CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Machine Learning

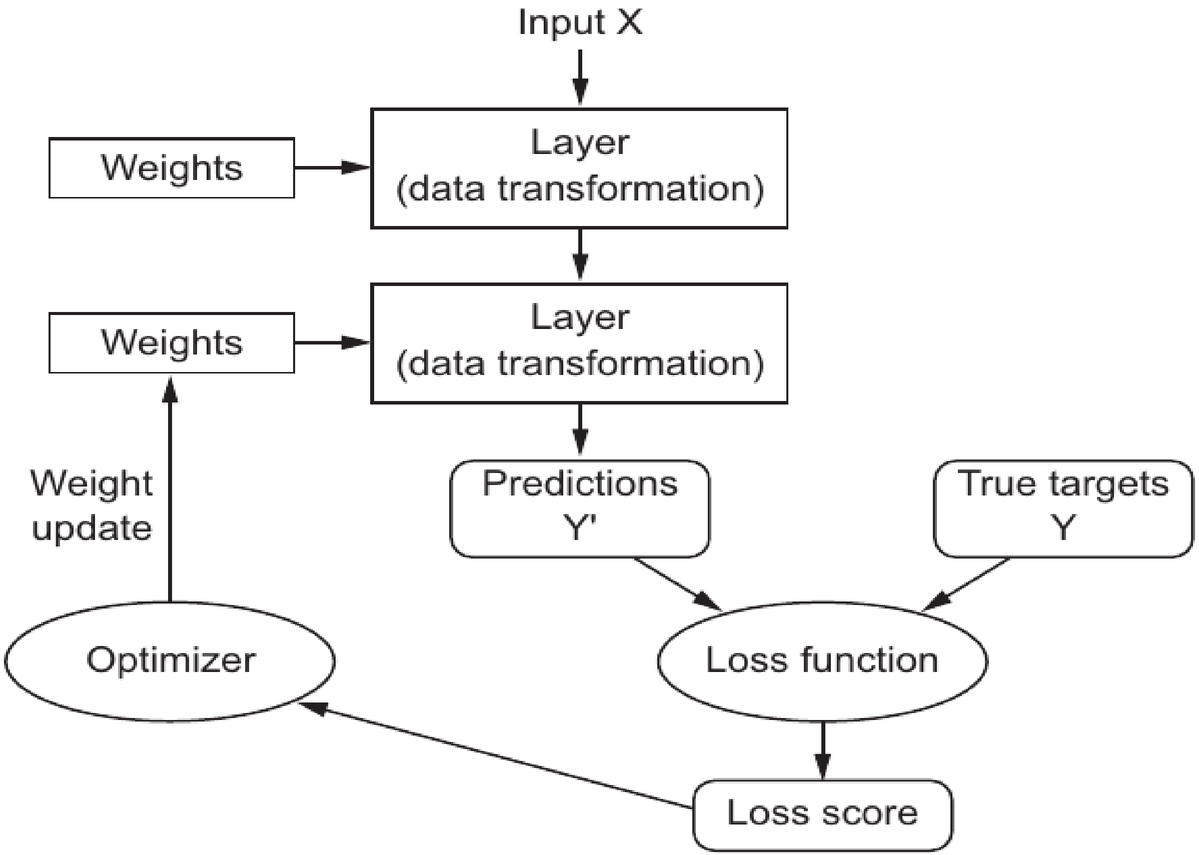
Machine Learning

Lecture: Linear and Multivariate Regression

Ted Scully

Contents

1. Introduction to Linear Regression
2. Applying Gradient Decent to Linear Regression
3. Multi-Variate Linear Regression
4. Review of Matrices and Broadcasting in Python
5. Logistic Regression
6. Vectorized Logistic Regression
7. Neural Networks


Input X
Weights
Layer
(data transformation)
Weights
Layer
(data transformation)
Weight
update
Predictions
Y'
True targets
Y
Optimizer
Loss function
Loss score

Linear Regression

1. Linear regression attempts to **model the relationship between two variables** by fitting a linear equation to observed data.

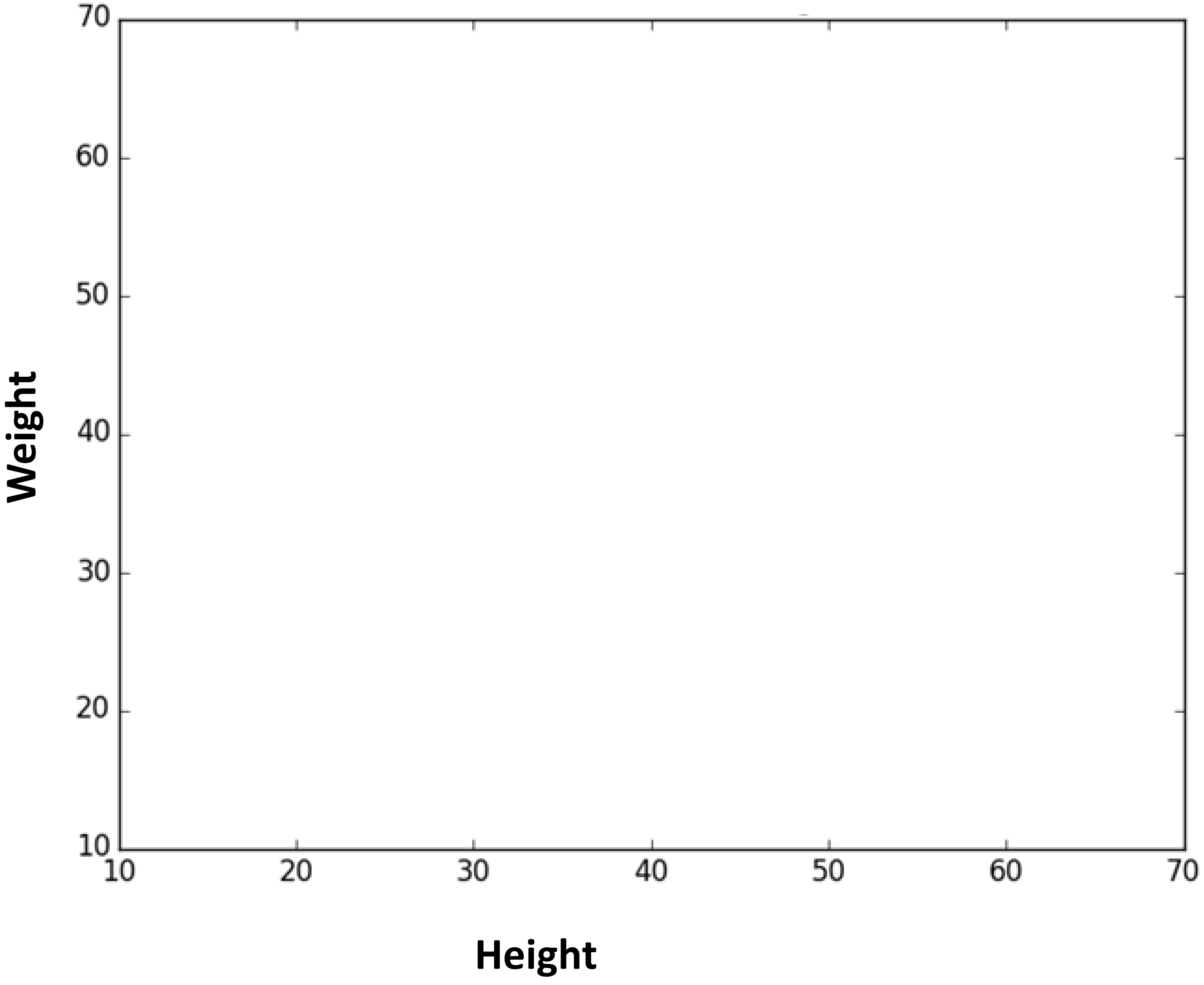
2. For example, you may want to relate the weights of individuals to their heights using a linear regression model or office rental prices with square footage.

3. Linear regression is an example of a supervised learning regression technique

Introduction to Linear Regression

In this example we look at a bivariate dataset.

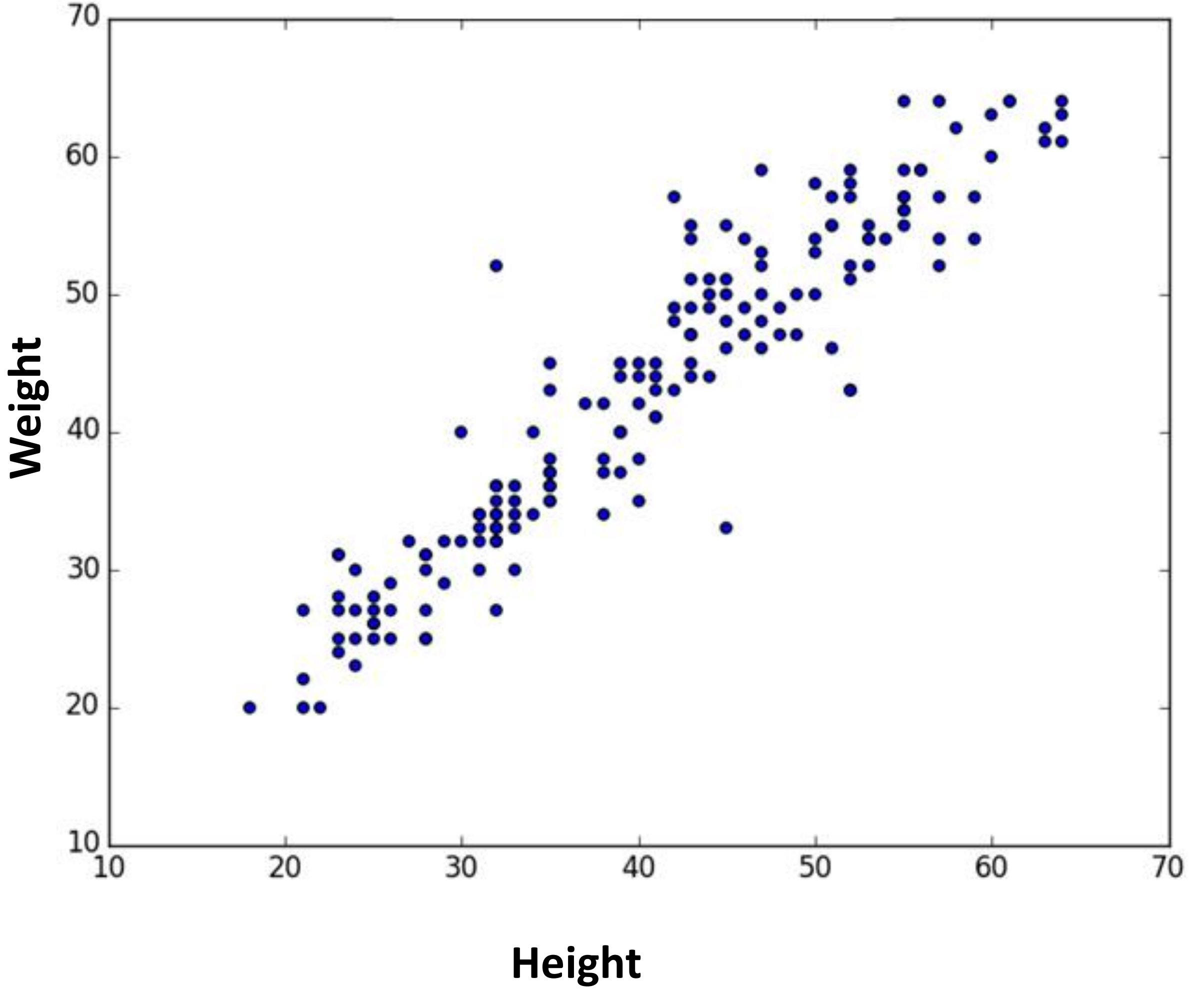
We examine the relationship between a persons height and weight



Introduction to Linear Regression

The data shows us the relationship between a persons height and weight.

It could be useful to create a **function** that accepts a persons height as input and allows us to estimate (based on this data) their weight.

