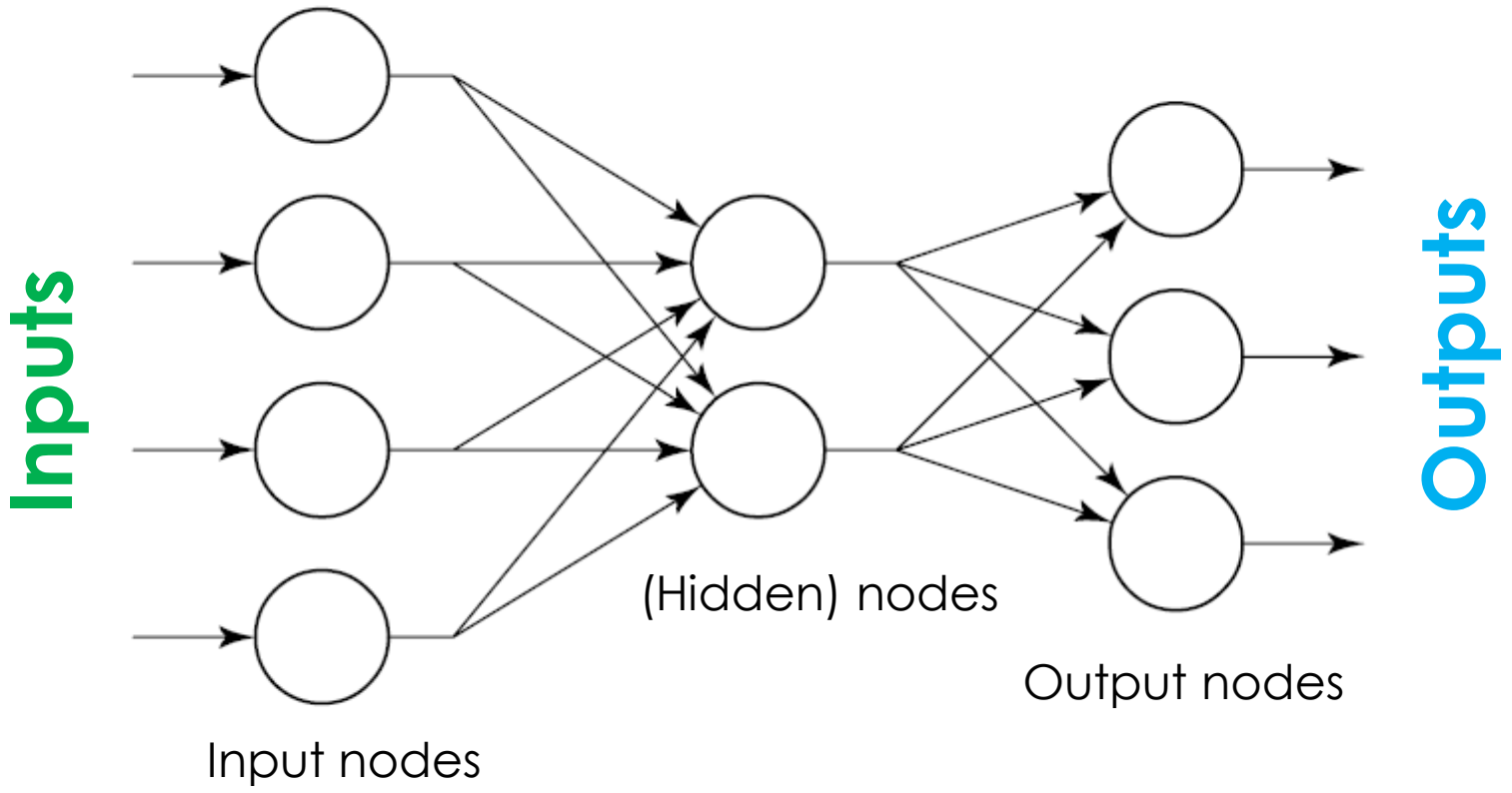


Architecture of a typical NN



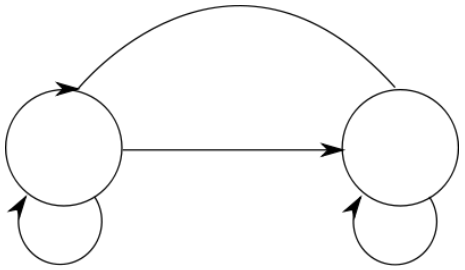
- A NN can have one or more layers (that is, single-layer and multi-layer neural networks).
- Each layer has one or more nodes
- The **input** layer provide information from the outside world to the network.
- The **output** later transfer information to the outside world
- The **middle** layers are called hidden layers, because they have no direct connection with the outside world. We can have 0 or more hidden layers

Architecture of a typical NN



Types of ANNs

- Feedforward networks
  - Like the NN in the previous slide, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes
- Recurrent networks
  - Nodes have feedback loops (back to themselves and/or to previous nodes)

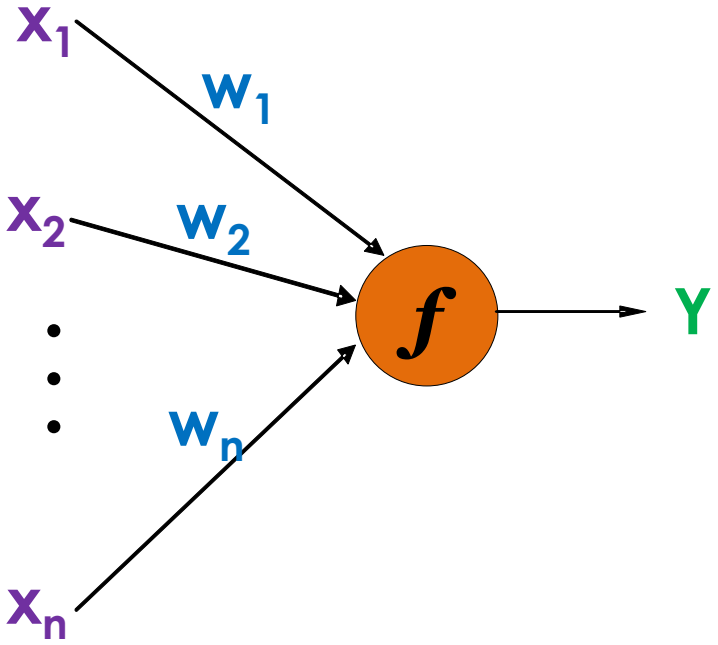


Perceptrons

- A perceptron is the simplest Neural Network structure
- It is a single-layer feed-forward NN
- Typically used for binary classification tasks

Perceptrons

- The general form of a Perceptron:
  - X_i is a vector of inputs
  - W_i is the weight (real numbers expressing the importance of the respective input to the output)
  - **F** is a function, called activation function
  - **Y** is the output, which is a single binary value



Thresholds in NNs

- The output Y (0 or 1) is determined by whether the weighted sum of the inputs:

$$\sum_i w_i x_i$$

… is less or greater than some **threshold** value:

$$Y = \begin{cases} 1, if \sum_i w_i x_i \geq threshold \\ 0, if \sum_i w_i x_i < threshold \end{cases}$$

Simple example

- There is going to be a cheese festival this weekend. I like cheese a lot, but I am trying to decide whether to go or not (output). I will make up my mind by weighting up three factors (inputs):
  1. Is the weather good (super important, so $w_1 = 6$)
  2. Is bae coming along (matters less, so $w_2 = 2$)
  3. Is there public transport (matters less, so $w_3 = 2$)

  I set my **threshold** to be **5**.

Simple

$$Y = \begin{cases} 1, if \sum_i w_i x_i \geq threshold \\ 0, if \sum_i w_i x_i < threshold \end{cases}$$

1. Is the weather good (super important, so $w_1 = 6$)
2. Is bae coming along (matters less, so $w_2 = 2$)
3. Is there public transport (matters less, so $w_3 = 2$)

I set my threshold to be 5.

- The weather **is** good, bae **is not** coming and there **is no** public transport.
- So what is the output? Should I go to the festival (1 means yes, 0 means no)?

