Implementation of Support Vector Machine

Lab Assignment 4

Sumedha Janani Siriyapuraju
Dept. of Electronics and Communication Engineering
Visvesvaraya National Institute of Technology
Nagpur, India
sumedhasjs@gmail.com

***Abstract*—Support Vector Machine (SVM) is a robust classification and regression technique that maximizes the predictive accuracy of a model without overfitting the training data. SVM is particularly suited to analyzing data with very large numbers (for example, thousands) of predictor fields.**

In this work we tried to use analytical solution to fit a curve to the fisheriris_matlab.mat Dataset.

***Index Terms*—Support Vector Machine, Linear Kernal, Polynomial Kernal, RBF Kernal, AUROC Score, ROC Curve, Confusion Matrix, Accuracy, Specificity, Sensitivity**

I. Introduction

Support vector machines (SVMs) are a set of related supervised learning methods, which are popular for performing classification and regression analysis using data analysis and pattern recognition. Methods vary on the structure and attributes of the classifier. The most commonly known SVM is a linear classifier, predicting each input's member class between two possible classifications. A more accurate definition would state that a support vector machine builds a hyperplane or set of hyperplanes to classify all inputs in a high-dimensional or even infinite space. The closest values to the classification margin are known as support vectors. The SVM's goal is to maximize the margin between the hyperplane and the support vectors.

Support vector machines are very popular and many consider them as the best off-the-shelf classifier. Furthermore, there are a wide selection of environments and toolboxes that implement SVMs. For these reasons we chose to apply SVMs to the problem of classifying infeasible test cases.

II. Method

A. Part A: SVM using 3 kernels

To construct the support vectors, 3 kernels—linear, radial basis function (RBF), and polynomial—were utilised. In all these 3 kernels, the hyper-parameters were tuned by using the gridSearchCV function The function accepted the grid parameters as a dictionary and returned a dictionary containing the values at which the model's efficiency would be at its peak. The tuned hyper-parameters for the linear and RBF kernels were C and gamma. Along with C and gamma, the degree of the polynomial was tweaked in polynomial kernels.

The SVM model was then trained using the obtained hyper-parameters. Using the testing dataset, the accuracy was evaluated after training. To determine if the value selected by gridSearch was appropriate or not, the relationship between hyper-parameter and accuracy was shown.

B. Part B: Accuracy, sensitivity and specificity

The confusion matrix was plotted for all 3 kernels separately. In each of the confusion matrix, the precision, recall, F1 was mentioned for all 3 target classes. Hence the shape came out to be 9*9. The accuracy of the model was given under the accuracy label. The TP, TN, FP and FN of a 3 or more class classifier is found out by :
TP = sum of diagonal elements of 3*3 confusion matrix
FP = (rowwise sum of array elements) - (sum of diagonal elements) of 3*3 confusion matrix
FN = (coloumnwise sum of array elements) - (sum of diagonal elements) of 3*3 confusion matrix
FP = sum of all elements of 3*3 confusion matrix - (FP+FN+TP)
The sensitivity and specificity are then found out using the formula
$sensitivity = tp/(tp + fn)$ and $specificity = tn/(tn + fp)$ respectively.

III. Results

	0	1	2	3
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

Fig. 1. Data Preprocessing - Initial Dataframe

	0	1	2	3	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Fig. 2. Data Preprocessing - Initial Dataframe with Classification

	Sepal Length	Sepal Width	Petal Length	Petal Width	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Fig. 3. Data Preprocessing - Final Dataframe

```
0.625000 with: {'C': 0.001, 'gamma': 'scale', 'kernel': 'linear'}
0.625000 with: {'C': 0.01, 'gamma': 'scale', 'kernel': 'linear'}
1.000000 with: {'C': 1, 'gamma': 'scale', 'kernel': 'linear'}
0.937500 with: {'C': 10, 'gamma': 'scale', 'kernel': 'linear'}
0.937500 with: {'C': 100, 'gamma': 'scale', 'kernel': 'linear'}
```