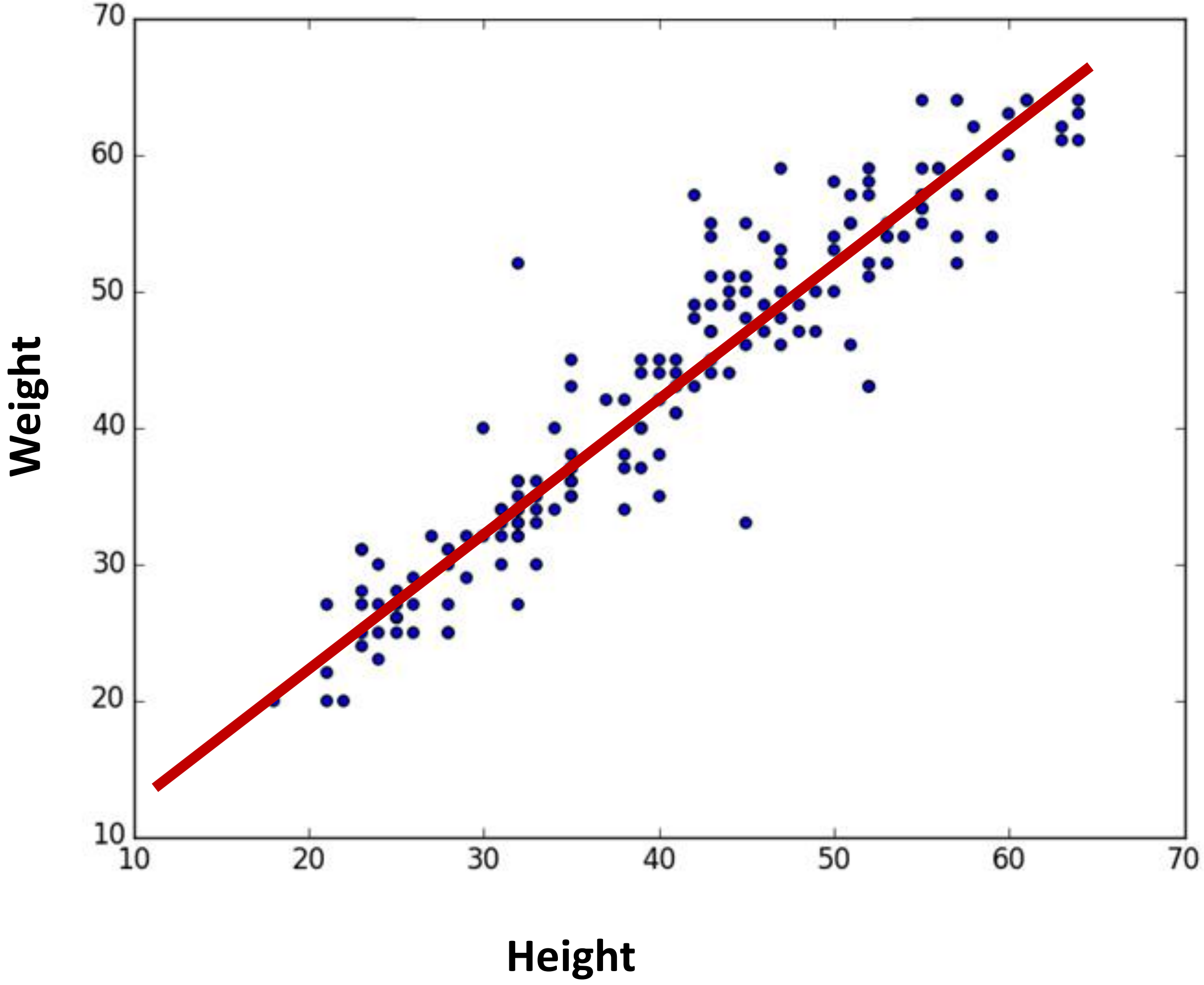


Introduction to Linear Regression

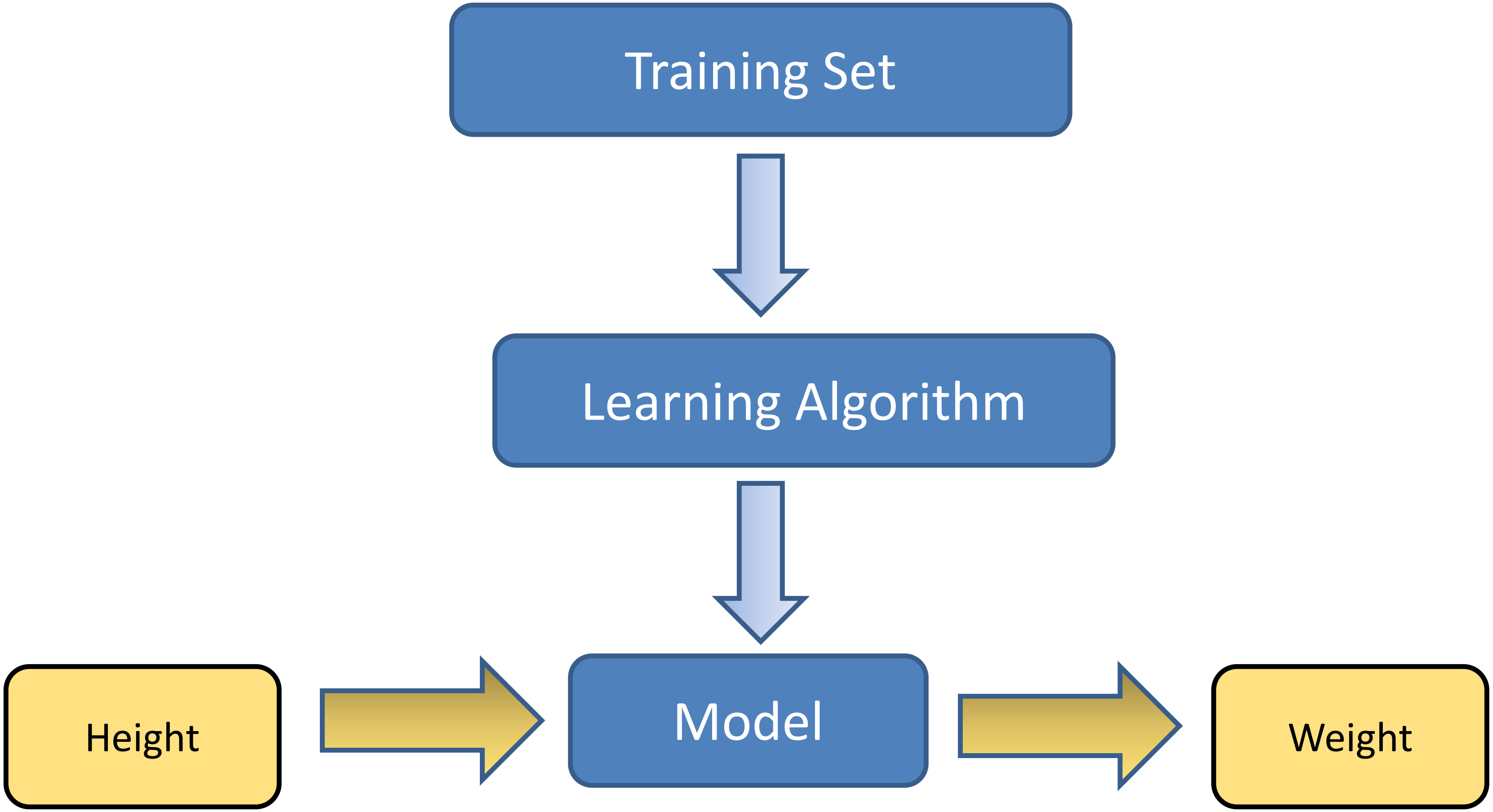
Linear regression allows us to do this.

Why is this supervised learning?

Why is it a regression problem?



Introduction to Linear Regression



Function h maps from a persons height to estimated weight

Introduction to Linear Regression

- Before attempting to fit a linear model to observed data you should first determine if there is a **relationship between the variables** of interest.

- There should be some significant association between the two variables.

- A **scatterplot** or a measure of **correlation** such as Pearson's Coefficient can be a helpful tool in determining the strength of the relationship between two variables.

- If there appears to be no association between the proposed variables (i.e., the scatterplot does not indicate any increasing or decreasing trends), then fitting a linear regression model to the data will not provide a useful model.

Notation

- **m** = Number of training examples
- **x** = Input variable (for example height of a person)
- **y** = Output variable (for example a persons weight)

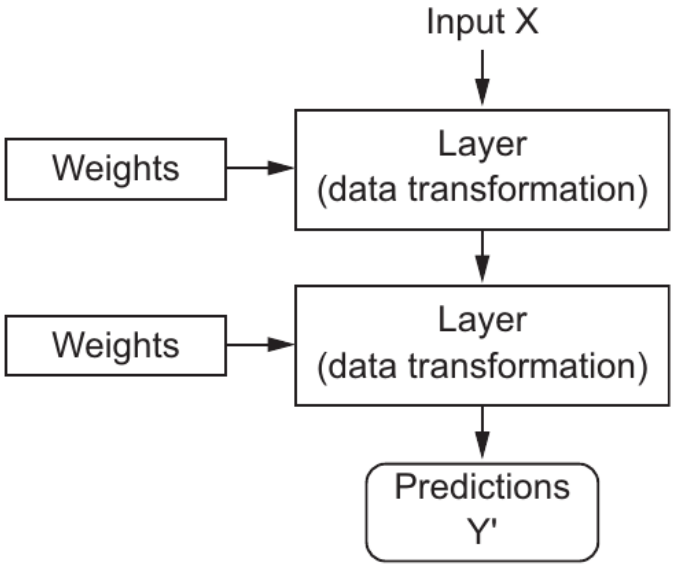
- **(x^i, y^i)** is the **i^{th}** training example from the dataset

- A linear regression line has an equation of the form:

  - ***$h(x) = \lambda_1 x + b$***

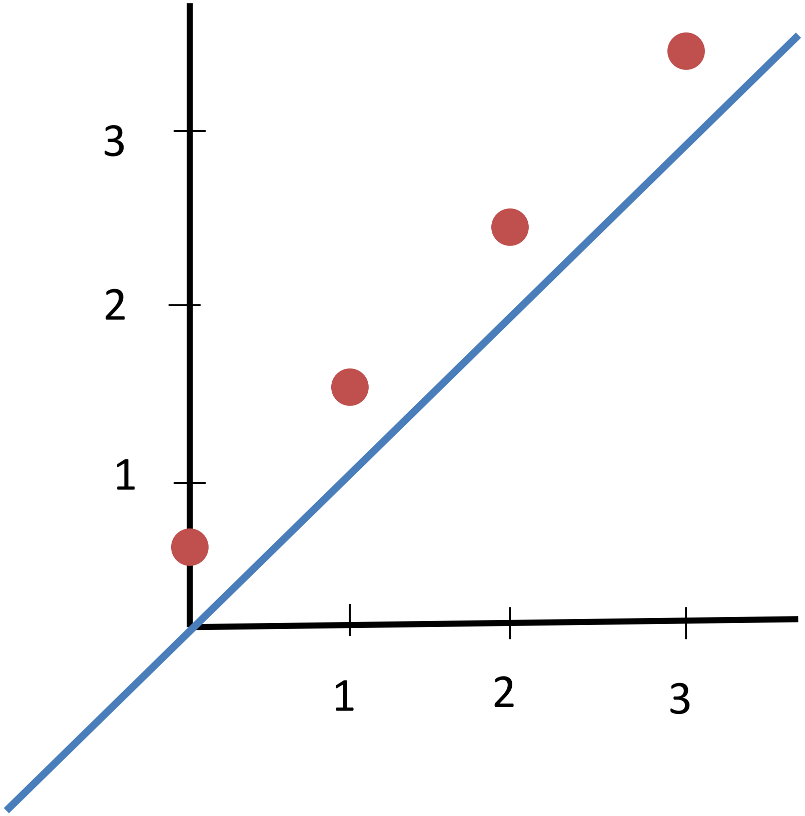
  - The **slope** of the line is λ_1 , and b is the **y** intercept (often referred to as bias).

- $h(x) = \lambda_1 x + b$



Formalising LR Problem

- The objective of a linear regression problem can be loosely described as assigning values to the variables λ_1 and ***b*** so as to **minimise predictive error**.
- So the first question we need to answer is **what do we mean by predictive error**? How would you measure it?

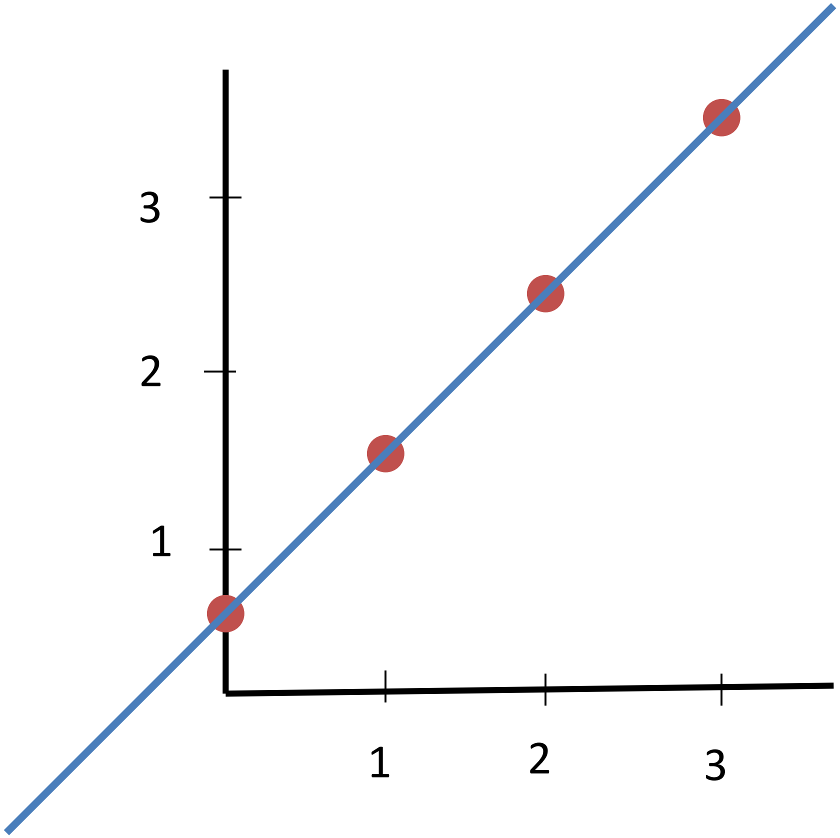


h(x) = λ_1 x + b

Let's assume we have the following values for our parameters:
b = 0
λ_1 = 1

Formalising LR Problem

- The objective of a linear regression problem can be loosely described as assigning values to the variables $\boldsymbol{\lambda_1}$ and ***b*** so as to **minimise error**.
- So the first question we need to answer is **what do we mean by error**? How would you measure it?



h(x) = $\boldsymbol{\lambda_1}$ x + b

Assume our linear regression gives us a value of

b = 0.5

$\boldsymbol{\lambda_1}$ = 1

Squared Error Cost Function

Here we obtain the difference between the actual value of $\mathbf{y}^i$ and our predicted value.

In this example $\mathbf{x}^i$ would be the size of a particular house from the dataset.

Our linear regression model produces the function **h**, which returns our estimated selling price for this house $\mathbf{h}(\mathbf{x}^i)$.

We then subtract this from the actual selling price $\mathbf{y}^i$.