Simple

$$Y = \begin{cases} 1, if \sum_i w_i x_i \geq threshold \\ 0, if \sum_i w_i x_i < threshold \end{cases}$$

1. Is the weather good (super important, so $w_1 = 6$)
2. Is bae coming along (matters less, so $w_2 = 2$)
3. Is there public transport (matters less, so $w_3 = 2$)

I set my threshold to be **5**.

- The weather **is** good, bae **is not** coming and there **is no** public transport.
- So what is the output? Should I go to the festival (1 means yes, 0 means no)?

(1 * 6) + (0 * 2) + (0 * 2) = **6**

6 > **5** (threshold)

So I am going!

Simple example variation

$$Y = \begin{cases} 1, if \sum_i w_i x_i \geq threshold \\ 0, if \sum_i w_i x_i < threshold \end{cases}$$

1. Is the weather good (super important, so w_1 = 6)
2. Is bae coming along (matters less, so w_2 = 2)
3. Is there public transport (matters less, so w_3 = 2)

I set my threshold to be **3**.

The weather **is not** good, bae **is** coming and there **is** public transport.

(0 * 6) + (1 * 2) + (1 * 2) = 4

4 > 3 (threshold)

So I am going!

We can choose a lower threshold or even no threshold, depending on whether we are more willing to go to the festival

Perceptrons

Moving the threshold (symbol: **θ**) to the other side of the inequality we get the perceptron's activation function :

$$\sum_i w_i x_i + \theta$$

$= w_1x_1 + w_2x_2 \ldots + w_nx_n + \theta$

where x_i is the ith input

w_i is associated weight

θ is the threshold value

Perceptrons

A Perceptron produces an output, ***Y***, based upon the computed output of an activation function called the **Sign Function**:

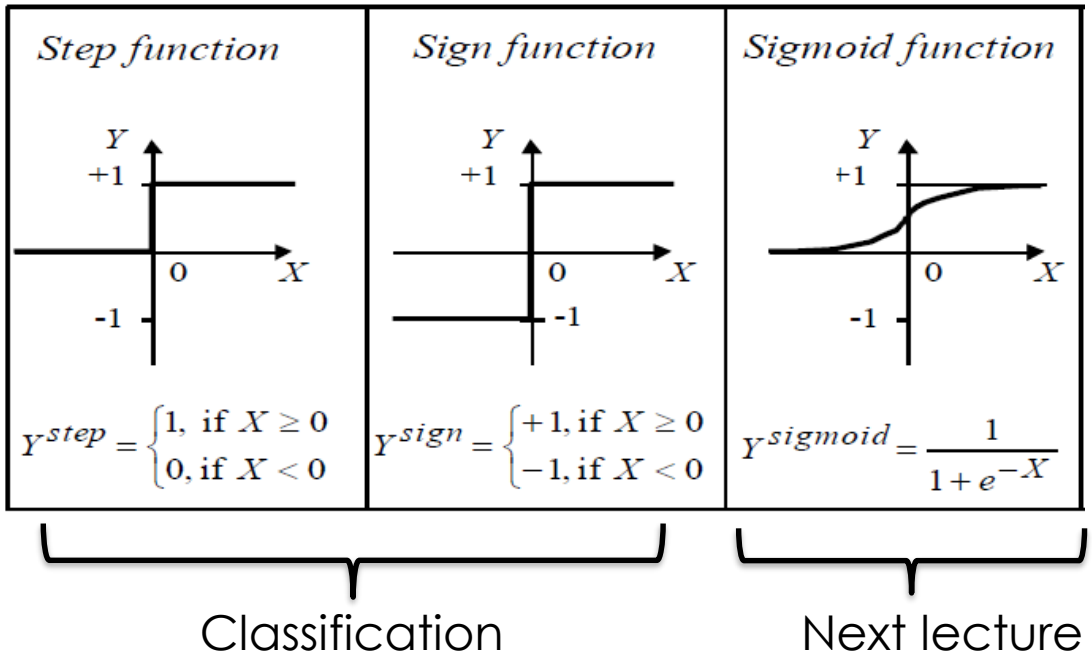
$$Y = \begin{cases} +1, if \sum_i w_i x_i + \theta \geq 0 \\ -1, if \sum_i w_i x_i + \theta < 0 \end{cases}$$

* Not the same activation function as the one in the simple example – see next slide…

Perceptrons

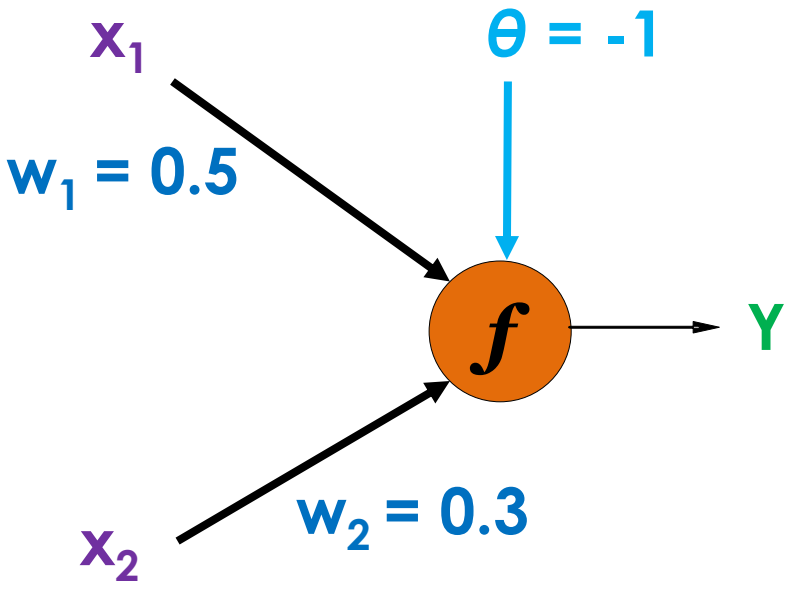
- Other functions available. For example:

  - Step, Sign, Sigmoid (takes input value, squashes it between 0 and 1), Tanh (takes input value and squashes it between -1 and 1)



Perceptrons

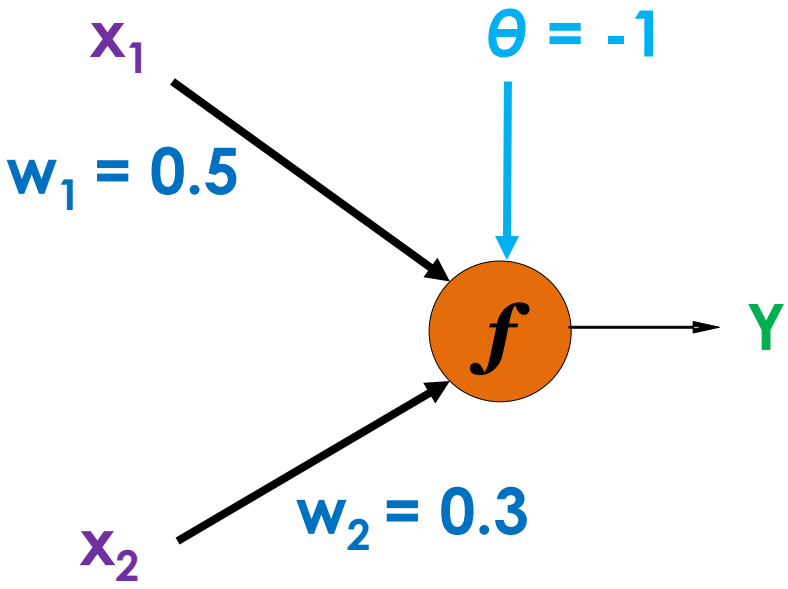
Let's look at a simple classification example using a perceptron with two inputs, $x_1 = 2$ and $x_2 = 1$



Perceptrons

$$Y = \begin{cases} +1, if \sum_i w_i x_i + \theta \geq 0 \\ -1, if \sum_i w_i x_i + \theta < 0 \end{cases}$$

- $x_1 = 2$ and $x_2 = 1$
- What will the output *Y* be?



