**Regression**

**Linear Regression**

Q. How does a linear regression model make a prediction?

A. By computing weighted sum of features plus a constant term called the bias/intercept



Q. What are some cost functions that can be used to train a linear regression model?

A. These cost functions can be minimized to train a linear regression model:

- MSE



- RMSE



- MAE



Q. What is the normal equation?

A. The normal equation is a closed form solution a.k.a. a mathematical way of computing the parameters if a linear regression problem



Q. What is the computational complexity of linear regression (in sklearn)?

A. O(n^2), where n is the number of features. Making predictions is O(n) operation.

**Gradient Descent**

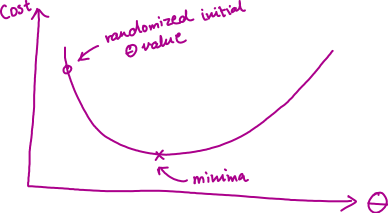
Q. Describe, in very simple terms, how gradient descent works?

A. The general idea of gradient descent is to tweak parameters gradually with the aim of minimizing the cost function:

- Initialize theta randomly or with 0s

- Tweak values in theta in steps such that the cost function is reduced

- Continue till the algorithm converges – i.e. a minima is reached



Q. What is the learning rate? Why is it important?

A. The learning rate is a hyperparameter that determines the step size the algorithm takes in an iteration to tweak the value of theta. If the learning rate is too small, the model will take too long to converge. If the learning rate is too high, the model may never converge and the value may diverge.

Q. What are 2 common challenges with gradient descent?

A.

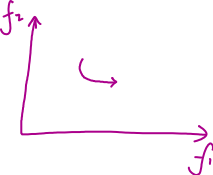
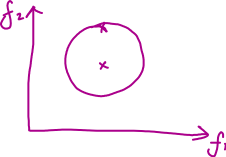


Q. Why do we not face the challenge of a local minima in MSE for linear regression?

A. MSE cost function for linear regression is a convex function, which means that there is only a global minima and no local minima.

Q. Describe the problem of differing scales of features?

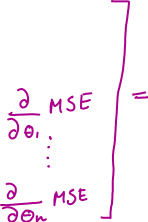
A. If feature 1 is on a scale of 1-100 while feature 2 is on a scale of 0-1, then this may cause the learning to take very long. We can use Sklearn’s StandardScaler.



**Batch Gradient Descent**

Q. How do you compute the ‘direction’ in which the value of theta will be changed by the update algorithm?

A. By computing the derivative/slope/gradient of the cost function. The partial derivatives of the cost function are computed.



Q. What is batch gradient descent?

A. When the gradient of the cost function is computed over the whole dataset X, it is called batch gradient descent aka full gradient descent.

Q. What is an advantage and disadvantage of batch gradient descent?

A. Advantage: It scales well with number of features as compared to Normal Equation or SVD decomposition

Disadvantage: It scales poorly with number of examples in the dataset

Q. What is a good way to set the number of iterations for which to run the gradient descent algorithm?

A. A good way is to set a high number of iterations but set a tolerance e. When the absolute value of the step\_size\*gradient becomes smaller than e, we stop the algorithm from updating.

Q. Describe the tolerance-convergence rate trade-off?

A. If the tolerance is too small, the algorithm will take a very long time to converge. If the tolerance is too large, the algorithm will stop quicker but then the solution will be further from the optimal solution. It takes O(1/e) time to converge. So if we set the tolerance to O(1/(e/10)) = O(10/e) it will take 10x longer.

**Stochastic Gradient Descent**