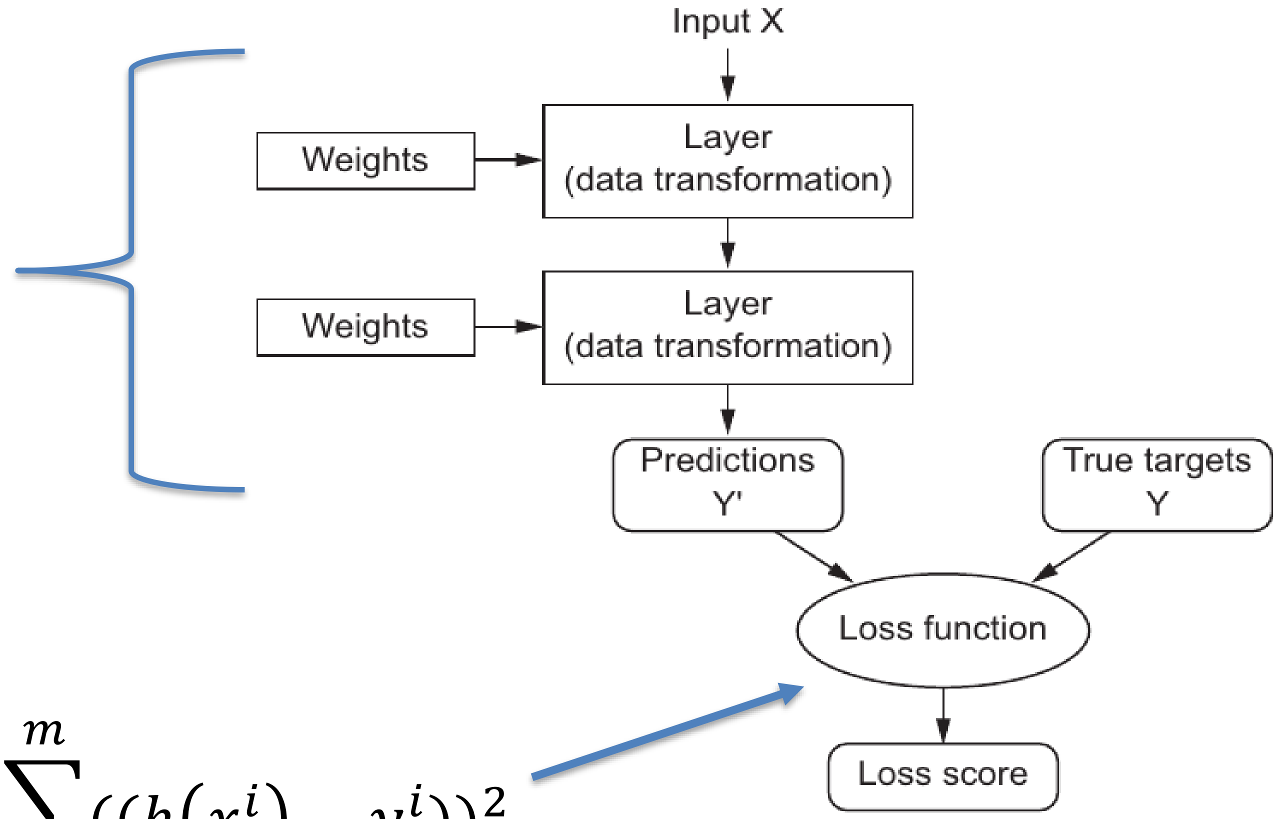
$$\frac{1}{m+2}\sum_{i=0}^{m}\boxed{((h(x^i)-y^i))^2}$$

Squared Error Cost Function

We then sum the squared difference (squared error) for each value in our training set. We then average the value by dividing by m (the number of training set examples).

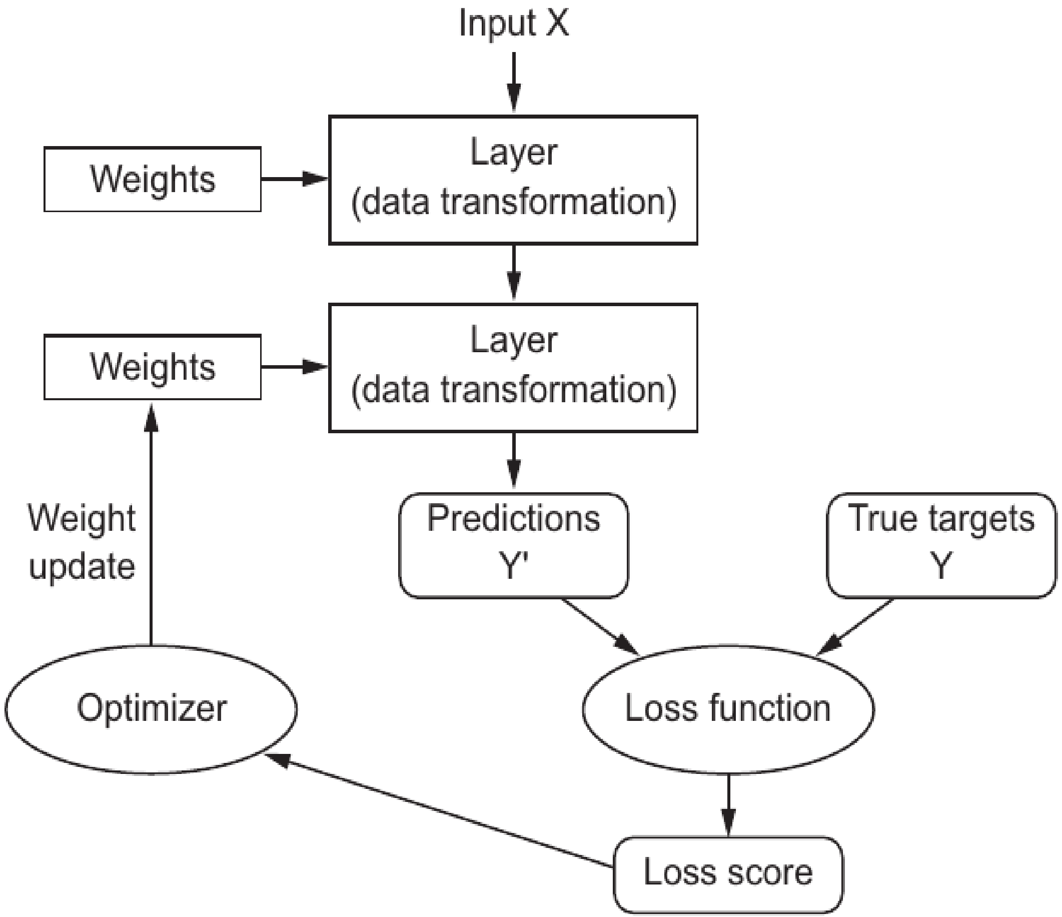
$$\frac{1}{m+2}\boxed{\sum_{i=0}^{m}((h(x^{i})-y^{i}))^{2}}$$

$$h(x) = \lambda_1\, x + b$$



$$\frac{1}{m+2}\sum_{i=0}^{m}((h(x^i) - y^i))^2$$

$h(x) = \lambda_1 x + b$



Input X

Weights

Layer
(data transformation)

Weights

Layer
(data transformation)

Weight
update

Predictions
Y'

True targets
Y

Optimizer

Loss function

Loss score

Gradient Descent

Optimizer

$$\frac{1}{m+2}\sum_{i=0}^{m}((h(x^{i}) - y^{i}))^{2}$$

Squared Error Cost Function

Remember our function is $h(x) = \lambda_1 x + b$. Therefore, our overall objective is to identify the values of b and λ_1 to plug into h such that we obtain the **minimal** average square error across all training examples. This is the objective or cost function for linear regression.

$$\boxed{minimise_{\lambda_1,\, b}} \frac{1}{m+2} \sum_{i=0}^{m} ((h(x^i) - y^i))^2$$

Squared Error Cost Function

$$\mathrm{c}(\lambda_1, b) \quad = \quad \frac{1}{m+2}\sum_{i=0}^{m}((h(x^i) - y^i))^2$$

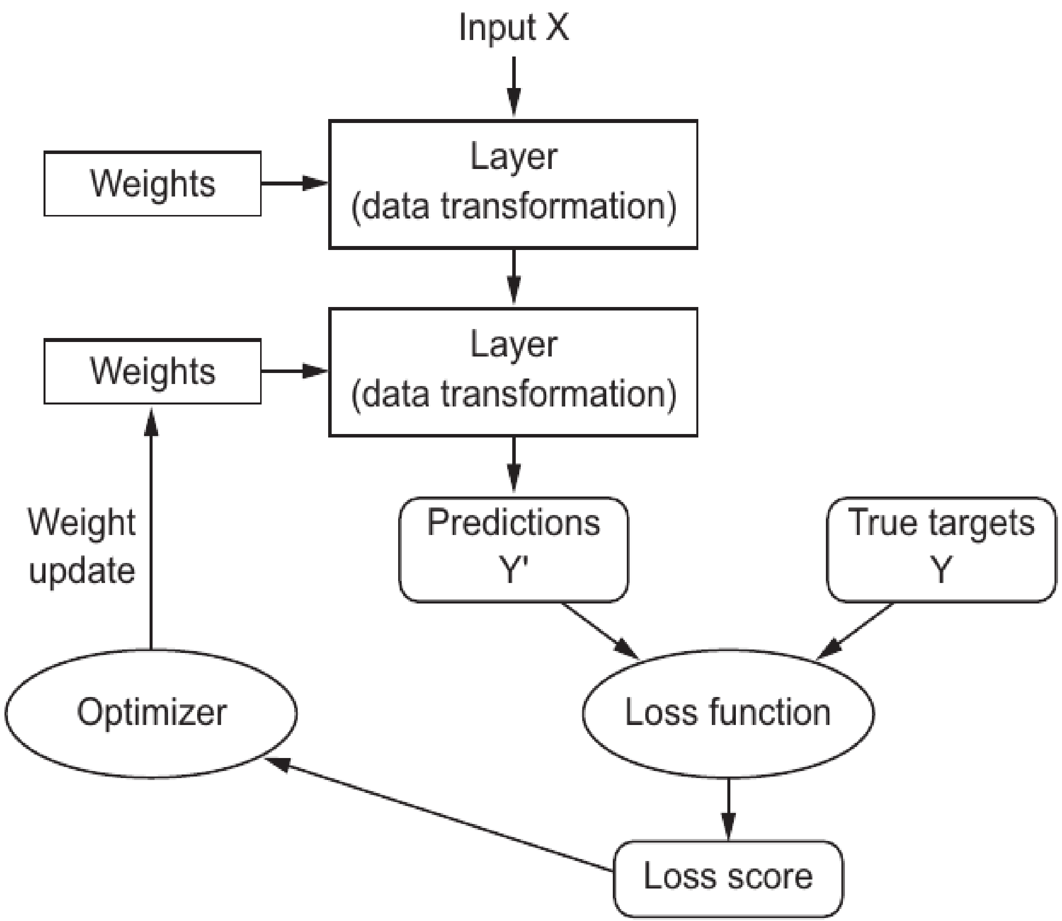
Just as a shorthand I’m going to rewire the above formula as follows:

$$minimise_{\lambda_1, b}\, C(\lambda_1, b)$$

Contents

1. Introduction to Linear Regression
2. Applying Gradient Decent to Linear Regression
3. Multi-Variate Linear Regression
4. Review of Matrices and Broadcasting in Python
5. Logistic Regression
6. Vectorized Logistic Regression
7. Neural Networks

$h(x) = \lambda_1 x + b$



Gradient Descent

Optimizer

$$\frac{1}{m+2}\sum_{i=0}^{m}((h(x^{i}) - y^{i}))^{2}$$

Squared Error Cost Function

Remember our function is *h(x) =* λ_1 *x + b*. Therefore, our overall objective is to identify the values of *b* and λ_1 to plug into h such that we obtain the **minimal** average square error across all training examples. This is the objective or cost function for linear regression.

$$\boxed{minimise_{\lambda_1,\, b}} \frac{1}{m+2}\sum_{i=0}^{m}((h\left(x^i\right) - y^i))^2$$

Squared Error Cost Function

$$\mathrm{c}(\lambda_1, b) \quad = \quad \frac{1}{m+2}\sum_{i=0}^{m}((h(x^i) - y^i))^2$$

Just as a shorthand I’m going to rewire the above formula as follows:

$$minimise_{\lambda_1, b}\, C(\lambda_1, b)$$

Gradient Descent and Linear Regression

- To understand and visualize the application of gradient descent we are going to **simplify the problem even further**.
- We are going to assume that the y-intercept (*b*) is always 0. Therefore, we only have one variable to worry about now (λ_1)
- ***h(x) =*** $\boldsymbol{\lambda_1}$ ***x***

$$\mathrm{c}(\lambda_1) \;=\; \frac{1}{m+2}\sum_{i=0}^{m}((h(x^i) - y^i))^2$$

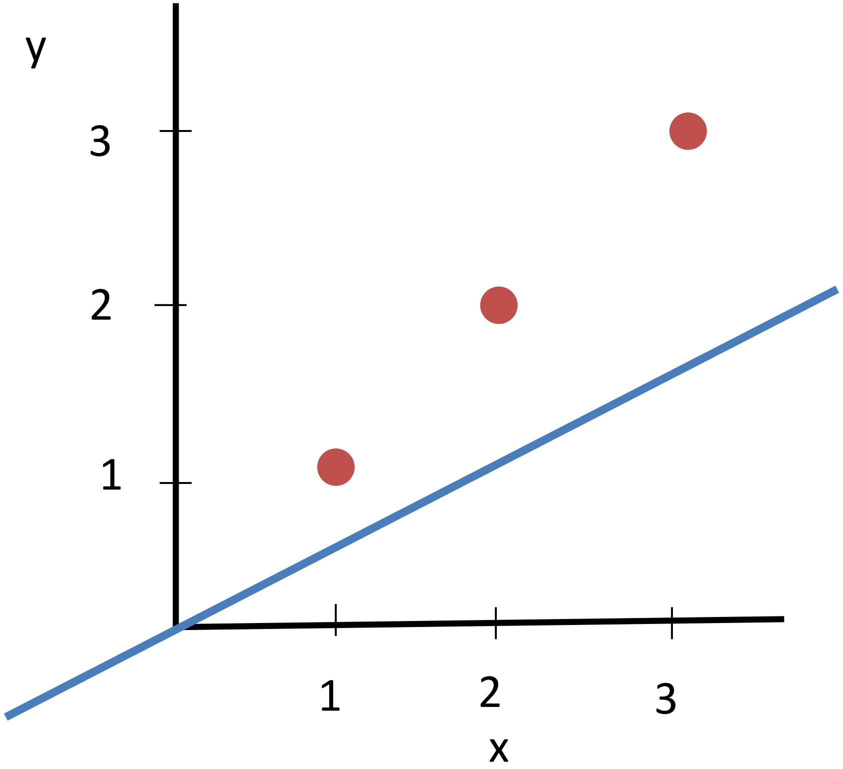
$$minimise_{\lambda_1}\; C(\lambda_1)$$

Over the next few slides we will pick a few different value for λ_1 and we will monitor the corresponding value of the cost function. The following is our simple dataset:

X	Y
1	1
2	2
3	3

$h(x) = \lambda_1 x$

Function h(x)



Let's examine what happens when I give **λ_1 a value of 0.5**.

This is not a good fit to data.

