VGP337 - Neural Network & Machine Learning

Instructor: Peter Chan

Supervised Learning

- In supervised learning, the goal is to train a system such that it can predict an output correctly based on never-before-seen data
- In order to train the system, we need to provide a dataset containing both inputs and outputs of our target model
- Each entry in the data set is sometimes referred as an instance or an example
- A data instance consists of a input feature vector X, and a label y

Dataset Example

- California Block Housing Prices in 1990:

	House Age	Average Rooms	Population	House Value
example	41	6.98412698	322	$452,600
	21	6.23813708	2401	$358,500
	52	8.28813559	496	$352,100
	52	5.8173516	558	$341,300
	52	6.28185328	565	$342,200
	52	4.76165803	413	$269,700
	52	4.93190661	1094	$299,200
	52	4.79752705	1157	$241,400
	42	4.29411765	1206	$226,700
		feature		label

Linear Regression

- One of the most fundamental algorithms in machine learning
- Usually one of the first algorithms to learn due its simplicity and how it leads into other algorithms like Logistic Regression and Neural Networks
- The fact is, linear regression was developed in the field of statistics and the goal is to understand the relationship between input and output variables using a linear predictor function
- It is borrowed in machine learning because it has the same goal in that it looks for a model that gives the most accurate prediction possible while minimizing the error

Linear Regression

- Given a dataset:

$$\{y_i,\, x_{i1},\ldots,x_{ip}\}_{i=1}^{n}$$

- A linear regression model assumes that the relationship between the dependent variable **y** and the independent variable vector **X** is linear

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^\mathsf{T}\boldsymbol\beta + \varepsilon_i, \qquad i = 1,\ldots,n,$$

- In other words, the model assumes the output value is a weighted sum of the input features plus some bias

Simple Linear Regression

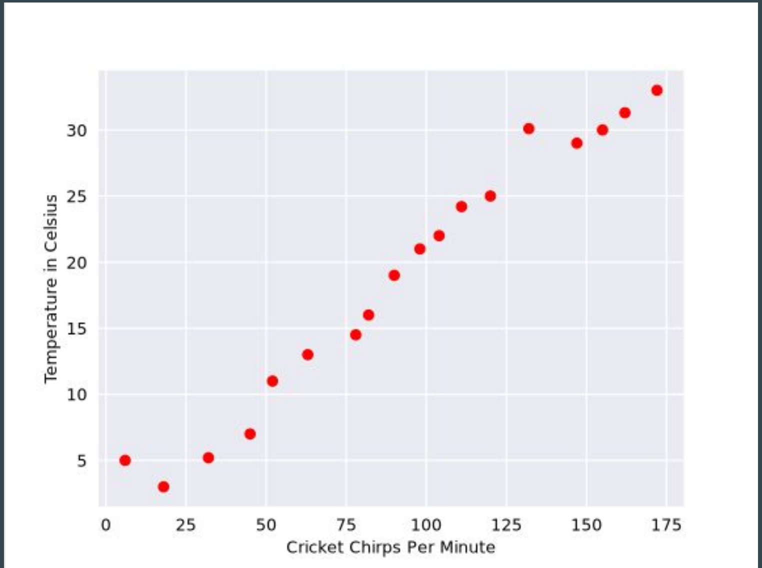
- It is the simplest form of linear regression where there is a single variable in the feature vector x
- In which case, the model function becomes the equation of a 2D line:

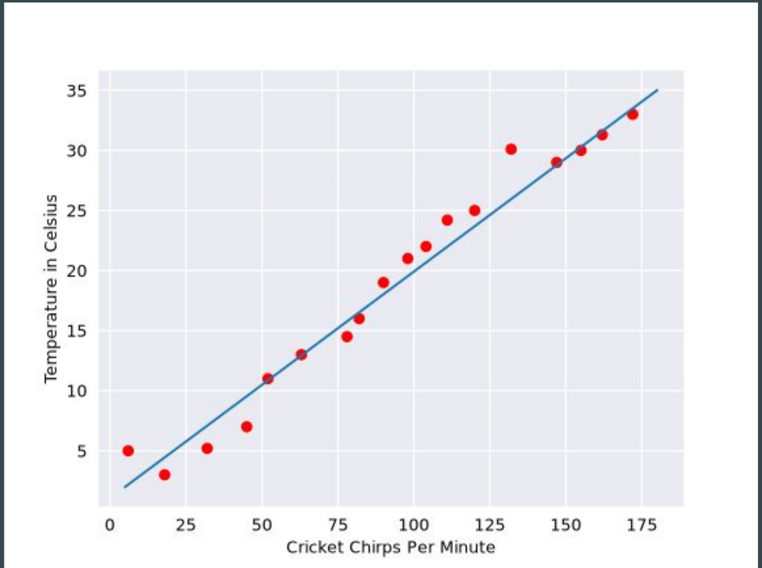
$$y = \beta_0 + \beta_1 x$$

- The goal is to estimate the coefficients β_0 and β_1 such that we have line of best fit which minimizes the error of our prediction

Simple Linear Regression

- But how do you know if you have a good line?





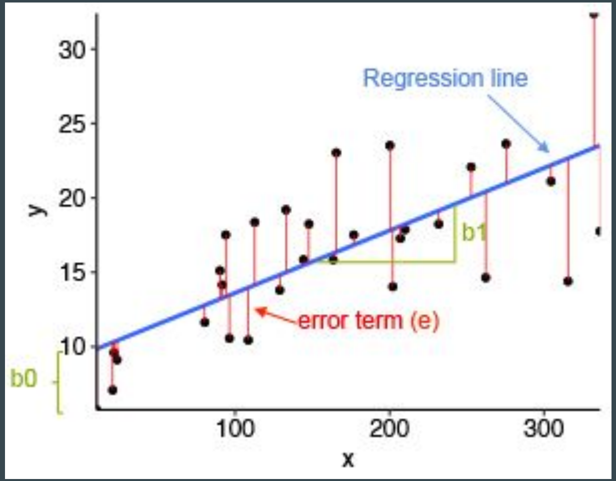
Cost Function

- Sometimes referred as the Loss function, is a way for us to measure how good our estimates are for the terms β_0 and β_1
- The most commonly used cost function for simple linear regression is probably the Mean Square Error (MSE)
- MSE computes the average squared loss per example over the entire dataset, here is the formula:

$$MSE = \frac{1}{N} \sum_{(x,y) \in D} (y - prediction(x))^2$$

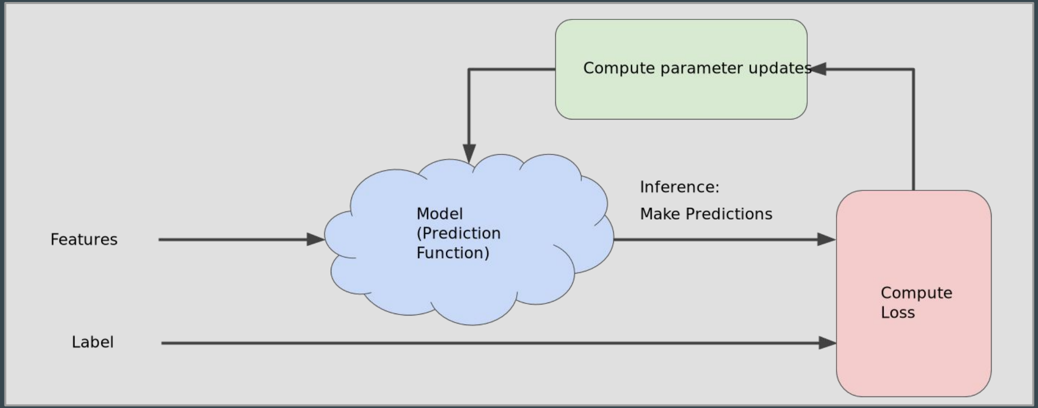
Mean Square Error

- Basically, it computes the average squared distance of each point to the line



Training to Reduce Loss

- Now that we have a way to tell how good our estimates are
- We need a way to improve our estimates so we can eventually find the best fit line
- There are many different methods of doing this, we will be looking at an iterative learning approach



