SWINBURNE UNIVERSITY OF TECHNOLOGY

COS30082
Applied Machine Learning

Lecture 2

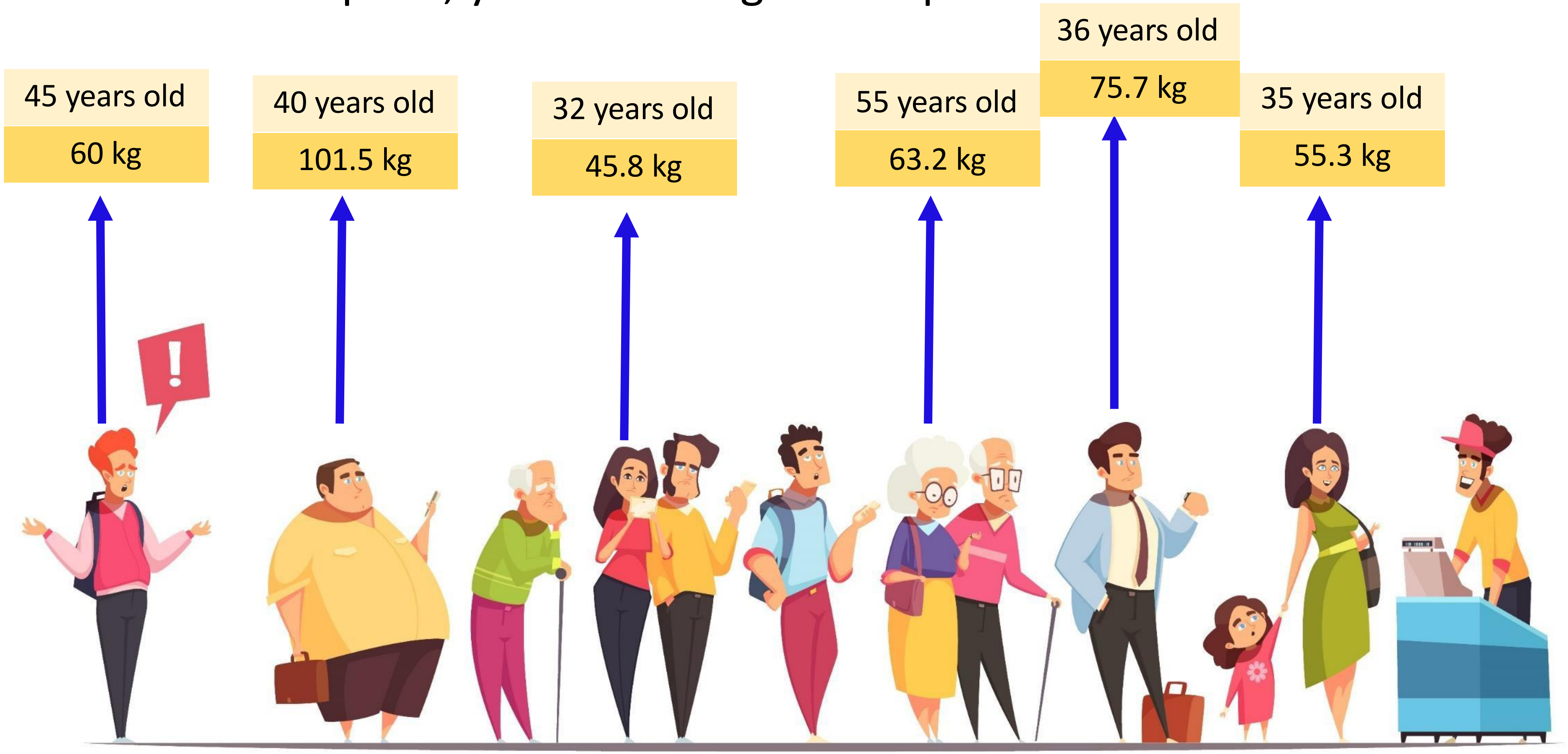
Linear Regression

Topics

- Regression Problems and Linear Regression
- Mathematical Foundations and Optimisation Techniques
- Variants of Linear Regression
- Overfitting and Regularisation Techniques

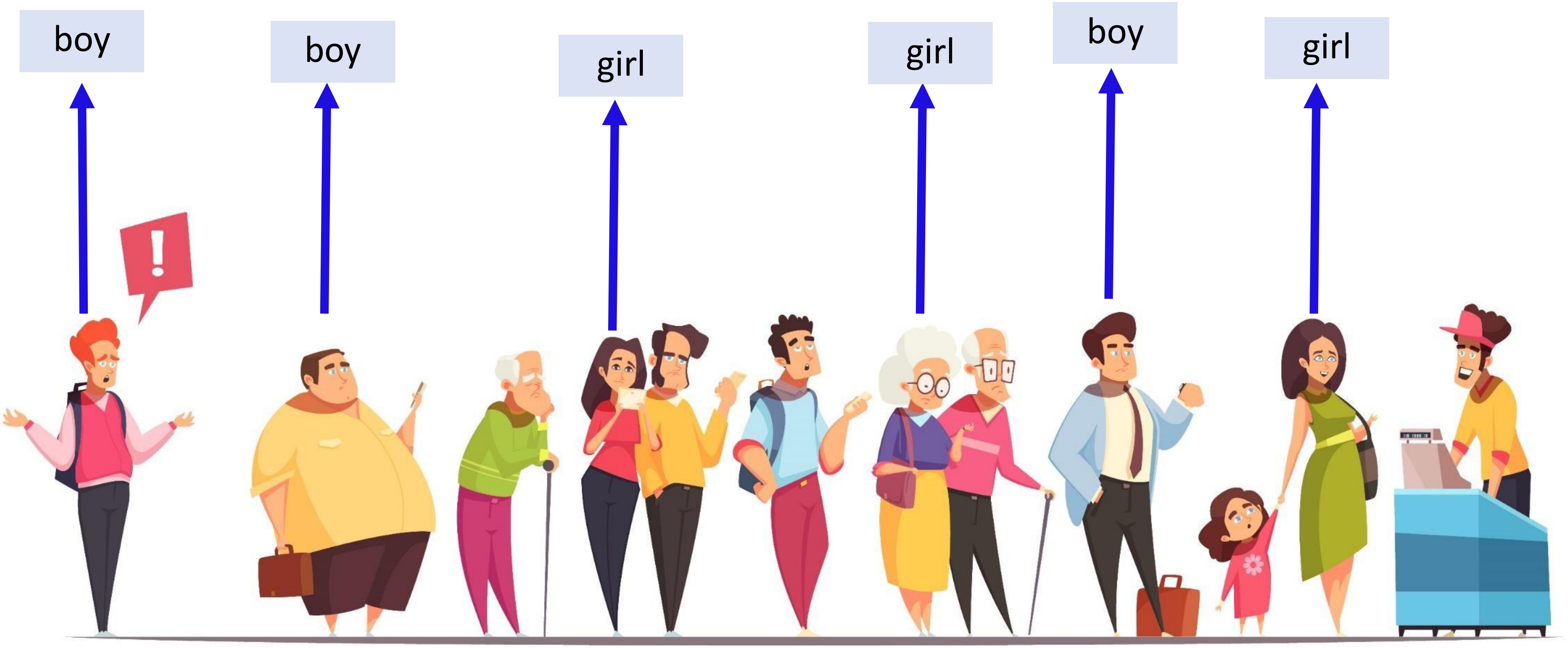
What is a Regression problem?

- A regression problem occurs when the target output is a real or continuous value. For example, when you aim to predict a person's age or weight or estimate a house price, you have a regression problem.



What is a Regression Problem?

- If you classify something into categories, such as predicting gender, that is not a regression problem.



Linear Regression

- Linear regression is a statistical method used to model the relationship between a dependent variable (**Y**) and one or more independent variables (**X**), assuming a linear relationship.

- For example, a company wants to predict sales revenue (Y) based on the advertising budget (X). Linear regression may model this relationship as:

$$Y = 100 + 1.5 X$$

  This equation indicates a linear relationship between Y and X, for every $1 increase in the advertising budget, sales revenue increases by $1.50.

Linear Regression

- **Simple linear regression** refers to a method used when dealing with a single **independent** variable (x).

- **Multiple linear regression** refers to a method used when dealing with multiple independent variables (x1, x2, x3,...).

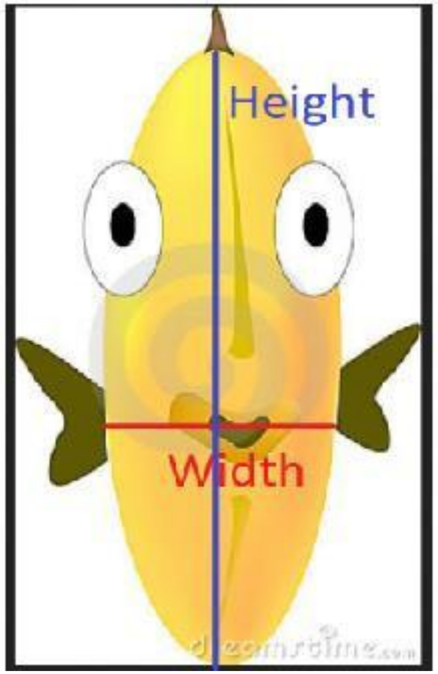
For example:

- If you predict a person's weight based on their height, this is simple linear regression because there is only one independent variable.
- If you predict a person's weight based on their height and age, this is multiple linear regression because there are two independent variables.

Linear Regression

- For example, from a series of N training set, we want to model the relationship between the *height* and *width* of sea bream (a type of fish species).

Height	Width
11.52	4.02
12.48	4.3056
12.3778	4.6961
12.73	4.4555
12.444	5.134
⋮	⋮
⋮	⋮



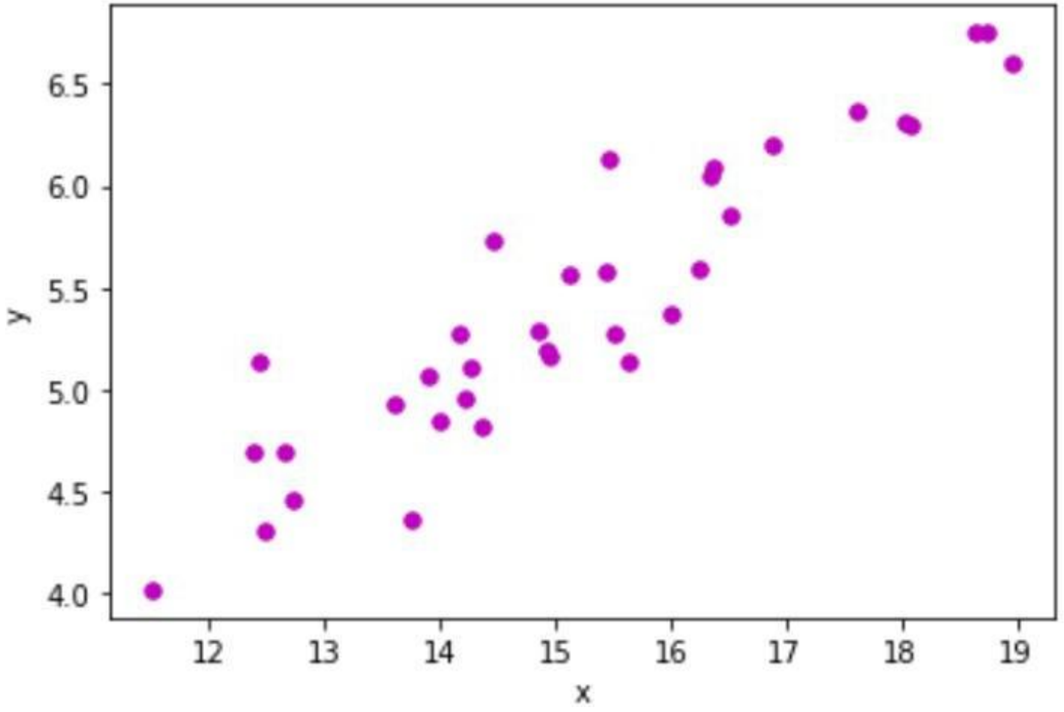
Source from [1]

[1] Aung Pyae, 2019, Fish Market database of common fish species for fish market, Kaggle, viewed 22 April 2020,<https://www.kaggle.com/aungpyaeap/fish-market>

Simple linear regression

- We assign *height* as the independent variable x, and *width as the* dependent variable, y .
- If we plot the data, we can see a positive relationship between the two variables.

