Support Vector Machines for Multiclass Classification

Grigorev Ilia
Innopolis University, Beijing Institute of Technology
Computer Science and Technology School
il.grigorev@innopolis.university

I. Introduction

Classification is a problem of assigning to an object a number from a finite set of values. This problem has been widely studied in the field of machine learning (ML) [1], and many approaches have been proposed. One of such methods is the Support Vector Machine (SVM). SVM is known to be a straightforward and robust solution for the binary classification of linearly separable data. The combination of multiple SVMs can also produce multiclass classification. In this paper, I intend to demonstrate the results of applied SVMs for multiclass classification of linearly separable data using a top-notch framework.

II. Method

A. Dataset

ML problems require training models on datasets related to the specific problem and assessing the results of a model. The last step helps to compare the results of one solution with other models and improve technology. Typically, datasets related to the well-known problem are already collected, and training can proceed directly with a downloaded dataset.

In the case of the multiclass classification of linearly separable data, many datasets exist, from which I decided to use the IRIS dataset [2]. This dataset represents the information on iris flowers, which consists of four features: the length and the width of the sepals and petals, in centimeters. The dataset maps these features to one of the three classes: iris setosa, iris virginica, and iris versicolor. Each group of flowers consists of exactly 50 individuals resulting in the total size of a dataset of 150 items.

To load the dataset, I used a powerful Python library `sklearn`. The library provides a method to load the dataset to the program directly.

B. Model selection

To solve the stated classification problem, a model should be proposed. In this paper, the SVM model is used. According to Rogers and Girolami [2], SVM is defined as follows:

$$a(x) = \text{sgn}(\langle \omega, x \rangle + b), \tag{1}$$

where x is the vector of features, ω is the vector of coefficients, and b is the bias term. The dataset $X_{n \times d}$ consists of row vectors of individuals, where columns are features.

The training process is used to find the best combination of ω and b:

$$\underset{\omega}{\text{argmax}} \frac{1}{||\omega||},$$
$$\text{subject to } y^i(\langle \omega, x^i \rangle + b) \geq 1 \text{ for all } 1 \leq i \leq n$$

Using Lagrangian duality, the problem is reduced to the following form:

$$\underset{\lambda}{\text{argmax}} \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j=1}^{n} \lambda_i \lambda_j y^i y^j \langle x^i, x^j \rangle$$

The last problem is quadratic, and solution can be obtained using quadratic programming solver. However, the solution will significantly overfit with the high noise in the dataset.

To overcome this problem, the soft-margin SVM is proposed:

$$\underset{\omega}{\text{argmin}} \frac{1}{2}||\omega||^2 + C \sum_{i=1}^{n} \xi_i$$
$$\text{subject to } y^i(\langle w, x^i \rangle + b) \geq 1 - \xi_i$$
$$\xi_i \geq 0 \text{ for all } 1 \leq i \leq n$$

Likewise, the dual problem is as follows:

$$\underset{\omega}{\text{argmax}} \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j=1}^{n} \lambda_i \lambda_j y^i y^j \langle x^i, x^j \rangle$$
$$\text{subject to } \sum_{i=1}^{N} \lambda_i y^i = 0 \text{ and } 0 \leq \lambda_i \leq C \text{ for all } i$$

The constant C is used to control how each support vector influences the position of the hyperplane that separates the points. As C decreases, the maximum potential influence of each training point gets smaller, so more and more of them become active in the decision function. This hyperparameter helps to make the model more resistant to noise, and, thus, less overfit.

C. Multiclass Classification

The binary SVM classifiers can be combined in several ways to produce multiclass classification [1].

D. Implementation

In the following Jupyter notebook, I implemented several approaches to build SVM models. The list includes: Lagrange dual problem solution using gradient descent or artificial bee colony, and Hinge Loss reduction using gradient descent.

E. Framework

Instead of composing the model and training it manually, I used the `sklearn` Python library. The library provides the class `LinearSVC` with the methods for model creation, fitting, and making predictions. In model creation, I specified decided to do the grid search for the hyperparameter C. The rest of the parameters are left to be default.

F. Metrics

The following metrics were used to evaluate the result of the classificator:

- Accuracy
- Precision (for each of three classes)
- Recall (for each of three classes)
- F1 (for each of three classes)

III. Experiment

The training of the model was handled with the Google colaboratory server with the following configurations: System RAM of 12.7 GB, Disk of 107.7 GB, 2vCPU @ 2.2GHz.

The model was trained using the train dataset which is the 70% of the initial dataset. Moreover, the K-fold cross-validation was used for the hyperparameter setup.

A. Source code

Below you can see the source code of the proposed model creation, training, and evaluation:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import
    train_test_split
from sklearn.model_selection import
    GridSearchCV
from sklearn.svm import SVC

data, target = load_iris(return_X_y=True)
X_train, X_test, Y_train, Y_test =
    train_test_split(data, target,
    test_size=.3)

parameters = {'C':[1, 10]}
svc = SVC()
clf = GridSearchCV(svc, parameters)
clf.fit(X_train, Y_train)

from sklearn.metrics import accuracy_score,
    precision_score, recall_score, f1_score

print(f"Best parameter value:
    {clf.best_params_}, best training score:
    {clf.best_score_}")
svc = clf.best_estimator_

Y_pred = svc.predict(X_test)
print(f"Metrics:\n\taccuracy:
    {accuracy_score(Y_test, Y_pred)}")
print(f"\tprecision: {precision_score(Y_test,
    Y_pred, average=None)}")
print(f"\trecall: {recall_score(Y_test,
    Y_pred, average=None)}")
print(f"\tf1: {f1_score(Y_test, Y_pred,
    average=None)}")
```

IV. Results

In this section, the results of the training are demonstrated.

First, the best training score during the hyperparameter search gained the value $C = 1$.

The following table shows the following metrics after evaluation in the test dataset.

TABLE I
Model Metrics Scores

Metrics									
Accuracy	Recall			Precision			F1		
	IS	IV	IV	IS	IV	IV	IS	IV	IV
0.956	1.0	0.9	0.95	1.0	0.9	0.95	1.0	0.9	0.95

V. Discussion

The results obtained demonstrate sufficient performance of the model. Compared to one of the recent studies related to SVM [3], the results on the same dataset are similar and demonstrate strong capabilities of the method for classifying linearly separable objects. Unlike Z. F. Hussain et al. [3], I used the grid search for hyperparameter tuning, which may be less efficient as compared to genetic programming. However, since the problem dataset lacks noise, hyperparameter tuning is excessive, which suggests further investigation on SVM capabilities.

VI. Conclusion

Overall, this paper demonstrated the approach to separating multidimensional data into classes using a hyperplane. The method of support vector machines is useful for classification if the data is linearly separable, although the kernel method can be used to solve the problem [1]. Finally, I report the limitation of the paper, which is the lack of rigorousness in the SVM noise capabilities research.

References

[1] S. Rogers and M. Girolami, "A First Course in Machine Learning," CRC Press, 2012

[2] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugenics*, 1936

[3] Z. F. Hussain et al., "A new model for iris data set classification based on linear support vector machine parameter's optimization," *Int. J. Elect. Comput. Eng.*, 2020