1. What is Gradient descent?

Gradient descent is an optimization algorithm commonly used in machine learning and deep learning to minimize a cost or loss function. The goal of gradient descent is to find the set of parameters (also called weights) that minimize the error between the predicted output of a model and the actual target values. It does this by iteratively adjusting the parameters in the direction that reduces the loss. The key idea is to use the gradient, which is the vector of partial derivatives of the loss function with respect to each parameter.

The gradient tells us the slope or steepness of the loss function at the current point in the parameter space. By moving the parameters slightly in the opposite direction of the gradient, we ensure that the loss decreases. This step size is controlled by a hyperparameter called the learning rate, which determines how big each step should be. If the learning rate is too large, the algorithm might overshoot the minimum and fail to converge. If it's too small, gradient descent will take a very long time to reach the minimum. The process starts with randomly initialized parameters, then repeatedly computes the gradient, updates the parameters, and checks if the loss is low enough or if a maximum number of iterations has been reached.

There are different variants of gradient descent: batch gradient descent uses all training examples to compute the gradient at once, stochastic gradient descent (SGD) uses one example at a time, and mini-batch gradient descent uses a small random subset of data. Mini-batch gradient descent is popular because it balances the accuracy of the gradient estimate with computational efficiency. Gradient descent can sometimes get stuck in local minima or saddle points where the loss isn't globally minimal. To address this, various techniques like momentum, adaptive learning rates (Adam, RMSProp), or learning rate schedules are used.

Gradient descent is fundamental because it enables neural networks and many other machine learning models to learn from data by improving their parameters step-by-step, leading to better predictions and generalization on unseen data. Without it, training complex models with millions of parameters would be impossible. Overall, gradient descent is a simple yet powerful tool that iteratively moves parameters downhill on the loss landscape until it finds an optimal or near-optimal solution.

2. What is regularization? What re the types of regularization”

Regularization is a technique used in machine learning and statistical modeling to prevent overfitting, which occurs when a model learns not only the underlying patterns in the training data but also the noise. Overfitting leads to poor generalization on unseen data. Regularization addresses this by adding a penalty term to the loss function, discouraging the model from becoming too complex. The idea is to constrain or regularize the coefficients of the model so that it remains simple and interpretable.

There are different types of regularization techniques, with L1 (Lasso) and L2 (Ridge) being the most common. L1 regularization adds the absolute value of the coefficients as a penalty, which can shrink some coefficients to zero, leading to sparse models that are useful for feature selection. L2 regularization adds the square of the coefficients as a penalty, which discourages large coefficients but does not necessarily eliminate them. Elastic Net is a combination of L1 and L2 regularization, benefiting from both feature selection and coefficient shrinkage. Regularization is particularly important in high-dimensional data where the number of features is large compared to the number of samples, as this increases the risk of overfitting. By penalizing large weights, regularization encourages the model to rely on the most informative features and ignore irrelevant ones.

This also makes the model more stable and less sensitive to small changes in the input data. In linear regression, regularization modifies the cost function by adding the penalty term, thereby altering the optimization landscape and leading to different solutions. In neural networks, regularization can be applied in various ways such as weight decay (which is essentially L2 regularization), dropout (randomly dropping units during training), and early stopping (halting training when validation performance begins to degrade). These methods help in preventing the network from fitting too closely to the training data. Regularization also improves the robustness of the model by smoothing the decision boundaries, especially in classification problems.

Without regularization, a model with high capacity like deep neural networks can memorize the training set and perform poorly on new data. Regularization is not limited to supervised learning; it is also used in unsupervised learning methods like clustering and dimensionality reduction. In ridge regression, the penalty term forces the coefficients to be small, distributing the effect of multicollinearity across correlated features. In contrast, Lasso regression can effectively remove redundant features by setting their coefficients to zero. Selecting the right amount of regularization is crucial and is often done using cross-validation to find the optimal regularization parameter (lambda or alpha). If regularization is too strong, the model may underfit, failing to capture the underlying patterns in the data. If it is too weak, the model may still overfit. Regularization is a fundamental concept that acts as a trade-off between bias and variance. A model with too much flexibility has low bias but high variance; regularization increases bias slightly while reducing variance significantly. This trade-off leads to better generalization performance. In practice, regularization helps create models that perform better on test data and are more reliable in production environments. It plays a key role in model interpretability, especially when dealing with complex models or large datasets.

The choice of regularization technique often depends on the problem domain, the nature of the data, and the specific goals of the analysis. In high-stakes applications like medical diagnosis or financial forecasting, regularization helps ensure that models are not only accurate but also interpretable and robust. From simple linear models to deep learning architectures, regularization remains a core component of model training and evaluation. As machine learning continues to evolve, new forms of regularization techniques are being developed to address emerging challenges in data complexity, model transparency, and generalization.