Fig. 4. Linear Kernel

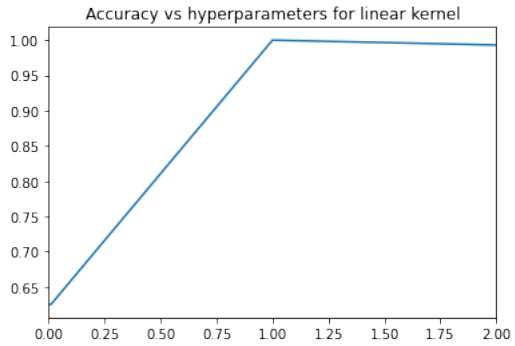

Fig. 5. Accuracy of Linear Kernel

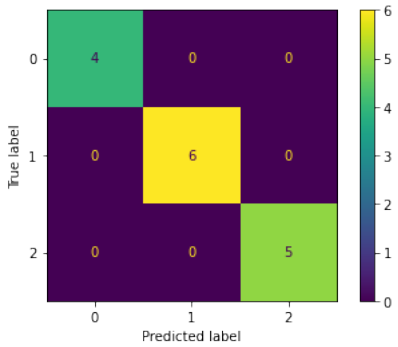

Fig. 6. Confusion Matrix of Linear Kernel

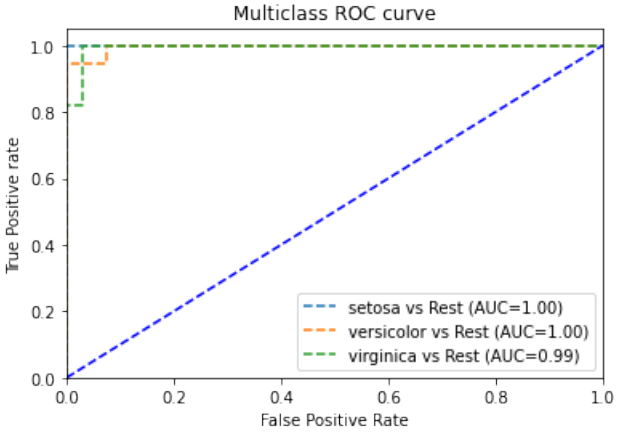

Fig. 7. ROC Curve of Linear Kernel

```
AUROC Score:  0.9968457266932212
```

Fig. 8. AUROC Score of Linear Kernel

```
0.500000 with: {'C': 0.001, 'gamma': 'scale', 'kernel': 'poly'}
0.500000 with: {'C': 0.01, 'gamma': 'scale', 'kernel': 'poly'}
1.000000 with: {'C': 1, 'gamma': 'scale', 'kernel': 'poly'}
1.000000 with: {'C': 10, 'gamma': 'scale', 'kernel': 'poly'}
1.000000 with: {'C': 100, 'gamma': 'scale', 'kernel': 'poly'}
```

Fig. 9. Polynomial Kernel

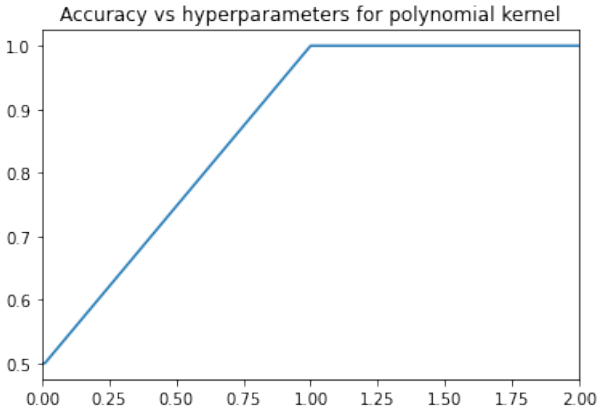

Fig. 10. Accuracy of Polynomial Kernel

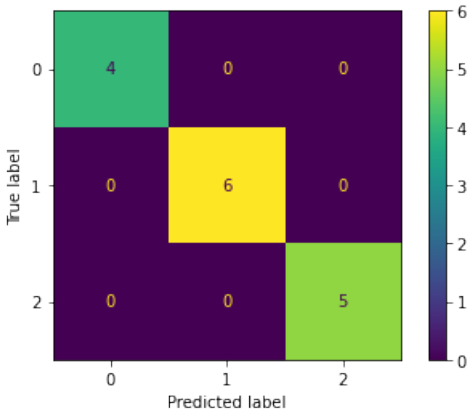

Fig. 11. Confusion Matrix of Polynomial Kernel

```
AUROC Score:  0.9952685900398318
```

