Introduction to Linear Regression

Algorithm Theory - Part Three
Cost Function

Linear Regression

- What we know so far:
  - Linear Relationships
    - y = mx+b
  - OLS
    - Solve simple linear regression
  - Not scalable for multiple features
  - Translating real data to Matrix Notation
  - Generalized formula for Beta coefficients

Linear Regression

- Recall we are searching for Beta values for a best-fit line.

$$\hat{y} = \sum_{i=0}^{n} \beta_i x_i$$

Linear Regression

- The equation below simply defines our line, but how to choose beta coefficients?

$$\hat{y} = \sum_{i=0}^{n} \beta_i x_i$$

Linear Regression

- We've decided to define a "best-fit" as **minimizing the squared error.**

$$\hat{y} = \sum_{i=0}^{n} \beta_i x_i$$

Linear Regression

- The residual error for some row ***j*** is:

$$y^j - \hat{y}^j$$

Linear Regression

- Squared Error for some row ***j*** is then:

$$\left(y^j - \hat{y}^j\right)^2$$

Linear Regression

- Sum of squared errors for **m** rows is then:

$$\sum_{j=1}^{m} \left(y^j - \hat{y}^j \right)^2$$

Linear Regression

- Average squared error for **m** rows is then:

$$\frac{1}{m}\sum_{j=1}^{m}\left(y^j - \hat{y}^j\right)^2$$

Linear Regression

- Exactly what we need for a **cost function!**

$$\frac{1}{m}\sum_{j=1}^{m}\left(y^j - \hat{y}^j\right)^2$$

Linear Regression

- Begin by defining a cost function **J.**

$$J(\beta)$$

Linear Regression

- A **cost function** is defined by some measure of error.

$$J(\beta)$$

Linear Regression

- This means we wish to **minimize** the cost function.

$$J(\beta)$$

Linear Regression

- Our cost function can be defined by the squared error:

$$J(\beta) = \frac{1}{2m} \sum_{j=1}^{m} \left(y^j - \hat{y}^j \right)^2$$

Linear Regression

- Note lowercase j is the specific data row.

$$J(\beta) = \frac{1}{2m} \sum_{j=1}^{m} \left(y^j - \hat{y}^j \right)^2$$

Linear Regression

- Want to minimize cost for set of Betas.

$$J(\beta) = \frac{1}{2m} \sum_{j=1}^{m} \left(y^j - \hat{y}^j \right)^2$$

Linear Regression

- Error between real **y** and predicted **ŷ**

$$J(\beta) = \frac{1}{2m} \sum_{j=1}^{m} \left(y^j - \hat{y}^j \right)^2$$

Linear Regression

- Squaring corrects for negative and positive errors.

$$J(\beta) = \frac{1}{2m} \sum_{j=1}^{m} \left(y^j - \hat{y}^j \right)^2$$

Linear Regression

- Summing error for m rows.

$$J(\beta) = \frac{1}{2m} \sum_{j=1}^{m} \left(y^j - \hat{y}^j \right)^2$$

Linear Regression

- Divide by **m** to get **mean**

$$J(\beta) = \frac{1}{2m} \sum_{j=1}^{m} \left(y^j - \hat{y}^j \right)^2$$

Linear Regression

- Additional ½ is for convenience for derivative.

$$J(\beta) = \frac{1}{2m} \sum_{j=1}^{m} \left(y^j - \hat{y}^j \right)^2$$

Linear Regression

- What is $\hat{y}$?

$$J(\beta) = \frac{1}{2m} \sum_{j=1}^{m} \left(y^j - \hat{y}^j \right)^2$$

Linear Regression

- It will be a function of **Betas** and **Features!**

$$J(\beta) = \frac{1}{2m} \sum_{j=1}^{m} \left(y^j - \hat{y}^j \right)^2$$

$$= \frac{1}{2m} \sum_{j=1}^{m} \left(y^j - \sum_{i=0}^{n} \beta_i x_i^j \right)^2$$

Linear Regression

- Recall from calculus to minimize a function we can take its derivative and set it equal to zero.

$$\frac{\partial J}{\partial \beta_k}(\boldsymbol{\beta}) = \frac{\partial}{\partial \beta_k}\left(\frac{1}{2m}\sum_{j=1}^{m}\left(y^j - \sum_{i=0}^{n}\beta_i x_i^j\right)^2\right)$$

$$= \frac{1}{m}\sum_{j=1}^{m}\left(y^j - \sum_{i=0}^{n}\beta_i x_i^j\right)(-x_k^j)$$

Linear Regression

- Recall from calculus to minimize a function we can take its derivative and set it equal to zero.

$$\frac{\partial J}{\partial \beta_k}(\boldsymbol{\beta}) = \frac{\partial}{\partial \beta_k}\left(\frac{1}{2m}\sum_{j=1}^{m}\left(y^j - \sum_{i=0}^{n}\beta_i x_i^j\right)^2\right)$$

$$= \frac{1}{m}\sum_{j=1}^{m}\left(y^j - \sum_{i=0}^{n}\beta_i x_i^j\right)(-x_k^j)$$

Linear Regression

- Recall from calculus to minimize a function we can take its derivative and set it equal to zero.

$$\frac{\partial J}{\partial \beta_k}(\boldsymbol{\beta}) = \frac{\partial}{\partial \beta_k}\left(\frac{1}{2m}\sum_{j=1}^{m}\left(y^j - \sum_{i=0}^{n}\beta_i x_i^j\right)^2\right)$$

$$= \frac{1}{m}\sum_{j=1}^{m}\left(y^j - \sum_{i=0}^{n}\beta_i x_i^j\right)(-x_k^j)$$

Linear Regression

- Unfortunately, it is not scalable to try to get an analytical solution to minimize this cost function.
- In the next lecture we will learn to use **gradient descent** to minimize this **cost function.**

Introduction to Linear Regression

Algorithm Theory - Part Four
Gradient Descent

Linear Regression

- We just figured out a **cost function** to minimize!
- Taking the cost function derivative and then solving for zero to get the set of Beta coefficients will be too difficult to solve directly through an analytical solution.

Linear Regression

- Instead we can describe this cost function through vectorized matrix notation and use **gradient descent** to have a computer figure out the set of Beta coefficient values that minimize the **cost/loss** function.

Linear Regression

- Our goals:
  - Find a set of Beta coefficient values that minimizes the error (cost function)
  - Leverage computational power instead of having to manually attempt to analytically solve the derivative.

Linear Regression

- Recall we now have the derivative of the cost function:

$$\frac{\partial J}{\partial \beta_k}(\boldsymbol{\beta}) = \frac{1}{m}\sum_{j=1}^{m}\left(y^j - \sum_{i=0}^{n}\beta_i x_i^j\right)(-x_k^j)$$

Linear Regression

- Also recall our data will be in the form of a matrix **X** with a vector of labels **y.**

$$\frac{\partial J}{\partial \beta_k}(\beta) = \frac{1}{m}\sum_{j=1}^{m}\left(y^j - \sum_{i=0}^{n}\beta_i x_i^j\right)(-x_k^j)$$

Linear Regression

- Which means we need a β for each feature, so we can express a vector of β values.

$$\frac{\partial J}{\partial \beta_k}(\boldsymbol{\beta}) = \frac{1}{m}\sum_{j=1}^{m}\left(y^j - \sum_{i=0}^{n}\beta_i x_i^j\right)(-x_k^j)$$

Linear Regression

- Use a **gradient** to express the derivative of the cost function with respect to each β

$$\nabla_{\beta} J = \begin{bmatrix} \frac{\partial J}{\partial \beta_0} \\ \vdots \\ \frac{\partial J}{\partial \beta_n} \end{bmatrix}$$

Linear Regression

- We can plug in our equation of the derivative of the loss function.

$$\nabla_{\beta} J = \begin{bmatrix} \frac{\partial J}{\partial \beta_0} \\ \vdots \\ \frac{\partial J}{\partial \beta_n} \end{bmatrix}$$

Linear Regression

- We also already know what this cost function derivative is equal to:

$$\nabla_{\beta} J = \begin{bmatrix} -\frac{1}{m} \sum_{j=1}^{m} \left(y^j - \sum_{i=0}^{n} \beta_i x_i^j \right) x_0^j \\ \vdots \\ -\frac{1}{m} \sum_{j=1}^{m} \left(y^j - \sum_{i=0}^{n} \beta_i x_i^j \right) x_n^j \end{bmatrix}$$

Linear Regression

- We also know we can vectorize our data:

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^1 & x_2^1 & \dots & x_n^1 \\ 1 & x_1^2 & x_2^2 & \dots & x_n^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^m & x_2^m & \dots & x_n^m \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_n \end{bmatrix}$$

Linear Regression

- We can split the gradient of the cost function into two parts:

$$\nabla_\beta J = \begin{bmatrix} -\frac{1}{m}\sum_{j=1}^{m}\left(y^j - \sum_{i=0}^{n}\beta_i x_i^j\right)x_0^j \\ \vdots \\ -\frac{1}{m}\sum_{j=1}^{m}\left(y^j - \sum_{i=0}^{n}\beta_i x_i^j\right)x_n^j \end{bmatrix}$$

Linear Regression

- We can split the gradient of the cost function into two parts:

$$\nabla_\beta J = \begin{bmatrix} -\frac{1}{m}\sum_{j=1}^{m}\left(y^j - \sum_{i=0}^{n}\beta_i x_i^j\right)x_0^j \\ \vdots \\ -\frac{1}{m}\sum_{j=1}^{m}\left(y^j - \sum_{i=0}^{n}\beta_i x_i^j\right)x_n^j \end{bmatrix}$$

Linear Regression

- We can split the gradient of the cost function into two parts:

$$\nabla_\beta J = \begin{bmatrix} -\frac{1}{m}\sum_{j=1}^{m}\left(y^j - \sum_{i=0}^{n}\beta_i x_i^j\right)x_0^j \\ \vdots \\ -\frac{1}{m}\sum_{j=1}^{m}\left(y^j - \sum_{i=0}^{n}\beta_i x_i^j\right)x_n^j \end{bmatrix}$$

Linear Regression

- This then results in the following:

$$\nabla_\beta J = -\frac{1}{m}\begin{bmatrix} \sum_{j=1}^{m} y^j x_0^j \\ \vdots \\ \sum_{j=1}^{m} y^j x_n^j \end{bmatrix} + \frac{1}{m}\begin{bmatrix} \sum_{j=1}^{m}\sum_{i=0}^{n} \beta_i x_i^j x_0^j \\ \vdots \\ \sum_{j=1}^{m}\sum_{i=0}^{n} \beta_i x_i^j x_n^j \end{bmatrix}$$

Linear Regression

- We can now calculate the gradient for any set of Beta values!

$$\nabla_\beta J = -\frac{1}{m}\begin{bmatrix} \sum_{j=1}^{m} y^j x_0^j \\ \vdots \\ \sum_{j=1}^{m} y^j x_n^j \end{bmatrix} + \frac{1}{m}\begin{bmatrix} \sum_{j=1}^{m}\sum_{i=0}^{n} \beta_i x_i^j x_0^j \\ \vdots \\ \sum_{j=1}^{m}\sum_{i=0}^{n} \beta_i x_i^j x_n^j \end{bmatrix}$$

Linear Regression

- In theory we could now guess and check Beta values that minimize this gradient.

$$\nabla_\beta J = -\frac{1}{m}\begin{bmatrix} \sum_{j=1}^{m} y^j x_0^j \\ \vdots \\ \sum_{j=1}^{m} y^j x_n^j \end{bmatrix} + \frac{1}{m}\begin{bmatrix} \sum_{j=1}^{m}\sum_{i=0}^{n} \beta_i x_i^j x_0^j \\ \vdots \\ \sum_{j=1}^{m}\sum_{i=0}^{n} \beta_i x_i^j x_n^j \end{bmatrix}$$

Linear Regression

- Note how the Beta coefficients are the only unknown variable here:

$$\nabla_\beta J = -\frac{1}{m}\begin{bmatrix} \sum_{j=1}^{m} y^j x_0^j \\ \vdots \\ \sum_{j=1}^{m} y^j x_n^j \end{bmatrix} + \frac{1}{m}\begin{bmatrix} \sum_{j=1}^{m}\sum_{i=0}^{n} \beta_i x_i^j x_0^j \\ \vdots \\ \sum_{j=1}^{m}\sum_{i=0}^{n} \beta_i x_i^j x_n^j \end{bmatrix}$$

Linear Regression

- What is the best way to “guess” at the correct Beta values that minimize the gradient?

$$\nabla_\beta J = -\frac{1}{m}\begin{bmatrix} \sum_{j=1}^{m} y^j x_0^j \\ \vdots \\ \sum_{j=1}^{m} y^j x_n^j \end{bmatrix} + \frac{1}{m}\begin{bmatrix} \sum_{j=1}^{m}\sum_{i=0}^{n} \beta_i x_i^j x_0^j \\ \vdots \\ \sum_{j=1}^{m}\sum_{i=0}^{n} \beta_i x_i^j x_n^j \end{bmatrix}$$

Linear Regression

- We can use gradient descent to computationally search for the coefficients that minimize this gradient.
- Let's visually explore what this looks like in the case of a single Beta value.

