Assignment (ML): Boosting

Theory Questions

1. What is Boosting in Machine Learning?

ANS: Boosting is an ensemble technique where multiple weak learners are trained sequentially. Each new learner focuses on correcting errors made by previous ones, iteratively improving overall model accuracy to transform weak learners into a strong learner.

2. How does Boosting differ from Bagging?

ANS: Boosting trains models sequentially, adjusting sample weights to focus on errors, primarily reducing bias. Bagging trains models independently on bootstrapped samples, primarily reducing variance.

3. What is the key idea behind AdaBoost?

ANS: The key idea behind AdaBoost is to sequentially train weak learners while adaptively assigning weights to training samples. Misclassified samples receive higher weights, forcing subsequent learners to focus on them, and final predictions are combined via a weighted majority vote.

4. Explain the working of AdaBoost with an example.

AdaBoost starts with equal sample weights. In each iteration, a weak learner is trained. The error rate is calculated, and based on this, the weak learner's "say" (alpha) is determined. Sample weights are then updated: misclassified samples get increased weights, and correctly classified samples get decreased weights. This process repeats, and the final model is a weighted sum of all weak learners.

5. What is Gradient Boosting, and how is it different from AdaBoost?

ANS: Gradient Boosting trains weak learners sequentially, but instead of adjusting sample weights, each new learner predicts the "residuals" or "errors" of the previous ensemble. It minimizes a user-defined loss function by predicting its negative gradient. Unlike AdaBoost, it's more flexible and applicable to various loss functions (regression and classification).

6. What is the loss function in Gradient Boosting?

ANS: The loss function in Gradient Boosting is user-defined and can be any differentiable function. Common examples include Mean Squared Error (MSE) for regression and Log Loss (Cross-Entropy) for classification. New weak learners are trained to predict the negative gradient of this loss function.

7. How does XGBoost improve over traditional Gradient Boosting?

ANS: XGBoost improves over traditional Gradient Boosting by incorporating regularization (L1 and L2), supporting parallel processing for faster training, having built-in handling for missing values, using advanced tree pruning (post-pruning), and employing shrinkage (learning rate) for better control against overfitting.

8. What is the difference between XGBoost and CatBoost?

ANS: The main difference is their handling of categorical features: CatBoost offers built-in, native handling using ordered target encoding to prevent target leakage and uses strictly symmetric (oblivious) trees for faster prediction and reduced overfitting. XGBoost typically requires manual pre-encoding of categorical features.

9. What are some real-world applications of Boosting techniques?

ANS: Real-world applications include credit scoring and fraud detection, ad click-through rate prediction, customer churn prediction, search ranking, recommendation systems, and medical diagnosis.

10. How does regularization help in XGBoost?

ANS: Regularization (L1 and L2) in XGBoost helps prevent overfitting by adding penalty terms to the objective function. This discourages complex models and large leaf weights, leading to simpler, more robust trees that generalize better to unseen data.

11. What are some hyperparameters to tune in Gradient Boosting models?
ANS: Key hyperparameters include n_estimators (number of trees), learning_rate (shrinkage), max_depth (tree depth), subsample (fraction of samples per tree), colsample_bytree (fraction of features per tree), min_child_weight (minimum samples in a leaf), and regularization terms (gamma, reg_alpha, reg_lambda).

12. What is the concept of Feature Importance in Boosting?
ANS: Feature Importance in Boosting assigns a score to each input feature, indicating its contribution to the model's predictive power. It's typically calculated based on how often a feature is used for splitting across all trees and how much it reduces impurity or error. It aids in feature selection and model interpretability.

13. Why is CatBoost efficient for categorical data?

ANS: CatBoost is efficient for categorical data due to its:

1. **Ordered Target Encoding:** Prevents target leakage by computing statistics based only on prior data points in a random permutation.
2. **Automatic Feature Combination:** Automatically generates new features by combining existing categorical features.

3. **Oblivious Trees:** Uses symmetric trees that are faster for prediction and inherently regularize the model, making them robust.