Height, x	Width, y
11.52	4.02
12.48	4.3056
12.3778	4.6961
12.73	4.4555
12.444	5.134
⋮	⋮
⋮	⋮

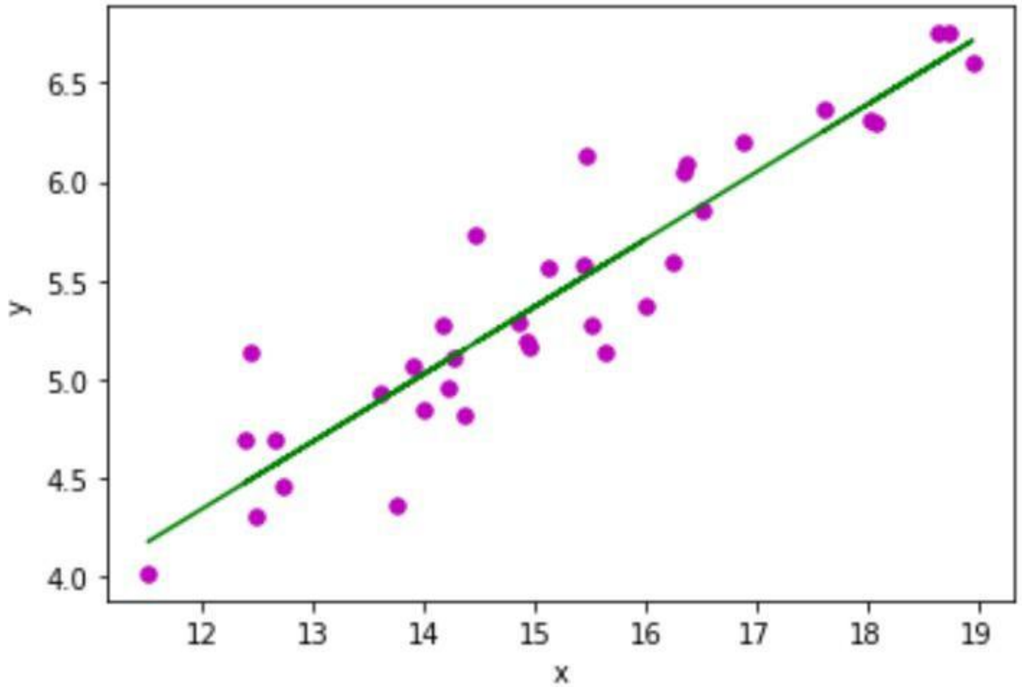


Simple linear regression

- The relationship between the two variables x and y can be modeled by a linear equation:

$$y = \theta_0 + \theta_1 x$$

given θ_0 is the bias (intercept) , θ_1 is the weight (coeficient) associated to x



Simple linear regression

$$y = \theta_0 + \theta_1 x$$

- Imagine if you fit a linear model and have the solutions for θ_0 and θ_1,

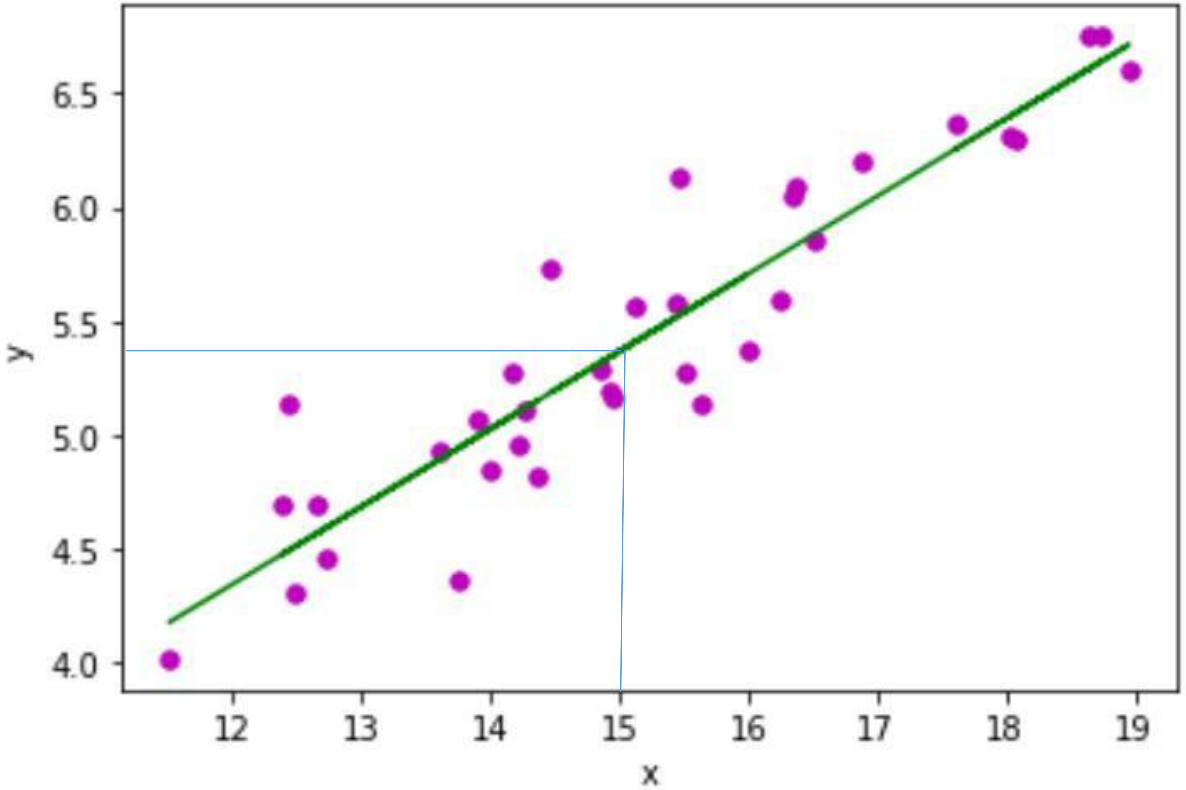
θ_0 = 0.261

θ_1 = 0.340

If x = 15,

$y =$ 0.261 + 0.340 (15)

= 5.361



Topics

- Regression Problems and Linear Regression
- Mathematical Foundations and Optimisation Techniques
- Variants of Linear Regression
- Overfitting and Regularisation Techniques

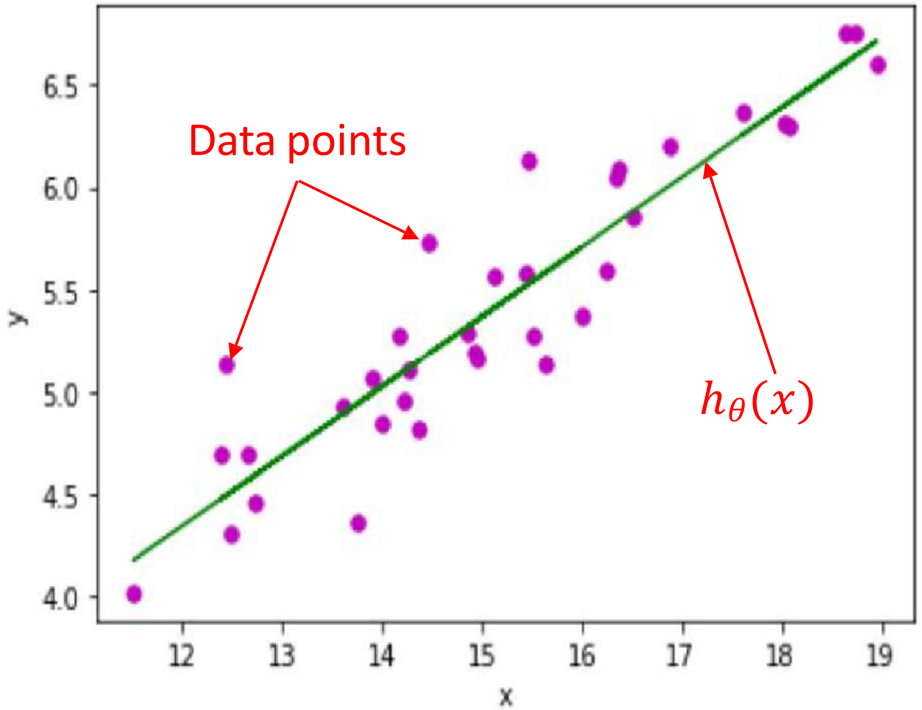
Finding Parameters in Linear Regression

$$h_\theta(x) = \theta_0 + \theta_1 x$$

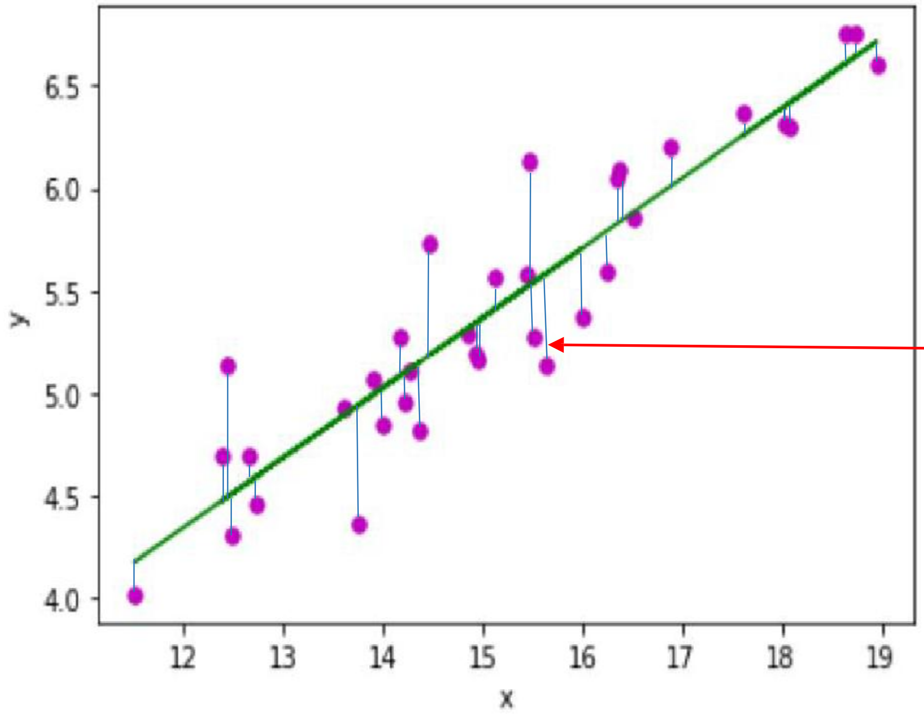
- The method used to determine the parameters in a linear regression equation is **Ordinary Least Squares (OLS)**, which minimizes the **sum of squared errors** between actual and predicted values.
- To solve for the optimal parameters, we can use either of the following approaches:
  - **Normal Equation**, which provides a **direct solution** using matrix operations.
  - **Gradient Descent**, an **iterative optimisation process** that updates parameters step by step.