Linear Regression

- Given a cost function of J(β) how can we computationally search for the correct value of β that minimizes the gradient of the cost function?
- What would the search process look like for single β value?

Linear Regression

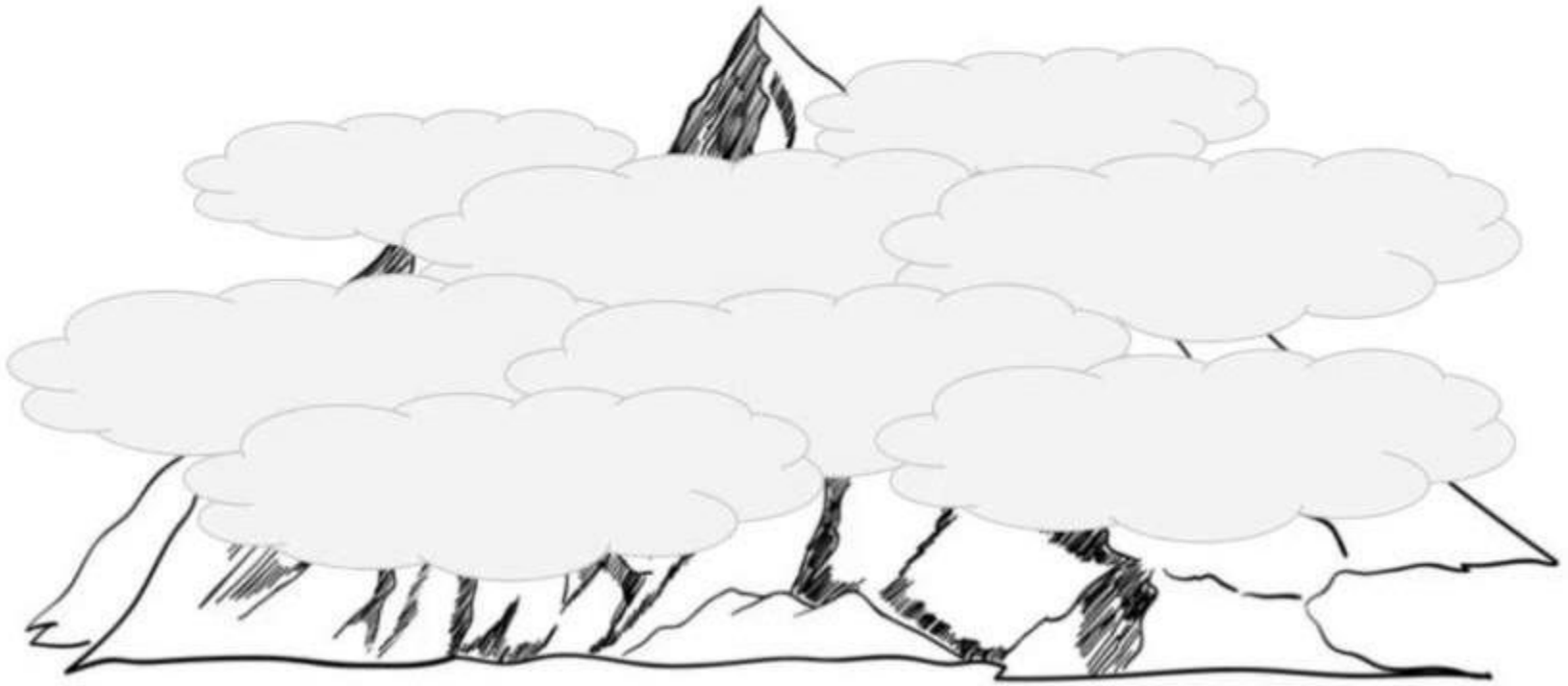
- Common mountain analogy

Linear Regression

- Common mountain analogy

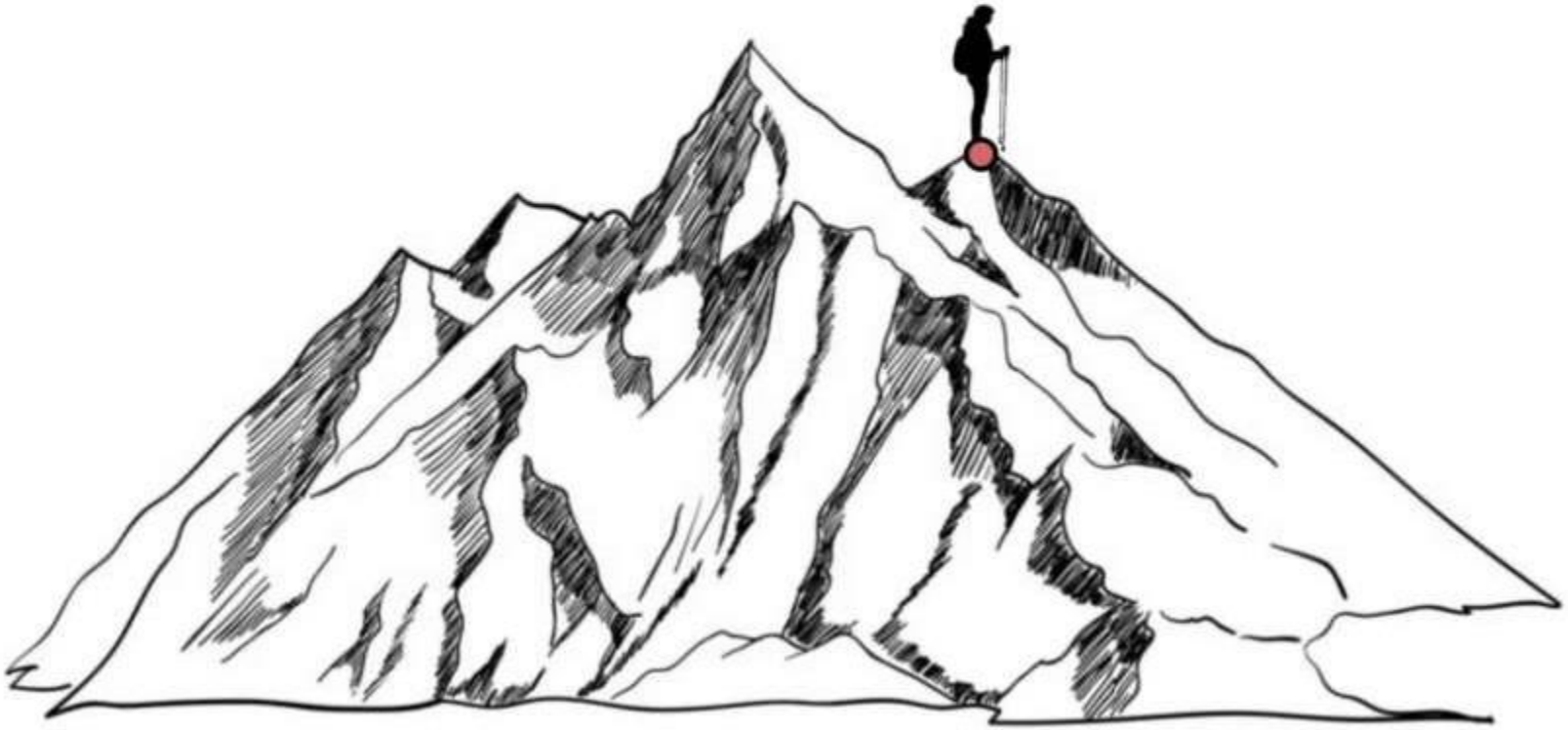
Linear Regression

- Common mountain analogy



Linear Regression

- Common mountain analogy

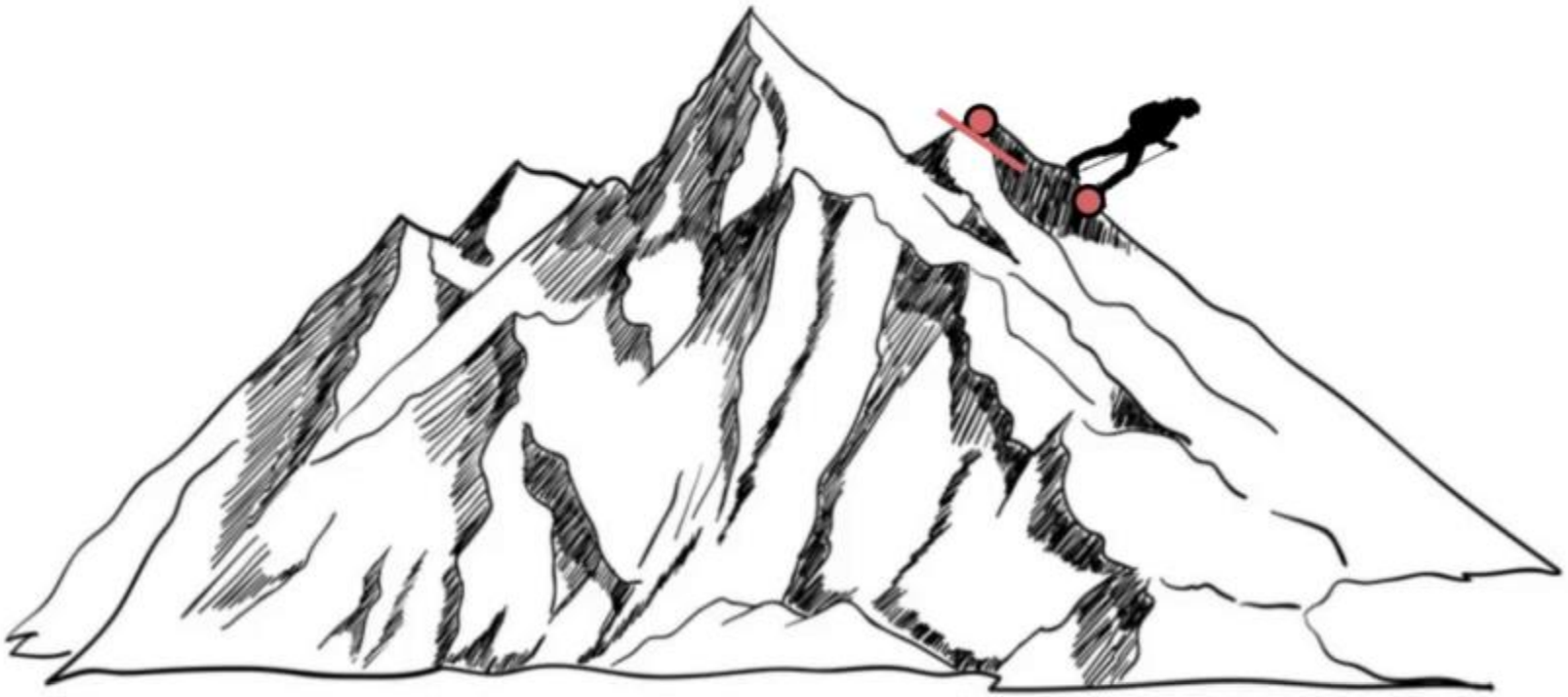


Linear Regression

- Common mountain analogy

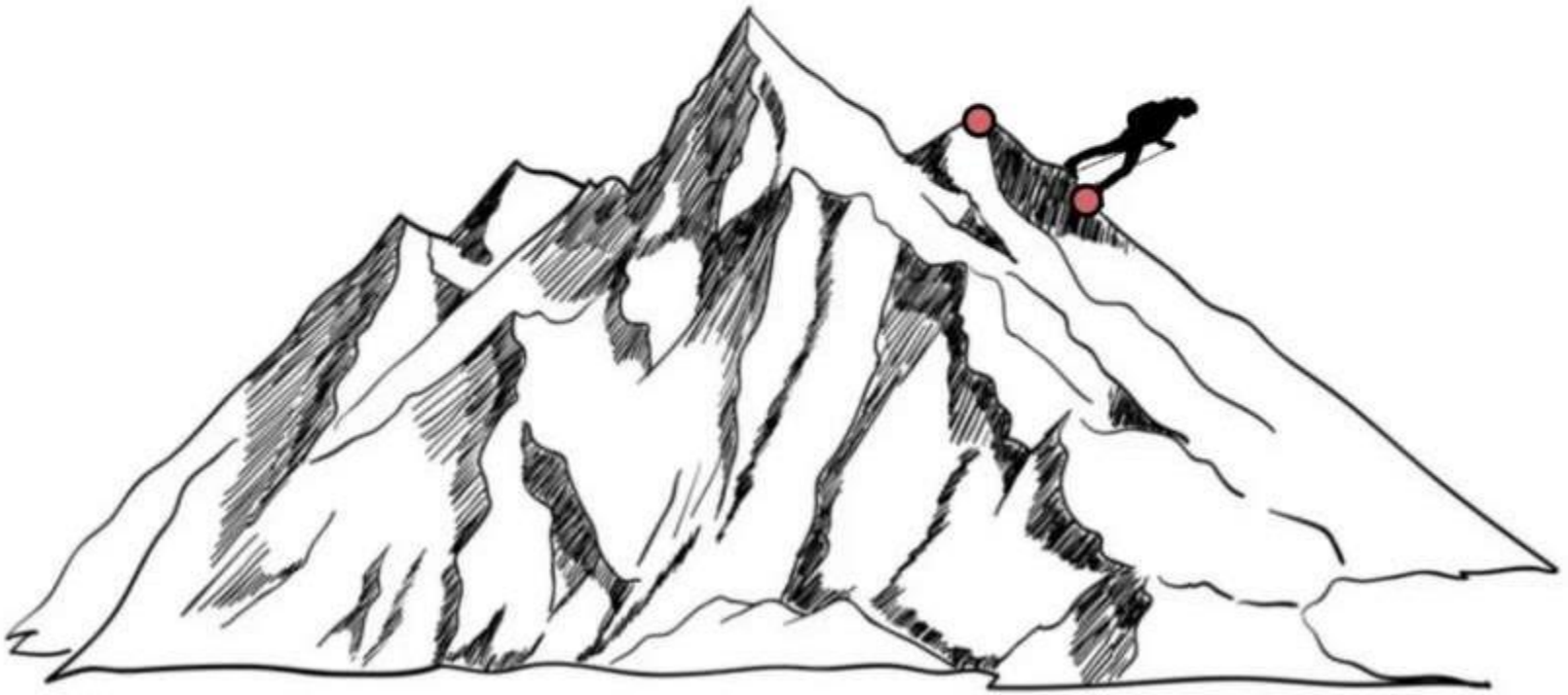
Linear Regression