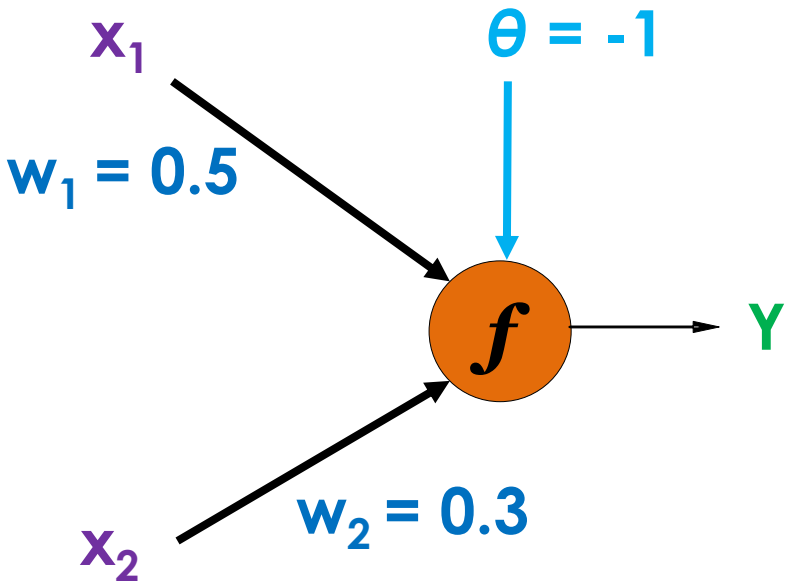
Perceptrons

$$Y = \begin{cases} +1, if \sum_i w_i x_i + \theta \geq 0 \\ -1, if \sum_i w_i x_i + \theta < 0 \end{cases}$$

- $x_1 = 2$ and $x_2 = 1$

$(0.5 * 2) + (0.3 * 1) - 1 = 0.3$

As 0.3 > 0, the output *Y* will be +1



Learning

- It learns through adjusting **weights**
- Learning is iterative. Each iteration is called an ***epoch***.
- The new updated weights and threshold are calculated at *epoch* ***p+1*** given the values at epoch *p*:

$$w_i(p+1) = w_i(p) + \Delta w_i(p)$$

$$\theta(p+1) = \theta(p) + \Delta\theta(p)$$

Learning

$$w_i(p+1) = w_i(p) + \Delta w_i(p)$$

$$\theta(p+1) = \theta(p) + \Delta\theta(p)$$

- An error correcting procedure is used:

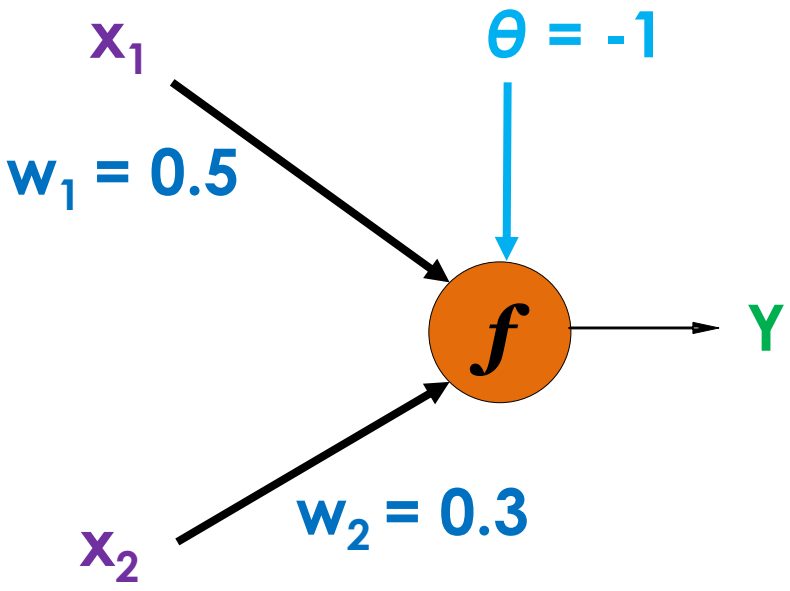
$$\Delta w_i(p) = (Y^d - Y)x_i$$

$$\Delta\theta(p) = (Y^d - Y)$$

- Where Y^d is the True answer and Y is the perceptron output

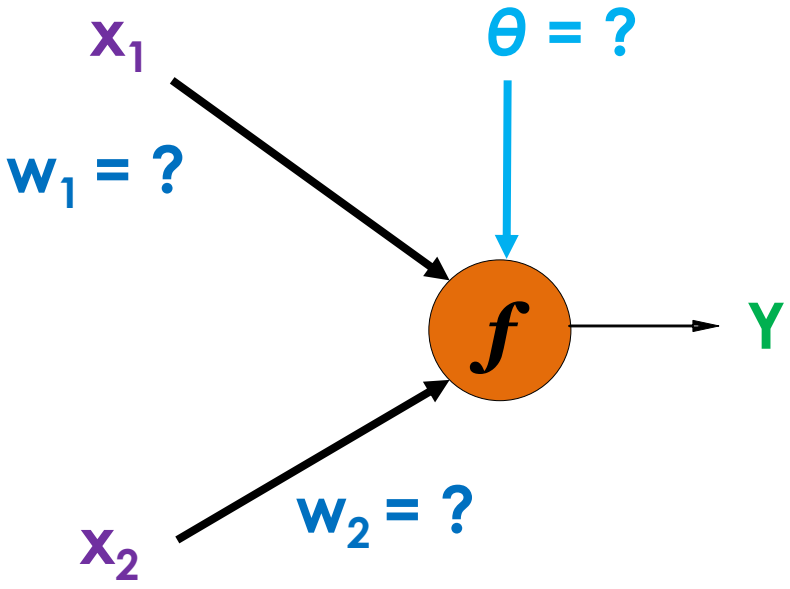
Learning – example

- *From the previous example…*
- $x_1 = 2$, $x_2 = 1$
  $w_1 = 0.5$, $w_2 = 0.3$, $\theta = -1$
- The output was calculated as **Y = 1**
- However, the true output is **$Y^d = 0$**



Learning – previous example

- $x_1 = 2, x_2 = 1$
  $w_1 = 0.5, w_2 = 0.3, \theta = -1$
- The output was calculated as **Y = 1**
- However, the true output is $\boldsymbol{Y^d = 0}$
- How should I update the weights and threshold?



Learning – previous example

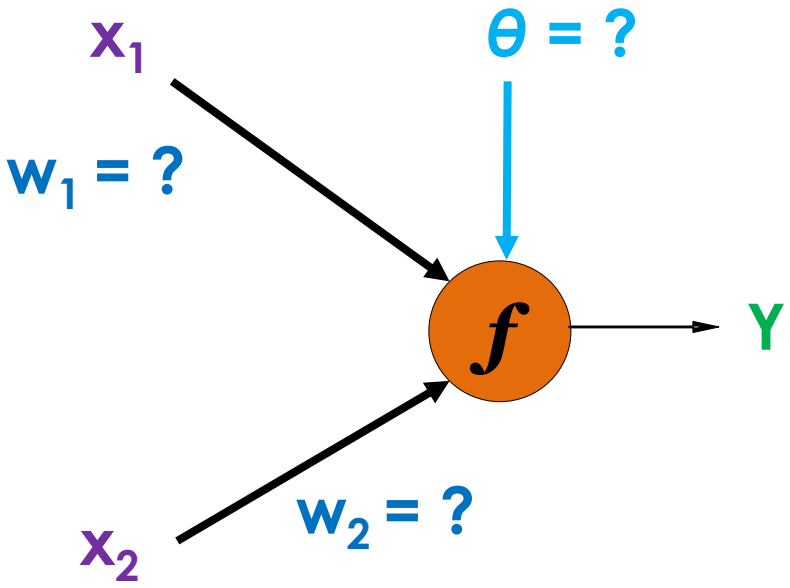
- $x_1 = 2$, $x_2 = 1$
  $w_1 = 0.5$, $w_2 = 0.3$, $\theta = -1$
- The output was calculated as **Y = 1**
- However, the true output is **$Y^d = 0$**
- How should I update the weights and threshold?

