geeksforgeeks.org

Multiclass classification using Support Vector Machines (SVM)

GeeksforGeeks

8–10 minutes

Last Updated : 26 Aug, 2024

Support Vector Machines (SVM) are widely recognized for their effectiveness in binary classification tasks. However, real-world problems often require distinguishing between more than two classes. This is where multi-class classification comes into play. While SVMs are inherently binary classifiers, they can be extended to handle multi-class classification problems. This article explores the techniques used to adapt SVMs for multi-class tasks, the challenges involved, and how to implement multi-class SVMs using scikit-learn.

Table of Content

Understanding Multi-Class Classification

In multi-class classification, the goal is to categorize an instance into one of three or more classes. For example, consider a problem where you need to classify an image as either a cat, dog, or bird. Unlike binary classification, which deals with two classes, multi-class classification must handle multiple possible outcomes.

Multi-Class Classification With SVMs

SVMs are designed to separate data points into two distinct classes by finding the optimal hyperplane that maximizes the margin between the classes. This binary nature makes SVMs particularly well-suited for two-class problems. However, when there are more than two classes, SVMs need to be adapted to handle the additional complexity. **Multi-class Classification Challenges:**

- In binary classification, SVMs are straightforward as they separate data into two classes.
- However, multi-class classification involves more complexity since there are more than two classes to consider. The challenge is to extend the binary classification capability of SVMs to handle multiple classes effectively.

Strategies for Multi-class Classification with SVM

When dealing with multi-class classification using Support Vector Machines (SVM), two primary strategies are commonly employed: **One-vs-One (OvO)** and **One-vs-All (OvA)**. Each approach has its own methodology and application scenarios, making them suitable for different types of classification problems.

1. One-vs-One (OvO) Approach

The One-vs-One (OvO) approach involves creating a binary classifier for every possible pair of classes. This method is particularly advantageous when the number of classes is relatively small, as it allows for a focused comparison between pairs of classes. Each classifier is responsible for distinguishing between two specific classes, effectively ignoring the others.

Implementation:

1. **Training Phase:**
   - For each pair of classes, a binary SVM classifier is trained. For instance, if there are three classes (A, B, and C), the OvO approach will train classifiers for (A vs B), (A vs C), and (B vs C).
   - Each classifier is trained using only the data points belonging to the two classes it is meant to distinguish.
2. **Prediction Phase:**
   - During prediction, each binary classifier provides a vote for one of the two classes it was trained on.
   - The final class prediction for a given instance is determined by a majority vote among all the classifiers.
   - This voting mechanism ensures that the class with the most

votes is selected as the final output.

2. One-vs-All (OvA) Approach

The One-vs-All (OvA) approach, also known as One-vs-Rest, involves training a single binary classifier for each class. In this method, each class is treated as the positive class, while all other classes are grouped together as the negative class. This approach is straightforward and scales linearly with the number of classes.

Implementation:

1. **Training Phase:**
- For each class, a binary SVM classifier is trained. For a dataset with n classes, this results in n classifiers.
- Each classifier is trained to distinguish its respective class from all other classes combined.
2. **Prediction Phase:**
- During prediction, each classifier outputs a score or probability indicating the likelihood of the instance belonging to its respective class.
- The class with the highest score is selected as the prediction for the instance.

Implementing Multi-Class SVMs in Scikit-Learn

To implement multi-class classification using SVM in Python, we can utilize libraries such as Scikit-learn, which provides built-in support for both OvO and OvA strategies.

- **One-vs-One (OvO) Implementation:**

```
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 Wine dataset
wine = datasets.load_wine()
X, y = wine.data, wine.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.3,
random_state=42)

# Initialize the SVM classifier with One-vs-One
strategy
svm_ovo = SVC(decision_function_shape='ovo')
svm_ovo.fit(X_train, y_train)

# Predict and evaluate the One-vs-One model
y_pred_ovo = svm_ovo.predict(X_test)
print("One-vs-One Accuracy:",
accuracy_score(y_test, y_pred_ovo))
```

Output:

One-vs-One Accuracy: 0.7592592592592593

- **One-vs-Rest (OvR) Implementation:**

```
# Initialize the SVM classifier with One-vs-All
```

```
strategy
svm_ova = SVC(decision_function_shape='ovr')
svm_ova.fit(X_train, y_train)

# Predict and evaluate the One-vs-All model
y_pred_ova = svm_ova.predict(X_test)
print("One-vs-All Accuracy:",
accuracy_score(y_test, y_pred_ova))
```

Output:

One-vs-All Accuracy: 0.7592592592592593

Explanation:

- **Wine Dataset**: This dataset contains 178 samples of wine, each with 13 features, and is divided into three classes. It is a well-known dataset for practicing classification algorithms.
- **Data Splitting**: The dataset is split into training and testing sets using a 70-30 split.
- **SVM Classifiers**: Two SVM classifiers are initialized, one for each strategy (OvO and OvA), using the `decision_function_shape` parameter to specify the strategy.
- **Evaluation**: The accuracy of each model is calculated using the `accuracy_score` function from Scikit-learn.

Choosing Between OvO and OvA

The choice between One-vs-One and One-vs-All strategies depends on several factors, including the number of classes, the size of the dataset, computational resources, and the

specific requirements of the application.

- **One-vs-One** is generally preferred when the number of classes is small and computational resources are sufficient, as it can provide higher accuracy through focused pairwise comparisons.
- **One-vs-All** is more suitable for scenarios with a large number of classes or when simplicity and computational efficiency are prioritized.

  Both strategies have their strengths and weaknesses, and the decision should be guided by the specific context of the problem being addressed.

Challenges in Multi-Class SVMs

While SVMs are powerful, multi-class classification using SVMs can be challenging due to:

- **Scalability:** As the number of classes increases, the complexity of training and prediction can grow significantly, particularly with the OvO approach.
- **Class Imbalance:** OvR can suffer from class imbalance, where the positive class is much smaller than the negative class. This can lead to biased models that favor the larger class.
- **Computational Cost**: The training time and memory requirements can increase substantially with the number of classes, particularly in OvO, where many classifiers are trained.

Conclusion

Support Vector Machines are inherently binary classifiers, but they can be effectively adapted for multi-class classification using strategies like One-vs-Rest and One-vs-One. Scikit-learn provides robust implementations of these strategies, making it easier to apply SVMs to multi-class problems. While challenges exist, such as scalability and computational cost, the choice between OvR and OvO depends on the specific needs of your application. By understanding these approaches and their trade-offs, you can leverage the power of SVMs for multi-class classification tasks.