In the Name of God

Statistical Pattern Recognition

Homework VI

Linear and Non-Linear Classification Using SVM

Assignment Date: **3 Bahman** Submission Deadline: **15 Bahman**

Contents

1 Objectives and Precautions

In this homework you will learn:

- The core idea of using SVM as a classifier and its other applications

Also note that:

- You can use any python libraries you want for this homework.
- The home work instructions and datasets will be shared with you in your **Quera class**.
- Save your plots, results, and answers in any format you want, this will be your report.
- Save your code files and your report as a zip file and upload in Quera. (naming format is "your names.zip" for example "Achraf_Hakimi_Mohamed_Salah.zip")
- Late submission strategy is: 70 percent score for one day delay and 50 percent for 2 days delay. Submissions after 2 days will not be graded.
- Use only the python programming language.
- Feel free to ask your questions in the telegram channel.
- Do not copy other works, write your own code.

2 Datasets

2.1 Linearly Separable: The Iris Dataset from Scikit-Learn

The Iris dataset consists of 3 different types of flowers (Setosa, Versicolour, and Virginica) with their features of petal and sepal length and width, as shown in Fig 1. You will use this dataset to train a linear SVM, therefore:

1. Load this dataset using sklearn.datasets.load_iris() and see the names of its feature and targets using data.feature_names and data.target_names.
2. Pick only 2 flower classes of Setosa and Versicolor and drop the remaining flower class.
3. Also select only 2 features of Petal Length and Petal Width and drop the rest.
4. Your linearly separable dataset is ready. Plot it in the 2d plane.

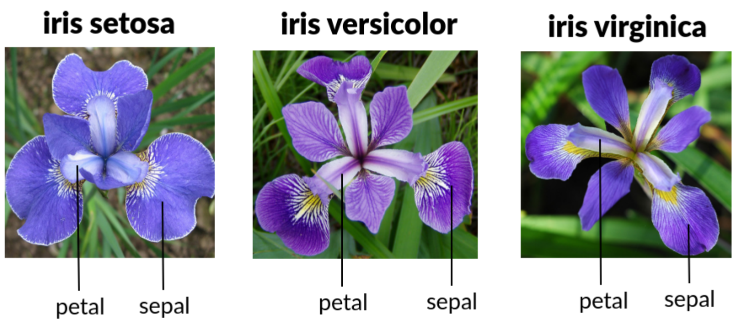

Figure 1: The Iris dataset contains 3 flower types.

2.2 Non-Linearly Separable: Make Your Own Moons

Scikit-learn includes various random sample generators to generate synthetic datasets for classification and clustering (more information can be found here). One of these sample generators is the sklearn.datasets.make_moons() which makes moons, two interleaving half circles. This datasets won's be linearly separable and is good choice to learn non-linear classification using SVM. Therefore:

1. Make moons with 500 samples, 0.15 noise and a constant random_state like 42 (so that you get the same dataset in each run).
2. That's it. Your non-linearly separable dataset is ready. Plot it in the 2d plane.

3 Support Vector Machines

SVM is a supervised algorithm used in applications of classification, regression, and novelty (or outlier) detection. It was introduced by Vladimir Vapnik and his coworkers in 1992, and been popular ever since.

It's good to note that SVM is not very well suited for large datasets due to its algorithmic complexity, and is also sensitive to feature scales. So don't forget to scale your datasets before modeling.

3.1 Linear Classification

You will be using the Iris dataset here. In order to create your classifier model, follow the instructions and answer the questions:

1. Split your data to training and test sets by a factor of 0.2.
2. Scale your dataset using sklearn.preprocessing.StandardScaler(). (fit the scalar to the training set then transform the test set)
3. Fit your linear SVM model to the training dataset. Use sklearn.svm.SVC() with a linear kernel.
4. Predict class labels for the test set and report recall, precision, and the confusion matrix. (You can use libraries from sklearn.metrics)
5. Plot the linear decision boundary and the marginal boundaries along with the training and test sets. Also circle around the support vectors. (you can use the plot_svm() function shared with you in the drive folder to avoid further implementations)
6. Your main task is to see how changing the C parameter will impact the result. Select different C values between 0 and 20 and plot your results using the plot function from the previous step.
7. How does the value of C impact the number of support vectors?
8. How does the value of C impact the margins? Search and see what soft and hard margin mean for a SVM model.
9. What is this C value anyways?

3.2 Non-Linear Classification

You will be using the moons dataset here. In order to create your classifier model, follow the instructions and answer the questions:

1. Split your data to training and test sets by a factor of 0.2.
2. Scale your dataset using sklearn.preprocessing.StandardScaler(). (fit the scalar to the training set then transform the test set)
3. You will be using 2 kernels for non-linear classification, polynomial (poly) and Gaussian Radial Basis Function (rbf). For each kernel, tune your model using a grid search or similar hyperparameter selection process, and report the best hyperparameters found. (you can use the tune() function shared with you in the drive folder to avoid further implementations)
4. Once you found the best model for each kernel, predict class labels for the test set and report recall, precision, and the confusion matrix. (You can use libraries from sklearn.metrics)
5. Plot the non-linear decision boundary and the marginal boundaries along with the training and test sets. Also circle around the support vectors. (you can use the plot_svm() function shared with you in the drive folder to avoid further implementations)
6. Your main task is to experiment how changing the parameters for each kernel, will impact the result. You can select different parameter values from the following ranges:
   - C: float 0 to 20
   - degree: integer 0 to 5

    - gamma: float 0 to 1
    - coef0: integer 0 to 5

7. In each experiment use the parameters from the best model and change only one parameter to see its impact on your plots and metrics. You can again use plot_svm() here.
8. See the impact in terms of number of support vectors and the shapes of decision and marginal boundaries. Try to reason the changes.

4 Bonus

4.1 SVM for Novelty Detection

Search the web and find one application where SVM is used for novelty detection (also referred to as anomaly detection or outlier detection). The algorithm is called **One-Class SVM** or OSVM for this particular application. You can search Kaggle or Github for code bases too.

You will get the bonus if only you can find an application of one-class SVM and a code base for it that runs, and also be able to explain how the algorithm works. Good luck.