$$w_i(p+1) = w_i(p) + \Delta w_i(p)$$

$$\theta(p+1) = \theta(p) + \Delta\theta(p)$$

$$\Delta w_i(p) = (Y^d - Y)x_i$$

$$\Delta\theta(p) = (Y^d - Y)$$



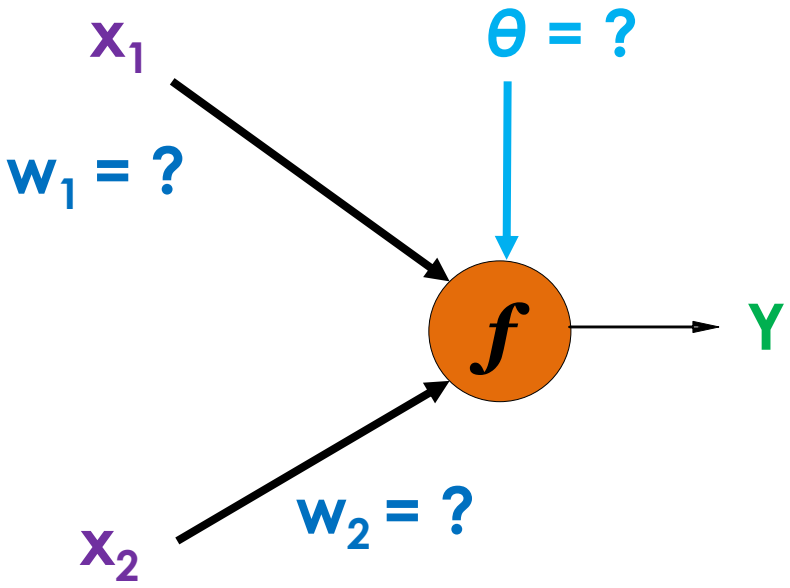
Learning – previous example

$$\Delta w_i(p) = (Y^d - Y)x_i \qquad \Delta\theta(p) = (Y^d - Y)$$

$$w_i(p+1) = w_i(p) + \Delta w_i(p) \qquad \theta(p+1) = \theta(p) + \Delta\theta(p)$$

- **$x_1 = 2$**, $x_2 = 1$
  $w_1 = \mathbf{0.5}$, $w_2 = 0.3$, $\theta = -1$
- The output was calculated as **$Y = 1$**
- However, the true output is **$Y^d = 0$**
- How should I update the weights and threshold?

$w_{1(p+1)} = 0.5 + (0 - 1) * 2 = -1.5$



Learning – previous example

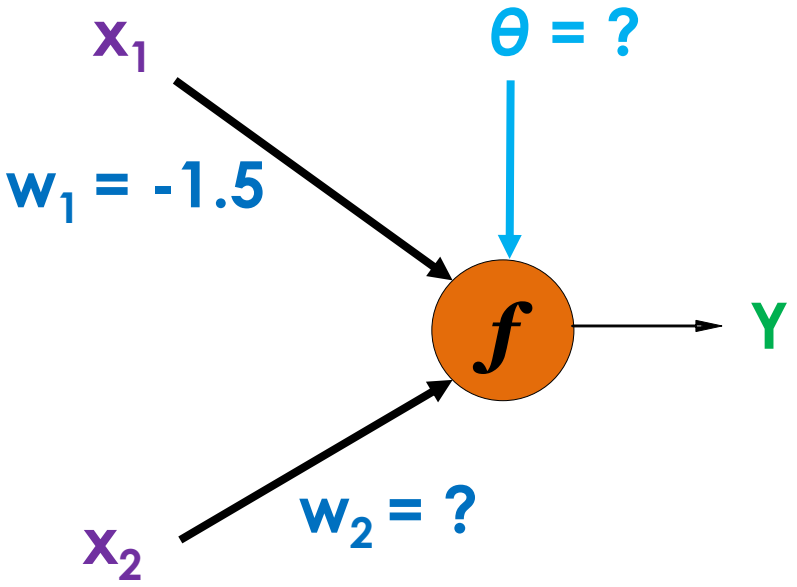
$$\Delta w_i(p) = (Y^d - Y)x_i \qquad \Delta\theta(p) = (Y^d - Y)$$

$$w_i(p+1) = w_i(p) + \Delta w_i(p) \qquad \theta(p+1) = \theta(p) + \Delta\theta(p)$$

- $x_1 = 2$, $x_2 = 1$
  $w_1 = 0.5$, $w_2 = 0.3$, $\theta = -1$
- The output was calculated as $Y = 1$
- However, the true output is $Y^d = 0$
- How should I update the weights and threshold?

$w_{1(p+1)} = 0.5 + (0 - 1) * 2 = -1.5$

So the new w_1 should be -1.5



Learning – previous example

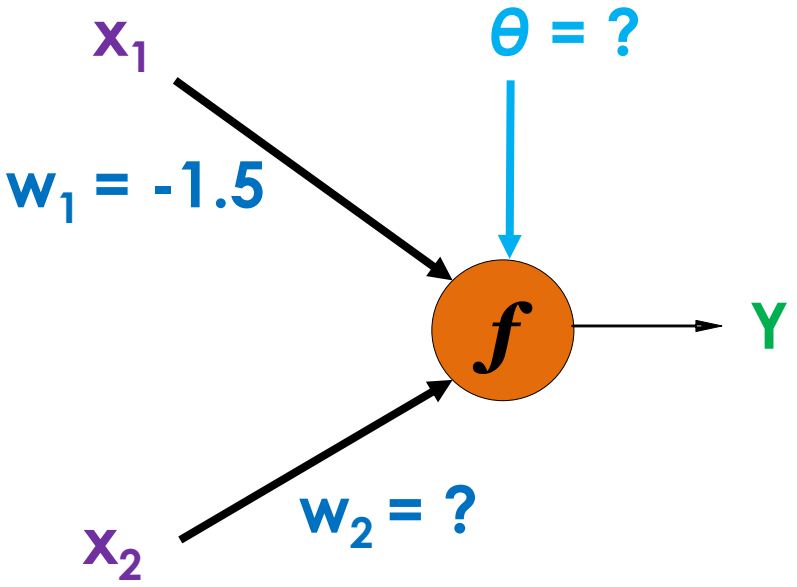
$$\Delta w_i(p) = (Y^d - Y)x_i \qquad \Delta\theta(p) = (Y^d - Y)$$

$$w_i(p+1) = w_i(p) + \Delta w_i(p) \qquad \theta(p+1) = \theta(p) + \Delta\theta(p)$$

- $x_1 = 2$, $x_2 = 1$
  $w_1 = 0.5$, $w_2 = 0.3$, $\theta = -1$
- The output was calculated as $Y = 1$
- However, the true output is $Y^d = 0$
- How should I update the weights and threshold?

$w_{1(p+1)} = 0.5 + (0 - 1) * 2 = -1.5$

Can you calculate the new w_2 ?



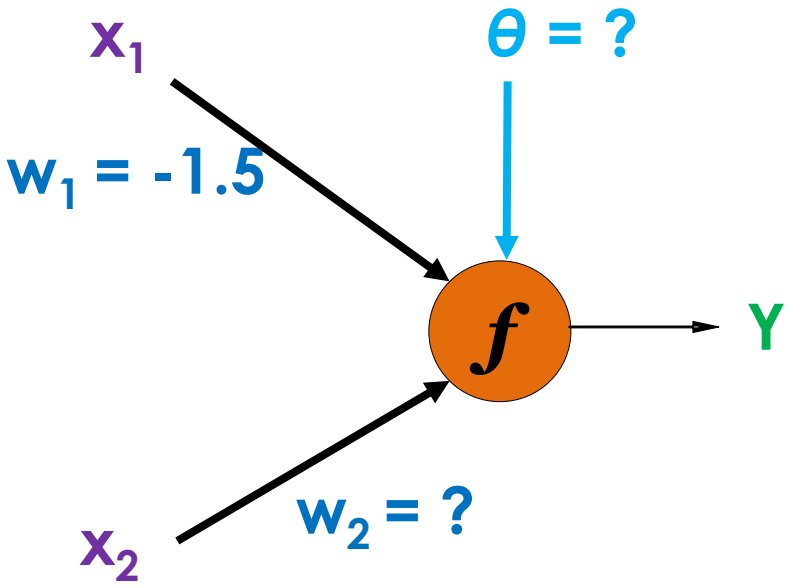
Learning – previous example

$$\Delta w_i(p) = (Y^d - Y)x_i \qquad \Delta\theta(p) = (Y^d - Y)$$

$$w_i(p+1) = w_i(p) + \Delta w_i(p) \qquad \theta(p+1) = \theta(p) + \Delta\theta(p)$$

- $x_1 = 2$, $x_2 = 1$
  $w_1 = 0.5$, $w_2 = 0.3$, $\theta = -1$
- The output was calculated as $Y = 1$
- However, the true output is $Y^d = 0$
- How should I update the weights and threshold?