- Common mountain analogy



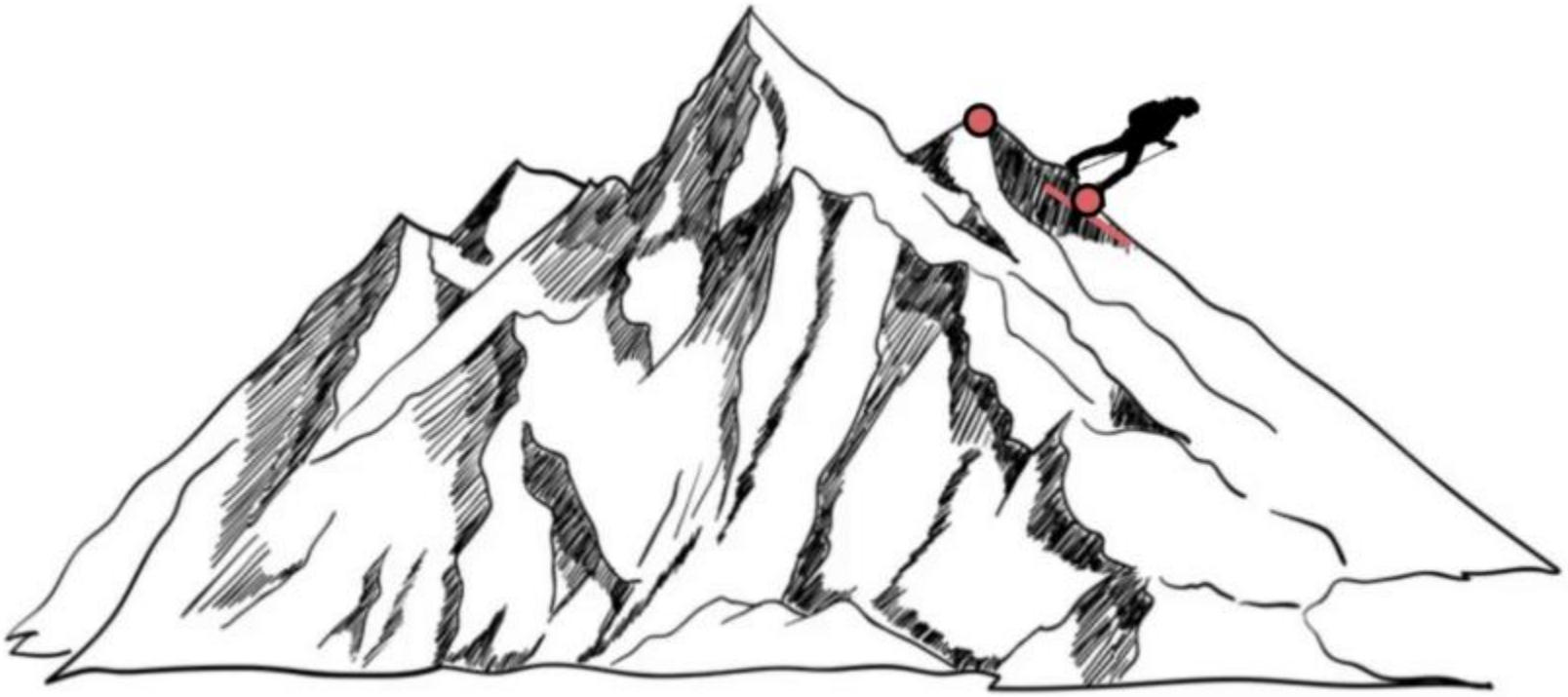
Linear Regression

- Common mountain analogy



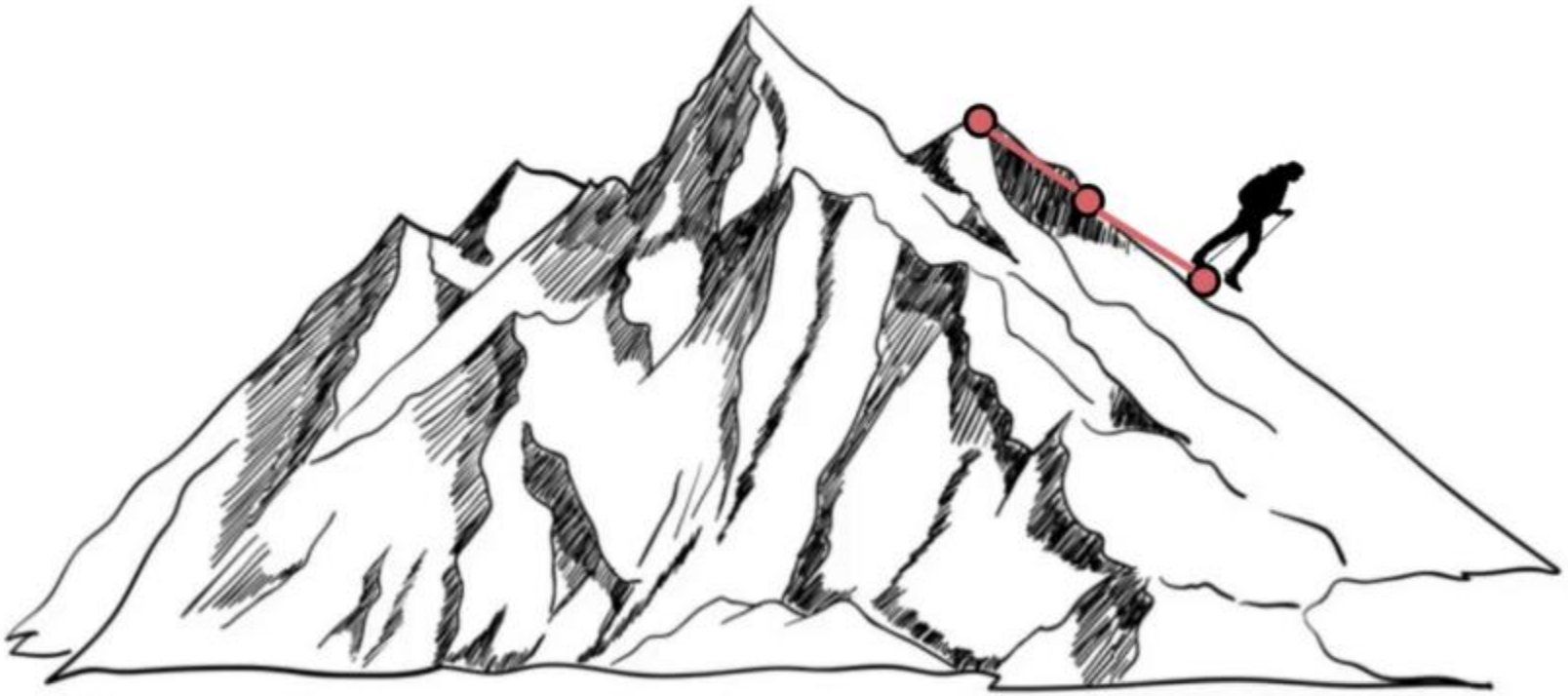
Linear Regression

- Common mountain analogy



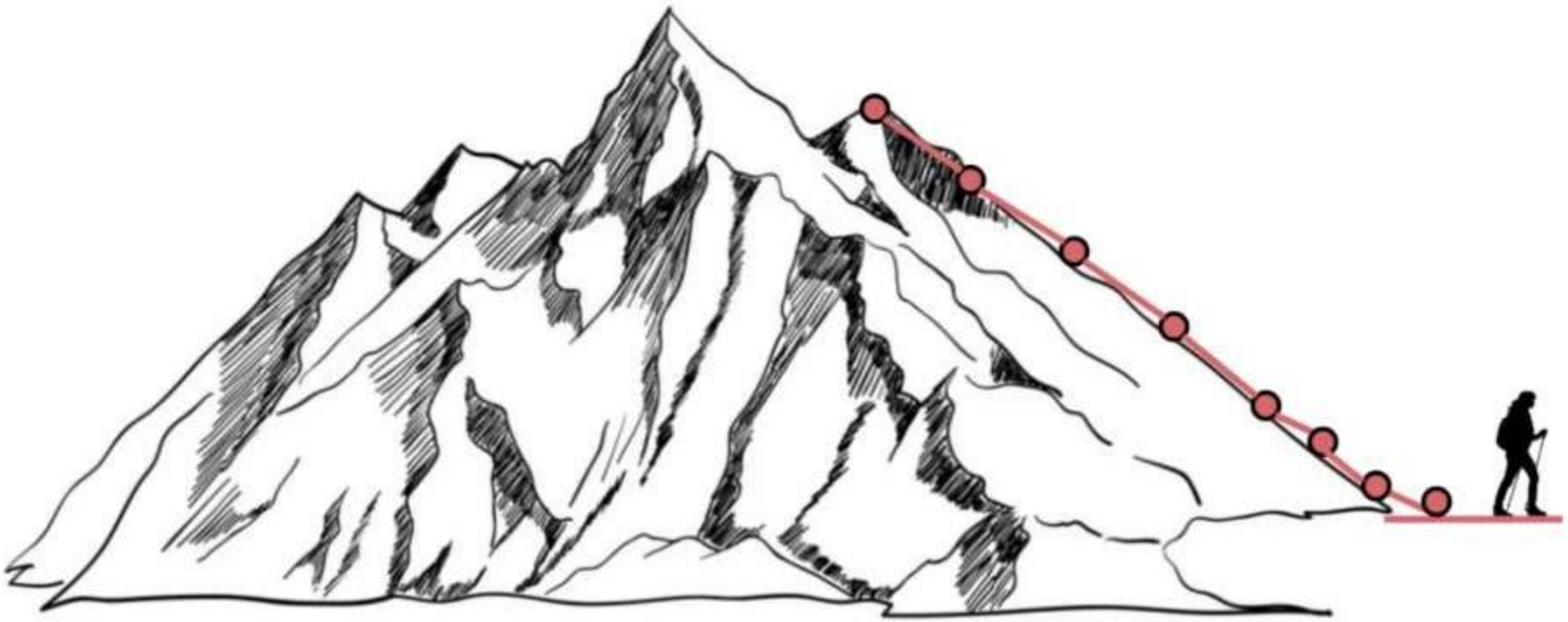
Linear Regression

- Common mountain analogy



Linear Regression

- Common mountain analogy

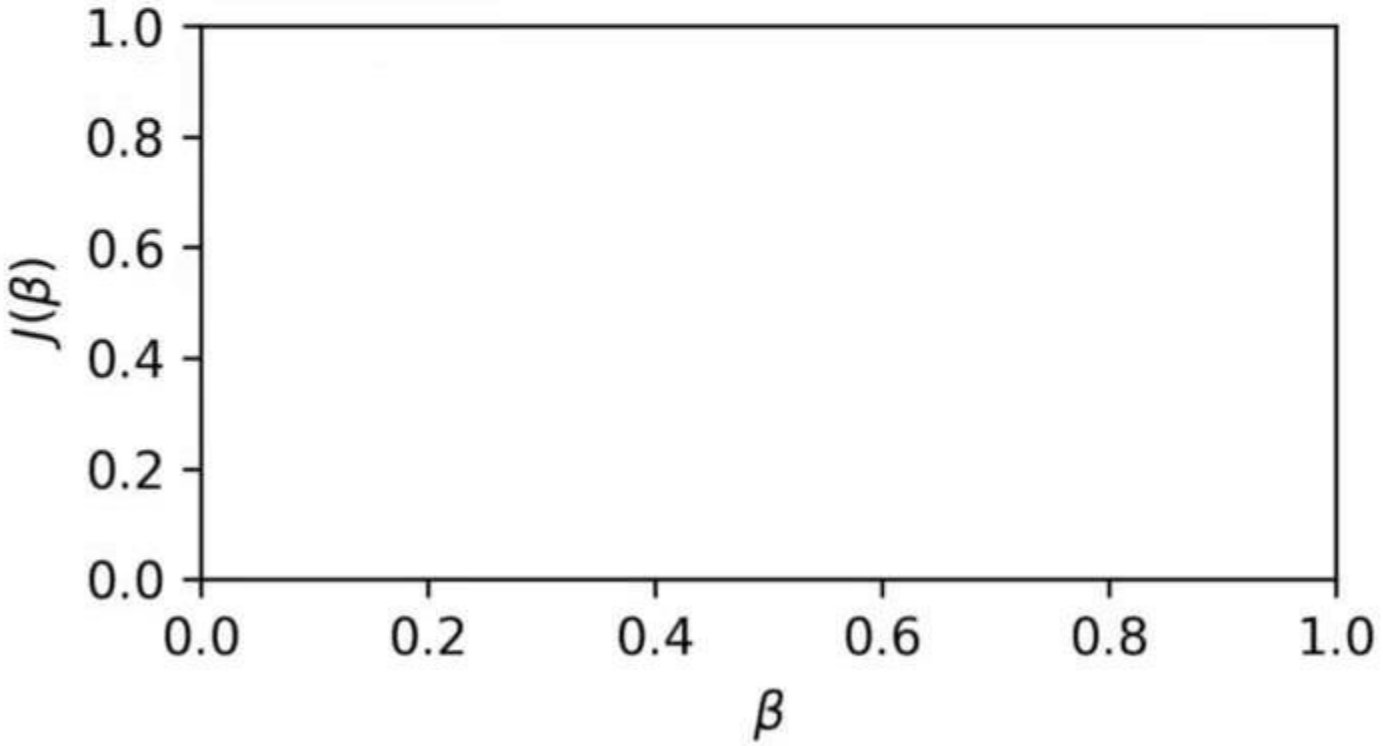


Linear Regression

- This is exactly what gradient descent does!
- It even looks similar for the case of a single coefficient search.

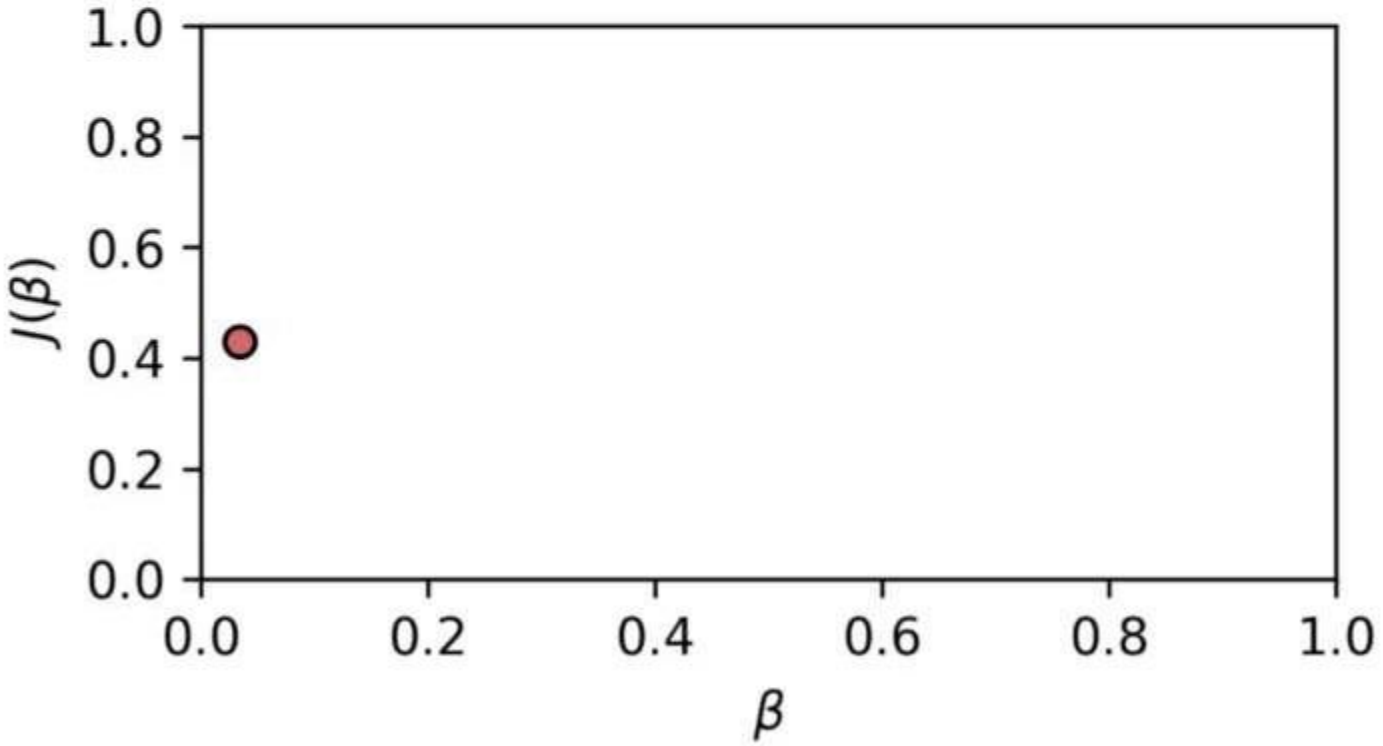
Linear Regression

- 1 dimensional cost function (single Beta)



