Support Vector Machine Report

Support Vector Machine Method

Support Vector Machines (SVM) is method for the classification of both linear and nonlinear data. It uses a nonlinear mapping to transform the original training data into a higher dimension. Within this new dimension, it searches for the linear optimal separating hyperplane (i.e., a "decision boundary" separating the tuples of one class from another). With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane. The SVM finds this hyperplane using support vectors ("essential" training tuples) and margins (defined by the support vectors).

Support Vectors

Support vectors are the data points that lie closest to the decision surface (or hyperplane). They are the data points most difficult to classify and they have direct bearing on the optimum location of the decision surface. The Support Vectors constrain the width of the margin. And since they are very less as compared to the total number of data points, they hand us many advantages.

Feature Selection

To select two best attributes, SelectKBest method is used from sklearn.feature_selection. This method takes as a parameter a score function, which must be applicable to a pair (X, y). The score function must return an array of scores, one for each feature X[:,i] of X (additionally, it can also return p-values, but these are neither needed nor required). SelectKBest then simply retains the first k features of X with the highest scores.

So, for example, if you pass chi2 as a score function, SelectKBest will compute the chi2 statistic between each feature of X and y (assumed to be class labels). A small value will mean the feature is independent of y. A large value will mean the feature is non-randomly related to y, and so likely to provide important information. Only k features will be retained.

One-vs-One and One-vs-All method

The difference between these two is the number of classifiers you have to learn, which strongly correlates with the decision boundary they create.

Assume you have N different classes. One vs all will train one classifier per class in total N classifiers. For class i it will assume i-labels as positive and the rest as negative. This often leads to imbalanced datasets meaning generic SVM might not work, but still there are some workarounds.

In one vs one you have to train a separate classifier for each different pair of labels. This leads to $N(N-1)/2$ classifiers. This is much less sensitive to the problems of imbalanced datasets but is much more computationally expensive.

<u>Method used for this task</u>

For this assignment, I have used 'one-vs-one' method which is define as an argument of SVC() method.

When we use the OVO strategy, we have to train a set of K(K −1)/2 binary classifiers between each pair of the classes. Then all the samples representing all classes are tested against these classifiers which vote for each sample. This brings us to the problem of incompetent classifiers. Another problem can be seen when the number of classes increases. The number of binary classifiers rises quadratically and all the samples have to be tested against each classifier during the testing phase.

The OVR strategy uses samples of all the classes to train each binary classifier. However, the samples from one distinguished class are treated as the class one, ω1, and all the other samples are considered to belong to the class rest, ωr. Compared with the OVO strategy, the number of binary classifiers which we have to train is quite small.

From the above explanation it is clear that as we have selected only two features, it is better to use OVO.

Results Interpretation

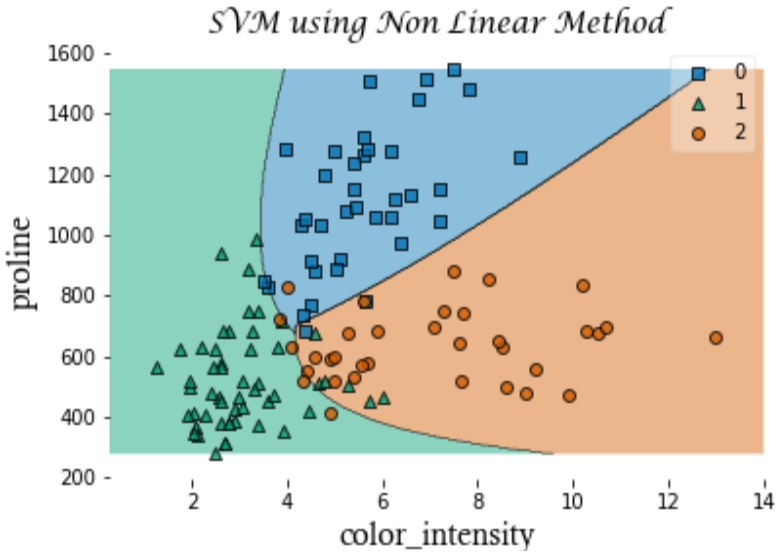
For splitting the train and test data in the assignment, 'train_test_split' is used from "sklearn.model_selection". This function splits the data every time in the random order and hence the output of the code would be different every time because of this function. Below are the accuracies of all three SVM methods.