Ordinary Least Square

- Given *N* number of data points (training set) = $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots\ldots, (x^{(N)}, y^{(N)})$
- Goal: To find a linear model ($h_\theta(x) = \theta_0 + \theta_1 x$) that best fit all the data.



Ordinary Least Square



$|y^{(n)} - h_\theta(x^{(n)})|$

- Cost function

$$J(\theta_0, \theta_1) = \Sigma_{n=1}^{N} \left(y^{(n)} - h_\theta(x^{(n)})\right)^2 = \Sigma_{n=1}^{N} \left(y^{(n)} - (\theta_0 + \theta_1 x^{(n)})\right)^2$$

Called least square cost function

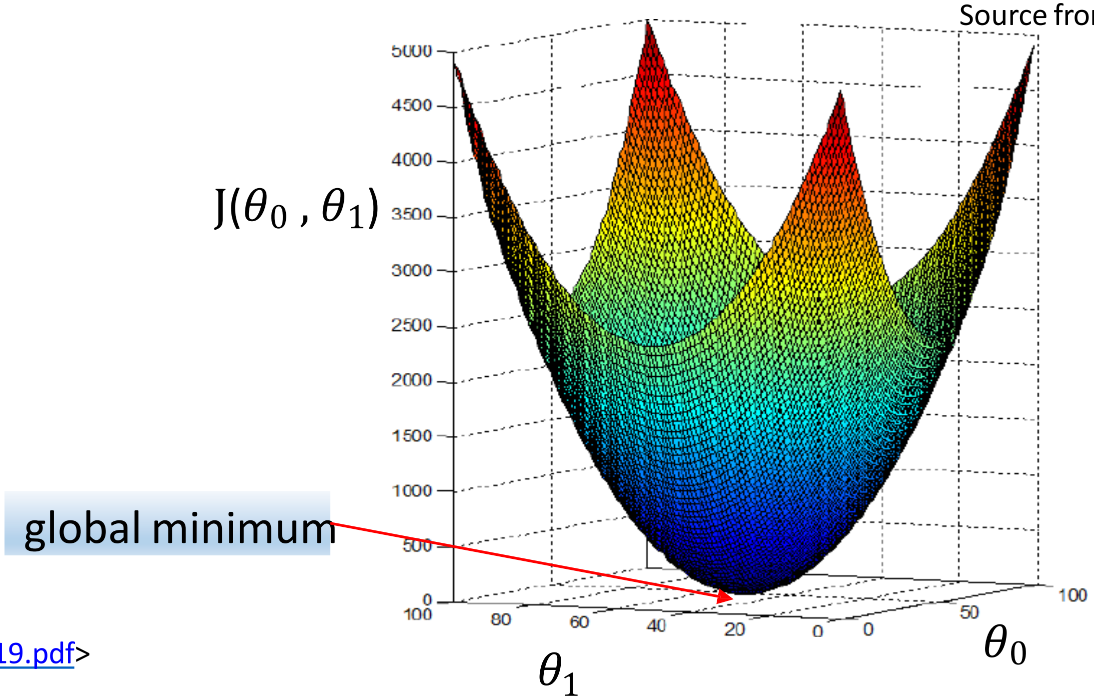
Ordinary Least Square

$$J(\theta_0, \theta_1) = \Sigma_{n=1}^{N} \left(y^{(n)} - (\theta_0 + \theta_1 x^{(n)})\right)^2$$

Characteristics of the Least Squares Cost Function:

- The Least Squares cost function is a convex function.
- A convex function has a single global minimum.

Source from [2]



[2] Optimization and Least Square, accessed 28 April 2020, <http://www.cs.cornell.edu/courses/cs1114/2009sp/lectures/CS1114-lec19.pdf>

Ordinary Least Square

- Method of least square directly computes the optimal choice of (θ_0 , θ_1) at the global minimum

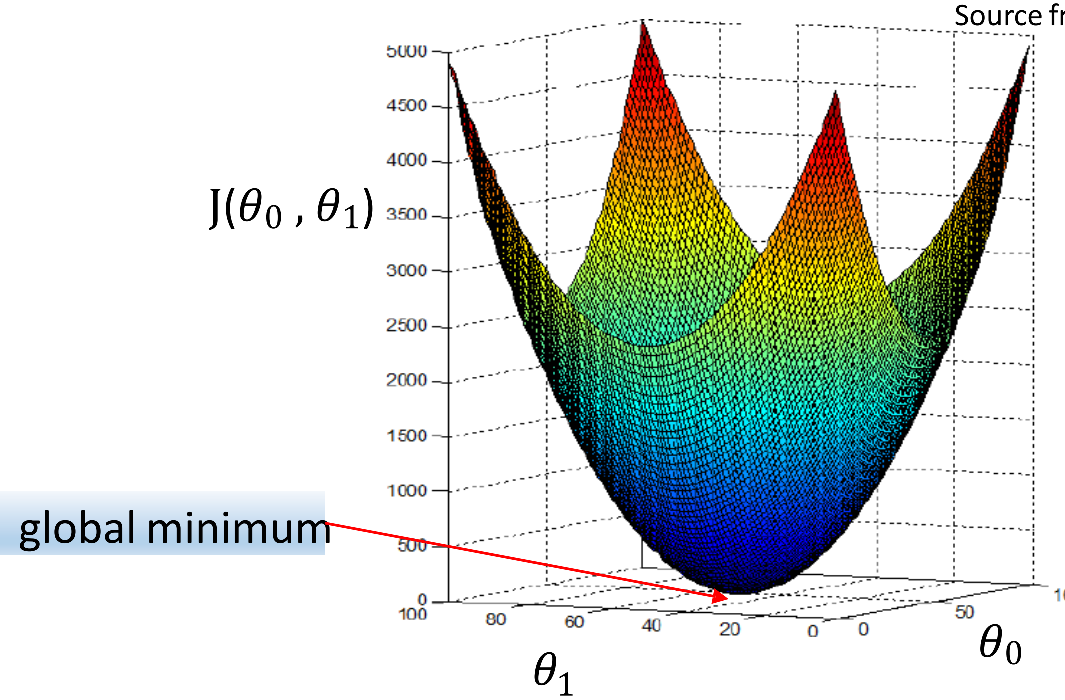
$$argmin_{\theta_0 , \theta_1} \; J(\theta_0 , \theta_1)$$

setting:

$$\frac{\partial J}{\partial \theta_0} = 0$$

$$\frac{\partial J}{\partial \theta_1} = 0$$

Source from [2]



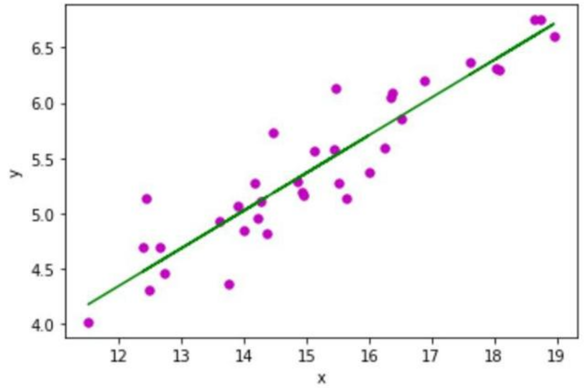
Ordinary Least Square

$$J(\theta_0, \theta_1) = \Sigma_{n=1}^{N} \left(y^{(n)} - h_\theta(x^{(n)})\right)^2 = \Sigma_{n=1}^{N} \left(y^{(n)} - (\theta_0 + \theta_1 x^{(n)})\right)^2$$

Differentiating wrt. θ_0 : $\frac{\partial J}{\partial \theta_0} = \sum_{n=1}^{N} 2\left(y^{(n)} - (\theta_0 + \theta_1 x^{(n)})\right) = 0$

Differentiating wrt. θ_1 : $\frac{\partial J}{\partial \theta_1} = \sum_{n=1}^{N} 2\left(y^{(n)} - (\theta_0 + \theta_1 x^{(n)})\right) . (-x^{(n)}) = 0$

$$\Rightarrow \quad \theta_0 = \frac{1}{N}\left[\Sigma_{n=1}^{N} y^{(n)} - \theta_1 \Sigma_{n=1}^{N} x^{(n)}\right]$$



$$\Rightarrow \quad \theta_1 = \frac{N \sum_{n=1}^{N} x^{(n)} y^{(n)} - \left(\sum_{n=1}^{N} x^{(n)}\right)\left(\sum_{n=1}^{N} y^{(n)}\right)}{N \sum_{n=1}^{N} (x^{(n)})^2 - \left(\sum_{n=1}^{N} x^{(n)}\right)^2}$$

Normal Equation

Based on this idea, there are two main methods to solve for the optimal parameters in linear regression:

- **Normal Equation** – A **direct method** that uses algebra to solve for the parameters.
- **Gradient Descent** – An **iterative method** that gradually adjusts the parameters until it reaches the minimum.

Normal Equation

- The Normal Equation is a direct method to find the optimal parameters in linear regression without iteration.
- It is derived by setting the derivative of the cost function to zero and solving for the parameters algebraically.
- This method works well for small datasets but becomes computationally expensive for large datasets due to matrix inversion.
- The equation is:

  - $\theta = (X^T X)^{-1} X^T y$

- This formula gives the best-fit parameters in one step.

Normal Equation

- For example, to optimise two parameters θ_0 and θ_1 :

$$J(\theta_0, \theta_1) = \sum_{n=1}^{N} (y^{(n)} - (\theta_0 + \theta_1 x^{(n)}))^2$$

Substitute X and Y into: $\theta = (X^T X)^{-1} X^T y$

where

$$X = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} \\ x_0^{(2)} & x_1^{(2)} \\ x_0^{(3)} & x_1^{(3)} \\ \vdots & \end{bmatrix}_{(N \times 2)} \qquad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \vdots \end{bmatrix}_{(N \times 1)}$$

Number of training data

Number of parameters

Gradient Descent

- Gradient Descent is an iterative method used to find the optimal parameters in linear regression by minimizing the cost function.
- It starts with initial parameter values and gradually updates them in the direction of the steepest decrease, guided by the derivative (gradient).
- The process repeats until it reaches the global minimum.
- This method is efficient for large datasets.
- It requires choosing a learning rate, which controls the step size for updates.