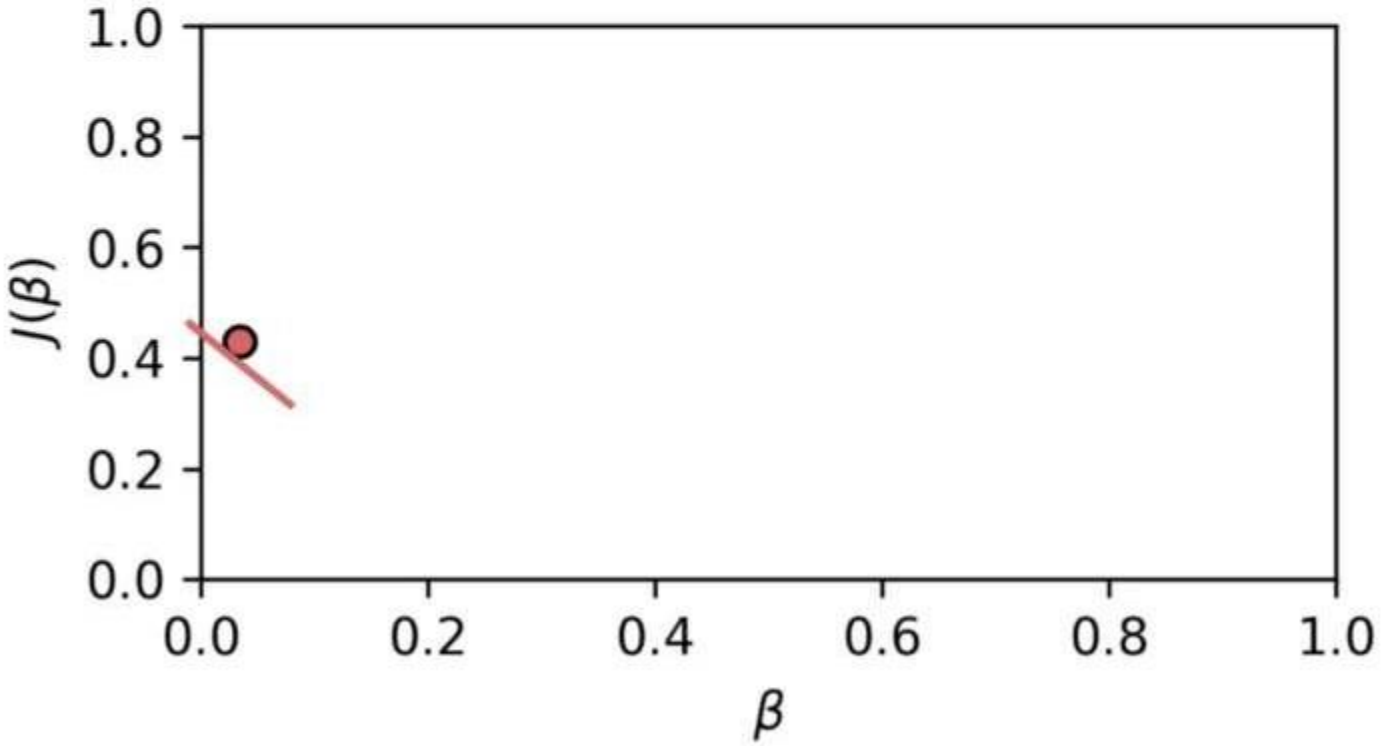
Linear Regression

- Choose a starting point



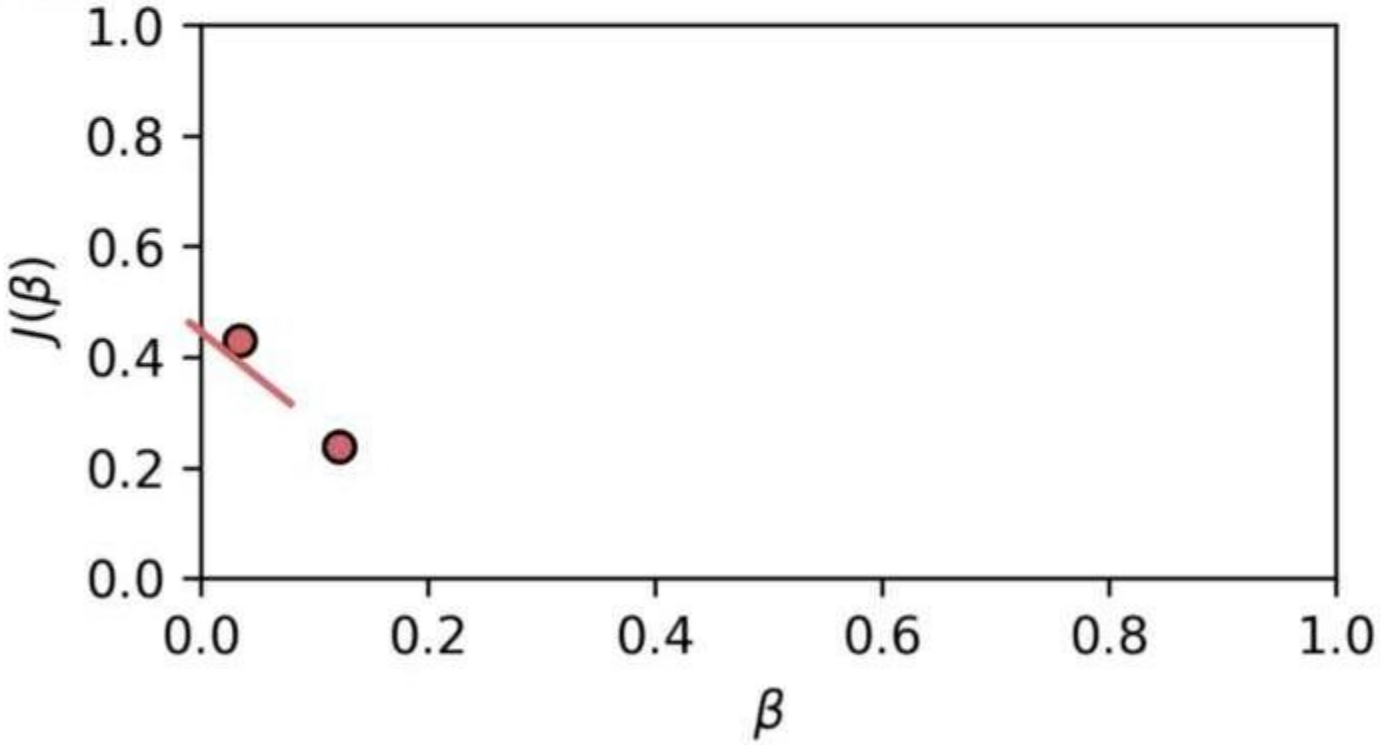
Linear Regression

- Calculate gradient at that point



Linear Regression

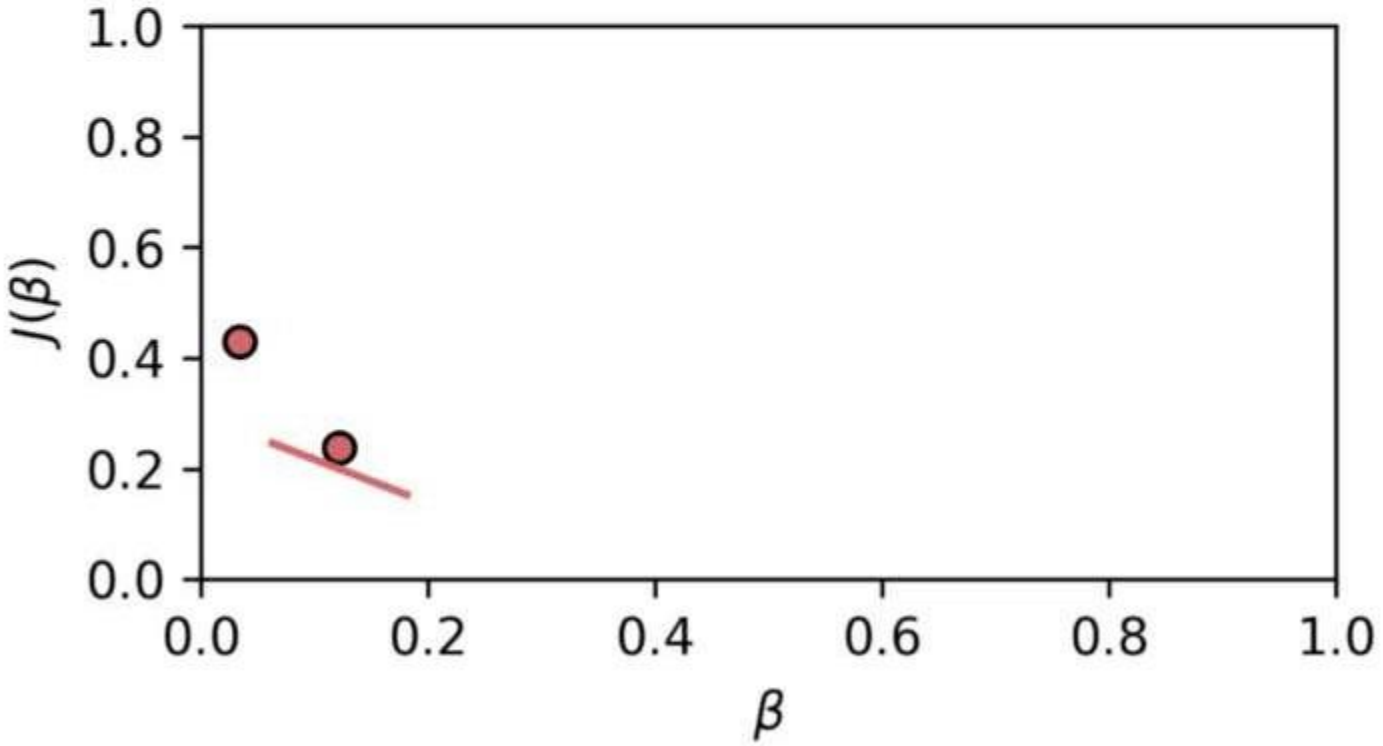
- Step forward proportional to **negative** gradient



$$\beta_{new} = \beta_{old} - \alpha \nabla f(\beta_{old})$$

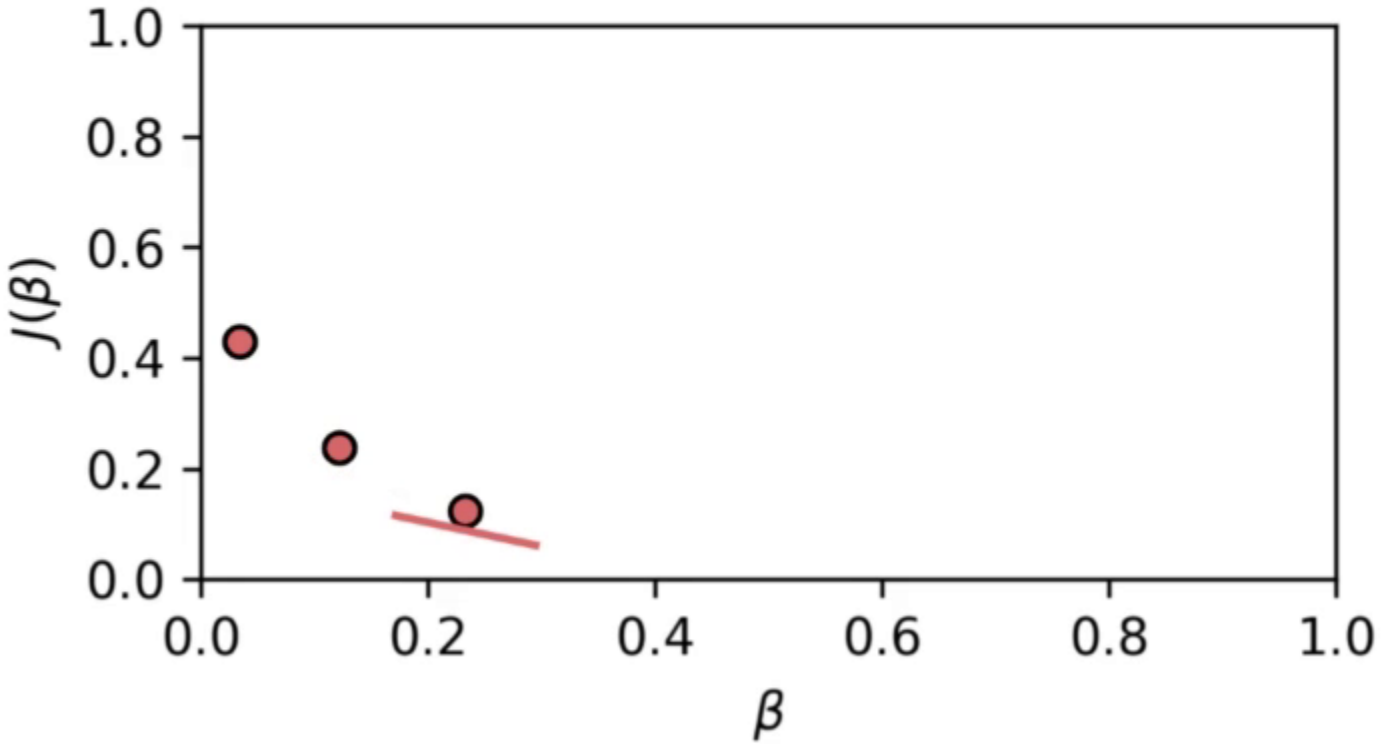
Linear Regression

- Repeat the steps



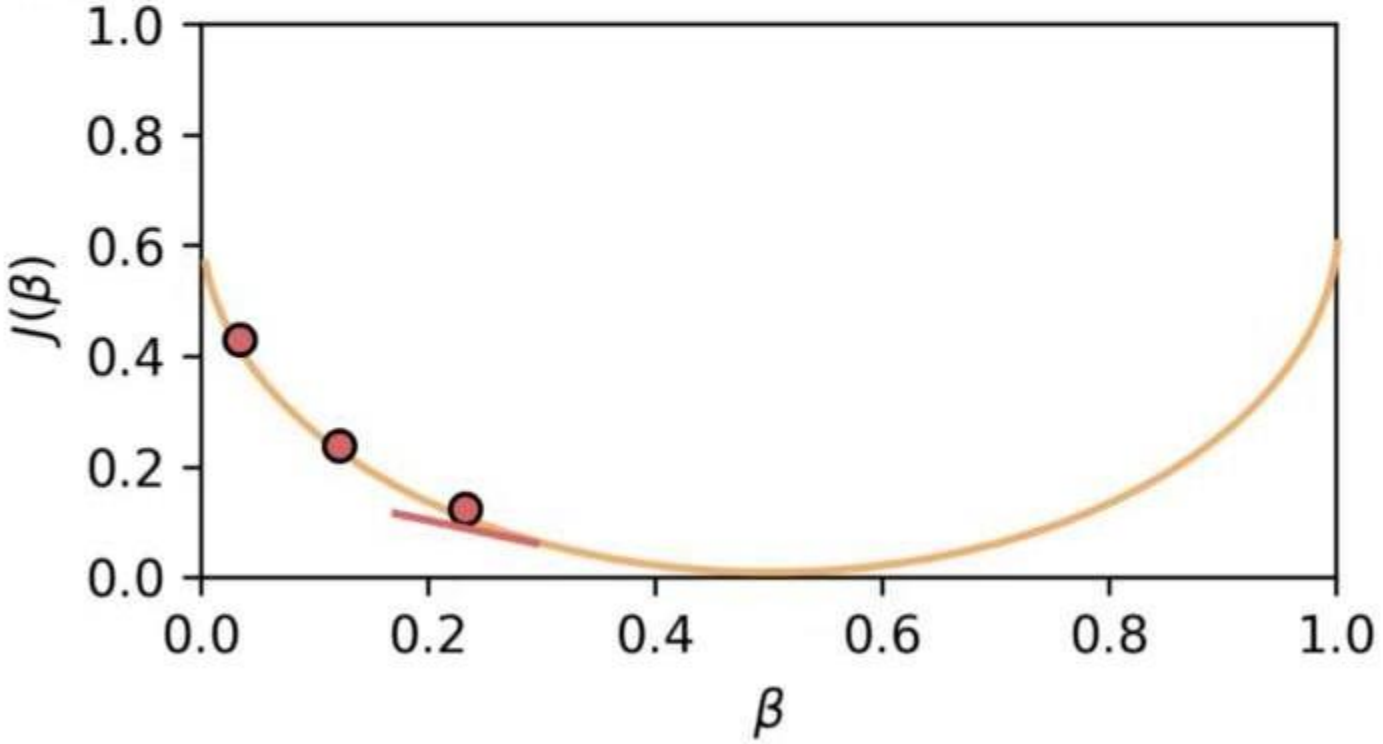
Linear Regression

- Repeat the steps



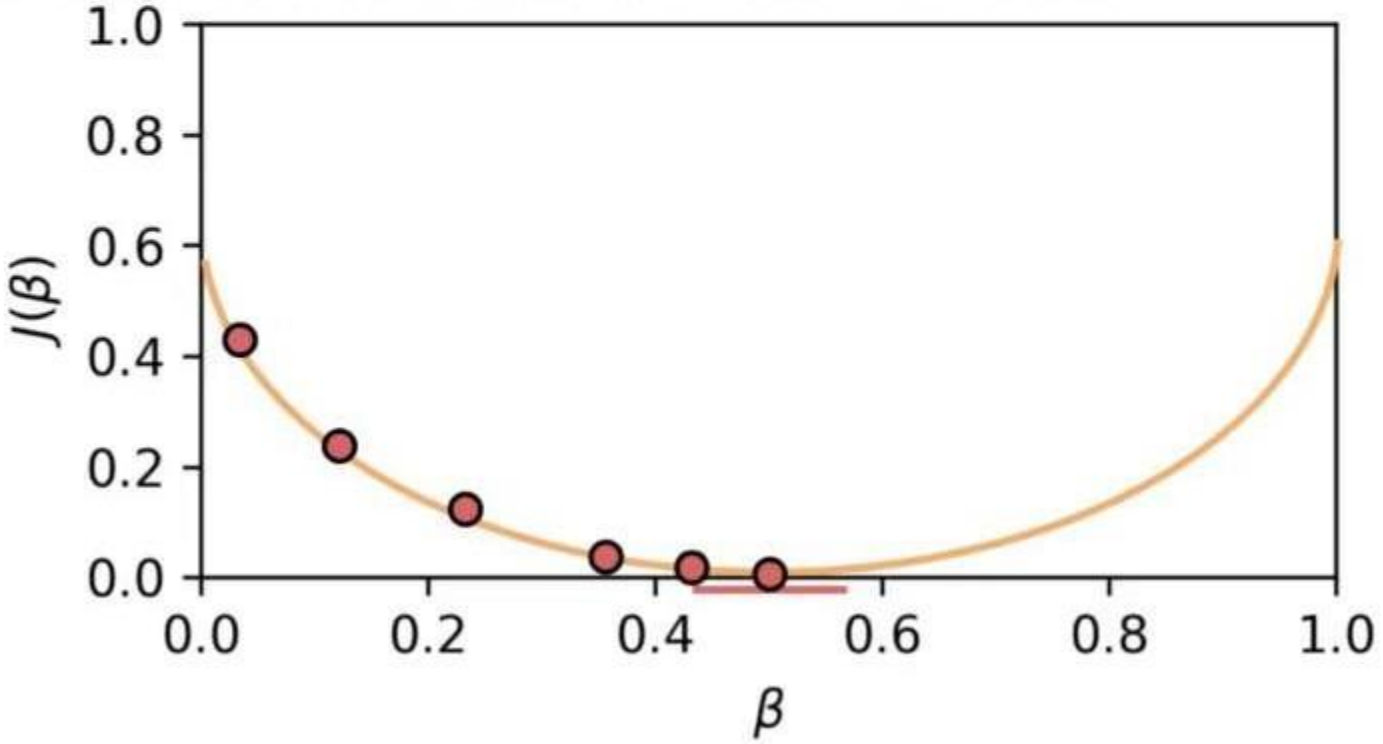
Linear Regression

- Note how we are essentially mapping the gradient!