Gradient Descent

- Gradient Descent is a popular optimization technique based on Calculus
- To use it, we first need to make some adjustments to our cost function

$$MSE = \frac{1}{N} \sum_{(x,y) \in D} (y - prediction(x))^2$$

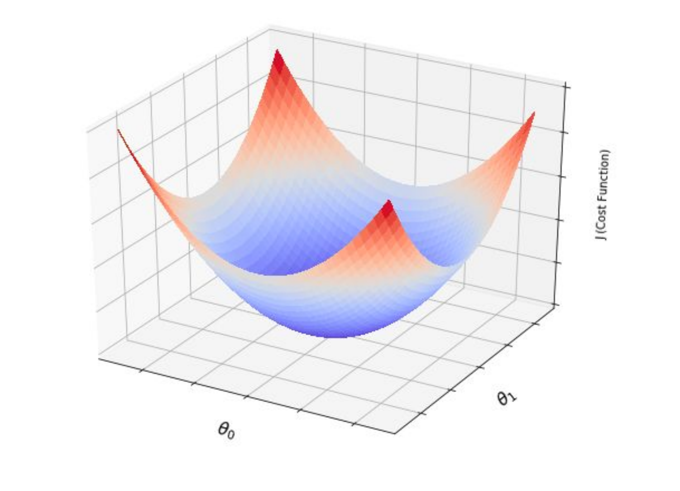
- We want to parameterize this on β_0 and β_1, this gives:

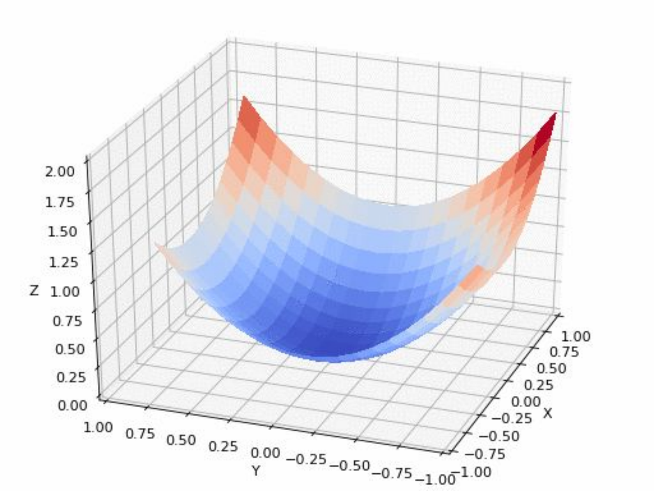
Cost Function

$$J(\Theta_0, \Theta_1) = \frac{1}{2m} \sum_{i=1}^{m} [\underbrace{h_\Theta(x_i)}_{\text{Predicted Value}} - \underbrace{y_i}_{\text{True Value}}]^2$$

Gradient Descent

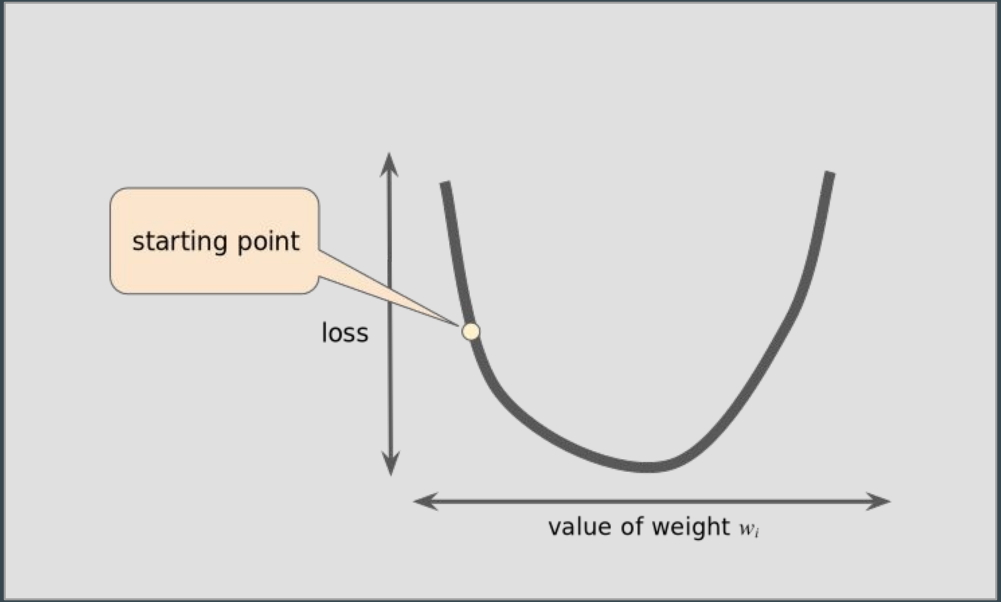
- This gives us a bowl-shaped function when plotted that has a global minimum
- The idea with gradient descent is to “ride the slope” until we reach this point, which tells us the values of β_0 and β_1 that will minimize the cost