Gradient Descent

- Hypothesis $h_\theta(x)$ is represented as:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

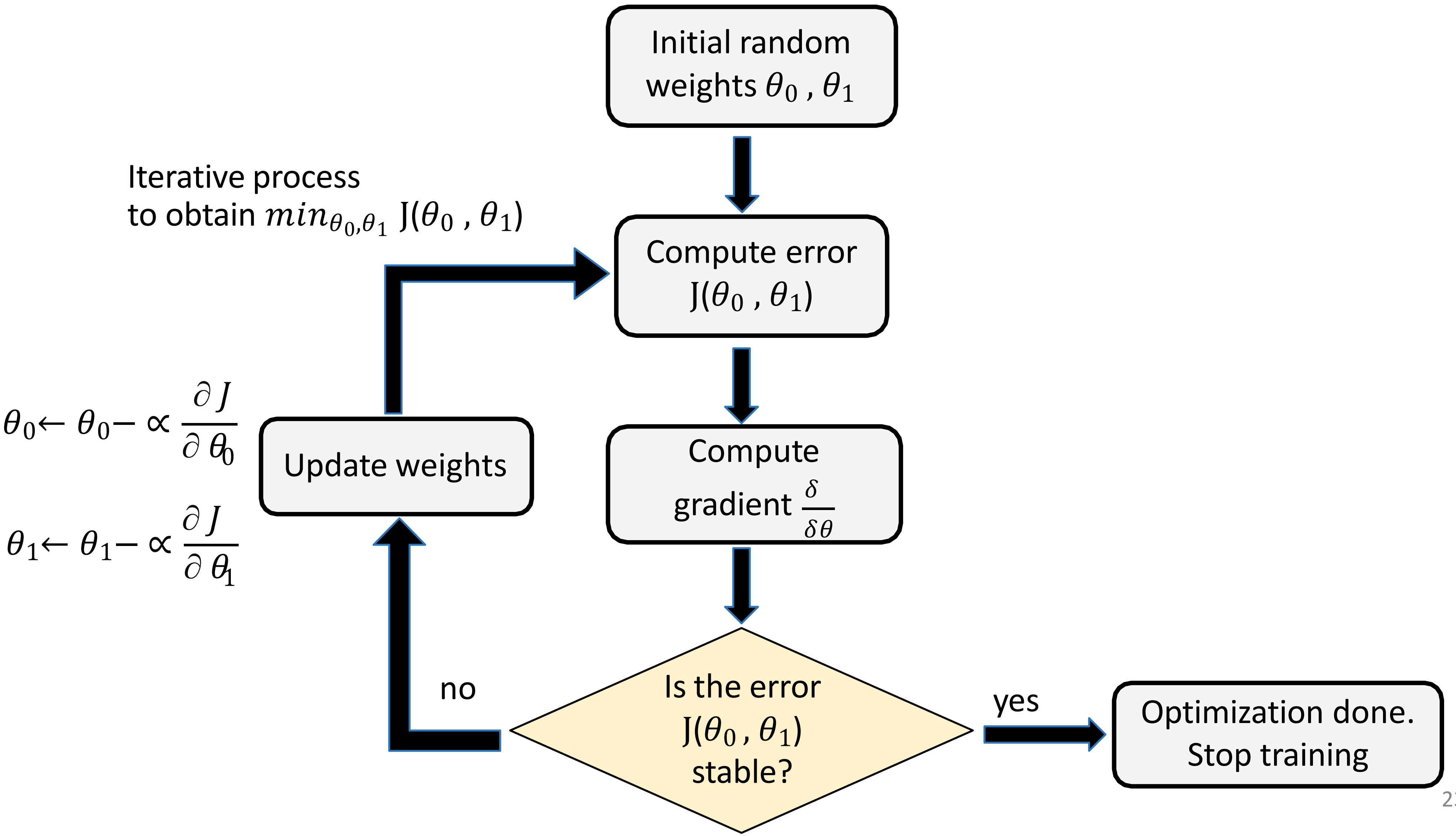
- Parameters: θ_0 and θ_1
- Cost function:

$$J(\theta_0, \theta_1) = \frac{1}{2N} \sum_{n=1}^{N} (h_\theta(x^{(n)}) - y^{(n)})^2$$

- Goal:

$$minimize \quad J(\theta_0, \theta_1)$$

How Gradient Descent works



How the Gradient Descent works

- Gradient descent update weights simultaneously:

Repeat

$$\theta_0 \leftarrow \theta_0 - \alpha \frac{\partial J}{\partial \theta_0}$$

$$\theta_1 \leftarrow \theta_1 - \alpha \frac{\partial J}{\partial \theta_1}$$

[1] $tmp_0 \leftarrow \theta_0 - \alpha \frac{\partial J}{\partial \theta_0}$

[2] $tmp_1 \leftarrow \theta_1 - \propto \frac{\partial J}{\partial \theta_1}$

[3] $\theta_0 \leftarrow tmp_0$

[4] $\theta_1 \leftarrow tmp_1$

Not in this way:

[1] $tmp_0 \leftarrow \theta_0 - \alpha \frac{\partial J}{\partial \theta_0}$

[2] $\theta_0 \leftarrow tmp_0$

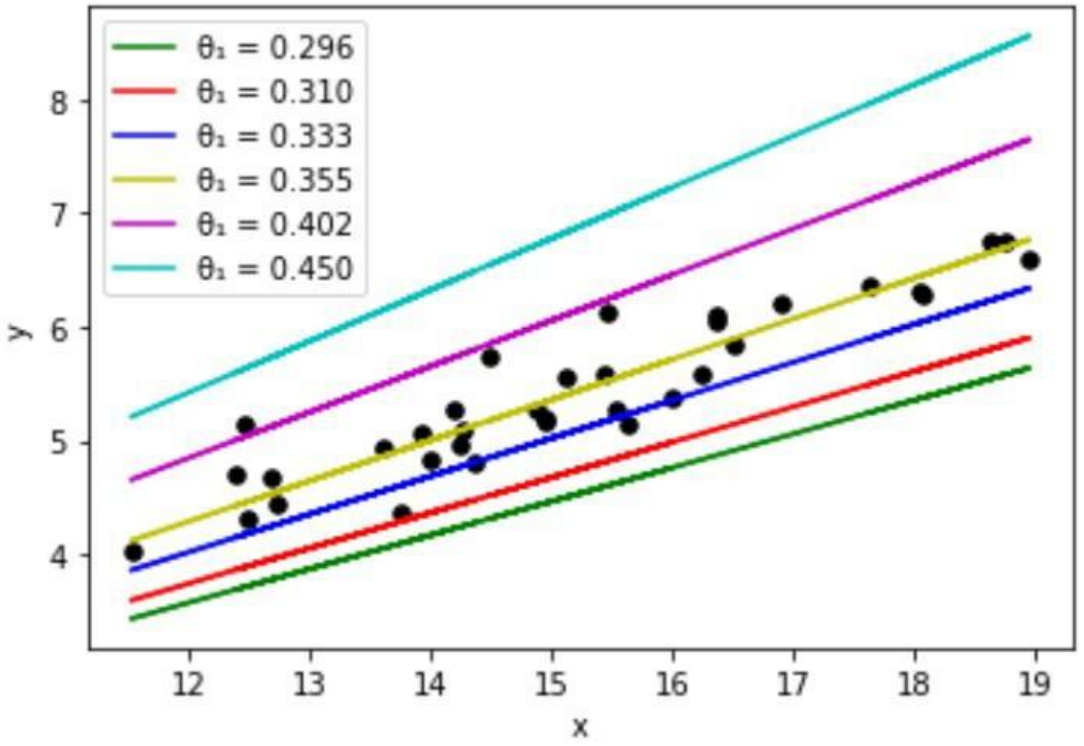
[3] $tmp_1 \leftarrow \theta_1 - \alpha \frac{\partial J}{\partial \theta_1}$

