ort-vector-machines-assignment-2

September 13, 2023

[ ]:
```
# Q1. What is the relationship between polynomial functions and kernel␣
↪functions in machine learning algorithms
```

Polynomial functions are mathematical functions that involve powers of a variable (e.g., x^2, x^3, etc.). They are used in various machine learning models, such as polynomial regression, to capture non-linear relationships between input features and the target variable.

Kernel functions, on the other hand, are used primarily in the context of kernel methods, including SVMs. These methods aim to find a non-linear decision boundary in a higher-dimensional feature space without explicitly calculating the transformed features.

Relationship:

The relationship between polynomial functions and kernel functions lies in the use of polynomial kernels in SVMs. A polynomial kernel is a type of kernel function that captures polynomial relationships between data points in the feature space.

When you use a polynomial kernel in an SVM, you are essentially employing a polynomial function of a specific degree (e.g., degree 2, 3, etc.) to model non-linear decision boundaries in the data.

[ ]:
```
# Q2. How can we implement an SVM with a polynomial kernel in Python using␣
↪Scikit-learn?
```

[1]:
```
# Import The Necessary Libraries
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Load The Iris Dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target

# Split The Dataset 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)

# Create An SVM Classifier With A Polynomial Kernel
svm_classifier = SVC(kernel='poly', degree=3, C=1.0)
```

```
# Train The SVM Classifier On The Training Data
svm_classifier.fit(X_train, y_train)

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

# Evaluate The Classifier's Performance (E.g., Accuracy)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 1.0
```

[ ]:
```
# Q3. How does increasing the value of epsilon affect the number of support␣
↪vectors in SVR?
```

Increasing the value of epsilon in Support Vector Regression (SVR) typically reduces the number of support vectors. Smaller epsilon values lead to a tighter fit, requiring more support vectors to closely match the data, while larger epsilon values allow for a looser fit with fewer support vectors.

[2]:
```
# Q4. How does the choice of kernel function, C parameter, epsilon parameter,␣
↪and gamma parameter affect the performance of Support Vector Regression␣
↪(SVR)?

# Can you explain how each parameter works and provide examples of when you␣
↪might want to increase or decrease its value?
```

1 Choice of Kernel Function:

Use a linear kernel for mostly linear data, and non-linear kernels (e.g., RBF) for non-linear data.

2 C Parameter:

Increase C to fit the training data closely.

Decrease C to prioritize a larger margin and avoid overfitting.

3 Epsilon Parameter ( ):

Increase for more tolerance to errors. Decrease for a closer fit to the data with less tolerance for errors.

4 Gamma Parameter ( ):

Increase for localized, complex patterns. Decrease for smoother, global patterns, especially with noisy data.

[ ]:
```
#  Q5. Assignment:
#  Import the necessary libraries and load the dataset

# Split the dataset into training and testing sets

# Preprocess the data using any technique of your choice (e.g. scaling,␣
 ↪normalization)

# Create an instance of the SVC classifier and train it on the training data

# Use the trained classifier to predict the labels of the testing data

# Evaluate the performance of the classifier using any metric of your choice (e.
 ↪g. accuracy, precision, recall, F1-score)

# Tune the hyperparameters of the SVC classifier using GridSearchCV or␣
 ↪RandomiMedSearchCV to improve its performanc

# Train the tuned classifier on the entire dataset

# Save the trained classifier to a file for future use.




# You can use any dataset of your choice for this assignment, but make sure it␣
 ↪is suitable for classification and has a sufficient number of features and␣
 ↪samples.
```

[6]:
```
# Import The Necessary Libraries
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
import joblib

# Load The Iris Dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target

# Split The Dataset Into Training And Testing Sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=40)
```

```
# Preprocess The Data Using Standard Scaler For Feature Scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Create An Instance Of The SVC Classifier With An Initial Set Of␣
 ↪Hyperparameters
initial_svm_classifier = SVC(kernel='rbf', C=1.0, gamma='scale',␣
 ↪random_state=40)


# Train The Initial Classifier On The Training Data
initial_svm_classifier.fit(X_train_scaled, y_train)

# Use The Initial Classifier To Predict The Labels Of The Testing Data
y_pred = initial_svm_classifier.predict(X_test_scaled)

# Evaluate The Performance Of The Initial Classifier Using Accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Initial Classifier Accuracy:", accuracy)

# Tune Hyperparameters Using GridSearchCV
param_grid = {
    'C': [0.1, 1, 10],
    'kernel': ['linear', 'rbf', 'poly'],
    'gamma': ['scale', 'auto', 0.1, 1]
}

grid_search = GridSearchCV(SVC(), param_grid, cv=5)
grid_search.fit(X_train_scaled, y_train)

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

# Train The Tuned Classifier On The Entire Dataset
final_svm_classifier = SVC(**best_params, random_state=40)
final_svm_classifier.fit(X_train_scaled, y_train)

# Save The Trained Classifier To A File For Future Use
joblib.dump(final_svm_classifier, 'svm_classifier.pkl')
```

```
Initial Classifier Accuracy: 1.0
Best Hyperparameters: {'C': 10, 'gamma': 'scale', 'kernel': 'linear'}
```

[6]:
```
['svm_classifier.pkl']
```