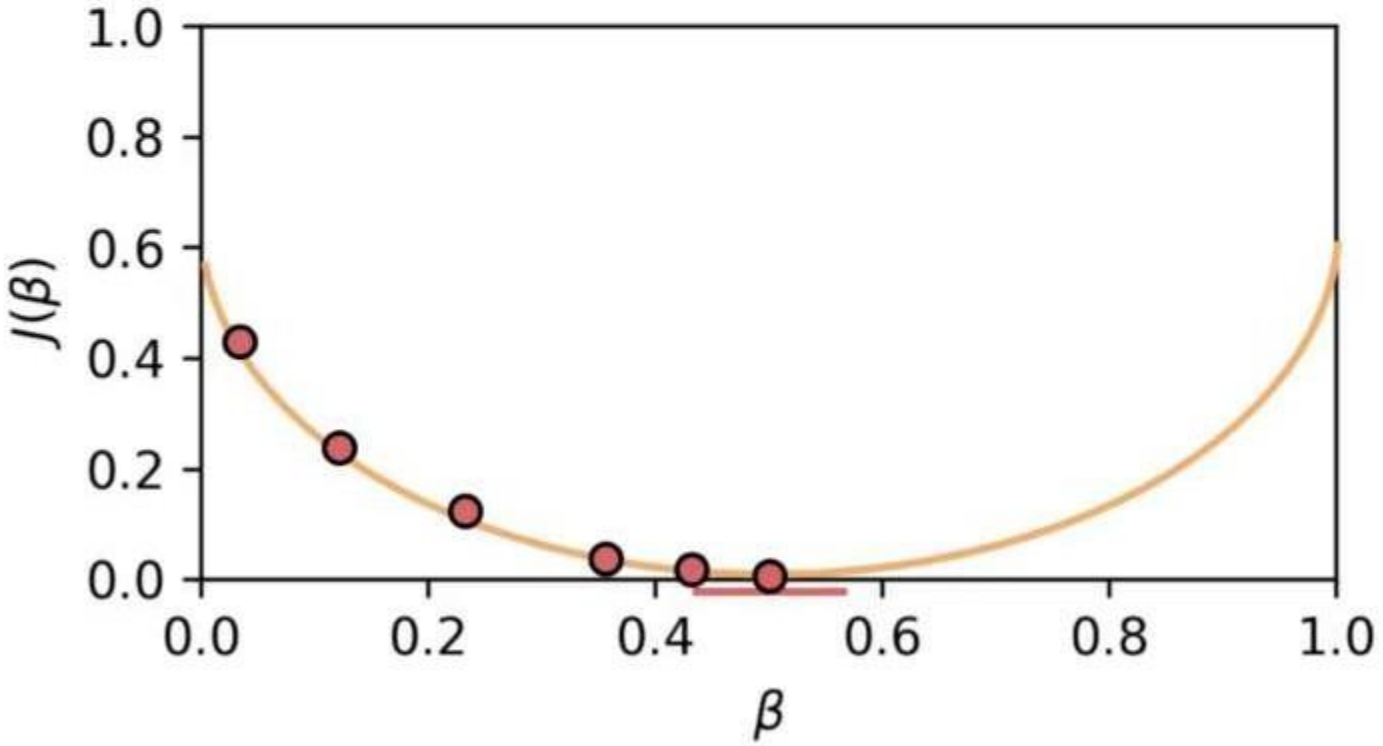
Linear Regression

- Eventually we will find the Beta that minimizes the cost function!



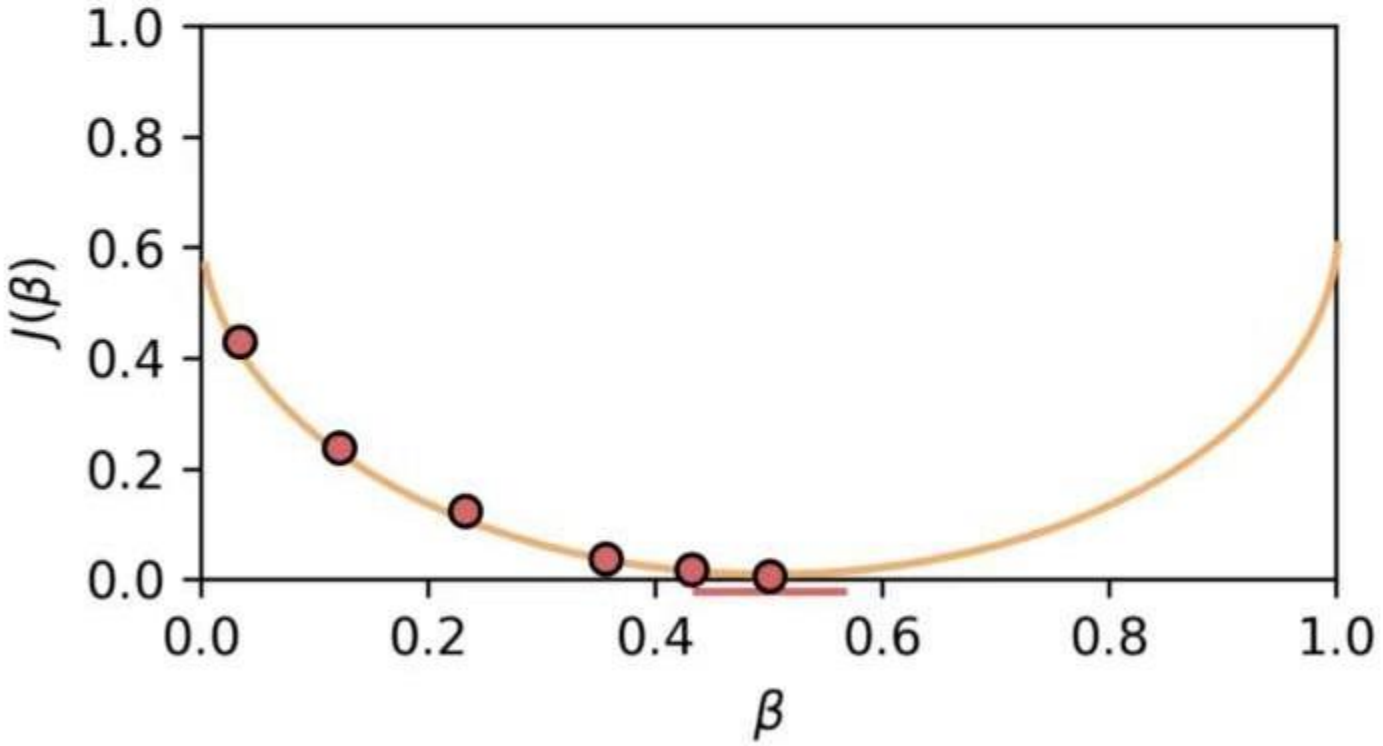
Linear Regression

- Steps are proportional to negative gradient!



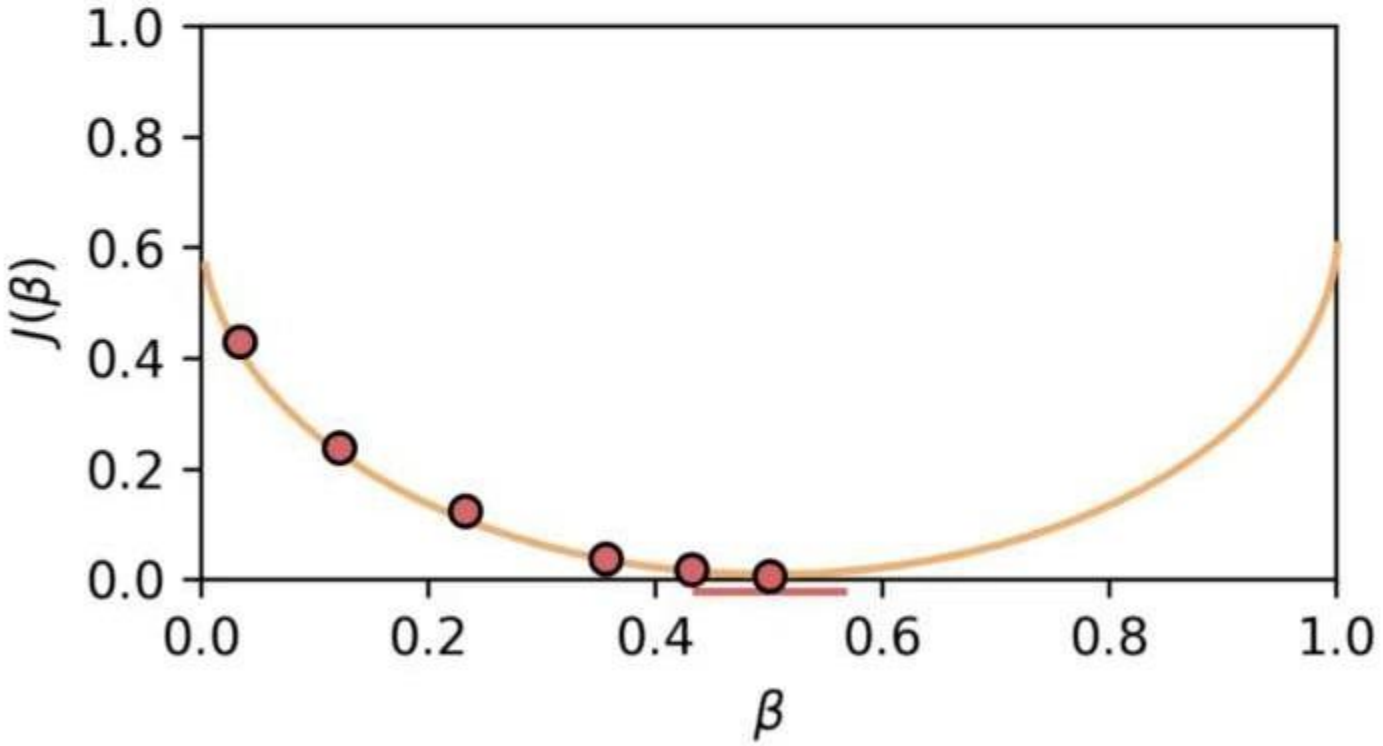
Linear Regression

- Steeper gradient at start gives larger steps.



Linear Regression

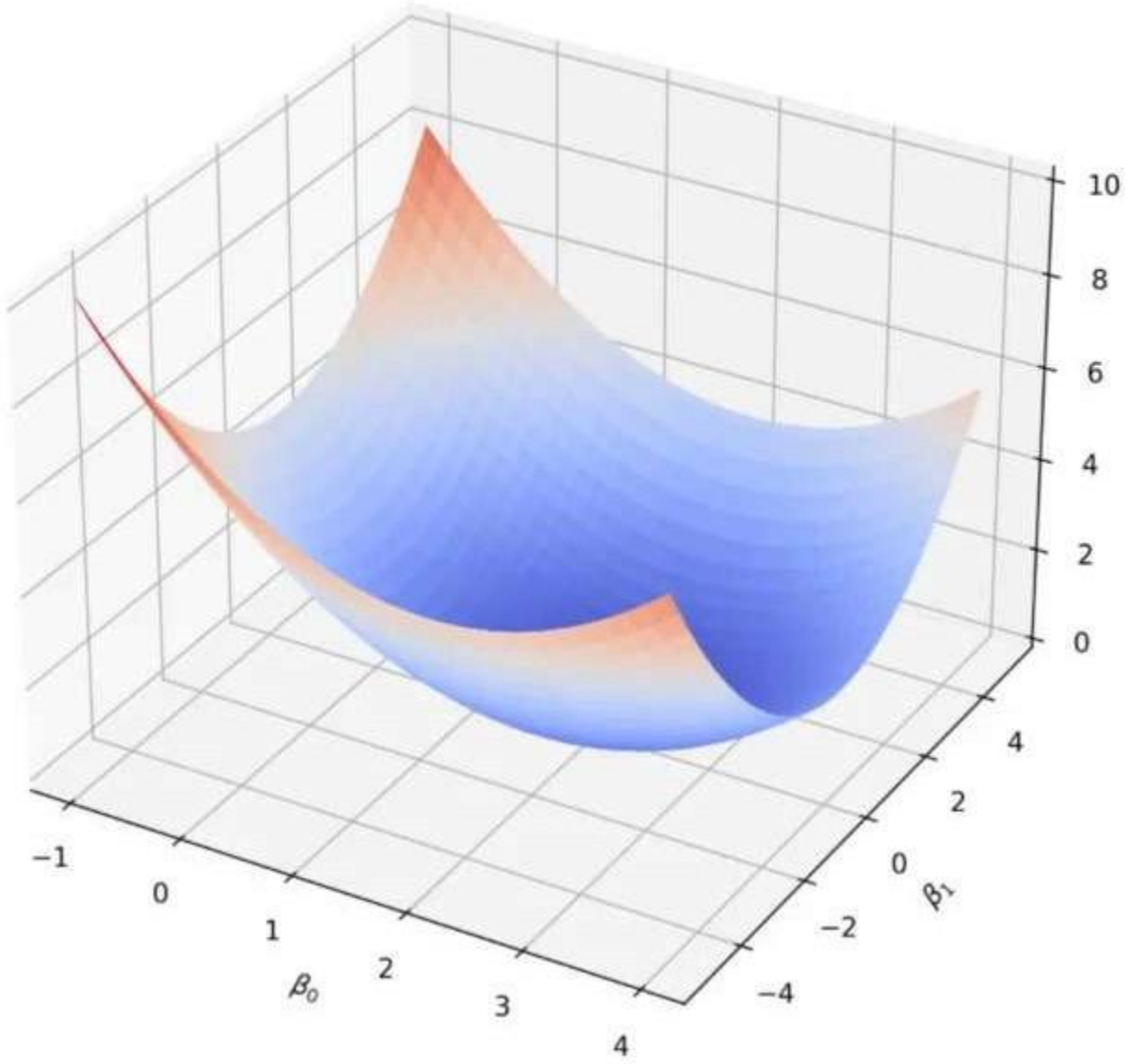
- Smaller gradient at end gives smaller steps.



