#### 1. What is gradient descent?

Gradient descent is a optimization algorithm used to find the values of parameters (such as weights and biases) that minimize a cost function. The cost function is a measure of how well the model predicts the target values. The algorithm works by iteratively updating the parameters in the direction that reduces the cost function.

#### 2. What’s the difference between batch gradient descent and stochastic gradient descent?

The main difference between batch gradient descent and stochastic gradient descent is that with batch gradient descent, the gradient is calculated using the entire dataset, while with stochastic gradient descent, the gradient is calculated using a single data point. This means that batch gradient descent is more computationally expensive, but it also means that the gradient is more accurate.

#### 3. What are some common cases where gradient descent may fail to converge?

There are a few reasons why gradient descent might fail to converge. One reason is if the function being optimized is not convex. If the function has multiple local minima, then gradient descent might get stuck in a local minimum that is not the global minimum. Another reason is if the step size is not chosen properly. If the step size is too large, then gradient descent might not converge. If the step size is too small, then gradient descent might converge slowly.

#### 4. Why is gradient descent needed when training a model?

Gradient descent is an optimization algorithm used to find the values of parameters (weights) that minimize a cost function. When training a model, gradient descent is used to find the values of the weights that minimize the error between the predicted values and the actual values.

#### 5. Is it possible to apply gradient descent to solve non-convex optimization problems?

Yes, it is possible to apply gradient descent to solve non-convex optimization problems, but it is important to keep in mind that doing so may lead to sub-optimal solutions. In general, gradient descent is more likely to find a local optimum when applied to non-convex optimization problems as opposed to the global optimum.

1. **what is the difference between Cost Function vs Gradient Descent?**

**Answer**

* A **Cost Function** is something we want to minimize. For example, our cost function might be the sum of squared errors over the training set.
* **Gradient Descent** is a method for finding the minimum of a function of multiple variables.

##### [Difference between convex and non convex function?](https://avatto.in/data-scientist/interview-questions/deep-learning/gradient-descent-method/page/2/#shrt-collapse-3)

A function is called a convex function when it has only one minimum value and a function is called a non-convex function when it has more than one minimum value.

##### [Explain the difference between gradient descent, stochastic gradient descent, and mini-batch gradient descent](https://avatto.in/data-scientist/interview-questions/deep-learning/gradient-descent-method/page/3/#shrt-collapse-2)

**Gradient descent -** Update the parameter of the network after iterating through all the data points present in the training set.  
**Stochastic gradient descent -**Update the parameter of the network after iterating through every single data points present in the training set.  
**Mini-batch gradient descent -** Update the parameter of the network after iterating through some n number of data points present in the training set.

##### [Why do we need momentum-based gradient descent?](https://avatto.in/data-scientist/interview-questions/deep-learning/gradient-descent-method/page/3/#shrt-collapse-3)

One problem we face with SGD and mini-batch gradient descent is that there will be too many oscillations in the gradient steps. This oscillation happens because we update the parameter of the network after iterating through every point or every n data points and thus the direction of the update will possess some variances causing oscillation in the gradient steps.  
This oscillation leads to slow training time and makes it's hard to reach the convergence. To avoid this issue we use momentum-based gradient descent.

##### [What is the issue faced in the momentum-based gradient descent?](https://avatto.in/data-scientist/interview-questions/deep-learning/gradient-descent-method/page/3/#shrt-collapse-4)

One issue we encounter with the momentum-based gradient descent method is that it causes us to miss out on the minimum value.  
Suppose, we are near to attaining convergence and when the value of momentum is high, then the momentum pushes the gradient step high and we miss out on the minimum value, that is we overshoot the minimum value.

1. **Explain how does the Gradient descent work in Linear Regression**

**Answer**

The **Gradient Descent** works by starting with random values for each coefficient in the linear regression model.

* After this, the sum of the squared errors is calculated for each pair of input and output values (loss function), using a learning rate as a scale factor.
* For each iteration, the coefficients are updated in the direction towards minimizing the error,
* then we keep repeating the iteration process until a minimum sum squared error is achieved or no further improvement is possible.

### Gradient descent vs gradient ascent

While gradient descent is used primarily for error prediction with a decreasing slope in a model, while gradient ascent is used primarily for improving optimization through an upward slope or increasing graph

Linear regression

**Name some Evaluation Metrics for Regression Model and when you would use one?**

**Answer**

**Mean absolute error (MAE)**: calculates the absolute difference between actual and predicted values. It can be used when we want that our model be robust to outliers, but this metric has the disadvantage of not being differentiable so we can't use it if we want to apply optimizers like Gradient descent.

**Mean squared error (MSE)**: calculates the squared difference between actual and predicted value. We can use this metric if we want to give bigger penalization to outliers and apply optimizers who require differentiation. MSE is a differentiable function that makes it easy to perform mathematical operations in comparison to a non-differentiable function like MAE.

**Root mean squared error (RMSE)**: This is simply the square root of mean squared error. This metric is not so robust to outliers as the mean absolute error but it has the advantage to be differentiable so we can use it if we want to apply gradient descent to minimize losses.

When to use one depends on your loss function:

* **When to use MAE**: If being off by ten is just twice as bad as being off by 5. it is better to use the MAE if you don't want your performance metric to be overly sensitive to outliers.
* **When to use RMSE**: In many circumstances, it makes sense to give more weight to points further away from the mean - that is, being off by 10 is more than twice as bad as being off by 5. In such cases, RMSE is a more appropriate measure of error.