$w_{2(p+1)} = 0.3 + (0 - 1) * 1 = -0.7$



Learning – previous example

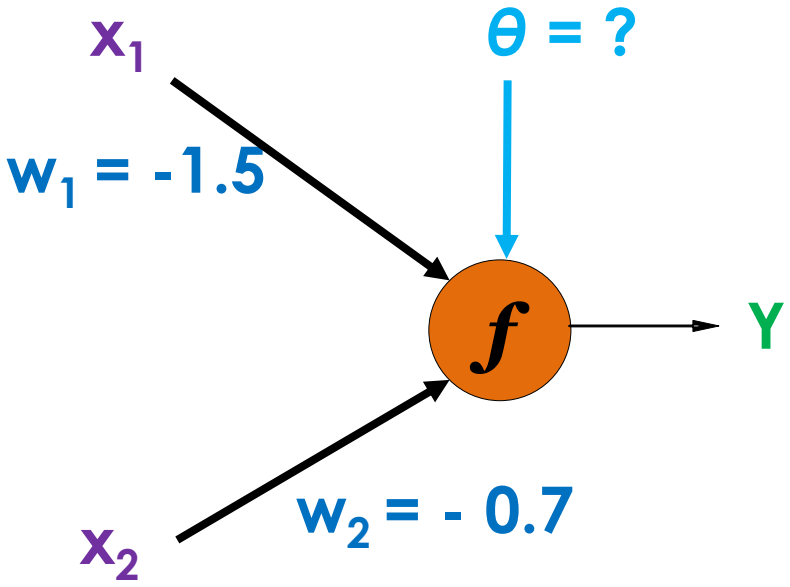
$$\Delta w_i(p) = (Y^d - Y)x_i \qquad \Delta\theta(p) = (Y^d - Y)$$

$$w_i(p+1) = w_i(p) + \Delta w_i(p) \qquad \theta(p+1) = \theta(p) + \Delta\theta(p)$$

- $x_1 = 2$, $x_2 = 1$
  $w_1 = 0.5$, $w_2 = 0.3$, $\theta = -1$
- The output was calculated as $Y = 1$
- However, the true output is $Y^d = 0$
- How should I update the weights and threshold?

$w_{2(p+1)} = 0.3 + (0 - 1) * 1 = -0.7$

So the new w_2 should be - 0.7



Learning – previous example

$$\Delta w_i(p) = (Y^d - Y)x_i \qquad \Delta\theta(p) = (Y^d - Y)$$

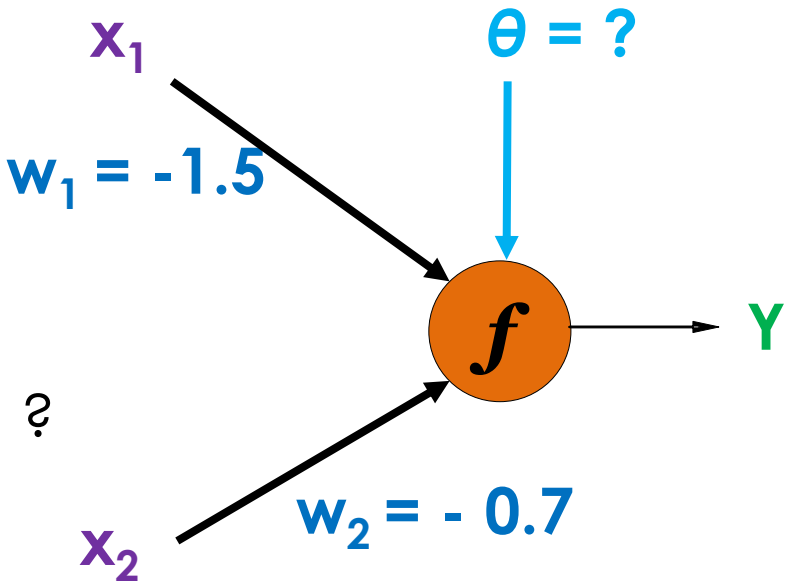
$$w_i(p+1) = w_i(p) + \Delta w_i(p) \qquad \theta(p+1) = \theta(p) + \Delta\theta(p)$$

- $x_1 = 2$, $\mathbf{x_2 = 1}$
  $w_1 = 0.5$, $w_2 = 0.3$, $\theta = -1$
- The output was calculated as **Y = 1**
- However, the true output is $\mathbf{Y^d = 0}$
- How should I update the weights and threshold?

$w_{2(p+1)} = 0.3 + (0 - 1) * 1 = - 0.7$

So the new w_2 should be - 0.7

Can you calculate the new θ ?



Learning – previous example

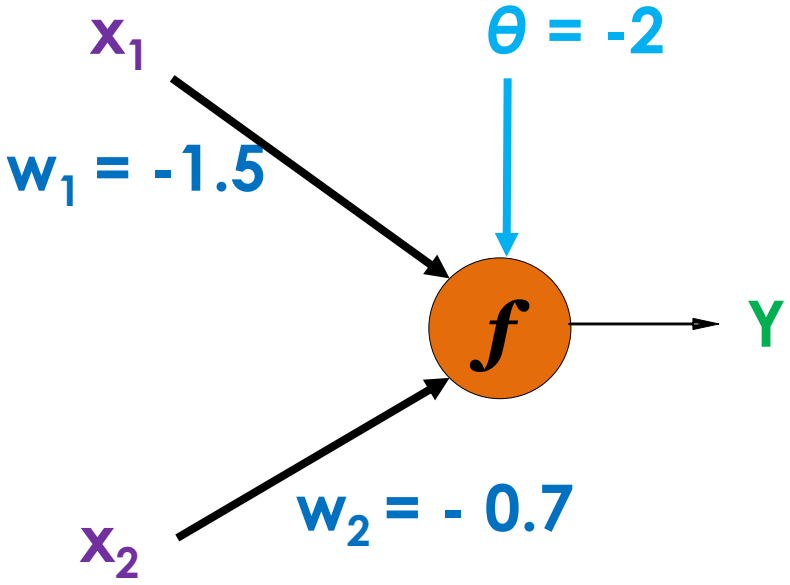
$$\Delta w_i(p) = (Y^d - Y)x_i \qquad \Delta\theta(p) = (Y^d - Y)$$

$$w_i(p+1) = w_i(p) + \Delta w_i(p) \qquad \theta(p+1) = \theta(p) + \Delta\theta(p)$$

- $x_1 = 2$, $x_2 = 1$
  $w_1 = 0.5$, $w_2 = 0.3$, **θ = -1**
- The output was calculated as **Y = 1**
- However, the true output is **Y^d = 0**
- How should I update the weights and threshold?