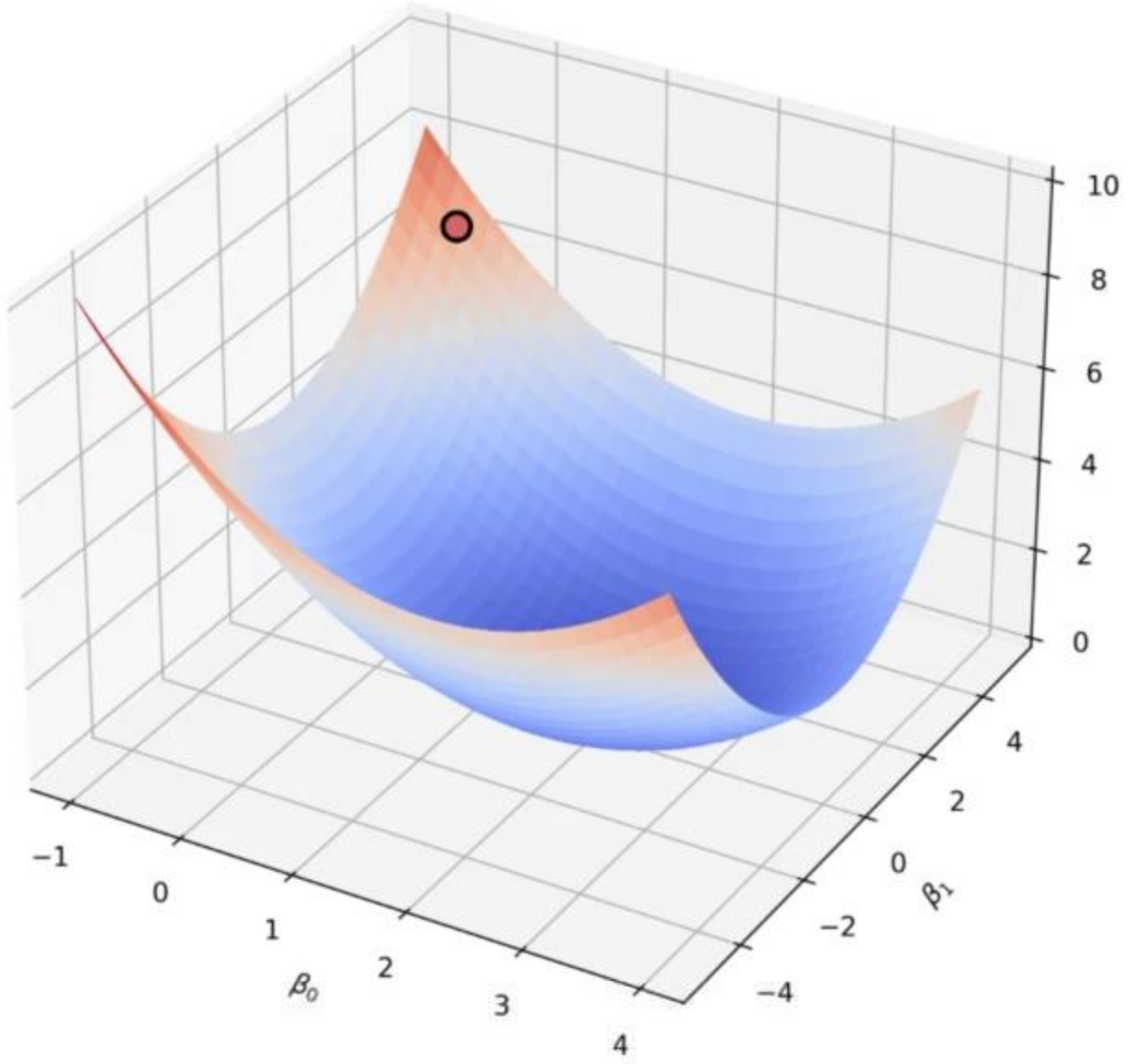
Linear Regression

- To further understand this, let's visualize this gradient descent search for two Beta values.
- Process is still the same:
  - Calculate gradient at point.
  - Move in a step size proportional to negative gradient.
  - Repeat until minimum is found.