[4] $\theta_1 \leftarrow tmp_1$

How the Gradient Descent works

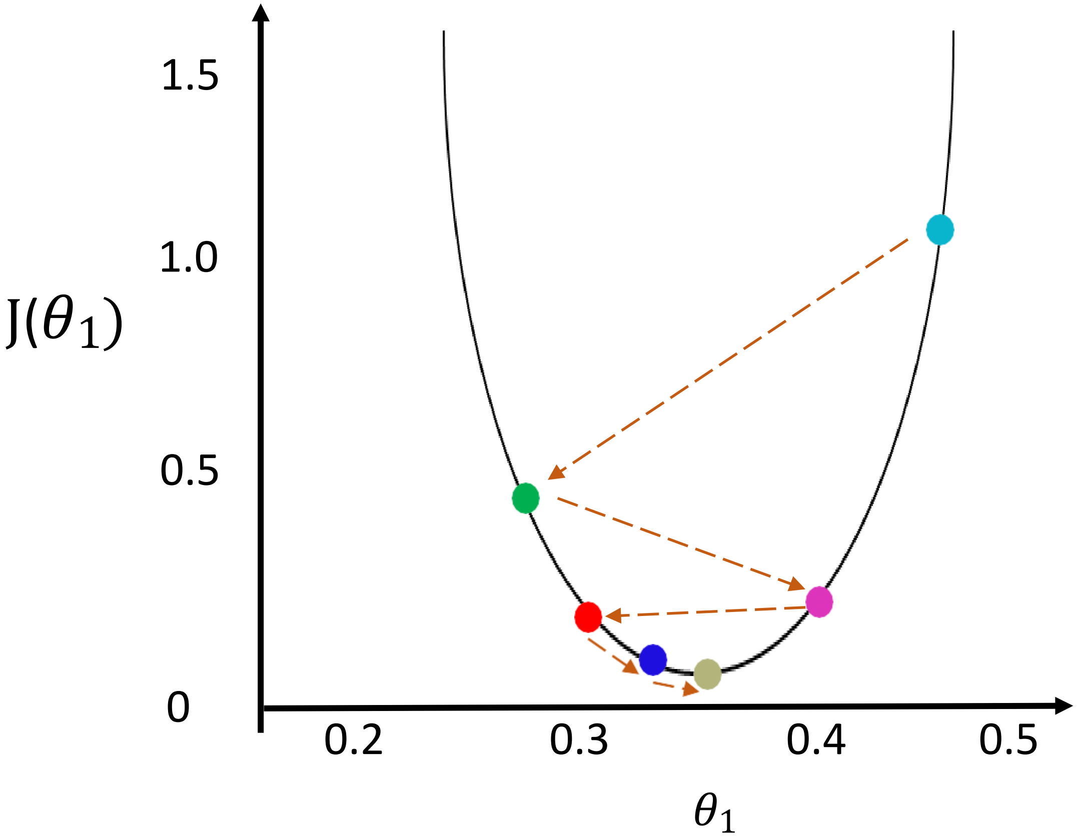
$h_\theta(x) = \theta_0 + \theta_1 x$

$h_\theta(x)$ for fixed θ_0

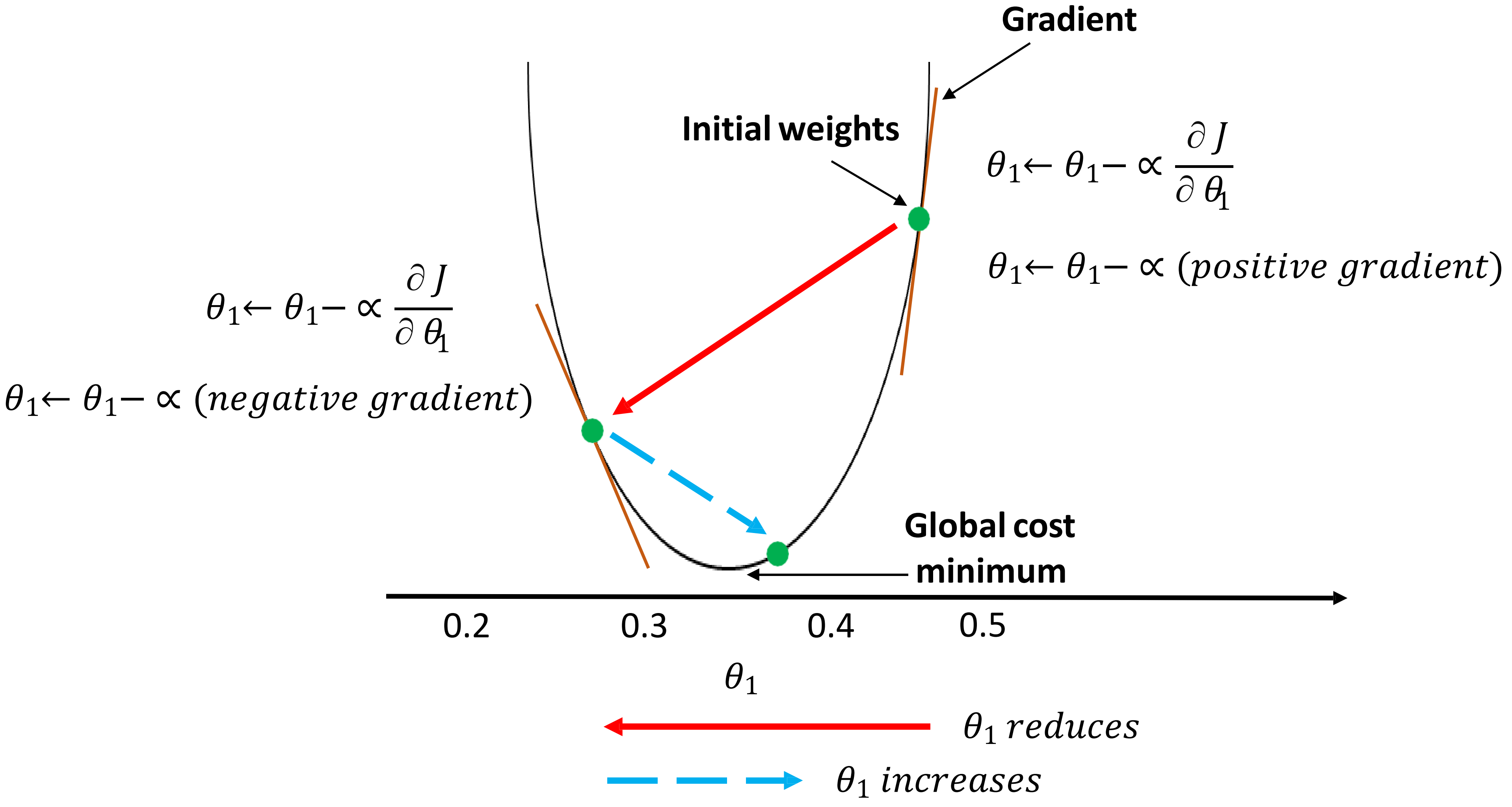


Iterative process to obtain $min_{\theta_1} \mathrm{J}(\theta_1)$

$$\theta_1 \leftarrow \theta_1 - \propto \frac{\partial J}{\partial \theta_1}$$

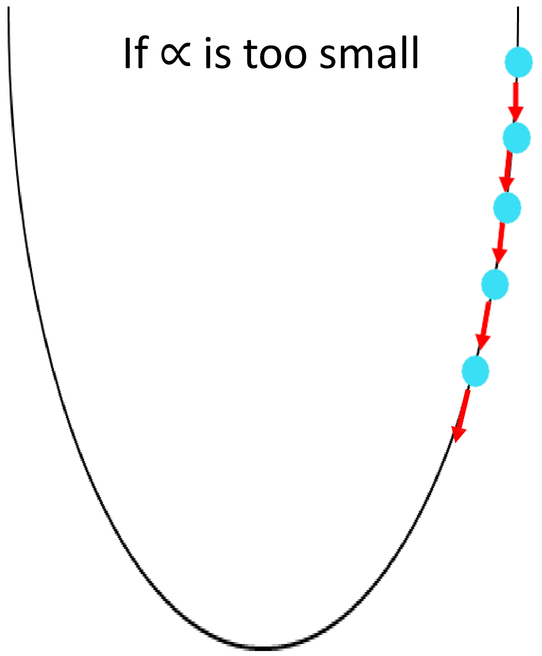


How the Gradient Descent works

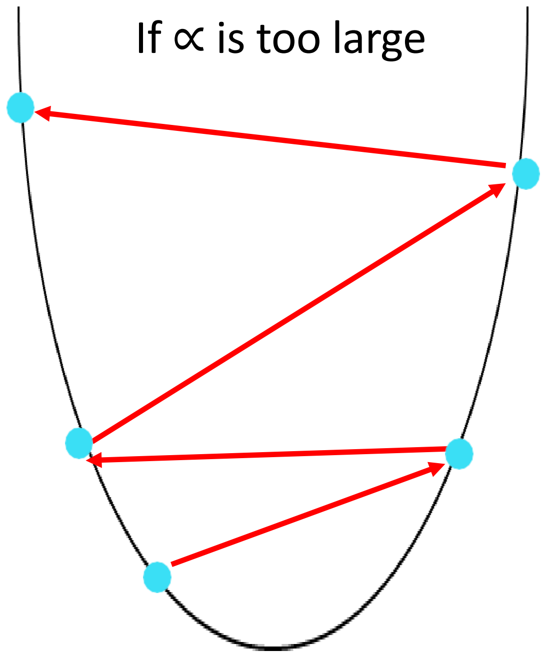


How the learning rate effects the learning

$$\theta_1 \leftarrow \theta_1 - \propto \frac{\partial J}{\partial \theta_1}$$



Too long time to take to converge



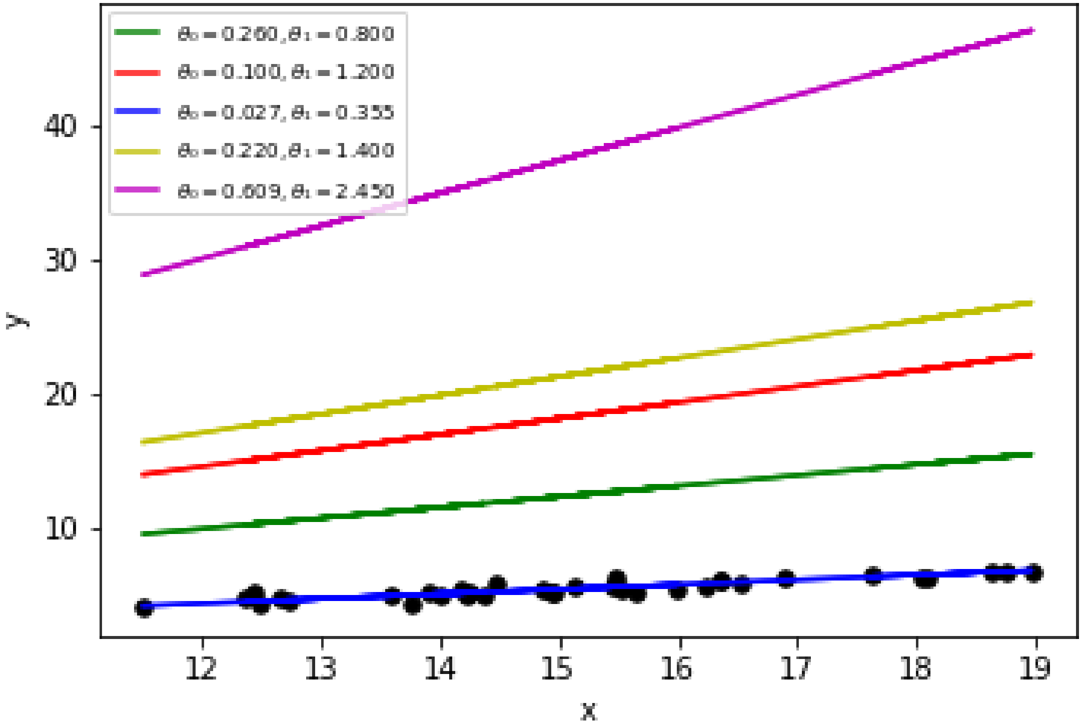
It may fail to converge, causing gradient explode (loss increases)

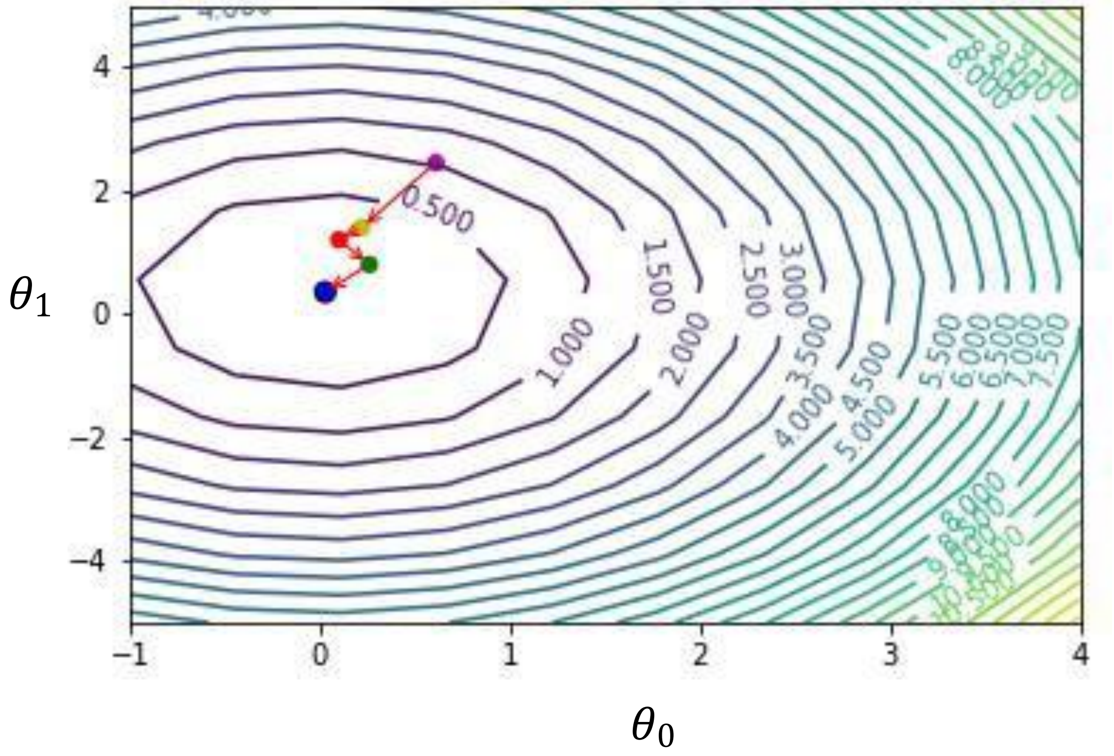
Gradient Descent to optimize $\boldsymbol{\theta_1}$ and $\boldsymbol{\theta_0}$

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Iterative process to obtain min_{θ_1} $J(\theta_1)$

$$\theta_0 \leftarrow \theta_0 - \propto \frac{\partial J}{\partial \theta_0} \qquad \theta_1 \leftarrow \theta_1 - \propto \frac{\partial J}{\partial \theta_1}$$





Normal equation vs Gradient descent

Normal Equation

- Pros
  - No need to adjust learning rate.
  - No iterative training.
- Cons

$$(X^T X)^{-1}$$

  - Computational expensive when number of parameters learned is too large (a hundred ~ ten thousand).
  - It is possible that $(X^T X)^{-1}$ is non-reversible if there are **redundant features** or **too many parameters.**

Gradient Descent

- Cons
  - Need to adjust learning rate.
  - Need iterative training.
- Pros
  - Still works efficiently when number of parameters learned is very large.

Topics

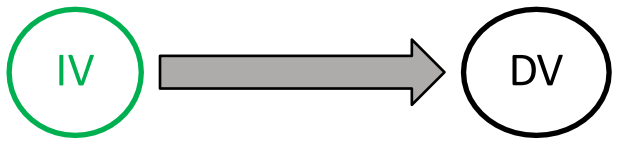
- Regression Problems and Linear Regression
- Mathematical Foundations and Optimisation Techniques
- Variants of Linear Regression
- Overfitting and Regularisation Techniques

Multiple linear regression

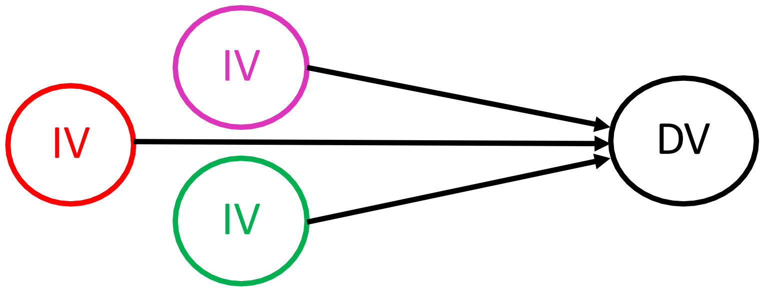
- Multiple linear regression (MLR) is a method used to model the linear relationship between **two or more** independent variables and a dependent variable.
    - The price of a house is corelated to its size in square feet and the number of bedrooms.
- MLR based on the assumption that the independent variables are not too highly correlated with each other.
    - The size and number of bedrooms are not highly correlated.