Fig. 12. AUROC Score of Polynomial Kernel

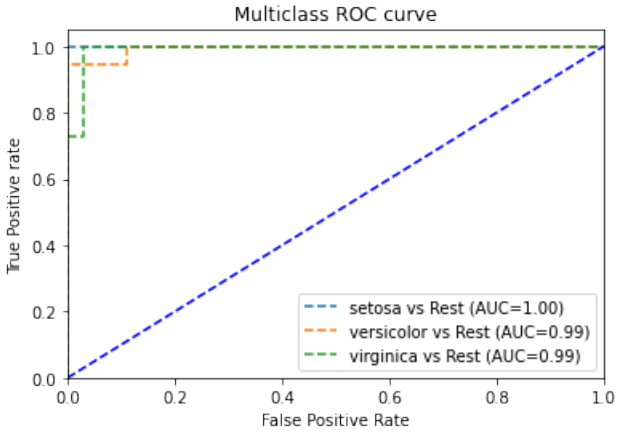

Fig. 13. ROC Curve of Polynomial Kernel

```
0.625000 with: {'C': 0.001, 'gamma': 0.1, 'kernel': 'rbf'}
0.625000 with: {'C': 0.001, 'gamma': 1, 'kernel': 'rbf'}
0.562500 with: {'C': 0.001, 'gamma': 10, 'kernel': 'rbf'}
0.625000 with: {'C': 0.01, 'gamma': 0.1, 'kernel': 'rbf'}
0.625000 with: {'C': 0.01, 'gamma': 1, 'kernel': 'rbf'}
0.562500 with: {'C': 0.01, 'gamma': 10, 'kernel': 'rbf'}
0.812500 with: {'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}
0.875000 with: {'C': 1, 'gamma': 1, 'kernel': 'rbf'}
0.562500 with: {'C': 1, 'gamma': 10, 'kernel': 'rbf'}
1.000000 with: {'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
0.875000 with: {'C': 10, 'gamma': 1, 'kernel': 'rbf'}
0.562500 with: {'C': 10, 'gamma': 10, 'kernel': 'rbf'}
0.937500 with: {'C': 100, 'gamma': 0.1, 'kernel': 'rbf'}
0.875000 with: {'C': 100, 'gamma': 1, 'kernel': 'rbf'}
0.562500 with: {'C': 100, 'gamma': 10, 'kernel': 'rbf'}
```

Fig. 14. RBF Kernel

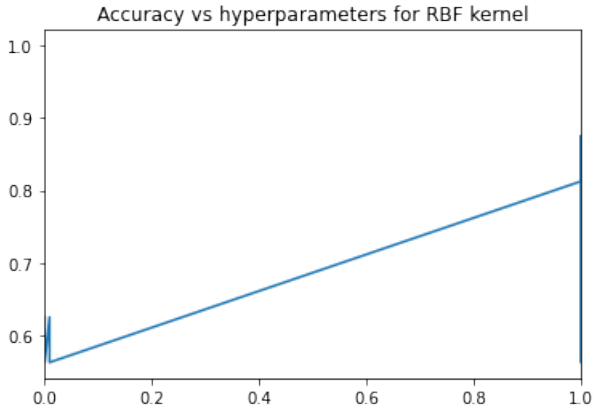

Fig. 15. Accuracy of RBF Kernel

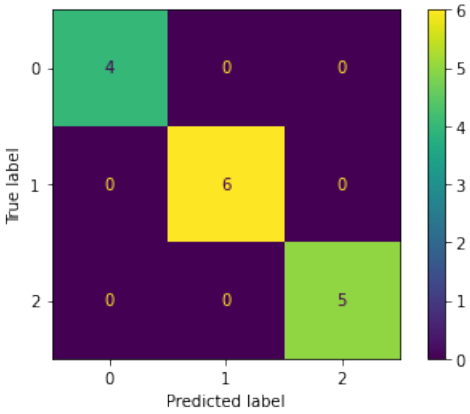

Fig. 16. Confusion Matrix of RBF Kernel

```
AUROC Score:  0.9968457266932212
```

Fig. 17. AUROC Score of RBF Kernel

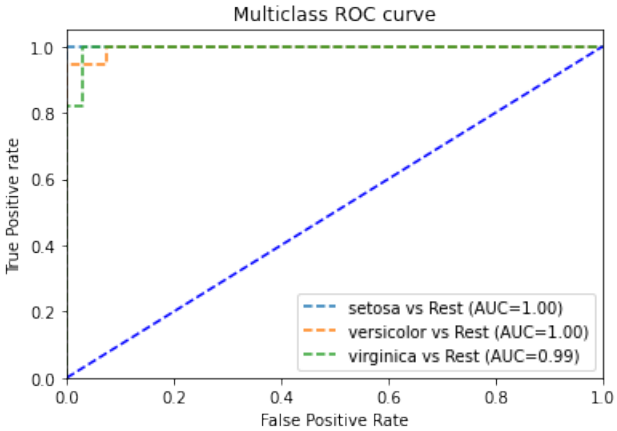

Fig. 18. ROC Curve of RBF Kernel

```
Sensitivity for Setosa:  1.0
Specificity for Setosa :  1.0
```

Fig. 19. Sensitivity and Specificity of Setosa for all 3 kernels

```
Sensitivity for Versicolor:  1.0
Specificity for Versicolor:  1.0
```

Fig. 20. Sensitivity and Specificity of Versicolor for all 3 kernels

IV. Discussion

- An accuracy of 87.5% was achieved.
- The value of C and gamma were taken as said by gridSearch
- The graphs of hyperparameter vs accuracy supported the results of gridSearch
- The value of true negative, true positive, false negative and false positive are found out from the confusion matrix using the formula stated above

V. Conclusion

Classification of iris was done successfully using support vector machine algorithm.

Appendix (Code)

The code is available here

```
Sensitivity for Virginia:  1.0
Specificity for Virginia:  1.0
```

Fig. 21. Sensitivity and Specificity of Virginia for all 3 kernels