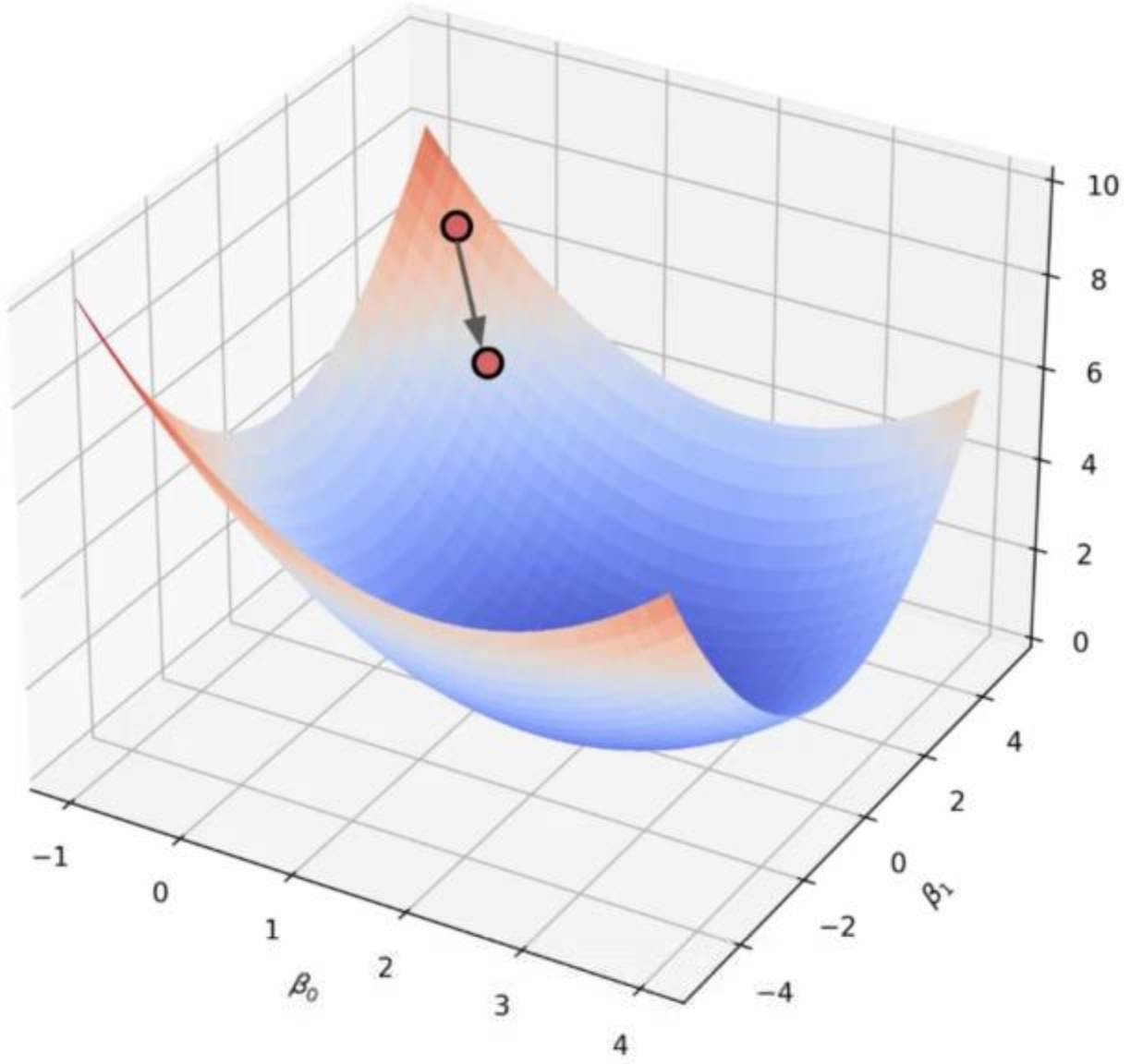
Linear Regression



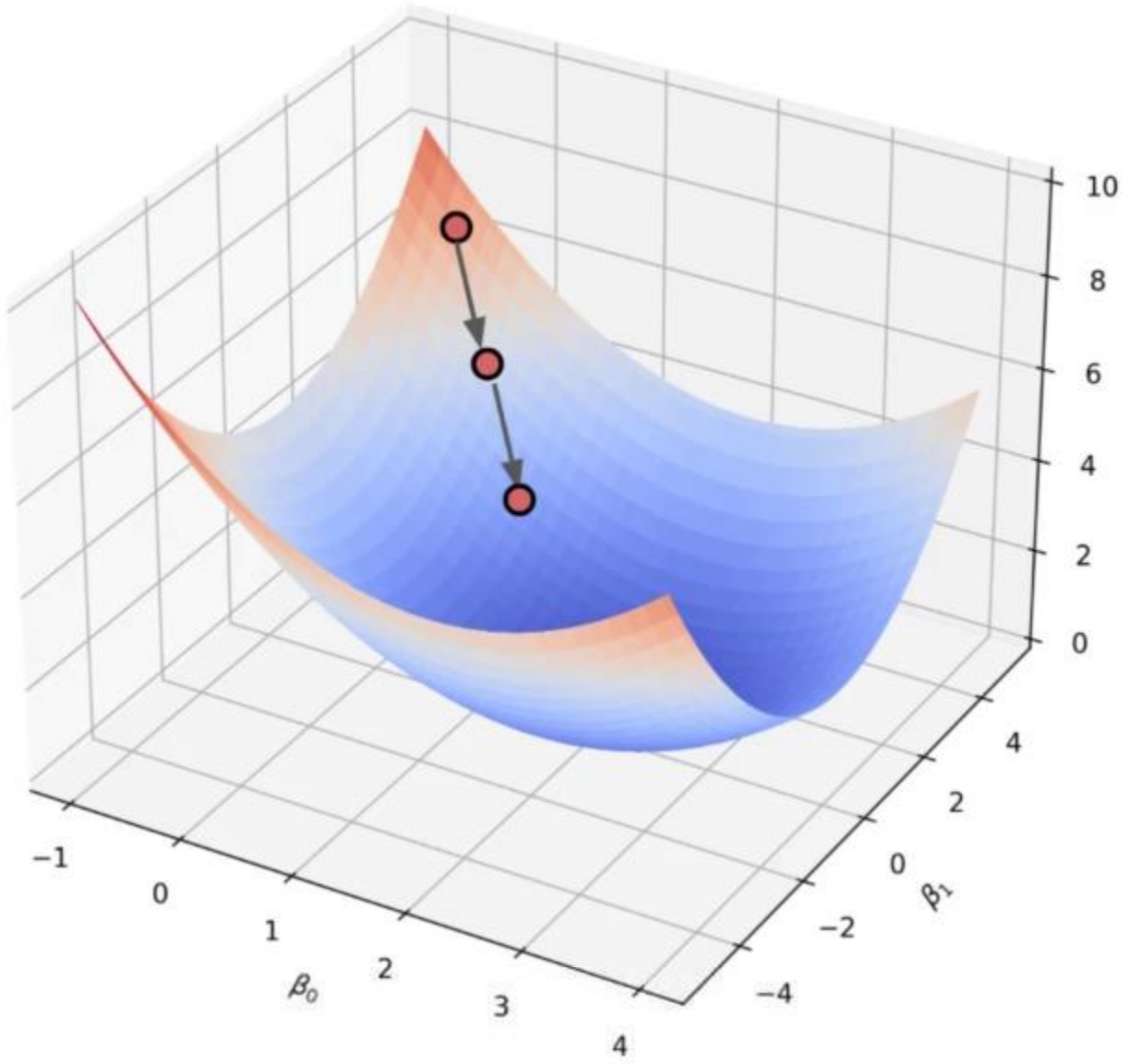
Linear Regression



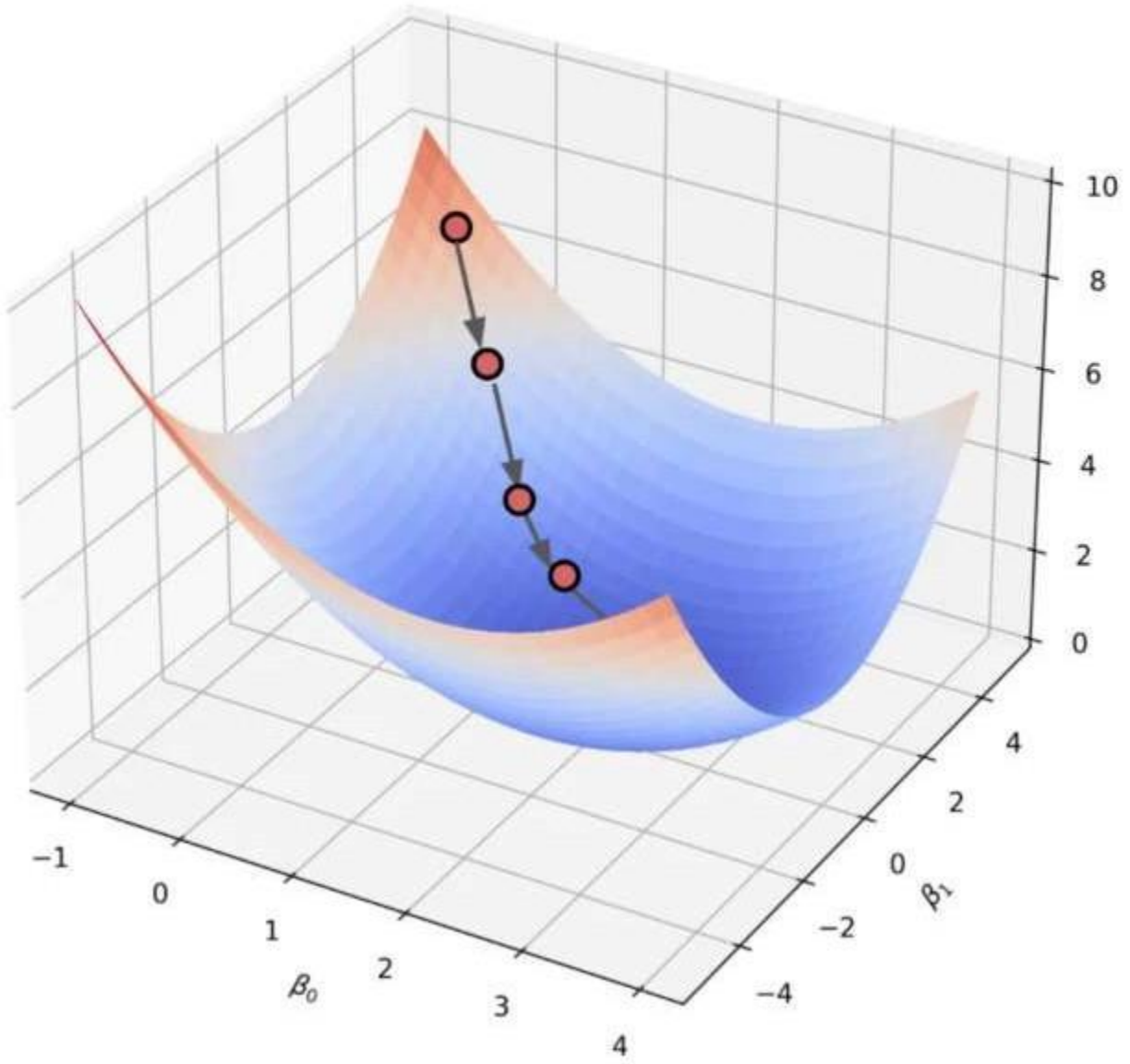
Linear Regression



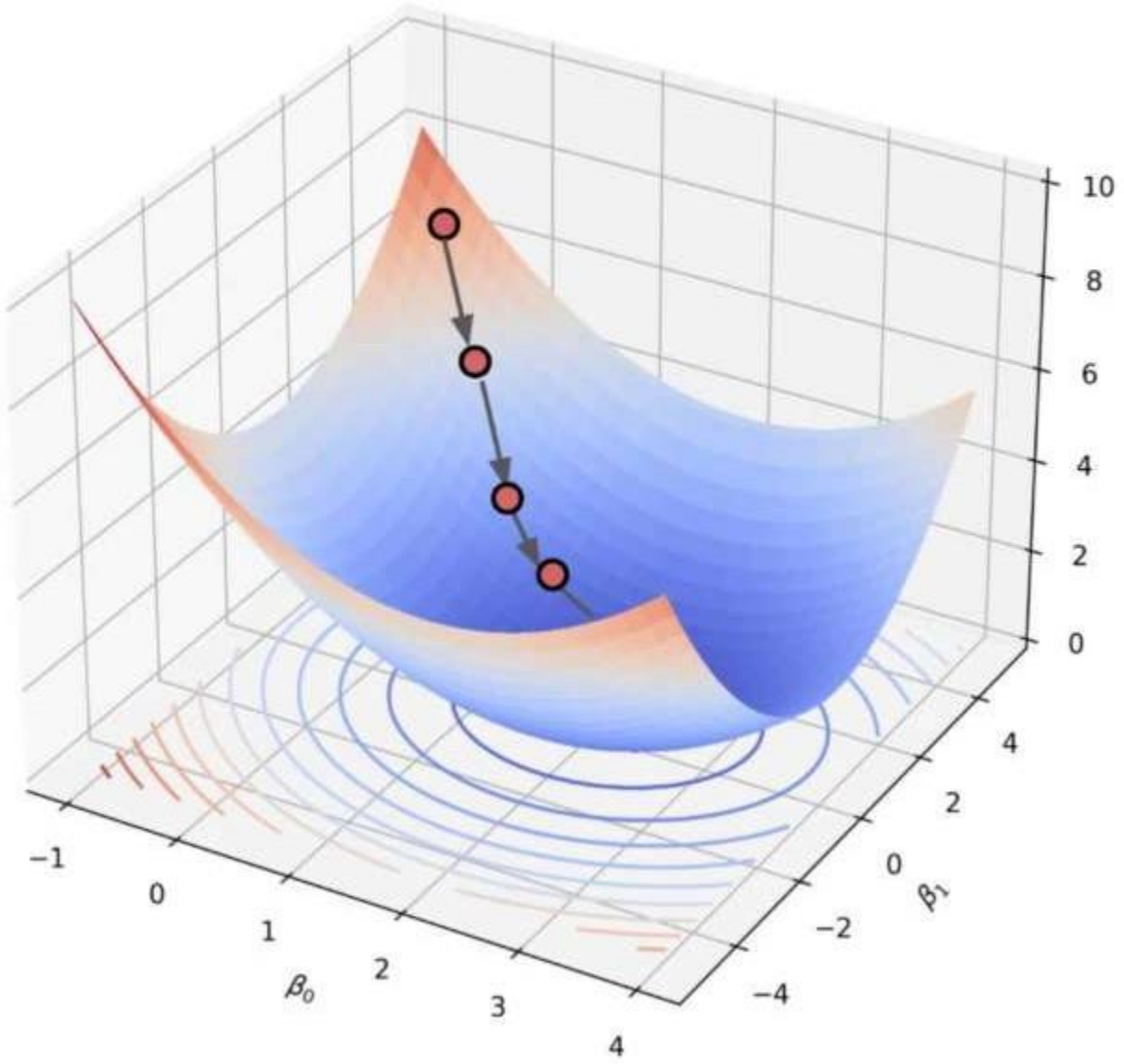
Linear Regression



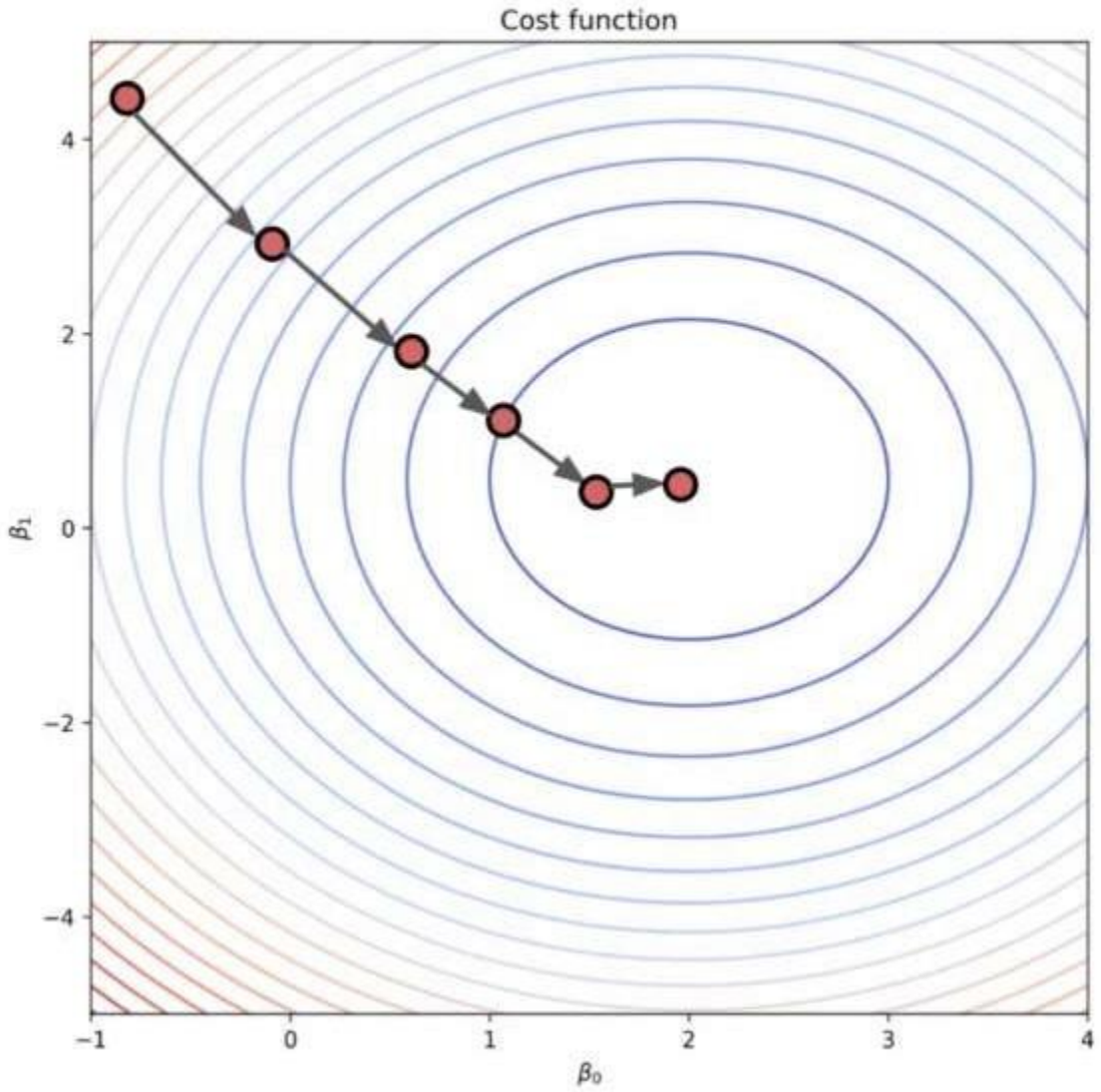
Linear Regression



Linear Regression



Linear Regression



Linear Regression

- Finally! We can now leverage all our computational power to find optimal Beta coefficients that minimize the cost function producing the line of best fit!
- We are now ready to code out Linear Regression!

Simple Linear Regression

Linear Regression

- Now that we understand what is happening “under the hood” for linear regression, let’s begin by coding through an example of **simple linear regression.**

Linear Regression

- To help with our understanding, we will start by directly using the advertising data set mentioned in Chapter 3 of ISLR.
- This lecture also serves to start motivating us to think about performance evaluation and multivariate regression.

Linear Regression

- Simple Linear Regression
  - Limited to one X feature (y=mx+b)
  - We will create a best-fit line to map out a linear relationship between total advertising spend and resulting sales.
- Let's head over to the notebook!

Linear Regression

- Simple Linear Regression
  - Limited to one X feature (y=mx+b)
  - We will create a best-fit line to map out a linear relationship between total advertising spend and resulting sales.
- Let's head over to the notebook!

Linear Regression

- Simple Linear Regression
  - Limited to one X feature (y=mx+b)
  - We will create a best-fit line to map out a linear relationship between total advertising spend and resulting sales.
- Let's head over to the notebook!

00-Intro-to-Simple-Linear-Regression [LEC3].ipynb