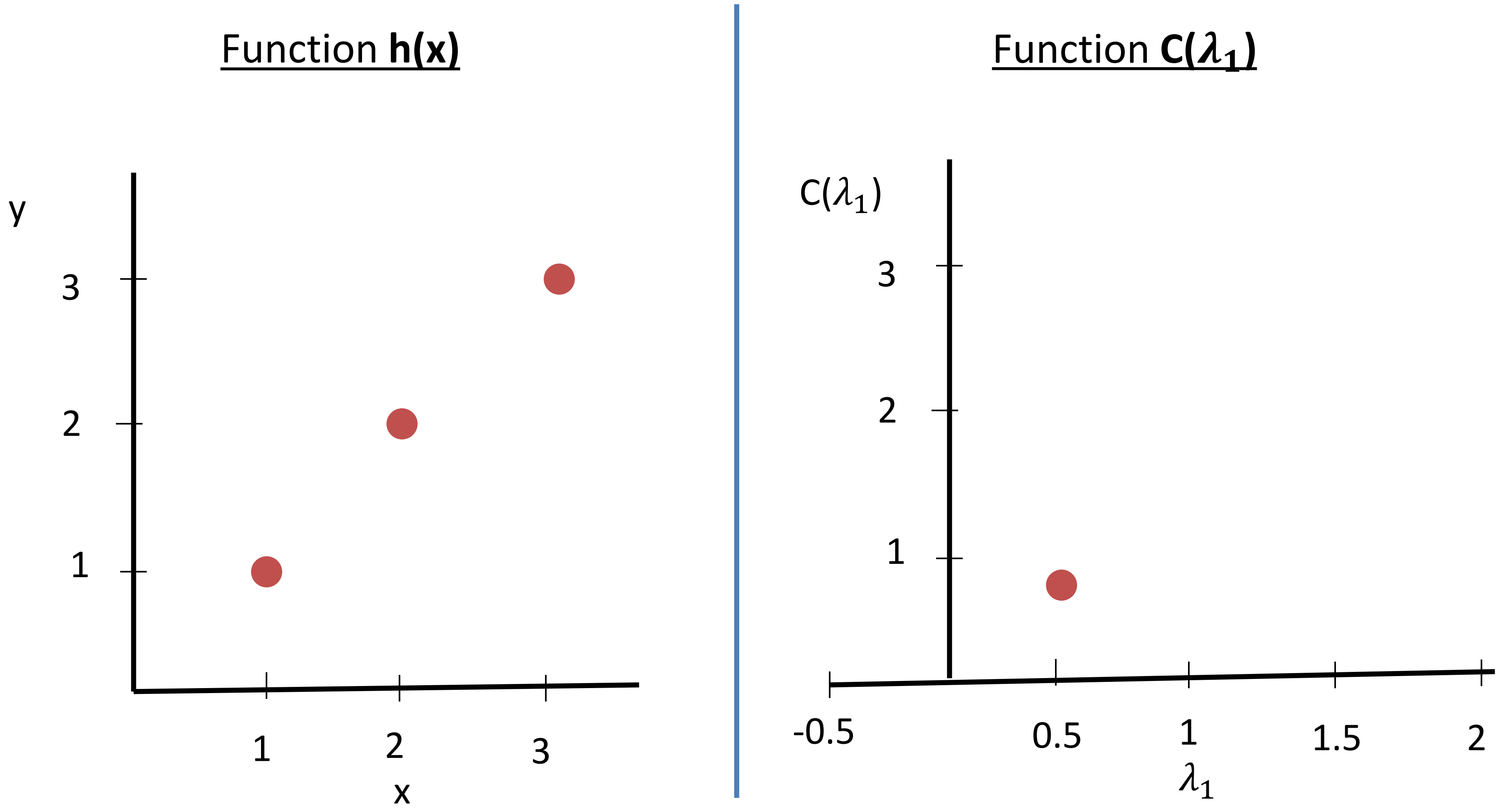
$$c(\lambda_1) \;=\; \frac{1}{m+2}\sum_{i=0}^{m}((h(x^i) - y^i))^2$$

$$\frac{1}{5}(\;(0.5-1)^2 + (1-2)^2 + (1.5-3)\;^2)$$

$$\frac{1}{5}(\;0.25 + 1 + 2.25)$$

$$= 0.7$$

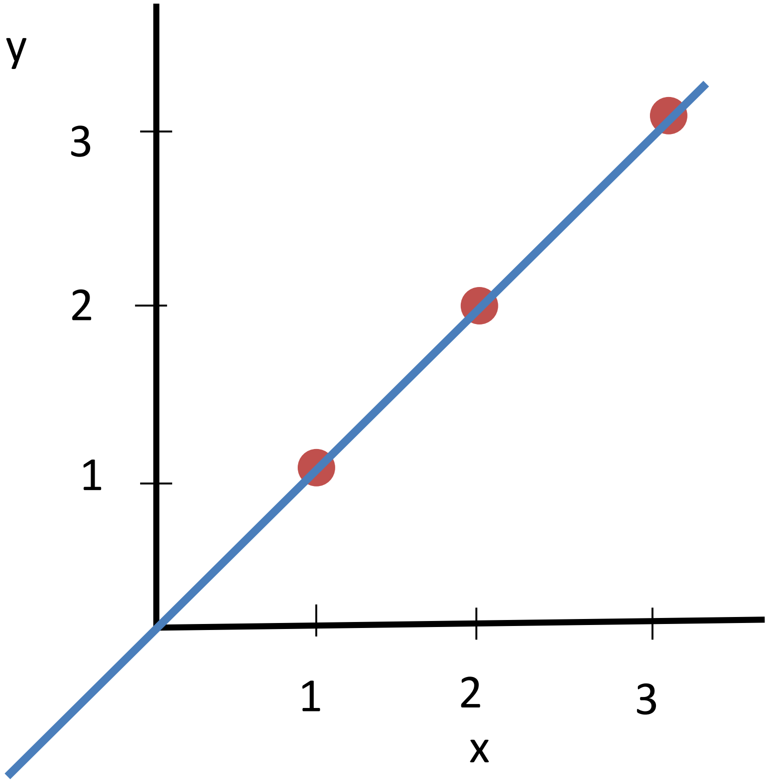
Notice we define our line only using a single parameter λ_1 . Controls the slope and must pass through the origin.



Map the value of C(λ_1) where $\lambda_1 = 0.5$

$$h(x) = \lambda_1 x$$

Function h(x)



Now we examine what happens when I give **λ_1 a value of 1**.

This provides an excellent fit to the data.

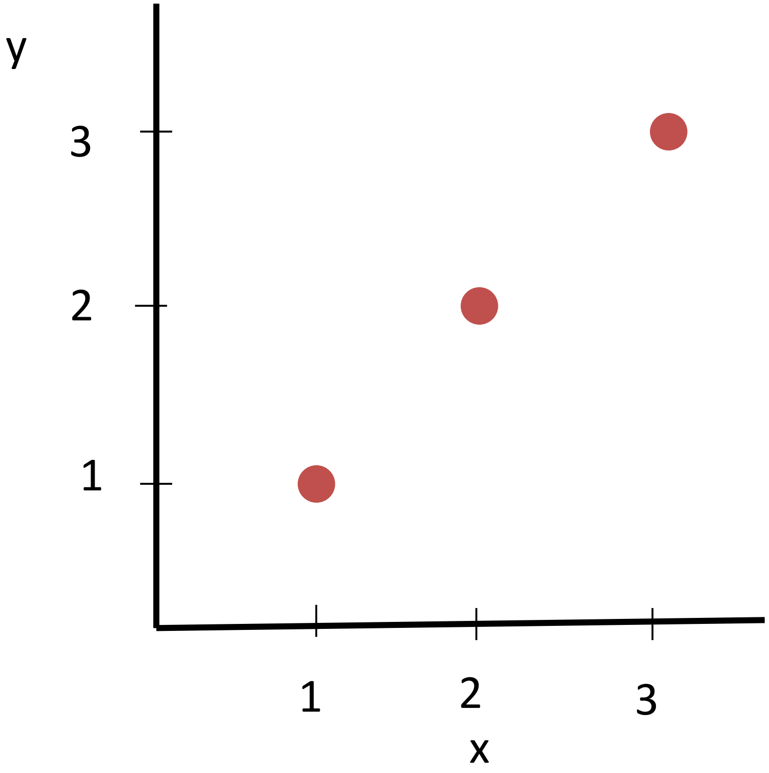
$$c(\lambda_1) = \frac{1}{m}\sum_{i=0}^{m}((h(x^i) - y^i))^2$$

$$\frac{1}{5}(\,(1-1)^2 + (2-2)^2 + (3-3)\,^2)$$

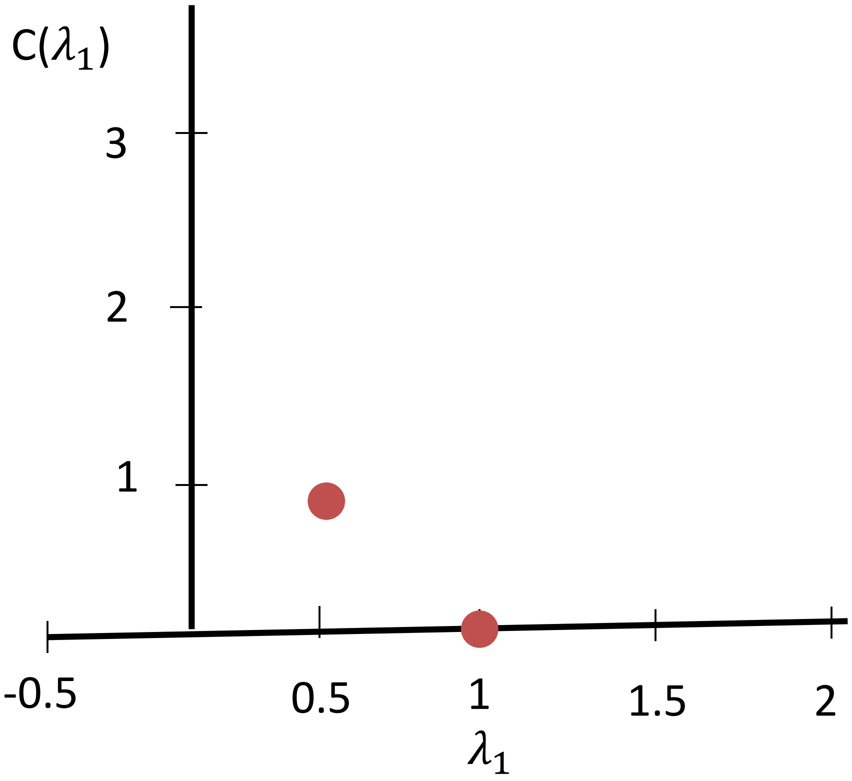
$$\frac{1}{5}(\,0)$$

$$= 0$$

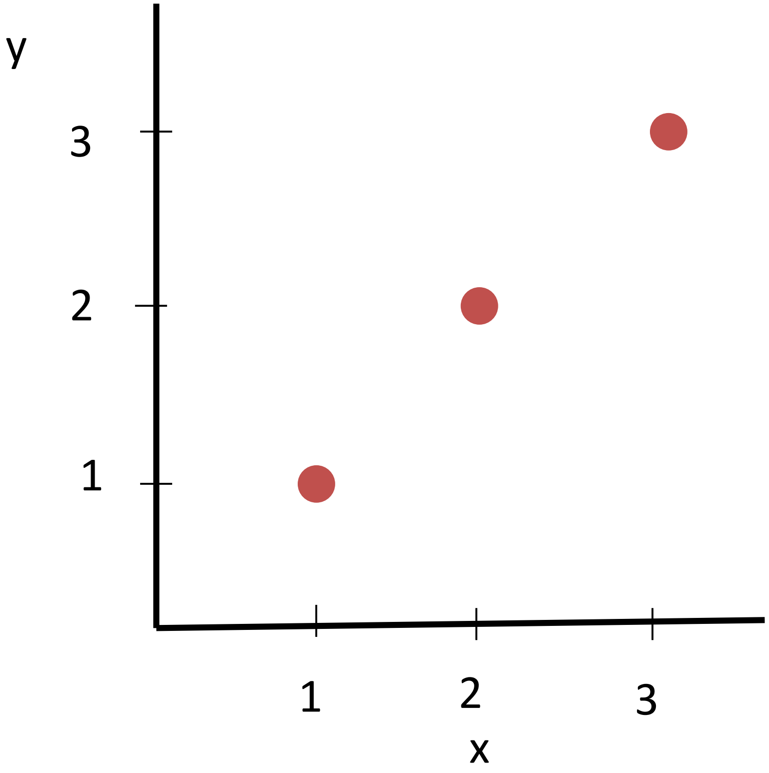
Function h(x)



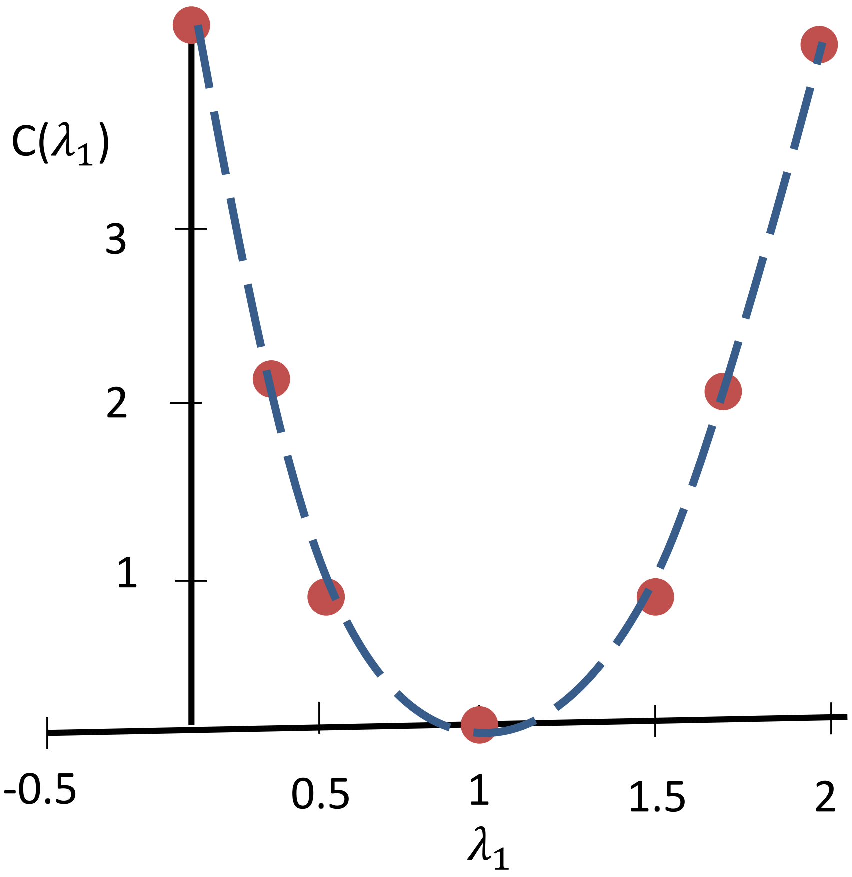
Function C(λ_1)