Simple vs. Multiple linear regression

- Simple linear regression - one-to-one



- Multiple linear regression - many-to-one



where,
IV: independent variables
DV: dependent variable

Multiple linear regression

- For example, from a series of N training set, to model the relationship between the *height* and *width* of sea bream.
- Hypothesis of a simple linear regression is represented as:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

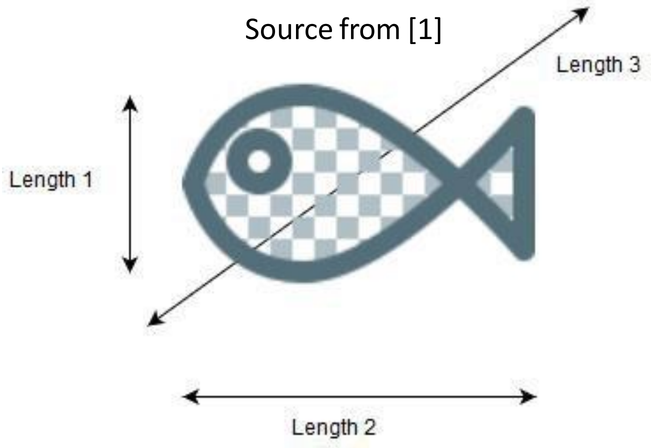
Height, x	Width, y
11.52	4.02
12.48	4.3056
12.3778	4.6961
12.73	4.4555
12.444	5.134
$\vdots$	$\vdots$
$\vdots$	$\vdots$

Multiple linear regression

- Other than *height*, to model the linear relationship between more independent variables such as the *weight*, *body height* and *diagonal length* with the dependent variable, *width*, the hypothesis is represented as follows:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 + \theta_5 x_5$$

Source from [1]



Weight, x_5	**Length 1 (Body Height), x_4**	**Length 2 (Total Length), x_3**	**Length 3 (Diagonal Length), x_2**	Height, x_1	Width, y
242	23.2	25.4	30	11.52	4.02
290	24	26.3	31.2	12.48	4.3056
340	23.9	26.5	31.1	12.3778	4.6961
363	26.3	29	33.5	12.73	4.4555
430	26.5	29	34	12.444	5.134
450	26.8	29.7	34.7	13.6024	4.9274
500	26.8	29.7	34.5	14.1795	5.2785
⋮	⋮	⋮	⋮	⋮	⋮

Multiple linear regression

- Hypothesis $h_\theta(x)$ is represented as:

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_M x_M$$

- Parameters: $\theta_0, \ldots, \theta_M$ Number of parameters = M + 1
- Cost function:

$$J(\theta) = \frac{1}{2N} \sum_{n=1}^{N} \left(h\ (x^{(n)}) - y^{(n)}\right)^2$$

Number of training data = N

- Goal:

$$minimize \quad J(\theta)$$

Multiple linear regression

- Two solutions for the MLR
  - Normal equation
  - Gradient Descents

Normal Equation

$$\theta = (X^T X)^{-1} X^T y$$

where X is a $N \times (M+1)$ matrix
Y is a $N \times 1$ matrix

Gradient Descent

Repeat

$$\{ \quad \boldsymbol{\theta_j} \leftarrow \boldsymbol{\theta_j} - \propto \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{\theta_j}}$$

$$\}$$

θ is updated simultaneously for j = 0,…,M

Polynomial regression

$$h_{\theta}(x) = \theta_0 x^0 + \theta_1 x^1 + \theta_2 x^2 + \; \ldots . + \theta_M \, x^M$$

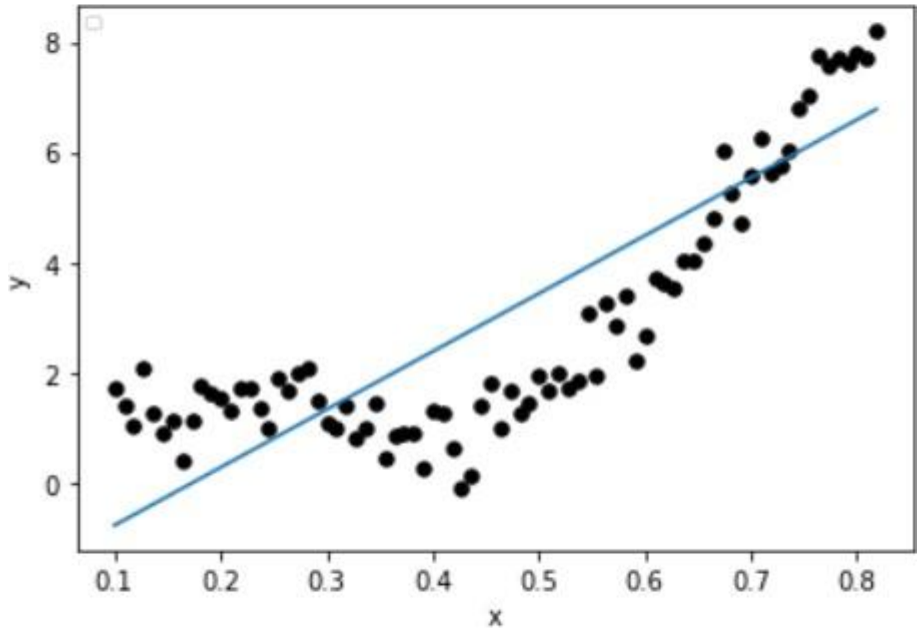
Polynomial regression is an extension of linear regression that models the relationship between the independent variable x and the dependent variable y using polynomial terms. Instead of fitting a straight line, it fits a curve by adding higher-degree terms of x.

Polynomial regression

- Simple linear regression model ($h_\theta(x) = \theta_0 + \theta_1 x$) could not fit the data well if the independent data, x exhibits nonlinear relationship with the dependent data, y

x	y
0.368	0.667
0.401	0.792
0.434	1.247
0.468	0.563
0.502	1.792
⋮	⋮
⋮	⋮

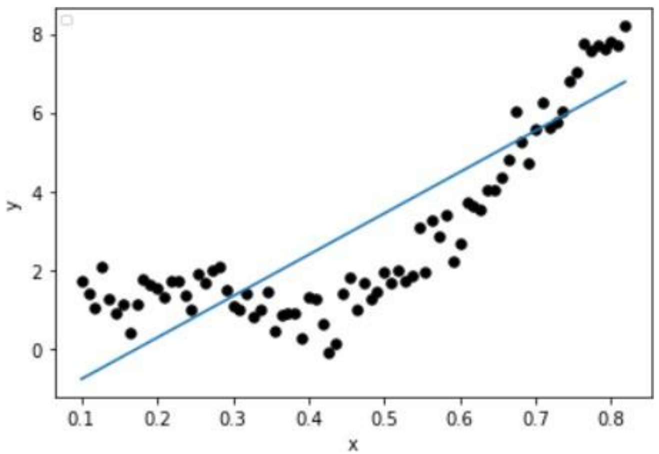
$h_\theta(x) = \theta_0 + \theta_1 x$



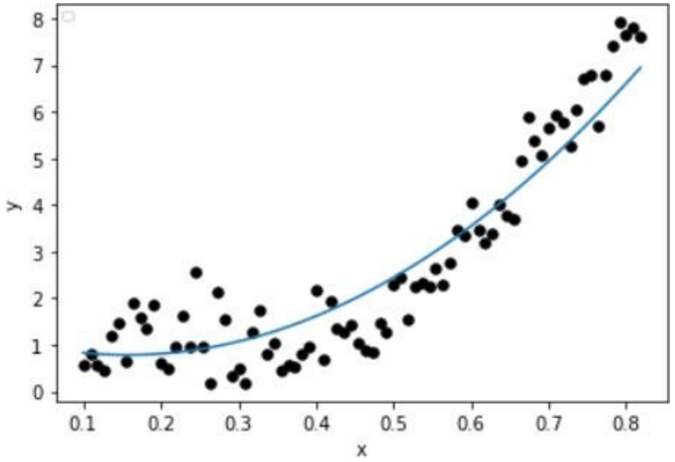
Polynomial regression

- Higher order Polynomial regression model can better fit the training data.

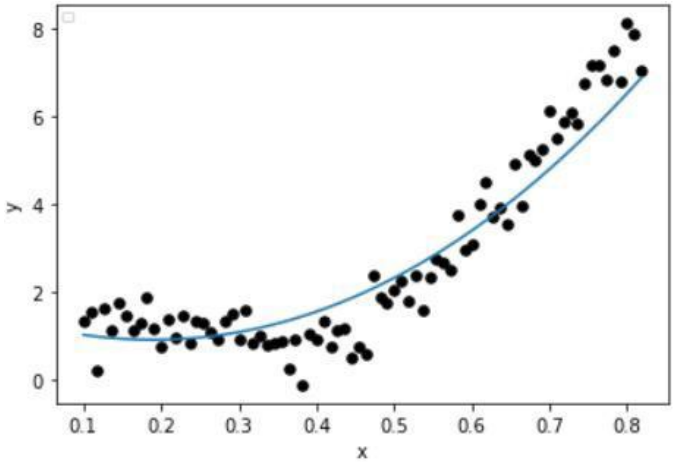
$h_\theta(x) = \theta_0 + \theta_1 x$



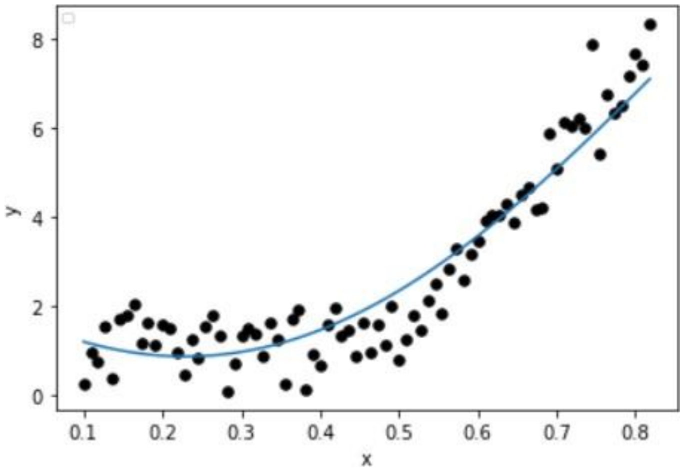
$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$



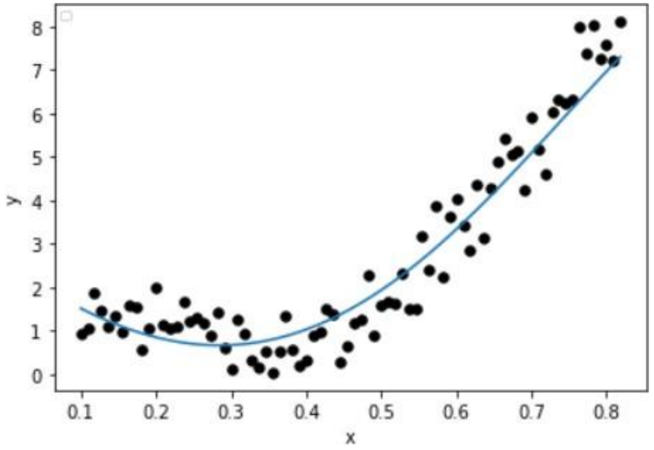
$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$



$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$



$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5$



Polynomial regression

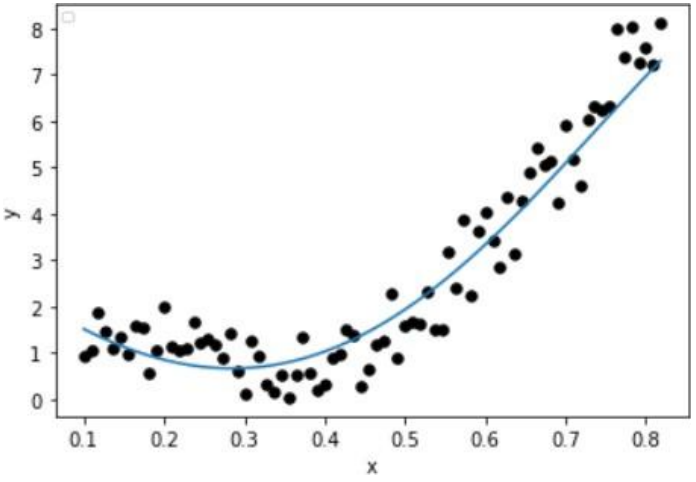
- Polynomial regression is consider a special case of multiple linear regression.
- Although polynomial regression fits a nonlinear model to the data, it is linear to the parameters $\theta_0, \theta_1, .., \theta_j$.