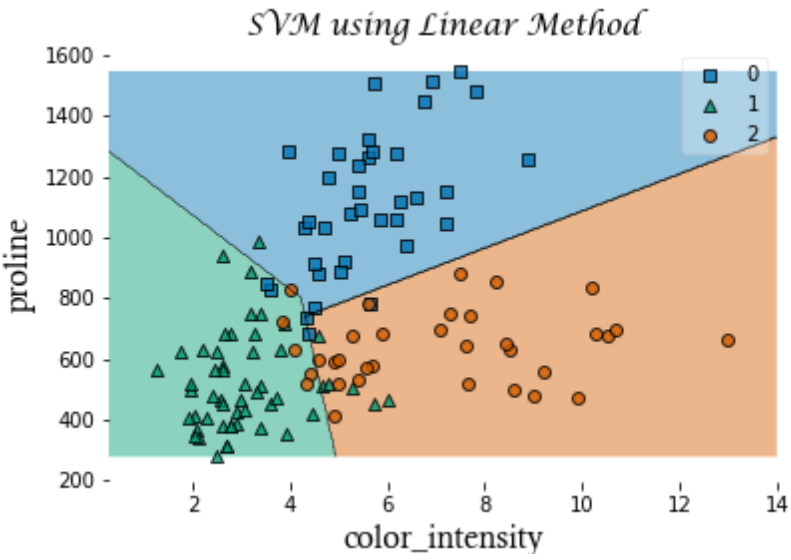
	Linear	**RBF**	**Non-Linear**
Accuracy	94.44 %	74.07 %	94.44 %

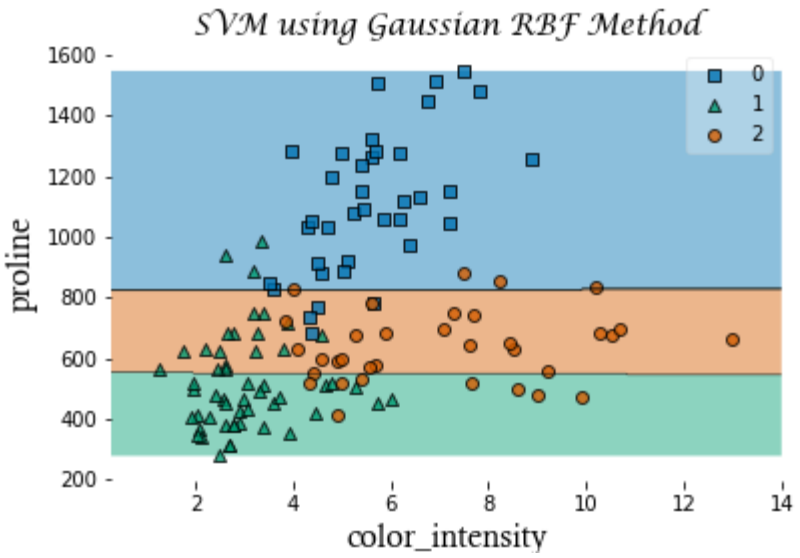
As a general, Usually the decision is whether to use linear or an RBF (aka Gaussian) kernel. There are two main factors to consider:

- Solving the optimization problem for a linear kernel is much faster.
- Typically, the best possible predictive performance is better for a nonlinear kernel (or at least as good as the linear one).

In practice, the linear kernel tends to perform very well when the number of features is large (e.g. there is no need to map to an even higher dimensional feature space). A typical example of this is document classification, with thousands of dimensions in input space. In those cases, nonlinear kernels are not necessarily significantly more accurate than the linear one. This basically means nonlinear kernels lose their appeal: they require way more resources to train with little to no gain in predictive performance, so why bother. From the above explanation it is clear that as we are using less number of features so the Nonlinear method should perform better than other two methods. Below are the output graphs of these methods.


SVM using Non Linear Method
1600
1400
1200
1000
800
600
400
200
proline
2
4
6
8
10
12
14
color_intensity
0
1
2


SVM using Linear Method
1600
1400
1200
1000
800
600
400
200
proline
2
4
6
8
10
12
14
color_intensity
0
1
2



Extra Imported Packages

- **Mlxtend:** Mlxtend (machine learning extensions) is a Python library of useful tools for the day-to-day data science tasks.

  Command:

  pip install mlxtend

References

i. Book: Data Mining Concepts and Techniques by Jiawei Han, Micheline Kamber, Jian Pei

ii. http://web.mit.edu/6.034/wwwbob/svm.pdf

iii. https://onionesquereality.wordpress.com/2009/03/22/why-are-support-vectors-machines-called-so/

iv. https://stats.stackexchange.com/questions/91091/one-vs-all-and-one-vs-one-in-svm

v. https://stats.stackexchange.com/questions/253086/selectkbest-feature-selection-python-scikit-learn

vi. https://github.com/rasbt/mlxtend

vii. https://www.degruyter.com/downloadpdf/j/amcs.2016.26.issue-1/amcs-2016-0013/amcs-2016-0013.pdf

viii. https://stats.stackexchange.com/questions/73032/linear-kernel-and-non-linear-kernel-for-support-vector-machine