Function h(x)

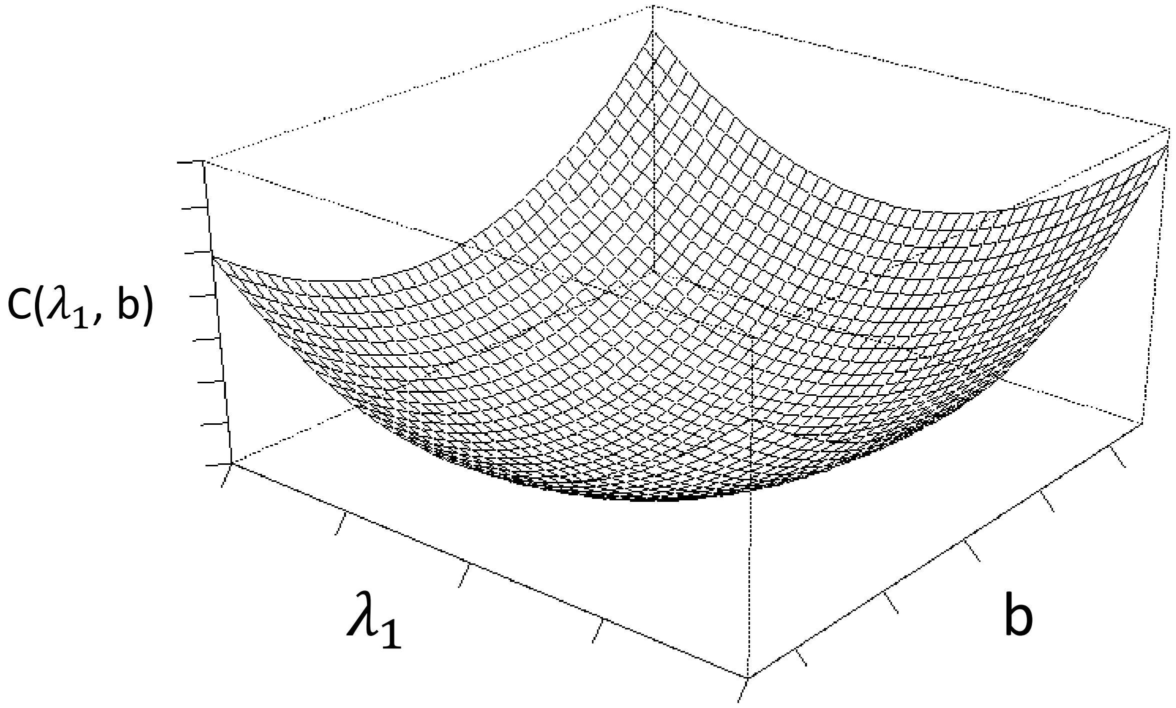


Function C(λ_1)



Linear Regression – A Search Problem

- As we add additional parameters (such as b) we increase the size of the search space. Below is an example **search space** when we have two parameters ($\underline{\lambda_1}$ and b)



Gradient Decent – An Optimization Algorithm

- The gradient descent method (also referred to as the method of steepest decent) is a way of finding a local minimum of a function.

- It begins with a **random guess** of the solution. We determine the **gradient of the function** at that point.

- We step the solution in the **negative direction** of the gradient and we repeat the process.

- The algorithm will eventually converge where the **gradient is zero** (which corresponds to a local minimum).

Gradient Decent – Algorithm

```
repeat {
    $\lambda_1 = \lambda_1 - \alpha \frac{\partial}{\partial \lambda_1} C(\lambda_1, b)$
    $b = b - \alpha \frac{\partial}{\partial b} C(\lambda_1, b)$
}
```

Where

- α is a numerical value that is called the **learning rate**. It controls the size of the decent that we make during each iteration. The larger the value of the greater the value of α
- $\frac{\partial}{\partial \lambda_1}$ and $\frac{\partial}{\partial b}$ are **derivative term** allowing us to calculate the slope of any point for the function

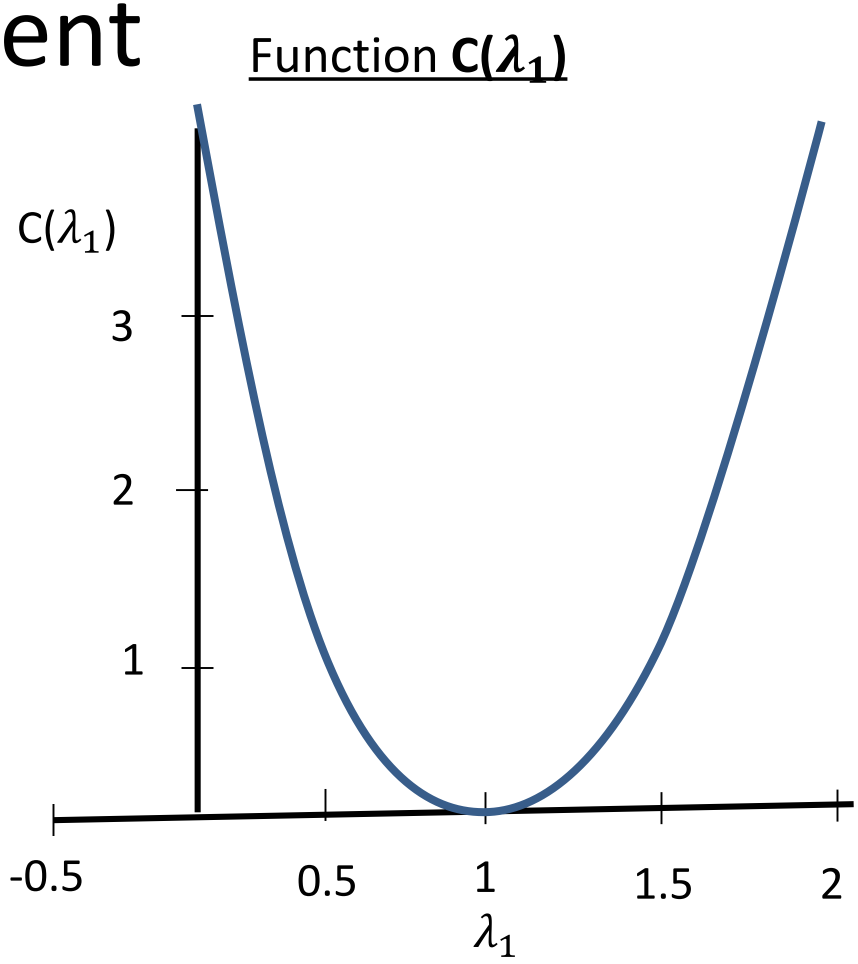
More Detail on Parameters of Gradient Decent

- Lets’ return to our simple example from earlier where we have

  $h(x) = \lambda_1 x$

- And our objective is:

$$minimise_{\lambda_1}\ C(\lambda_1)$$

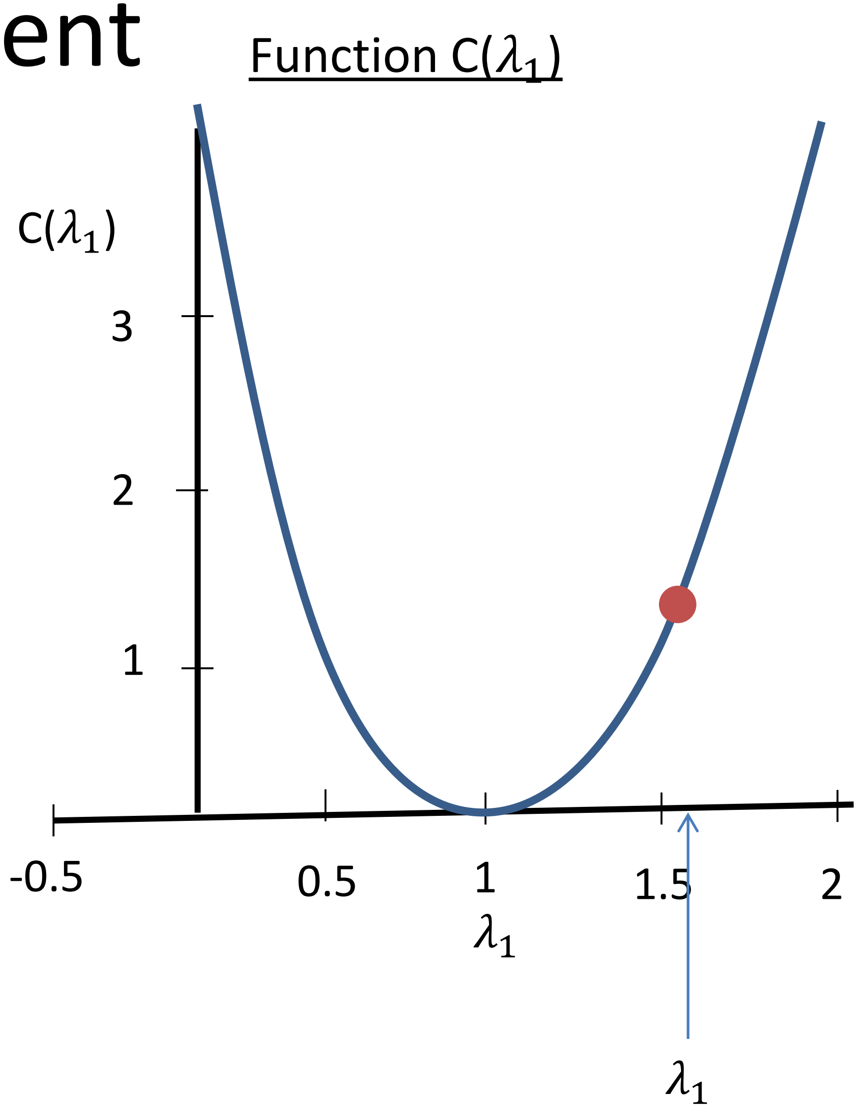


More Detail on Parameters of Gradient Decent

- Our rule updating λ_1 according to the Grad. Dec. algorithm is:

$$\lambda_1 = \lambda_1 - \alpha \frac{\partial}{\partial \lambda_1} C(\lambda_1)$$

- We randomly pick an initial value for λ_1 as seen in the graph.



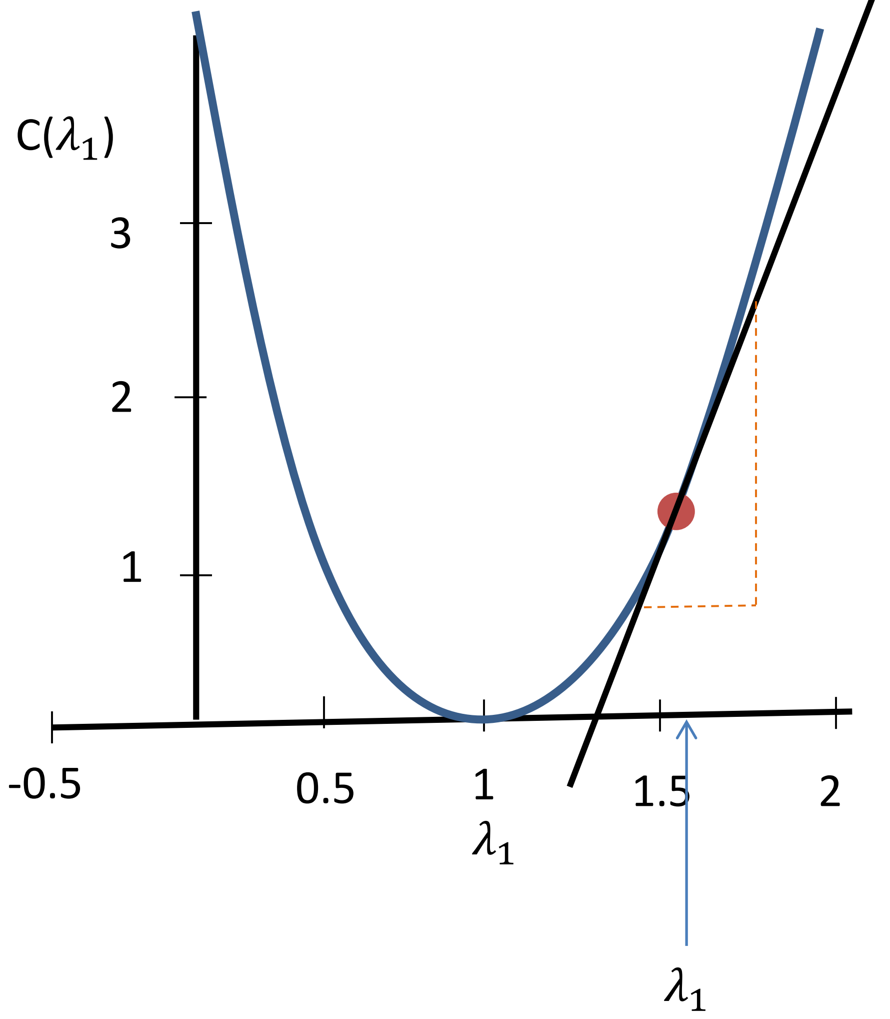
Function C(λ_1)

More Detail on Parameters of Gradient Decent

$$\lambda_1 = \lambda_1 - \alpha \boxed{\frac{\partial}{\partial \lambda_1} C(\lambda_1)}$$

This derivative term allows us to calculate **the slope of a line that forms a tangent to the point** we have selected. We can see the slope of this line is a positive number (slope obviously being height divided by horizontal – dashed lines)

Function C(λ_1)