Polynomial linear regression | Multiple linear regression

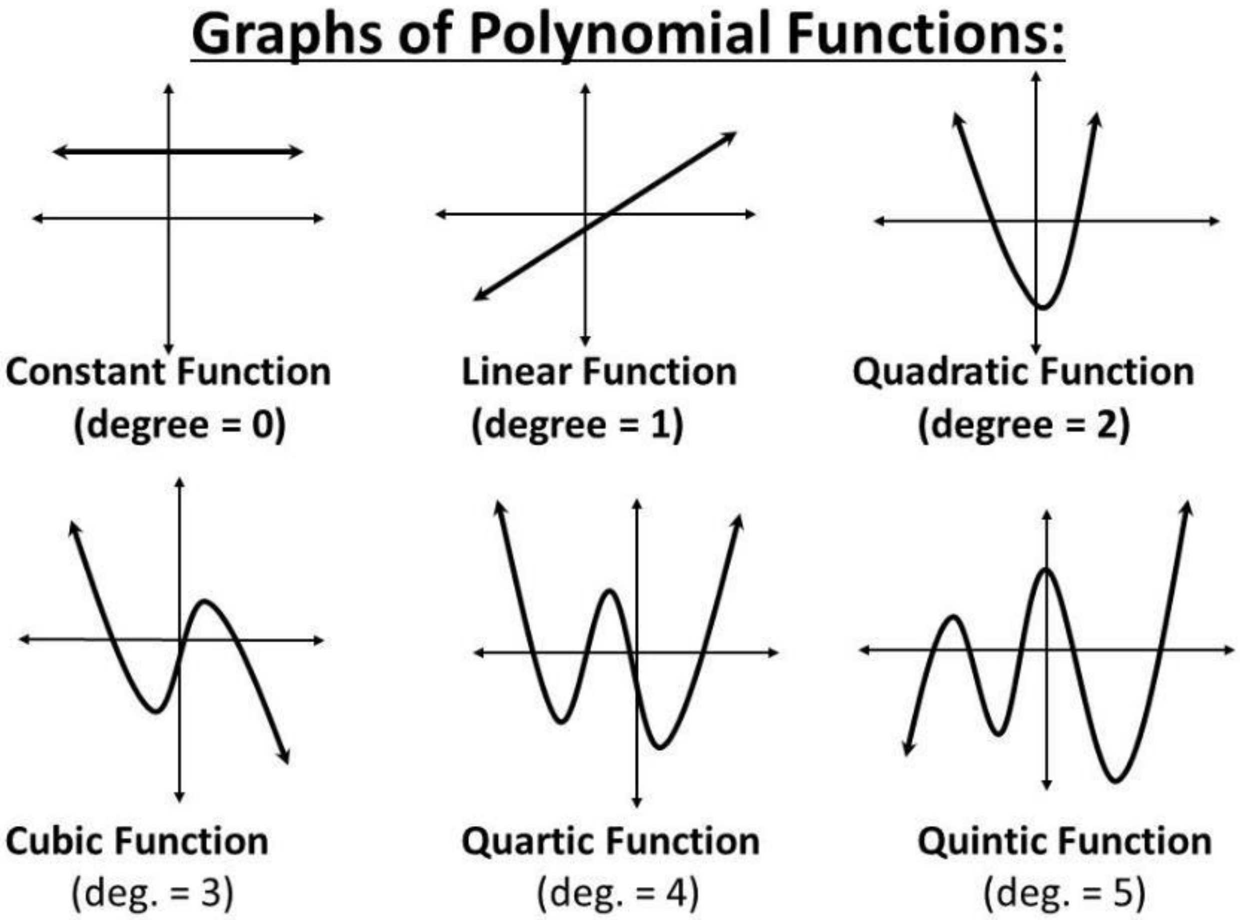
$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \ldots. + \theta_5 x^5 \subseteq \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots. + \theta_5 x_5$$

$(x)^j$ is using the same variable x

$(x)^2$	x	y
$(0.368)^2$	0.368	0.667
$(0.401)^2$	0.401	0.792
$(0.434)^2$	0.434	1.247
$(0.468)^2$	0.468	0.563
$(0.502)^2$	0.502	1.792
⋮	⋮	⋮
⋮	⋮	⋮



Polynomial Functions with varying degree



Different types of polynomial functions

[4] Brandon John Grenier, (2017, January 27), Multivariate Polynomial Regression <http://brandon.ai/2017/01/27/multivariate-polynomial-regression.html>

Topics

- Regression Problems and Linear Regression
- Mathematical Foundations and Optimisation Techniques
- Variants of Linear Regression
- Overfitting and Regularisation Techniques

Overfitting and Underfitting

Overfitting and underfitting are two common problems that occur when training machine learning models. They affect the model's ability to generalise well to new, unseen data.

- **Overfitting** (Too complex, memorises noise)
- **Underfitting** (Too simple, fails to learn)

Overfitting

Overfitting occurs when a model learns the noise and details of the training data too well, capturing unnecessary patterns that do not generalize to new data. The model performs exceptionally well on training data but poorly on validation/test data.

Causes:

- High model complexity – Too many parameters capture noise.
- Small dataset – Not enough data to generalise.
- Noisy/irrelevant features – Model learns unimportant patterns.
- Too many epochs – Model memorises training data.
- … …

Underfitting

Underfitting occurs when a model is too simple to capture the underlying pattern in the data. The model performs poorly on both training and test data, meaning it has not learned enough from the data.

Causes:

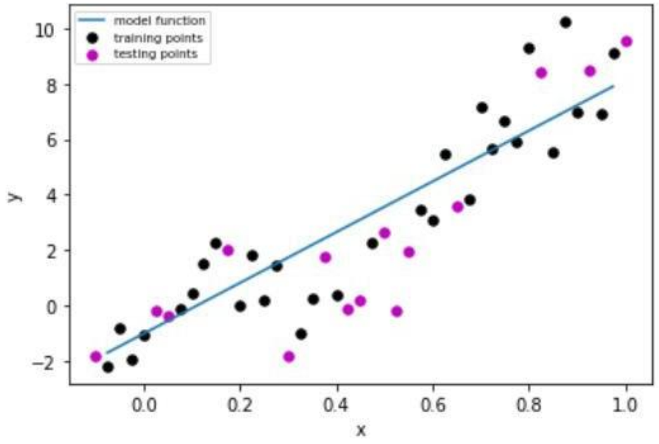
- Simple model – Lacks capacity for complex patterns.
- Few features – Important information missing.
- Insufficient training – Not enough epochs.
- Wrong model choice – Basic model for complex data.
- … …

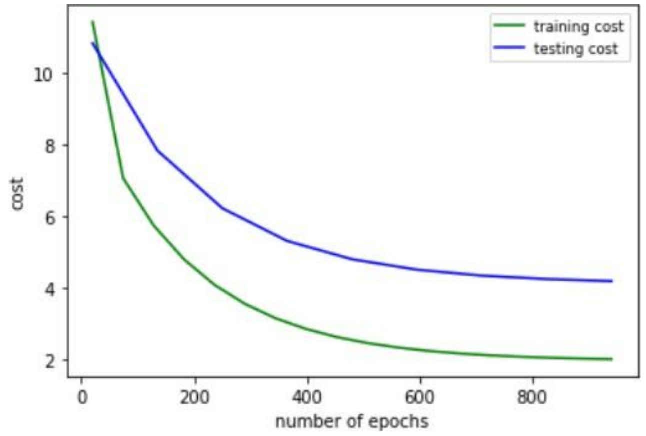
Examples of Overfitting and Underfitting

Underfitting: Model performs badly in both training and testing data

degree = 1

```
Epoch: 700 - Train Error: 2.1303 - Test Error: 4.3432
Epoch: 720 - Train Error: 2.1123 - Test Error: 4.3209
Epoch: 740 - Train Error: 2.0961 - Test Error: 4.3016
Epoch: 760 - Train Error: 2.0815 - Test Error: 4.2837
Epoch: 780 - Train Error: 2.0684 - Test Error: 4.2672
Epoch: 800 - Train Error: 2.0567 - Test Error: 4.2514
Epoch: 820 - Train Error: 2.0461 - Test Error: 4.2380
Epoch: 840 - Train Error: 2.0366 - Test Error: 4.2269
Epoch: 860 - Train Error: 2.0281 - Test Error: 4.2143
Epoch: 880 - Train Error: 2.0204 - Test Error: 4.2039
Epoch: 900 - Train Error: 2.0135 - Test Error: 4.1946
Epoch: 920 - Train Error: 2.0073 - Test Error: 4.1854
Epoch: 940 - Train Error: 2.0017 - Test Error: 4.1781
Converged.
```

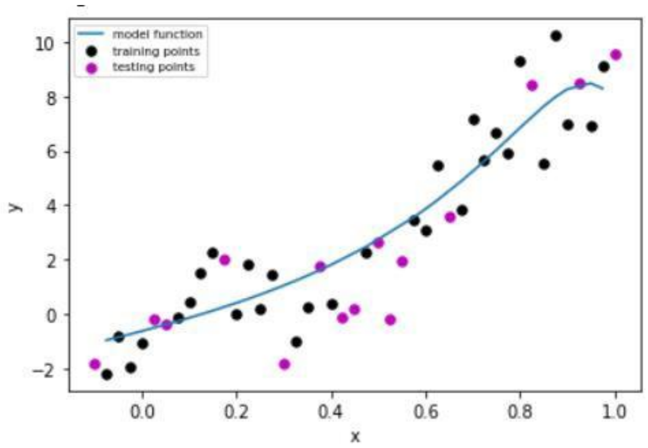


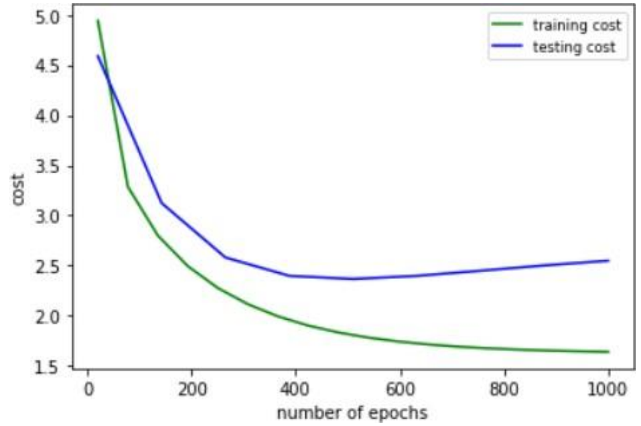


Overfitting: Model performs too well on the training data but the performance drops significantly over the testing data

degree = 19

```
Epoch: 740 - Train Error: 1.6817 - Test Error: 2.4412
Epoch: 760 - Train Error: 1.6761 - Test Error: 2.4500
Epoch: 780 - Train Error: 1.6711 - Test Error: 2.4589
Epoch: 800 - Train Error: 1.6665 - Test Error: 2.4677
Epoch: 820 - Train Error: 1.6624 - Test Error: 2.4763
Epoch: 840 - Train Error: 1.6586 - Test Error: 2.4847
Epoch: 860 - Train Error: 1.6552 - Test Error: 2.4937
Epoch: 880 - Train Error: 1.6521 - Test Error: 2.5019
Epoch: 900 - Train Error: 1.6493 - Test Error: 2.5104
Epoch: 920 - Train Error: 1.6468 - Test Error: 2.5181
Epoch: 940 - Train Error: 1.6445 - Test Error: 2.5260
Epoch: 960 - Train Error: 1.6423 - Test Error: 2.5337
Epoch: 980 - Train Error: 1.6404 - Test Error: 2.5406
Epoch: 1000 - Train Error: 1.6387 - Test Error: 2.5478
Converged.
```