$\theta_{(p+1)}$ = **-1** + (0 – 1) = - 2

So the new θ should be - 2

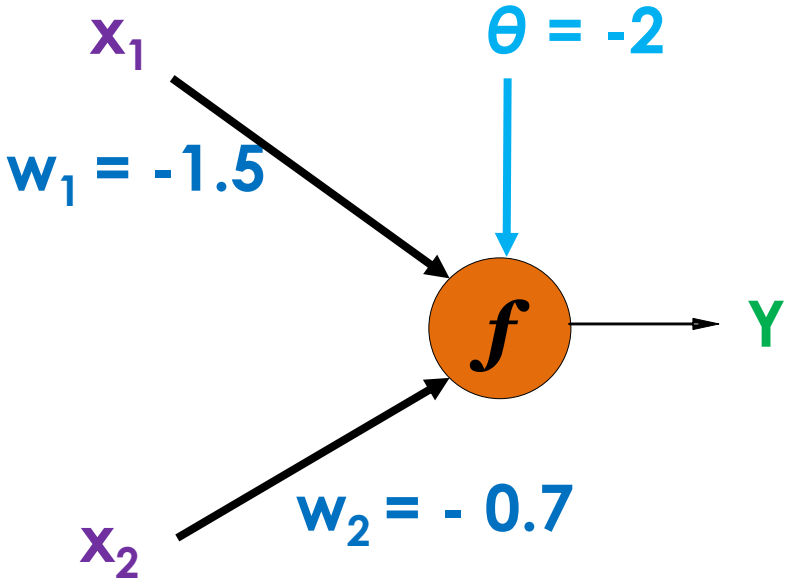


Learning – previous example

$$Y = \begin{cases} +1, if \sum_i w_i x_i + \theta \geq 0 \\ 0, if \sum_i w_i x_i + \theta < 0 \end{cases}$$

- $x_1 = 2$, $x_2 = 1$
  $w_1 = -1.5$, $w_2 = -0.7$, $\theta = -2$
- The output was calculated as **Y = 1**
- However, the true output is **Y^d = 0**
- How should I update the weights and threshold?

What is the output Y now?



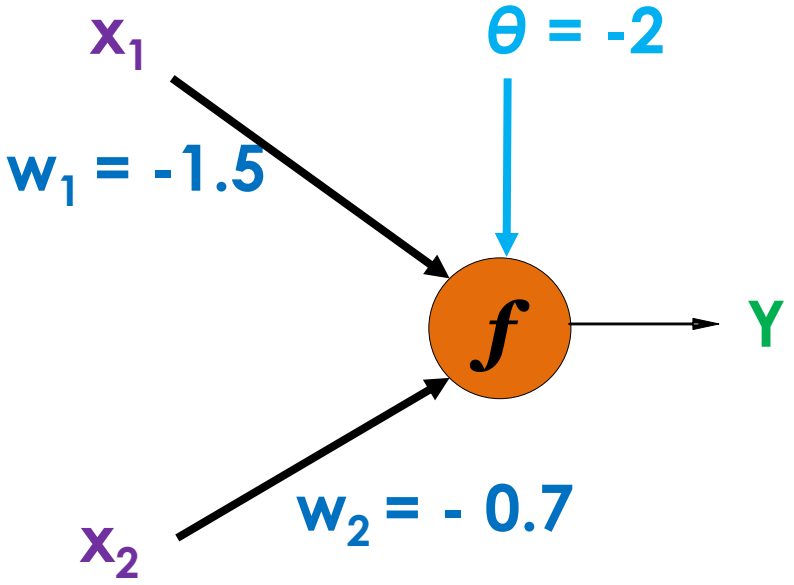
Learning – previous example

$$Y = \begin{cases} +1, if \sum_i w_i x_i + \theta \geq 0 \\ 0, if \sum_i w_i x_i + \theta < 0 \end{cases}$$

- $x_1 = 2, x_2 = 1$
  $w_1 = -1.5, w_2 = -0.7, \theta = -2$
- The output was calculated as **Y = 1**
- However, the true output is **Y^d = 0**
- How should I update the weights and threshold?

What is the output Y now?

(-1.5 * 2) + (- 0.7 * 1) – 2 =
- 3 – 0.7 – 2 = - 5.7
Y = ?



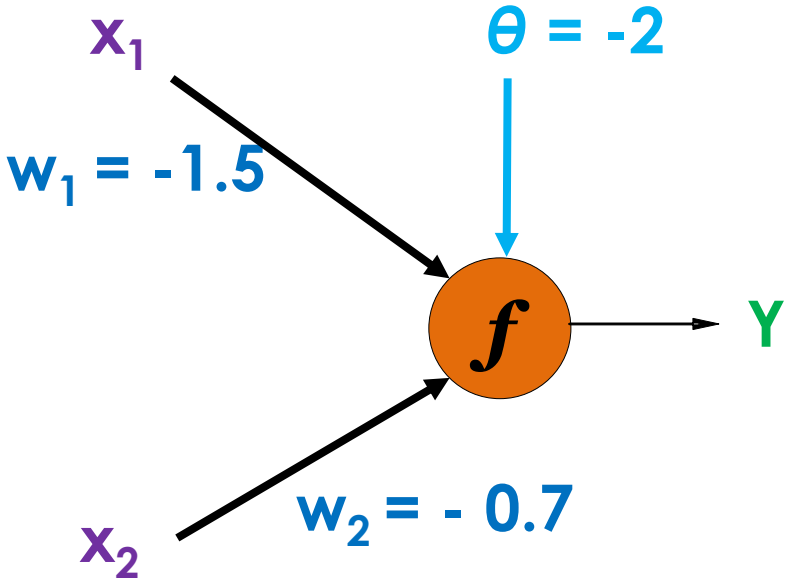
Learning – previous example

$$Y = \begin{cases} +1, if \sum_i w_i x_i + \theta \geq 0 \\ 0, if \sum_i w_i x_i + \theta < 0 \end{cases}$$

- $x_1 = 2, x_2 = 1$
  $w_1 = -1.5, w_2 = -0.7, \Theta = -2$
- The output was calculated as **Y = 1**
- However, the true output is **Y^d = 0**
- How should I update the weights and threshold?

What is the output Y now?

(-1.5 * 2) + (- 0.7 * 1) – 2 =
- 3 – 0.7 – 2 = - 5.7
Y = 0 so we stop

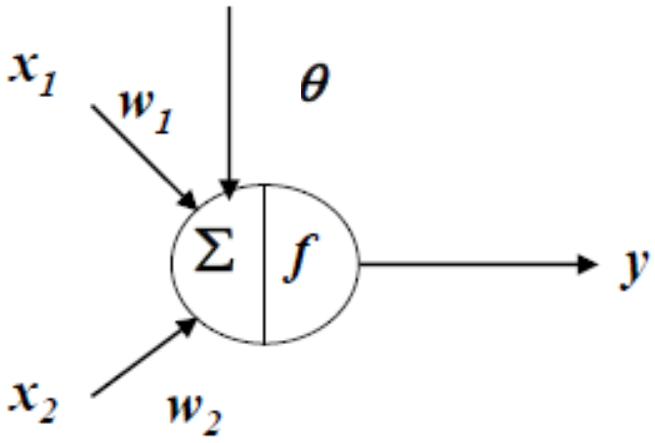


Perceptron example

- Example: logical AND operator
- Weights and threshold:

$w_1 = 1$, $w_2 = 1$, $\theta = -1.5$

- Activation function: Step

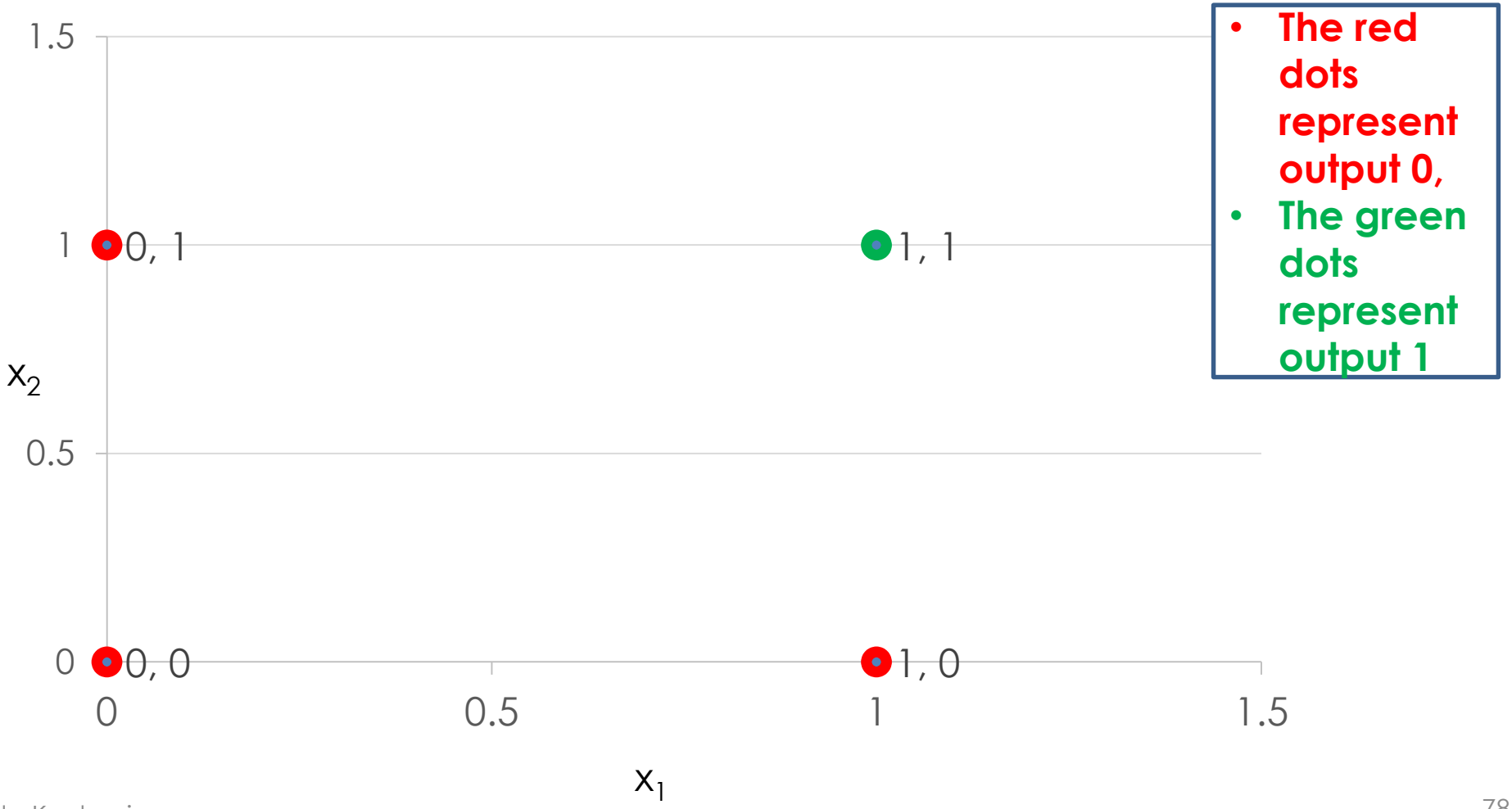


$$Y = \begin{cases} +1, if \sum_i w_i x_i + \theta \geq 0 \\ 0, if \sum_i w_i x_i + \theta < 0 \end{cases}$$

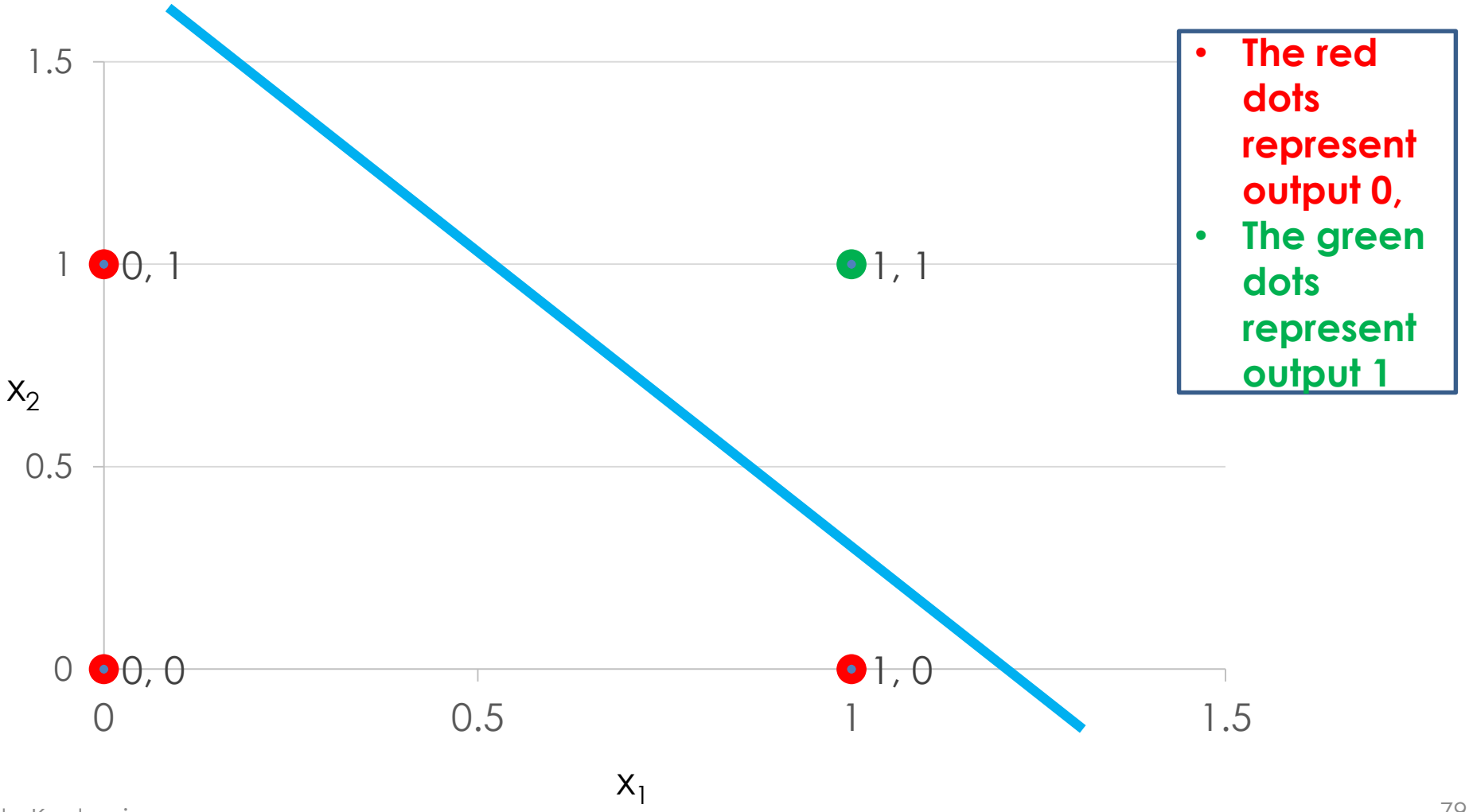
- For inputs 0/1, it correctly **classifies** to 0/1
  - remember the truth-tables