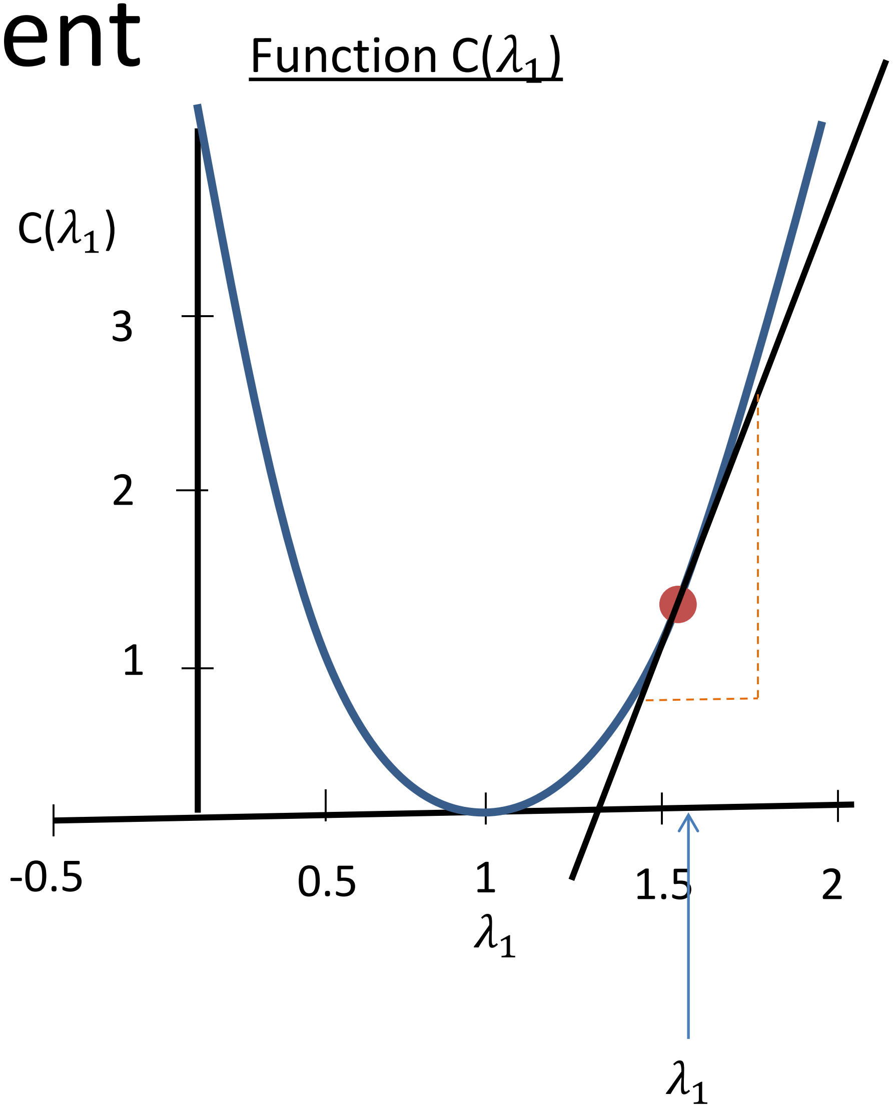


More Detail on Parameters of Gradient Decent

$$\lambda_1 = \lambda_1 - \alpha \boxed{\frac{\partial}{\partial \lambda_1} C(\lambda_1)}$$

The derivative value in this case will be a **positive number**, which is then multiplied by the α (the learning rate).

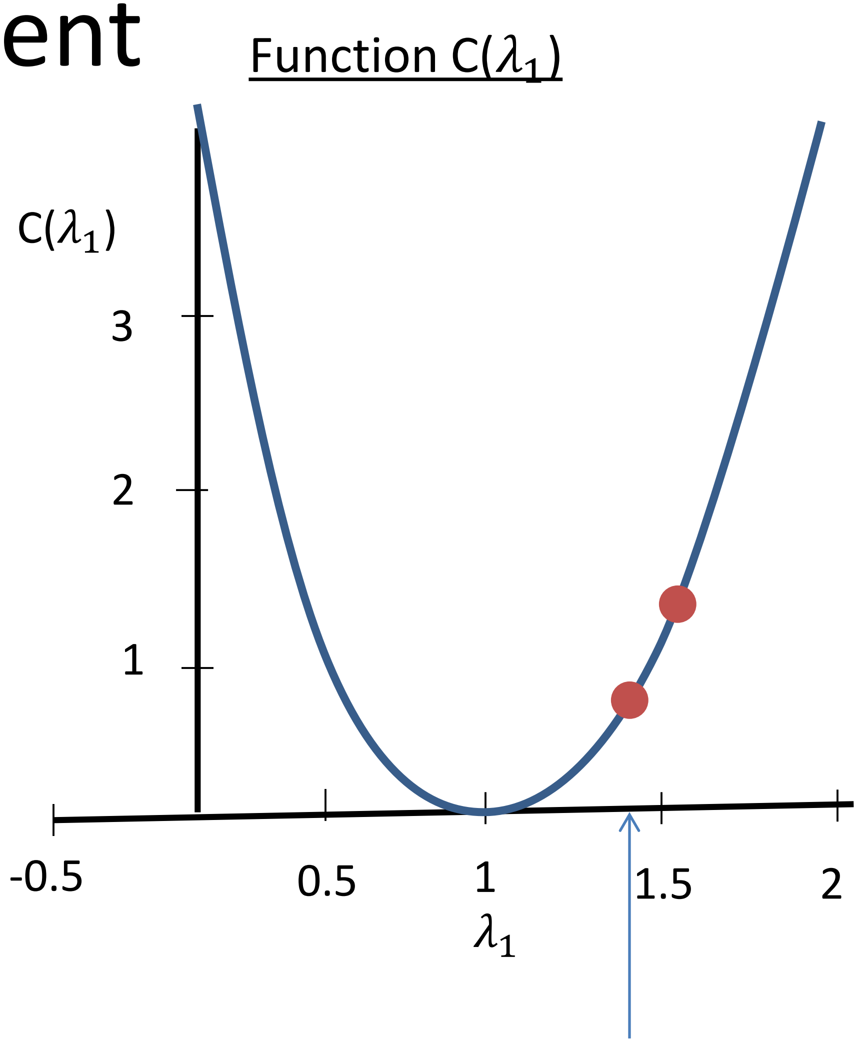
Therefore, this ultimately reduces the value of λ_1 as we are subtracting a small positive number.



More Detail on Parameters of Gradient Decent

$$\lambda_1 = \lambda_1 - \alpha \boxed{\frac{\partial}{\partial \lambda_1} C(\lambda_1)}$$

Notice the value of λ_1 is reduced



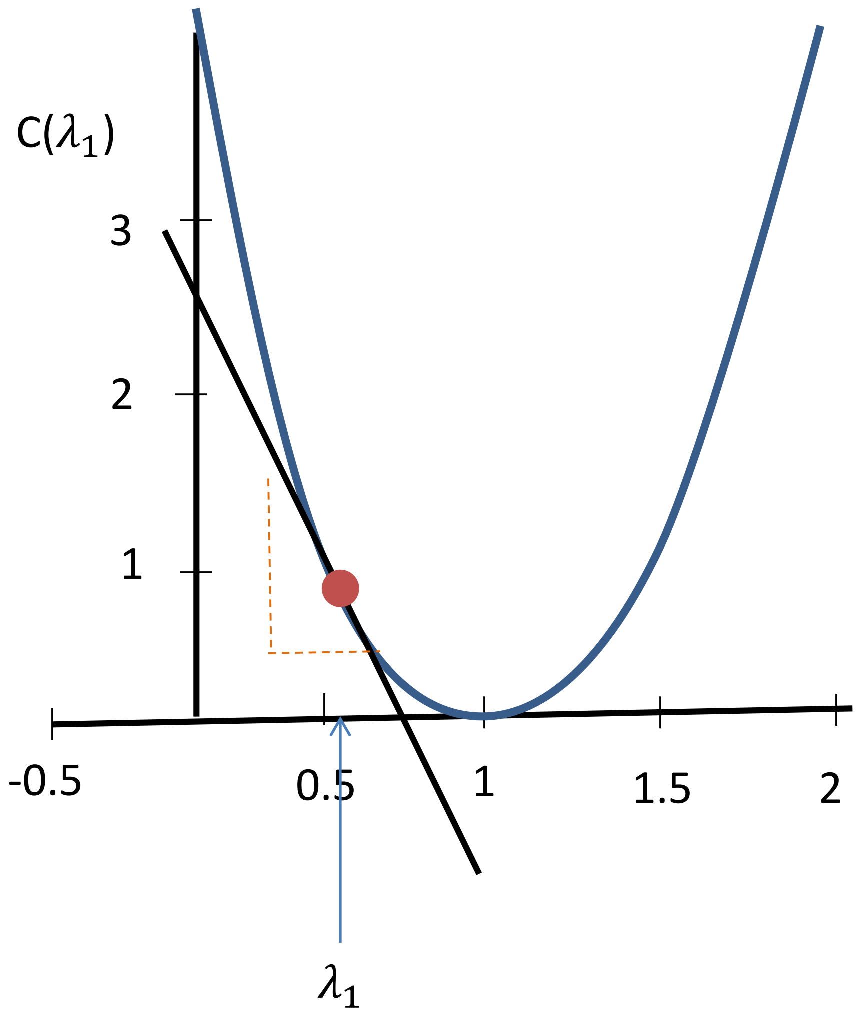
New value of λ_1

More Detail on Parameters of Gradient Decent

$$\lambda_1 = \lambda_1 - \alpha \boxed{\frac{\partial}{\partial \lambda_1} C(\lambda_1)}$$

What happens to the value of λ_1 if we select an initial value of λ_1 as shown in the graph?

Function $C(\lambda_1)$

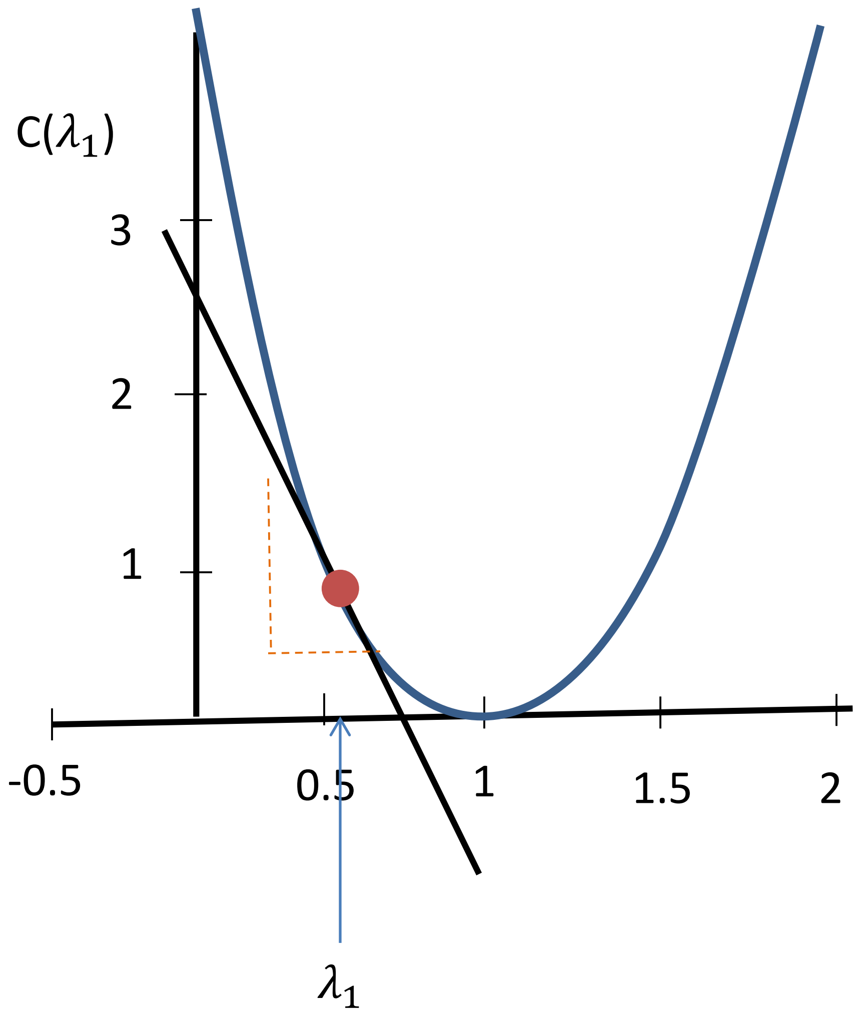


More Detail on Parameters of Gradient Decent

$$\lambda_1 = \lambda_1 - \alpha \boxed{\frac{\partial}{\partial \lambda_1} C(\lambda_1)}$$

The derivative of this line is negative. It is then multiplied by α and subtracted from λ_1. This has the effect of increasing the value of λ_1

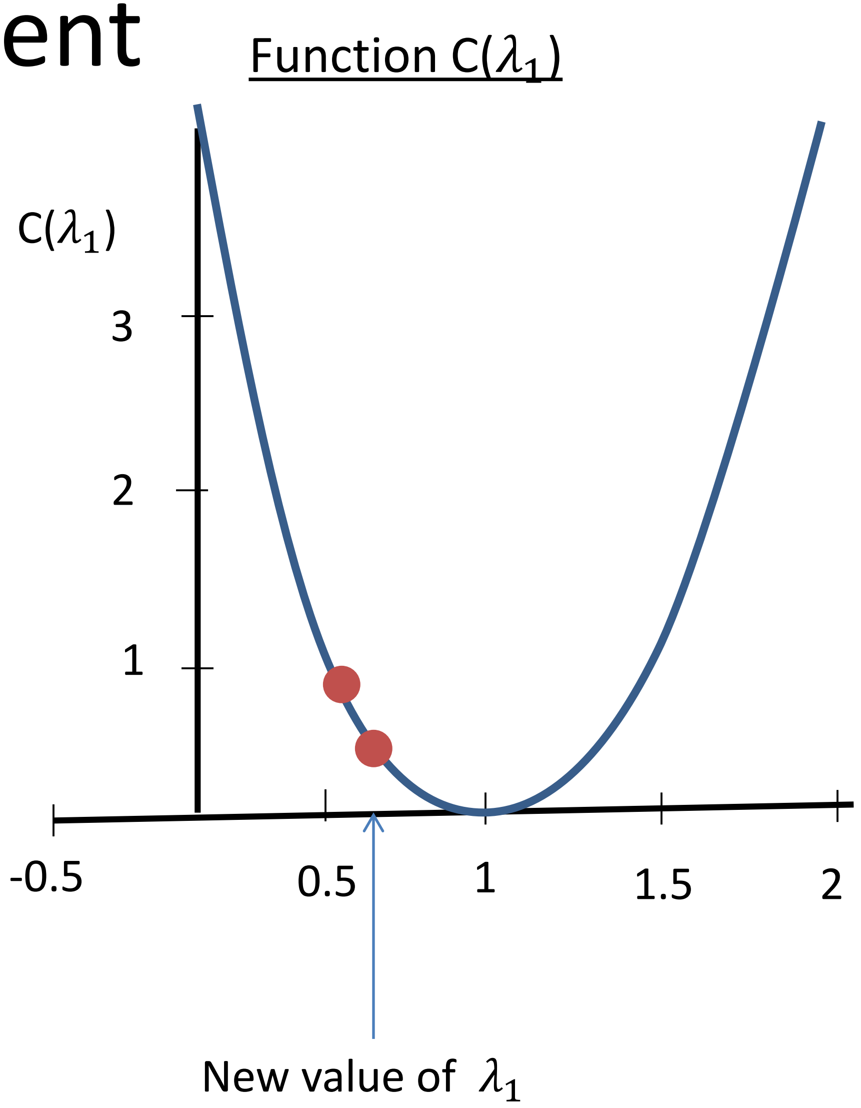
Function C(λ_1)