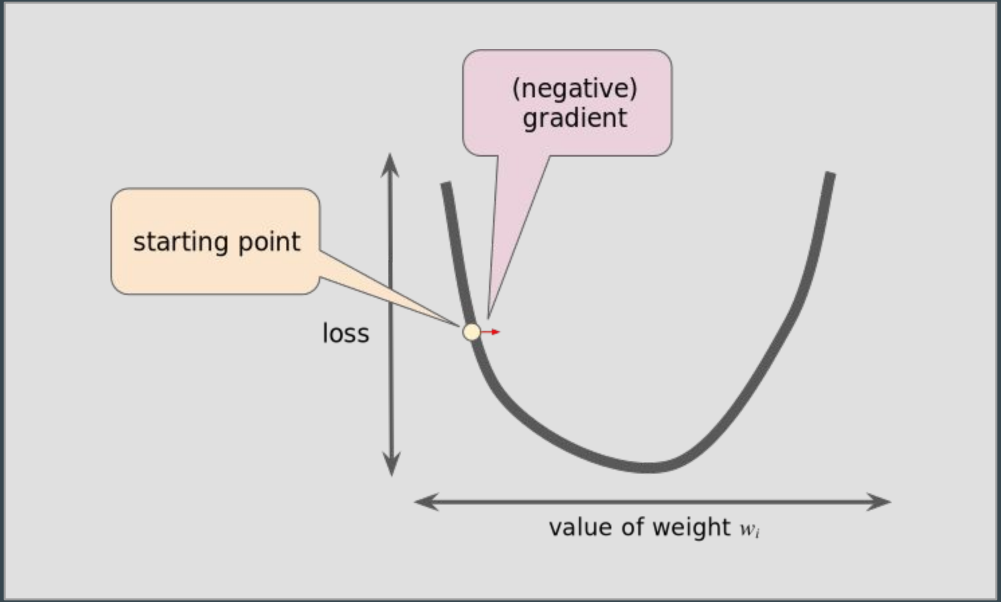
Gradient Descent

- To run gradient descent, you first need to pick a random starting point



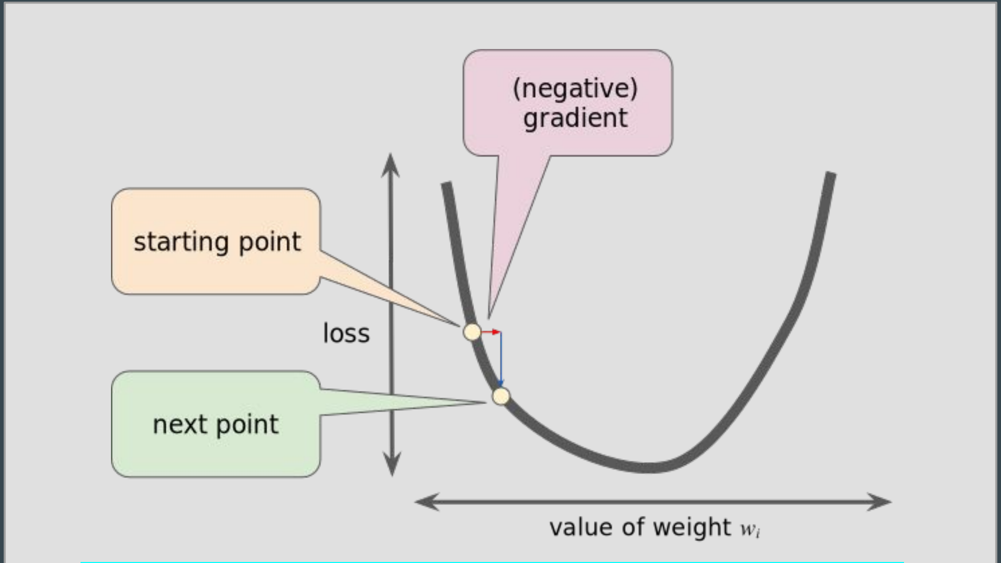
Gradient Descent

- Next determine the gradient (slope) at this point



Gradient Descent

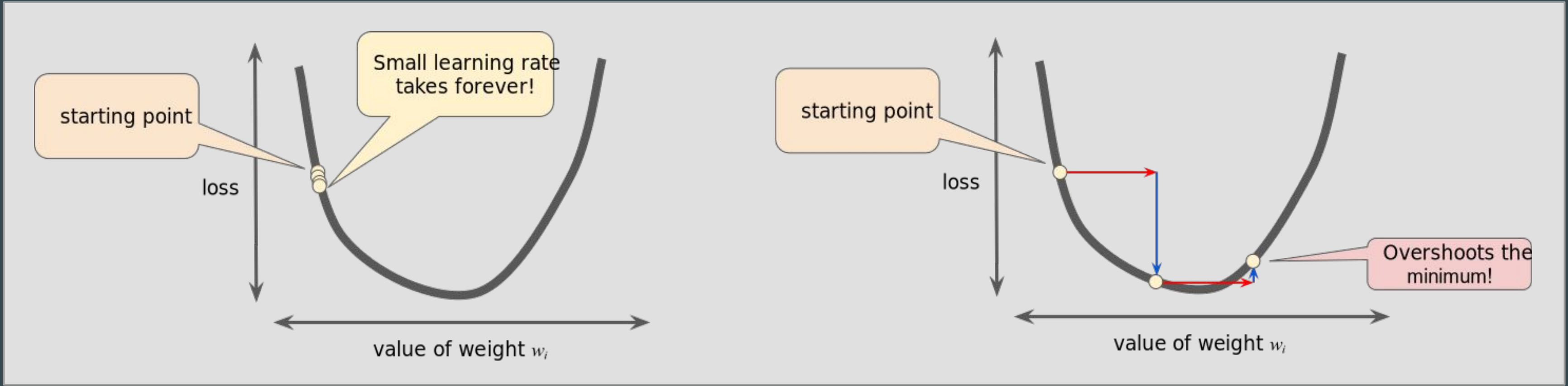
- Finally, take a step toward that direction and update your value



next point = starting point - (**α** * slope)

Learning Rate

- How big of a step do you take?
- This is controlled by a learning rate **α** that you select
- Picking the correct learning rate can affect the processing time and accuracy of your training



Gradient Descent

- Here is the formula that you need to apply for each gradient descent step

Gradient Descent

$$\boxed{\Theta_j = \Theta_j - \underset{\substack{\uparrow \\ \text{Learning Rate}}}{\alpha} \frac{\partial}{\partial \Theta_j} J\left(\Theta_0, \Theta_1\right)}$$

- You can either run this a fixed number of steps or until convergence
- To get the slope at a given point, you will need to use ~~Calculus~~ magic

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x^{(i)}$$

Here are some good references

Descending into ML: Linear Regression

Linear Regression Simplified - Ordinary Least Square vs Gradient Descent

Gradient Descent in Linear Regression

Algorithms From Scratch: Linear Regression (follow this for homework!)