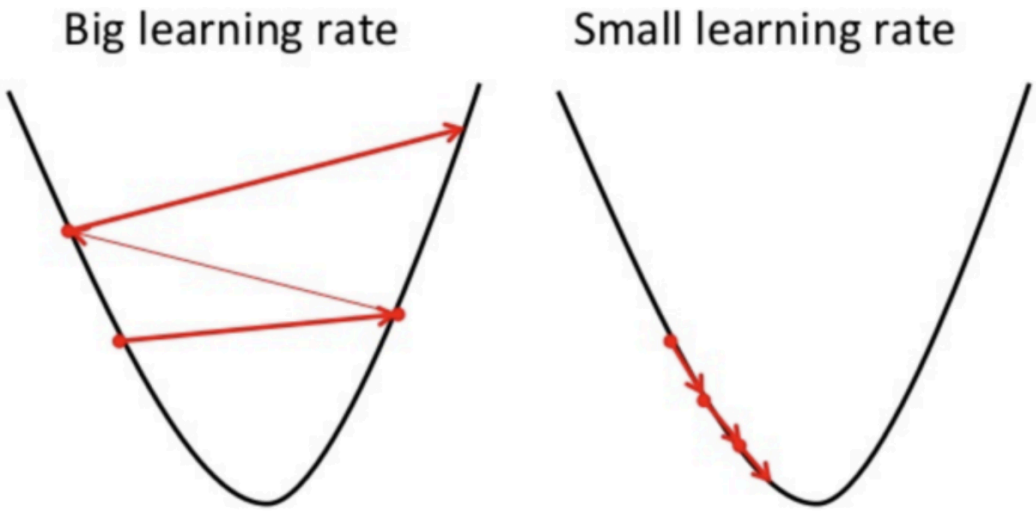
– Why is gradient descent important in machine learning?

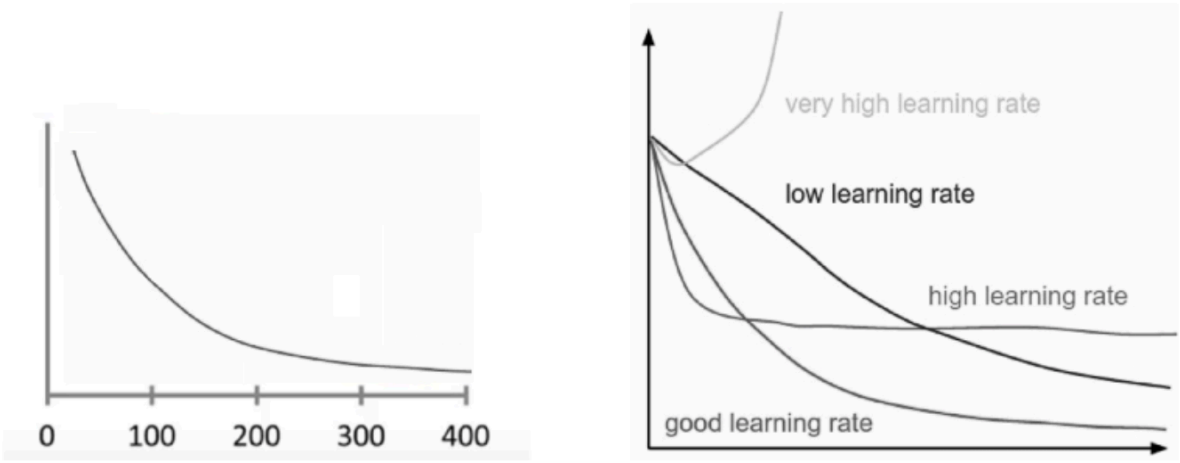
The ultimate goal of Machine Learning is to find the best parameter to minimize the loss function. Loss functions are used to measure the error in predictions of an ML model in order to increasing the accuracy of the model. We adjusted the parameters (the weights and biases) of the model by iterating over a training data set. Loss function helps in correcting the steps as to how much the algorithm must take so that it helps in reducing the mistakes (Biswas, 2019).

Gradient descent is an efficient optimization algorithm that attempts to minimize the loss function. The minimum value of a loss function can be treated as lowest point of the valley, the whole process of gradient descent would be a journey of hiking down a hill from an initial starting point, while choosing a direction to advance using small steps along the way toward the deepest valley. The gradient is the slope of the mountain, and descent is downhill. In order to find the path, we need to know the gradient of the surface which indicates the direction of nearest hill, then we just need to follow the opposite direction which can reach the bottom of the mountain, effectively. Therefore, at each step we need to measure the slope of the surface created and take the opposite way until we reach the valley. It is important to check and control the step size, which is the learning rate, we need to make sure to take the correct steps to reach the global minimum. If the step size is too big which can result in overshooting the valley; if the step size is too tiny which can make the journey time infinite long as graph 1.



Graph 1

In order to find the most efficient point when we reached the global minimum point, we can draw a graph while put learning rate as Y-axis and iteration times as X-axis as per Graph 2, if the Gradient Descent working correctly, the loss function will decrease after each iteration. Hence, we just need to locate the point with reasonable iterations times to find out the learning rate.



Comparison of different learning rates

Graph 2

References:
https://medium.com/towards-artificial-intelligence/gradient-descent-in-layman-language-d4028b486103

https://hackernoon.com/gradient-descent-aynk-7cbe95a778da

– How does plain vanilla gradient descent work?

Plain vanilla gradient descent also called Batch gradient descent, which is the simplest form of gradient descent. It takes small steps in the direction of the gradient, which means it calculates the error for each sample in the data and updates the parameters of the model only after the training example has been calculated.

If a training samples has ith data points.

1. We need to calculate the gradient of the loss function for the i-th training example includes every weight and bias.
2. Add the gradients of the weights and bias calculated to a separate accumulator vector which after the iteration of each training sample.
3. Calculating the average gradients for all weights and bias by dividing the accumulator variables of them by the number of training examples, which is called updated accumulators (UAs).
4. Using the UA to replace the weights and bias, respectively and stopping the whole process until convergence.

– Two modifications to plain vanilla gradient descent.

Stochastic Gradient Descent (SGD) not only calculates the error for each training sample data set but also updates the parameter of the model one by one. It is faster than Plain Vanilla Gradient Descent (PVGD) but also computationally costly as well. As it will take into all the data points (include the noise) into consideration which require more iteration times to reach the minima.

Another modification to plain vanilla gradient descent is called Mini Batch Gradient Descent. It is a combination approach of SGD and PVGD. It splits the training data into small batches of a fixed size and then updates the parameters for each of the batches accordingly. It creates a

balance between the efficiency of batch gradient descent and the robustness of stochastic gradient descent.

	Process	Advantage	Disadvantage
Plain Vanilla Gradient Descent (PVGD)	Calculates the error for each example within the training set. Updates the model parameters after accessing the whole training examples	Computationally efficient, produces a stable error gradient and convergence	The model generated might not achieve the best performance
Stochastic Gradient Descent (SGD)	It calculates the error for each example and updates the parameter straight away after each error to a single training example.	Faster than PVGD, provides detailed rate of improvement.	Computationally expensive than PVGD. The error of noise might affect the performance of GD
Mini Batch Gradient Descent (MBGD)	Combination of SGD and PVGD. Separates the training set into small batches and performs an update for each of these batches.	Idea for large data sets. *Creates a balance between the robustness of stochastic gradient descent and the efficiency of batch gradient descent.*	