More Detail on Parameters of Gradient Decent

$$\lambda_1 = \lambda_1 - \alpha \boxed{\frac{\partial}{\partial \lambda_1} C(\lambda_1)}$$

The value of λ_1 is increased and is making it ways towards the minimum. Notice that we continue to move towards the minimum value of λ_1



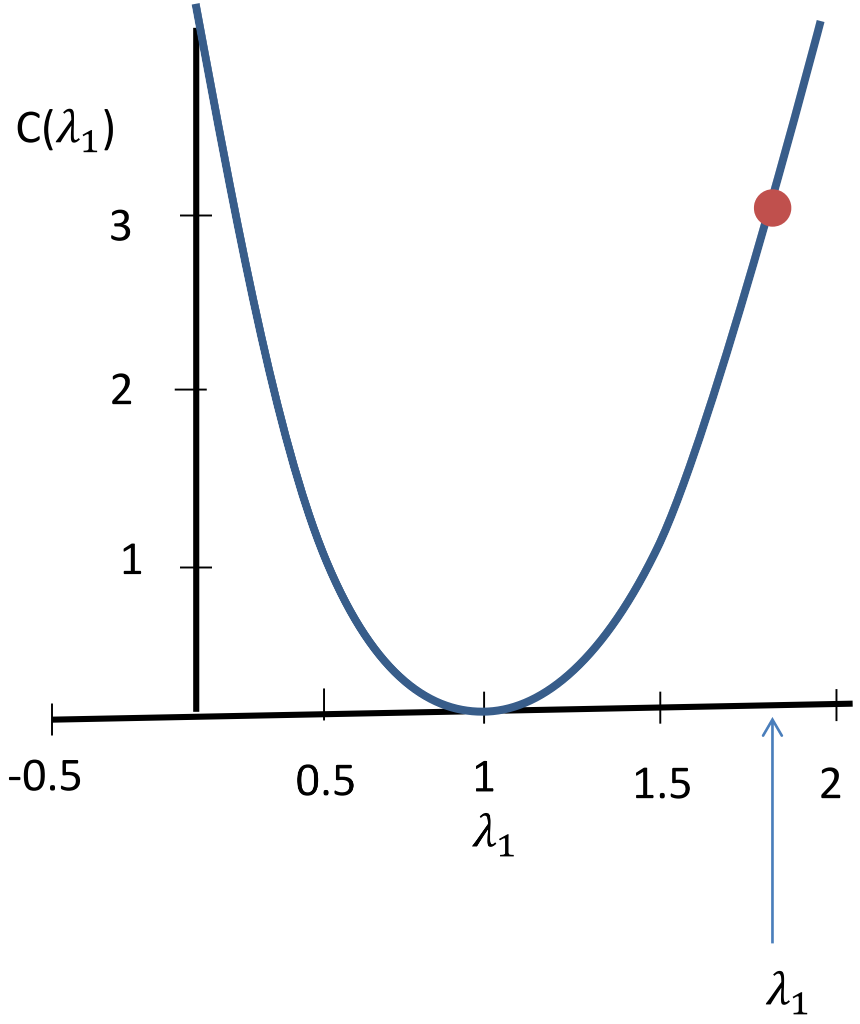
More Detail on Parameters of Gradient Decent

- As we mentioned before **the value of $\boldsymbol{\alpha}$** controls the rate of decent.

$$\lambda_1 = \lambda_1 - \alpha \frac{\partial}{\partial \lambda_1} C(\lambda_1)$$

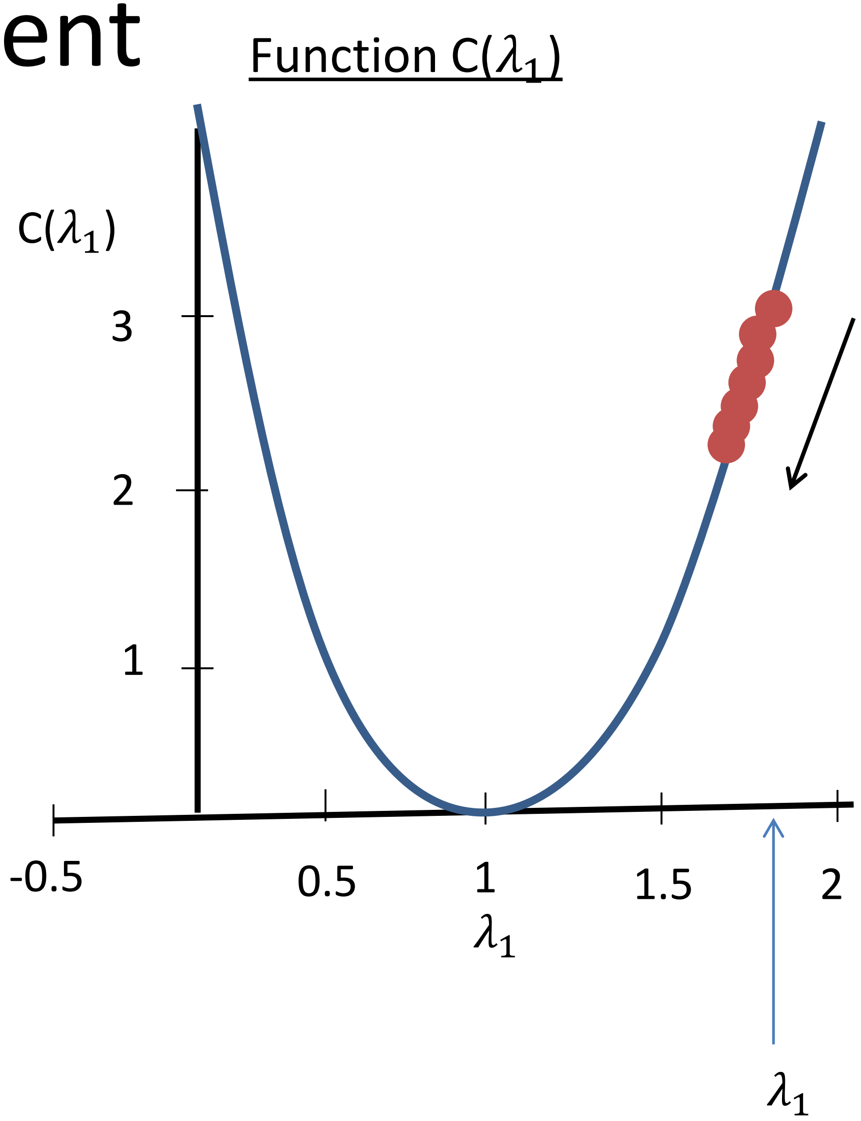
- Consider the scenario where the value of α is very small. What effect do you think it would have on the update of λ_1

Function C(λ_1)



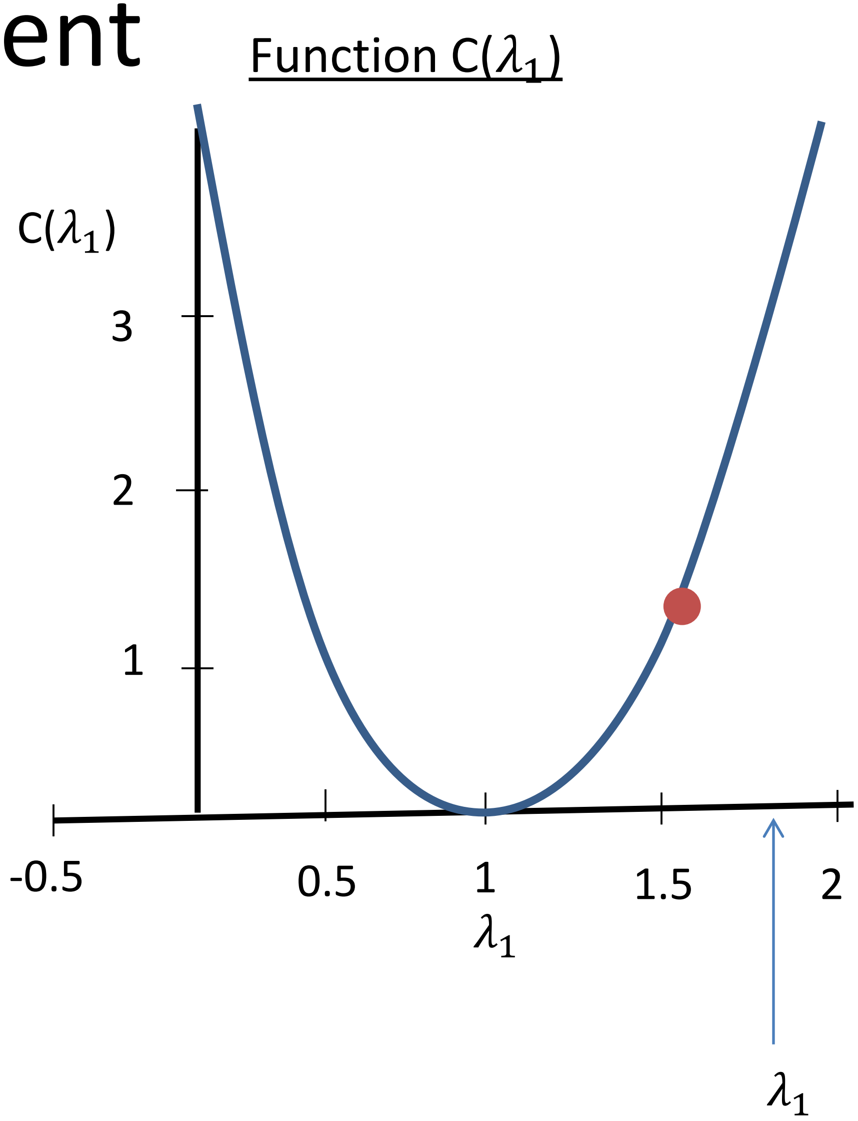
More Detail on Parameters of Gradient Decent

- This would result in very small movement in the value of λ_1 and would cause a long time until we reach convergence.



More Detail on Parameters of Gradient Decent

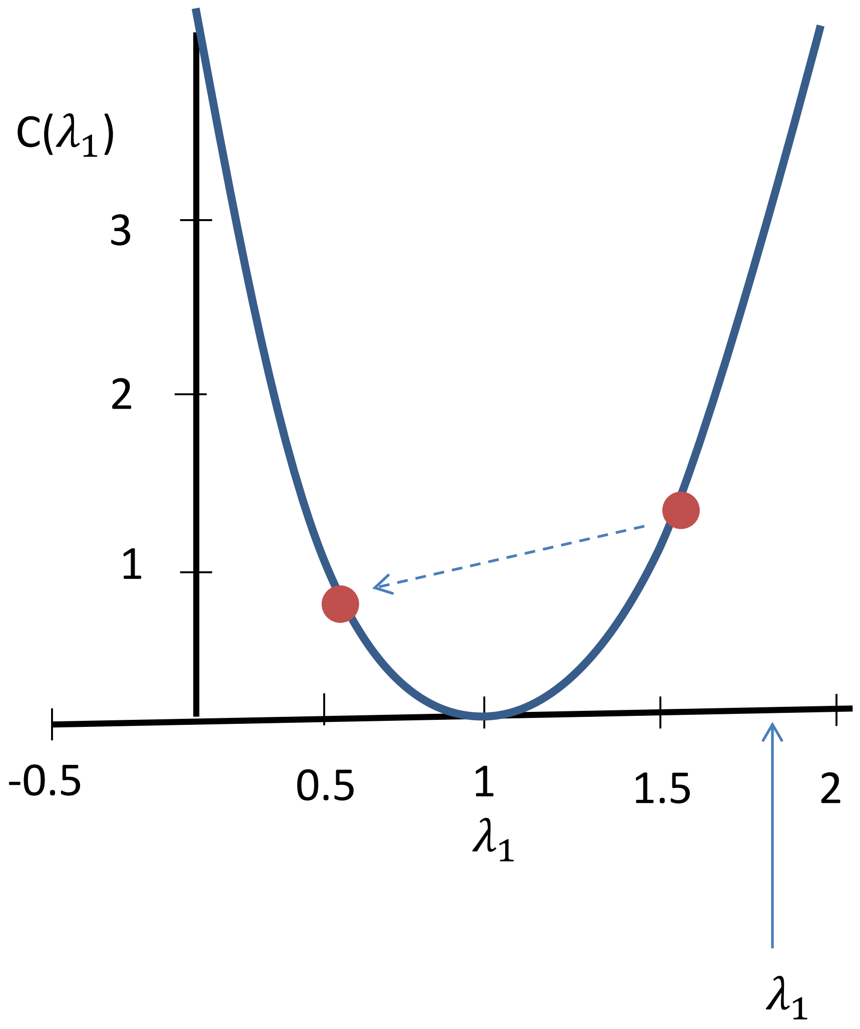
- Conversely if the value of α is very large then the change in value of λ_1 will be very substantial. Why would this be a problem?



More Detail on Parameters of Gradient Decent

- It can cause very large jumps in the value of λ_1 and we could potentially miss the minimum as shown on the graph.

