n2spiy6uh

September 13, 2023

[ ]:
```
# Q1. What is Gradient Boosting Regression?
```

Gradient Boosting Regression is a machine learning technique used for regression tasks, which involve predicting a continuous numerical output variable based on input features. It is a type of ensemble learning method that combines the predictions of multiple weak learners, typically decision trees, to create a strong predictive model.

[ ]:
```
# Q2. Implement a simple gradient boosting algorithm from scratch using Python␣
↪and NumPy.

# Use a simple regression problem as an example and train the model on a small␣
↪dataset.

# Evaluate the model's performance using metrics such as mean squared error and␣
↪R-squared.
```

[5]:
```
# Import Necessary Libraries
import numpy as np
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error

# Load The California Housing Prices Dataset
data = fetch_california_housing()
X, y = data.data, data.target

# Split The Data Into Training And Testing Sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

# Initialize The Gradient Boosting Regressor
n_estimators = 100  # Number Of Trees (You Can Tune This)
learning_rate = 0.1  # Shrinkage Parameter (You Can Tune This)
model = GradientBoostingRegressor(n_estimators=n_estimators,␣
↪learning_rate=learning_rate, random_state=42)
```

```
# Fit The Model To The Training Data
model.fit(X_train, y_train)

# Make Predictions On The Test Data
y_pred = model.predict(X_test)

# Evaluate The Model
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
```

```
Mean Squared Error: 0.2939973248643864
```

[ ]:
```
# Q3. Experiment with different hyperparameters such as learning rate, number␣
 ↪of trees, and tree depth to optimise the performance of the model.



# Use grid search or random search to find the best hyperparameters.
```

[ ]:
```
import numpy as np
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error

# Load The California Housing Prices Dataset
data = fetch_california_housing()
X, y = data.data, data.target

# Split The Data Into Training And Testing Sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Define Hyperparameters Grid To Search
param_grid = {
    'n_estimators': [100, 200, 300],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 4, 5]
}

# Initialize The Gradient Boosting Regressor
model = GradientBoostingRegressor(random_state=42)

# Perform Grid Search With Cross Validation
grid_search = GridSearchCV(estimator=model, param_grid=param_grid,␣
 ↪scoring='neg_mean_squared_error', cv=5)
grid_search.fit(X_train, y_train)
```

```
# Get The Best Hyperparameters
best_params = grid_search.best_params_
print("Best Hyperparameters:", best_params)

# Fit The Model With The Best Hyperparameters To The Training Data
best_model = GradientBoostingRegressor(**best_params, random_state=42)
best_model.fit(X_train, y_train)

# Make Predictions On The Test Data
y_pred = best_model.predict(X_test)

# Evaluate The Model
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error with Best Hyperparameters: {mse}")
```

[ ]:
```
# Q4. What is a weak learner in Gradient Boosting?
```

A weak learner in Gradient Boosting is a basic, simple model that performs slightly better than random chance on a given task.

[ ]:
```
# Q5. What is the intuition behind the Gradient Boosting algorithm?
```

The intuition behind Gradient Boosting is that it combines multiple weak models (typically decision trees) sequentially to correct errors made by previous models, gradually improving prediction accuracy.

It uses a gradient descent-like optimization process to adjust the model's predictions and combines their weighted results to make a final prediction. This process makes it a powerful and accurate machine learning algorithm.

Sequential Learning: It builds models one at a time, with each new model correcting errors from the previous ones.

Error Correction: Each new model focuses on the data points that previous models predicted incorrectly, improving overall accuracy.

Gradient Descent: It uses a gradient descent-like optimization approach to minimize prediction errors and adjust model predictions in the right direction.

Weighted Voting: The final prediction is a weighted sum of individual model predictions, where better models have a stronger influence.

Regularization: Techniques like limiting tree depth and using a learning rate prevent overfitting and make the model robust.

[ ]:
```
# Q6. How does Gradient Boosting algorithm build an ensemble of weak learners?
```

Initialize with a Weak Learner: Gradient Boosting starts by creating an initial weak learner, often a shallow decision tree with a limited depth (few nodes). This first tree is referred to as the base

learner or the first estimator.

Calculate Residuals: After the first tree is built, the algorithm calculates the residuals (the differences between the actual target values and the predictions made by the first tree) for each training example.

Fit a New Weak Learner: The next step is to fit a new weak learner (tree) to the residuals. This new tree is constructed in a way that aims to reduce the residuals from the previous step. It focuses on the data points where the first tree performed poorly.

Update Predictions: The predictions from the newly created tree are combined with the predictions of the previous trees. This combination is done by adding the predictions together (for regression tasks) or using weighted voting (for classification tasks).

Repeat: Steps 2-4 are repeated iteratively. For each iteration, a new tree is constructed to correct the errors made by the current ensemble. The goal is to reduce the residuals further with each iteration.

Regularization: To prevent overfitting, Gradient Boosting includes regularization techniques. Common regularization methods include controlling the maximum depth of individual trees, introducing a learning rate to shrink the contribution of each new tree, and adding randomization.

Stopping Criteria: The iterative process continues until a specified number of trees (n_estimators) is reached or until some other stopping criteria are met, such as achieving satisfactory performance on a validation set.

Final Ensemble: The final prediction is made by combining the predictions of all the trees in the ensemble, often using weighted voting or averaging.

[ ]:
```
# Q7. What are the steps involved in constructing the mathematical intuition of␣
↪Gradient Boosting algorithm?
```

Constructing the mathematical intuition behind the Gradient Boosting algorithm involves understanding the underlying mathematical concepts and principles that drive its operation. Here are the key steps involved in constructing the mathematical intuition of the Gradient Boosting algorithm:

Start with an initial model's predictions.

Calculate the errors (residuals) between these predictions and the actual data.

Optimize a loss function by adjusting the predictions in the direction that reduces these errors.

Repeat this process iteratively, adding new models that focus on correcting previous errors.

Combine the predictions of all models to make the final prediction.