X1	X2	Y
0	0	0
0	1	0
1	0	0
1	1	1

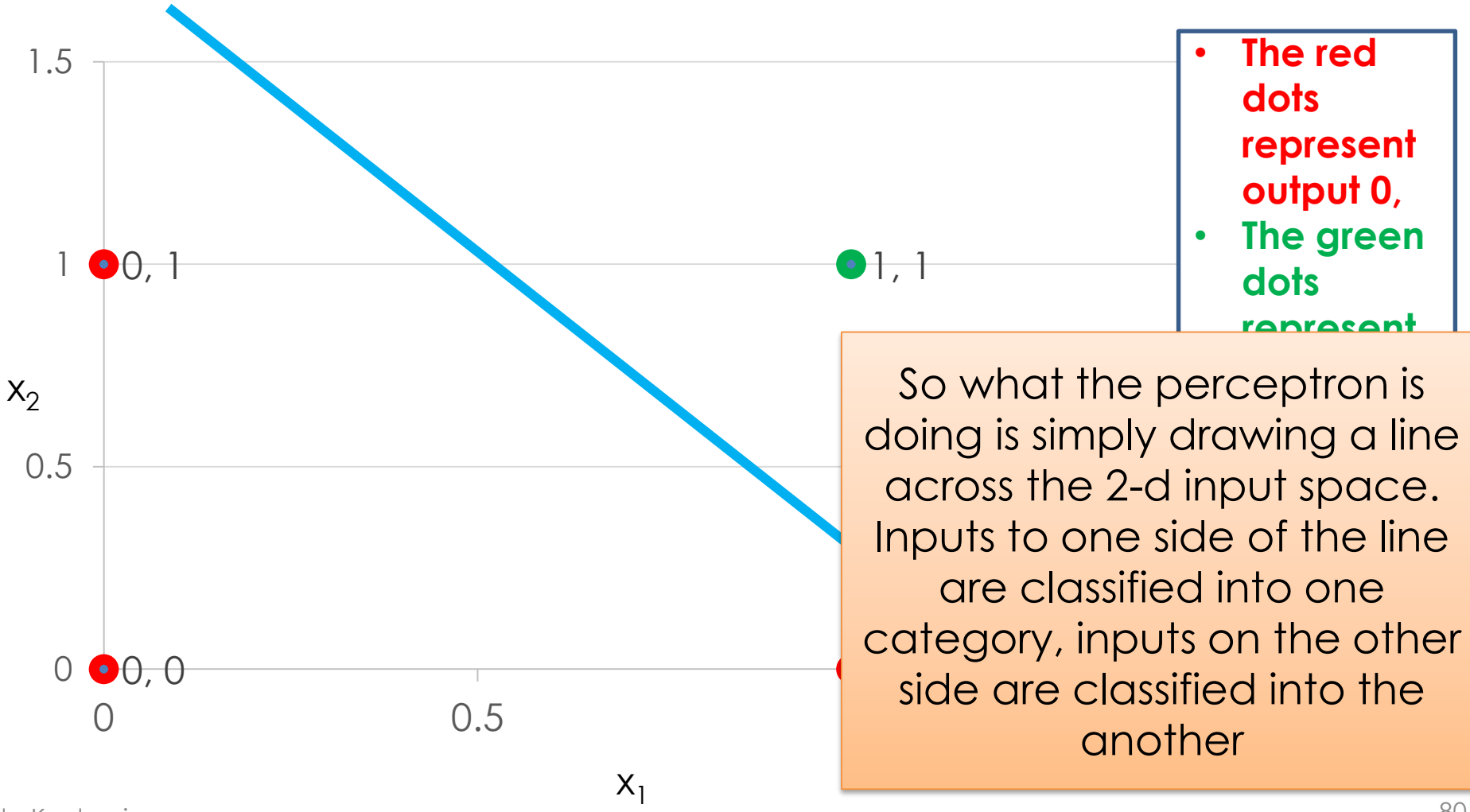
Let's plot these two inputs



Let's separate the outputs



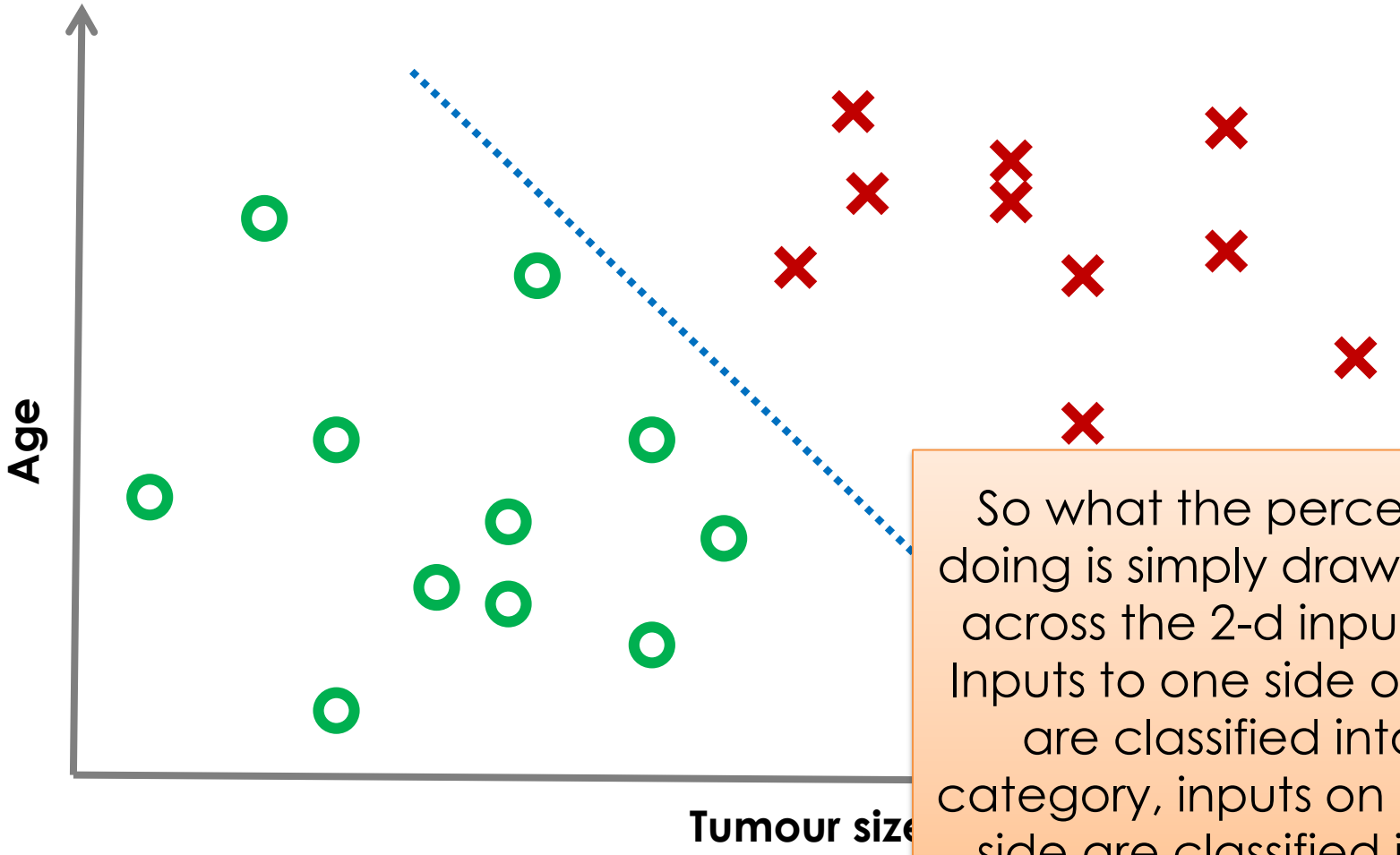
This is an example of linearly separable inputs!



- **The red dots represent output 0,**
- **The green dots represent**

So what the perceptron is doing is simply drawing a line across the 2-d input space. Inputs to one side of the line are classified into one category, inputs on the other side are classified into the another

Linearly separable inputs



So what the perceptron is doing is simply drawing a line across the 2-d input space. Inputs to one side of the line are classified into one category, inputs on the other side are classified into the another

Learning - summary

- Usually the perceptron is initialised with random weights between 0 and 1
- Cases are presented to the perceptron one by one and the weights updated
- If at least one error has been made during this *epoch* then the entire set of cases are presented again
- Repeated until no error is made
- This learning procedure is guaranteed to converge to a solution if the problem is linearly separable

Perceptron Learning Algorithm

1. Initialise weights to random numbers
2. set iteration $p = 1$;
3. Repeat
   4. Calculate activation function
   5. Update weights (and threshold):
   6. $p = p + 1$
7. Until convergence

Neural Networks

- Today we saw the simplest neural network which is used as a binary classifier
- But Neural Networks can be used for all types of Machine Learning
  - Unsupervised, supervised, and reinforcement learning
  - More in 2 weeks (last ML lecture)

Software implementations of ML algorithms

- R *
- Weka *
- ScipY and scikit-learn python libraries *
- Orange *
- MATLAB
- SPSS
- SAS
- STATA
- SQL Server Analysis Services

* Free software/OS licence

Additional resources

Reading:

- Linear regression (and more stats):

https://onlinecourses.science.psu.edu/stat501/node/250

- Perceptrons:
  - Russell and Norvig textbook, section 19.3

Watching:

- Andrew Ng's Stanford Machine Learning lectures.

Brilliant module – it covers everything in ML!