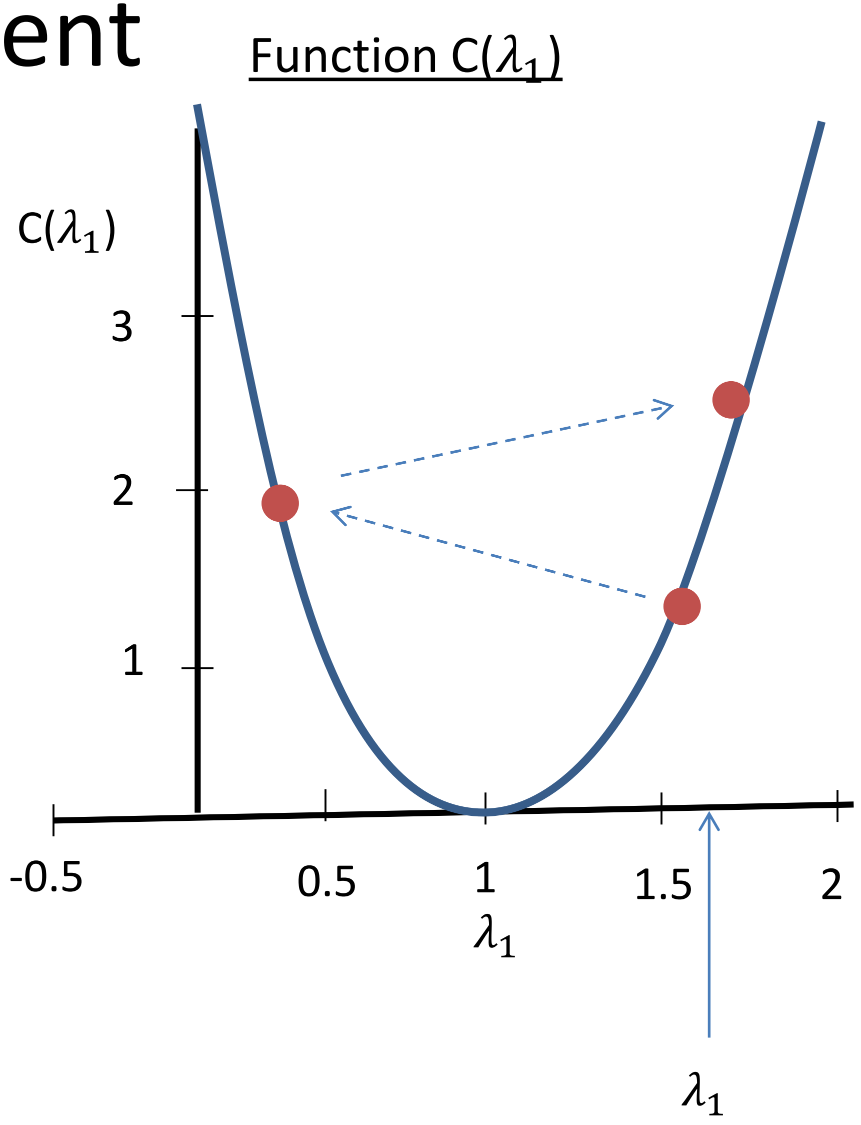
Function C(λ_1)



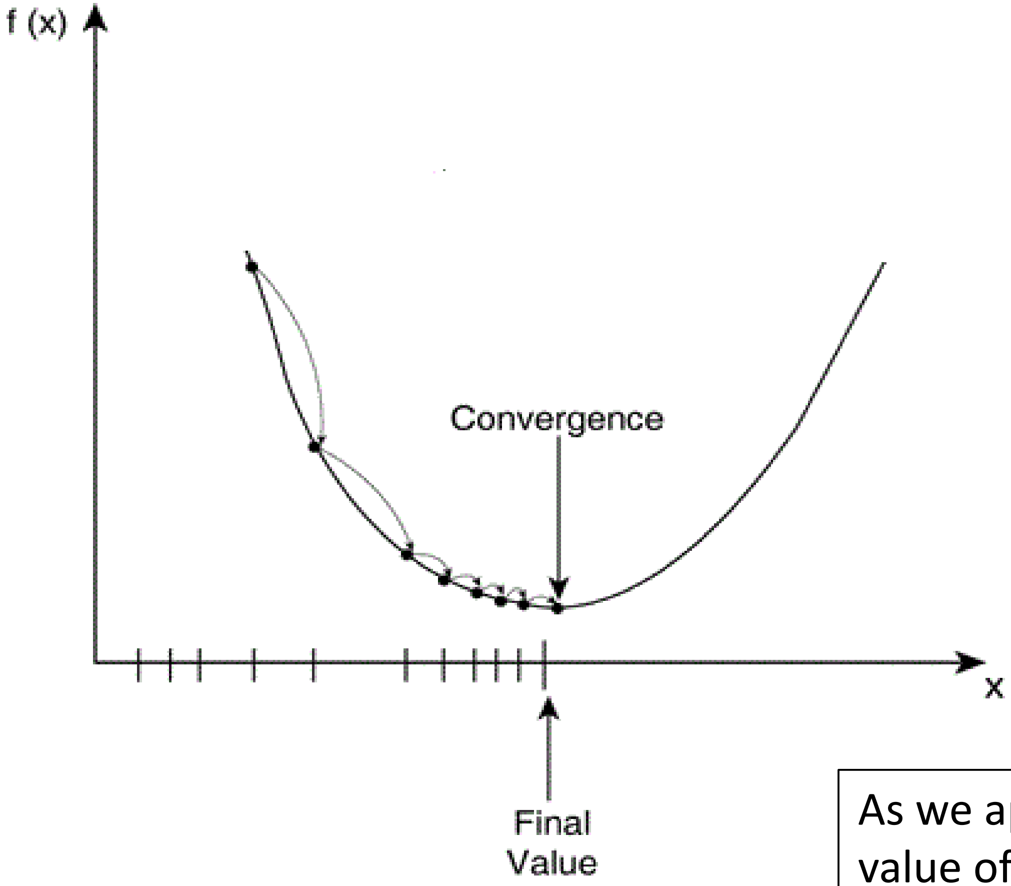
More Detail on Parameters of Gradient Decent

- It can cause very large jumps in the value of λ_1 and we could potentially miss the minimum as shown on the graph.

- **What do you think will happen if we pick λ_1 such that it has the minimum value?**



More Detail on Parameters of Gradient Decent



Can you explain why the size of the steps taken by gradient decent reduce as we move towards the minimum value in this example?

As we approach the local minimum the value of the derivative gets smaller and smaller (change in x with respect to y is smaller therefore the slope value is smaller)

Recap : Applying Gradient Decent to Linear Regression

$$\mathrm{c}(\lambda_1,\ b\ ) \quad = \quad \frac{1}{m+2}\sum_{i=0}^{m}((h(x^i) - y^i))^2$$

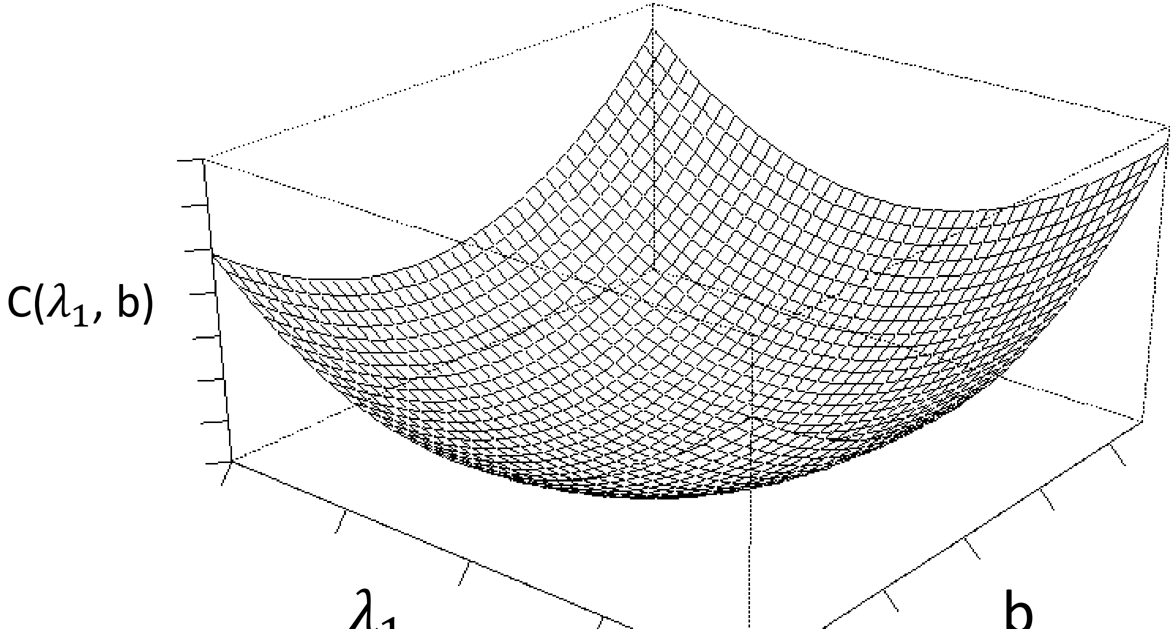
$$minimise_{\lambda_1 b}\mathrm{c}(\lambda_1, b)$$

Our linear regression problem is formalised using this equation. We will solve this using the gradient decent algorithm.

The first thing we need is the derivatives of the above linear regress problem for λ_1 and b

$$\lambda_1 = \lambda_1 - \alpha \boxed{\frac{\partial}{\partial \lambda_1}\mathrm{C}(\lambda_1, b)}$$

$$b = b - \alpha \boxed{\frac{\partial}{\partial b}\mathrm{C}(\lambda_1, b)}$$


C(λ_1, b)
λ_1
b

Derivatives for Linear Regression

$$\lambda_1 = \lambda_1 - \alpha \boxed{\frac{\partial}{\partial \lambda_1} C(\lambda_1, b)}$$

$$b = b - \alpha \boxed{\frac{\partial}{\partial b} C(\lambda_1, b)}$$

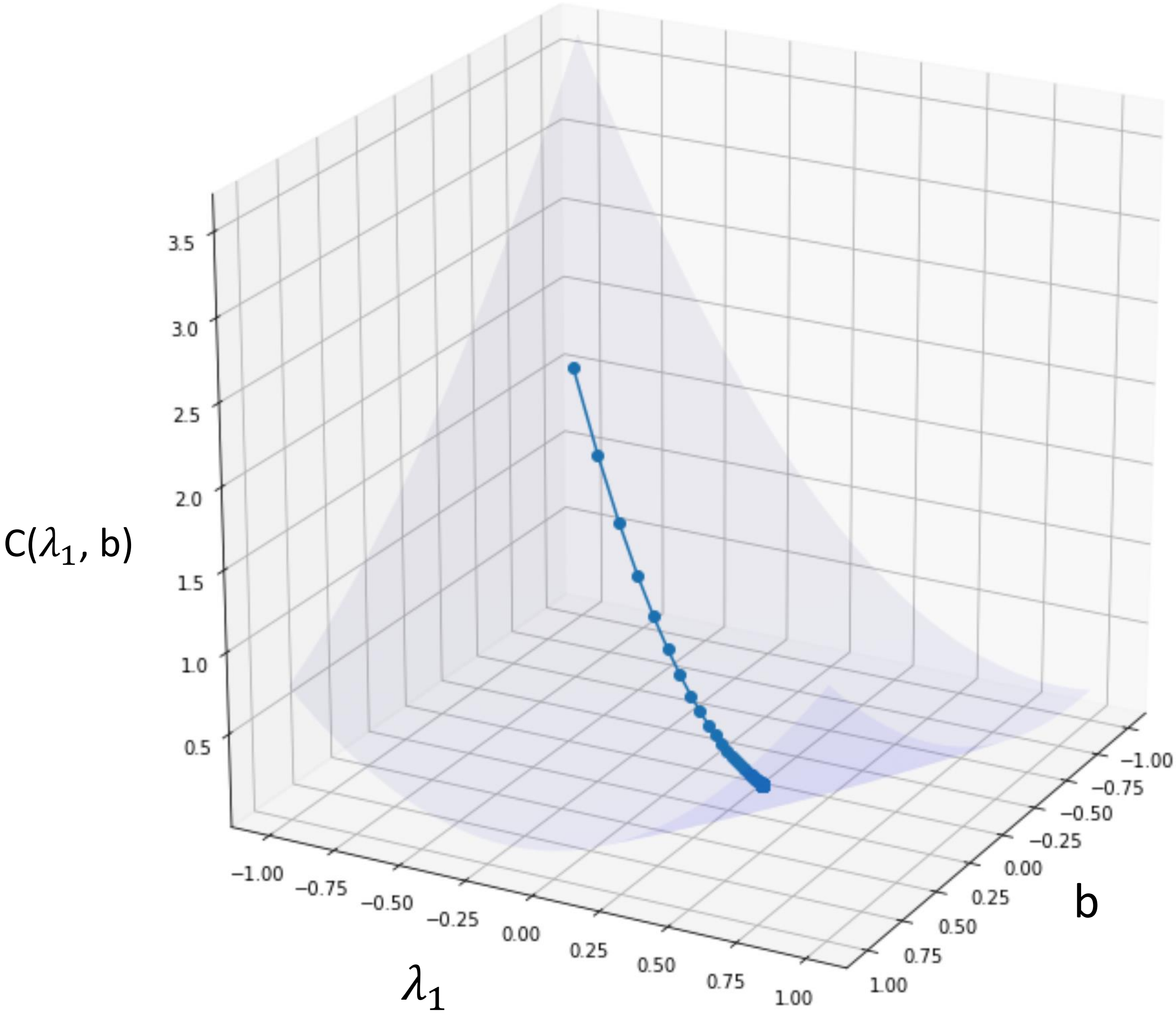
$$\frac{\partial}{\partial \lambda_1} C(\lambda_1,\ b) = \frac{1}{2m} \sum_{i=0}^{m} ((h(x^i) - y^i))(x^i)$$

$$\frac{\partial}{\partial b} C(\lambda_1,\ b) = \frac{1}{2m} \sum_{i=0}^{n} ((h(x^i) - y^i))$$

Gradient Decent Algorithm

```
repeat {
    $\lambda_1 = \lambda_1 - \alpha \frac{\partial}{\partial \lambda_1} C(\lambda_1, b)$
    $b = b - \alpha \frac{\partial}{\partial b} C(\lambda_1, b)$}

repeat {
    $\lambda_1 = \lambda_1 - \alpha \frac{1}{2m} \sum_{i=0}^{m} ((h(x^i) - y^i))(x^i)$
    $b = b - \alpha \frac{1}{2m} \sum_{i=0}^{m} ((h(x^i) - y^i))$
}
```


3.5
3.0
2.5
2.0
1.5
1.0
0.5
C(λ_1, b)
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
λ_1
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
b