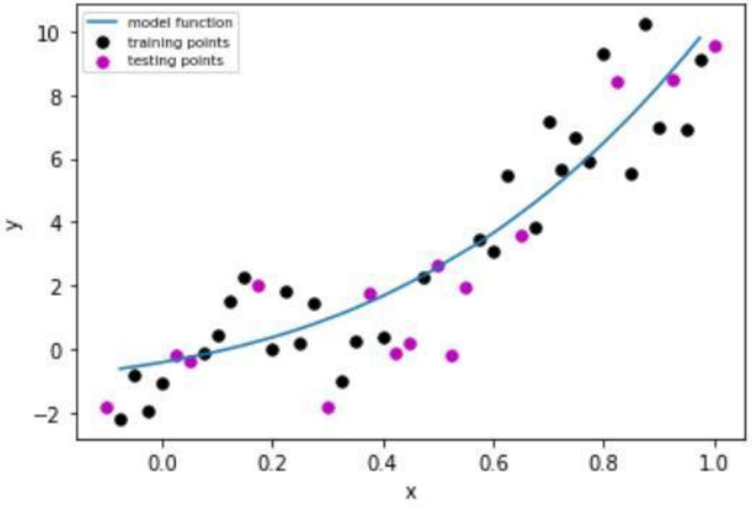


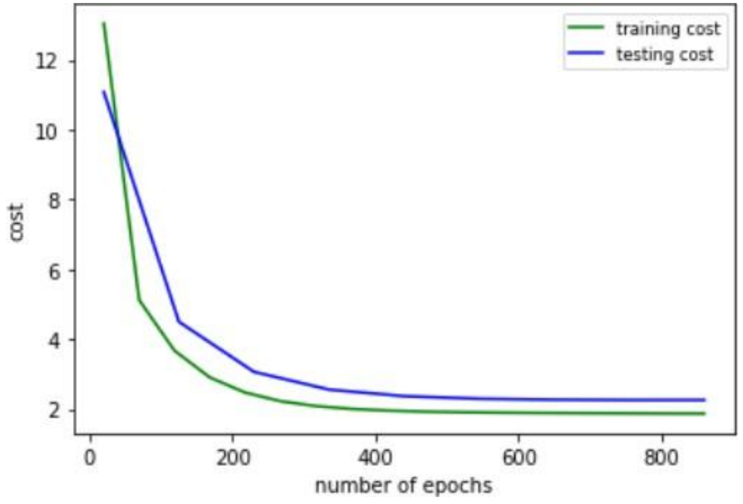
Examples of Overfitting and Underfitting

An acceptable model

Degree = 3

```
Epoch: 640 - Train Error: 1.8837 - Test Error: 2.2646
Epoch: 660 - Train Error: 1.8814 - Test Error: 2.2605
Epoch: 680 - Train Error: 1.8793 - Test Error: 2.2590
Epoch: 700 - Train Error: 1.8773 - Test Error: 2.2563
Epoch: 720 - Train Error: 1.8755 - Test Error: 2.2550
Epoch: 740 - Train Error: 1.8738 - Test Error: 2.2532
Epoch: 760 - Train Error: 1.8722 - Test Error: 2.2528
Epoch: 780 - Train Error: 1.8706 - Test Error: 2.2526
Epoch: 800 - Train Error: 1.8691 - Test Error: 2.2520
Epoch: 820 - Train Error: 1.8677 - Test Error: 2.2521
Epoch: 840 - Train Error: 1.8663 - Test Error: 2.2522
Epoch: 860 - Train Error: 1.8649 - Test Error: 2.2530
Converged.
```





Methods to overcome Overfitting

- **Reducing model complexity** – Overly complex models can memorise noise instead of learning patterns.
  - Manual feature selection: Remove irrelevant or redundant features to simplify the model.
  - Model selection: Choose a model that balances complexity and performance.
- **Adding regularisation penalties** – Regularisation prevents overfitting by shrinking the magnitude of the model's weights, making the model less sensitive to small fluctuations in training data.
- **Collecting more training data** – More diverse data helps the model learn general patterns rather than memorising specifics.

Regularised linear regression

Regularisation reduces model complexity by adding a penalty to large weights.

There are mainly two types:

- **L1 Regularization (Lasso) -** Adds absolute values of weights to the loss function.
- **L2 Regularization (Ridge)** - Adds squared values of weights to the loss function.

L2 Regularisation

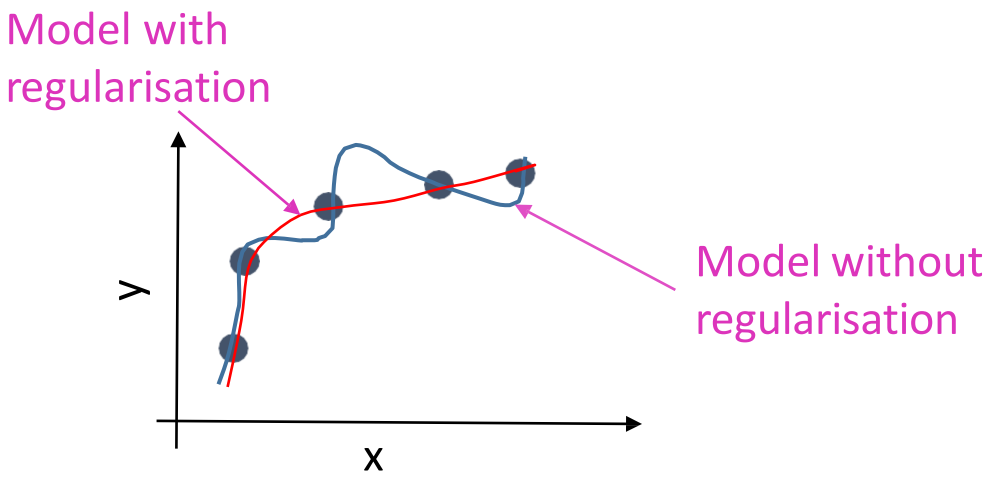
$$J(\theta) = \frac{1}{2N}\left[\sum_{n=1}^{N}\left(h\ (x^{(n)}) - y^{(n)}\right)^2 + \boxed{\lambda \sum_{j=1}^{M} \theta_j^{\ 2}}\right]$$

regularization term

Regularisation parameter (weight decay)

- Goal: minimise J(θ)

- To minimize J(θ), the learned model will try to shrink the regularisation term by reducing the θ_j towards zero.
- The values of θ_j decrease and become smaller, leading to a simpler hypothesis/model.



L2 Regularisation

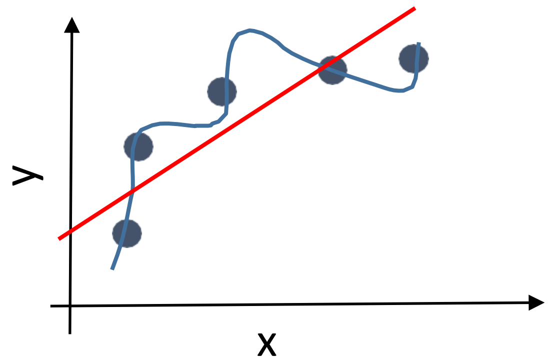
Regularisation parameter (weight decay)

$$J(\theta) = \frac{1}{2N}\Big[\sum_{n=1}^{N}\big(h\ (x^{(n)}) - y^{(n)}\big)^2 + \boxed{\lambda \sum_{j=1}^{M} \theta_j{}^2\Big]}$$

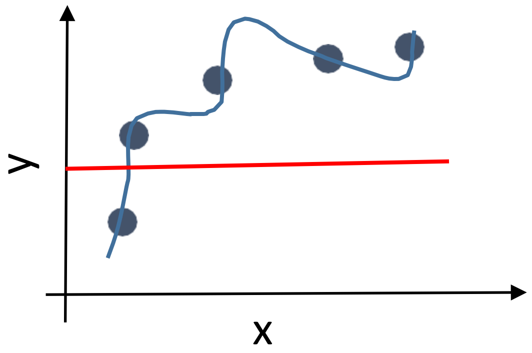
regularization term

- When λ is too *large*, the learned model will force the θ_j to shrink to a larger extent towards zero. The hypothesis will become almost linear.

$$h_\theta(x) = \theta_0 + \theta_1 x + \overset{\approx 0}{\cancel{\theta_2 x^2}} + \overset{\approx 0}{\cancel{\theta_3 x^3}}$$



$$h_\theta(x) = \theta_0 + \overset{\approx 0}{\cancel{\theta_1 x}} + \overset{\approx 0}{\cancel{\theta_2 x^2}} + \overset{\approx 0}{\cancel{\theta_3 x^3}}$$



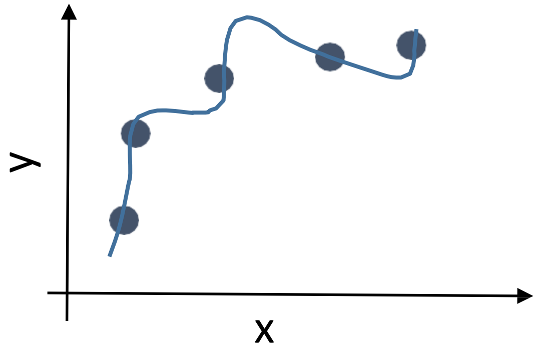
- A very large λ value may cause underfitting in the training set.

L2 Regularisation

$$J(\theta) = \frac{1}{2N}\left[\Sigma_{n=1}^{N}\left(h_\theta(x^{(n)}) - y^{(n)}\right)^2 + \lambda \Sigma_{j=1}^{M} \theta_j^2\right]$$

- If λ is too small, the regularisation term has little to no effect on regularising θ_{j} .

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$



L1 Regularisation

$$J(\theta) = \frac{1}{2N} \left[\sum_{n=1}^{N} \left(h(x^{(n)}) - y^{(n)}\right)^2 + \boxed{\lambda \sum_{j=1}^{M} |\theta_j| } \right]$$

L1 regularization term

- L1 regularisation is also called **Lasso** regularisation. It uses the **absolute values** of weights as a penalty term.
- A regression model that applies L1 regularisation is called **Lasso Regression**. Lasso has the unique property of shrinking some weights to **zero**, effectively removing less important features from the model.
- Thus, Lasso can be used for **feature selection**, identifying the most relevant features in a dataset.

Use L1 regularisation when feature selection is needed because it encourages sparsity by setting some weights to zero. Use L2 regularisation when reducing overfitting while keeping all features, as it shrinks weights smoothly.

Next Lecture

❖ Logistic regression