Improving Results

Course:
INFO-6145 Data Science and Machine Learning

Revised by:
Mohammad Noorchenarboo

November 28, 2024

Contents

Current Section

What is Cross Validation?

Cross Validation: A statistical method used to evaluate the performance of a machine learning model by testing it on unseen data.

Why is Cross Validation Important?

- Prevents overfitting by testing the model on unseen data.
- Provides a more accurate estimate of model performance.
- Ensures the model generalizes well to new data.

Key Idea: Split the data into multiple subsets, train on some subsets, and validate on others.

Types of Cross Validation

1. K-Fold Cross Validation

- Split the data into **K** subsets (folds).
- Train the model on **K-1** folds and validate on the remaining fold.
- Repeat this process **K** times, each time using a different fold for validation.
- Average the performance across all folds.

Example: K=5

- Split data into 5 folds.
- Use 4 folds for training, 1 fold for validation.
- Rotate the validation fold for each iteration.

Types of Cross Validation

2. Leave-One-Out Cross Validation (LOOCV)

- Special case of K-Fold Cross Validation where K = number of samples.
- Train the model on all data except one instance, then validate on the left-out instance.
- Repeat this for all instances.

3. Stratified K-Fold Cross Validation

- Ensures each fold has a similar distribution of classes (for classification problems).
- Prevents imbalances in the training and validation sets.

Cross Validation Workflow

1. Split the dataset into **K** folds.
2. For each fold:
   - Train the model on **K-1** folds.
   - Validate the model on the left-out fold.
3. Compute the performance metric (e.g., accuracy, RMSE) for each fold.
4. Average the performance metrics to estimate the model's overall performance.

Cross Validation Workflow

Example with Python Code: K-Fold Cross Validation

```python
from sklearn.model_selection import KFold
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import numpy as np
# Sample data
X = [[1], [2], [3], [4], [5]]
y = [0, 1, 0, 1, 0]
# K-Fold Cross Validation
kf = KFold(n_splits=5)
model = LogisticRegression()
scores = []
for train_idx, test_idx in kf.split(X):
    X_train, X_test = np.array(X)[train_idx], np.array(X)[
        test_idx]
    y_train, y_test = np.array(y)[train_idx], np.array(y)[
        test_idx]
    model.fit(X_train, y_train)
    preds = model.predict(X_test)
    scores.append(accuracy_score(y_test, preds))
print("Cross-Validation Accuracy:", np.mean(scores))
```

Advantages and Disadvantages of Cross Validation

Advantages

- Provides a robust estimate of model performance.
- Reduces the risk of overfitting compared to a single train-test split.
- Works well with small datasets by making full use of available data.

Disadvantages

- Computationally expensive, especially with large datasets or complex models.
- May not work well with time-series data due to temporal dependencies.

When to Use Cross Validation?

Use Cross Validation When:

- You have a limited amount of data and want to maximize its usage.
- Evaluating the performance of machine learning models.
- Comparing multiple models to select the best one.

Avoid Cross Validation When:

- Working with time-series data. Use **Time Series Split** instead.
- The dataset is extremely large, as cross validation can be computationally expensive.

Current Section

What are Hyperparameters?

Hyperparameters:

- Parameters that control the behavior of a machine learning model but are not learned from the data.
- They are set before the training process begins and remain constant during training.
- Examples include the learning rate, the number of hidden layers, and the number of trees in a random forest.

Why are Hyperparameters Important?

- Control the complexity and capacity of the model.
- Influence the performance and generalizability of the model.
- Poorly chosen hyperparameters can lead to underfitting or overfitting.

What are Hyperparameters?

Examples of Hyperparameters

1. Decision Tree:

- Maximum depth of the tree.
- Minimum number of samples per leaf node.

2. Neural Networks:

- Learning rate.
- Number of hidden layers and neurons.

3. Support Vector Machines (SVM):

- Kernel type (e.g., linear, RBF).
- Regularization parameter (`C`).

What is Hyperparameter Tuning?

Hyperparameter Tuning:

- The process of finding the best set of hyperparameters for a machine learning model.
- It aims to optimize model performance (e.g., accuracy, RMSE) on unseen data.
- A trade-off between computational cost and model performance.

Why is Hyperparameter Tuning Needed?

- Default hyperparameters may not be optimal for a given dataset.
- Proper tuning can improve model accuracy and generalization.
- Helps prevent overfitting or underfitting.

Techniques for Hyperparameter Tuning

1. Grid Search

- Tests all possible combinations of a predefined set of hyperparameter values.
- Suitable for small search spaces.

Example:

- Hyperparameters: `learning rate` = [0.01, 0.1], `batch size` = [16, 32].
- Total combinations: 4.

Techniques for Hyperparameter Tuning

2. Random Search

- Randomly samples hyperparameter combinations from a predefined range.
- Faster than grid search, especially for large search spaces.

Example:

- Test 10 random combinations of hyperparameters instead of all possible values.

3. Bayesian Optimization

- Uses probabilistic models to find the best hyperparameters.
- Balances exploration (trying new combinations) and exploitation (refining known good combinations).

Workflow of Hyperparameter Tuning

1. Define the hyperparameters to tune and their ranges or values.
2. Select a tuning technique (e.g., grid search, random search).
3. Split the dataset into training and validation sets.
4. For each combination of hyperparameters:
   - Train the model on the training set.
   - Evaluate performance on the validation set.
5. Choose the combination with the best performance metric.
6. Test the selected hyperparameters on a separate test set.

Workflow of Hyperparameter Tuning

Example with Scikit-learn's GridSearchCV

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

# Define hyperparameters
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [10, 20, None]}

# Initialize the model and GridSearchCV
model = RandomForestClassifier()
grid_search = GridSearchCV(model, param_grid, cv=5, scoring=
    'accuracy')

# Fit the model
grid_search.fit(X_train, y_train)

# Best hyperparameters and score
print("Best Params:", grid_search.best_params_)
print("Best Score:", grid_search.best_score_)
```

Advantages and Disadvantages of Hyperparameter Tuning

Advantages

- Improves model performance and generalization.
- Systematically explores the parameter space.
- Helps tailor the model to specific datasets.

Disadvantages

- Computationally expensive, especially for large datasets or complex models.
- Time-consuming for large search spaces or high-dimensional hyperparameters.
- Risk of overfitting to the validation set if not carefully managed.