# Artificial Intelligence & Machine Learning
# Homework 2 - Report

Antonio Tavera
243869

**Abstract**

In this homework is requested to familiarize with the basic concepts and options of the Support Vector Machines (SVMs) and to perform data validation, cross validation and grid search in order to find the best parameters to evaluate the model.
After describing how to prepare data, this report, as the source code, is divided into three different sections: first it deals with the train and the analysis of different C values for a Linear SVM, then it does the same thing but for a RBF kernel and it analyzes the results of a grid search to find the best parameters to setup the model; last it repeats the analysis but with a k-fold cross validation.

## 1. Data Preparation

For each section of the source code, the first thing that is done is the data preparation. That is handled by the *load_dataset()* function that receives as input parameters the wanted size of the validation and test set. In this homework we deal with the Iris Dataset, and this is imported by calling the respective *load_iris()* function of the scikit-learn library. Only the first two dimension are selected and then the dataset is split into three different parts according to the proportion received. When the *train_test_split()* function do its job, the three dataset, one for training, one for validation and one for test, and their relative labels are returned to the user and ready to use.

## 2. Linear SVM

### *2.1. About*

The Support Vector Machine, a.k.a. SVM, is a linear model for classification and regression problems. It can solves linear and non-linear problems. The idea of SVM is simple: the algorithm creates a line or a hyperplane which separates the data into classes. The most important parameter for a Linear SVM is C, also know as the regularization parameter, is a value that tells the SVM optimization how much we want to avoid misclassifying each training sample, is the penalty parameter of the error term, it controls the trade off between smooth decision boundary and classifying the training points correctly. When C is small, the classifier is okay with misclassified data points (high bias, low variance). When C is large, the classifier is heavily penalized for misclassified data and therefore bends over backwards to avoid any misclassified data points (low bias, high variance).

### *2.2. Algorithm*

The first choice offered by the menu is to work with a Linear SVM; if the data are not already loaded the algorithm performs what is described in the Section 1 and defines the C values. Then, for each value of C, it:

1. Trains a linear SVM through the *train()* function;
2. Plots the data and the decision boundaries calling the *plot_boundaries()* function, results in Figure [1];
3. Evaluates the model on the validation set through the *test_model()* function.

All the results that comes from the tests performed on the validation set are plotted in a graph, as it can be seen in Figure [2].
The last thing done is to use the best value of C, the one with whom is obtained the highest accuracy on the validation set, setup and train the Linear SVM model with that value and test on the test set. The accuracy that is achieved, as expected, is approximately of 84%.

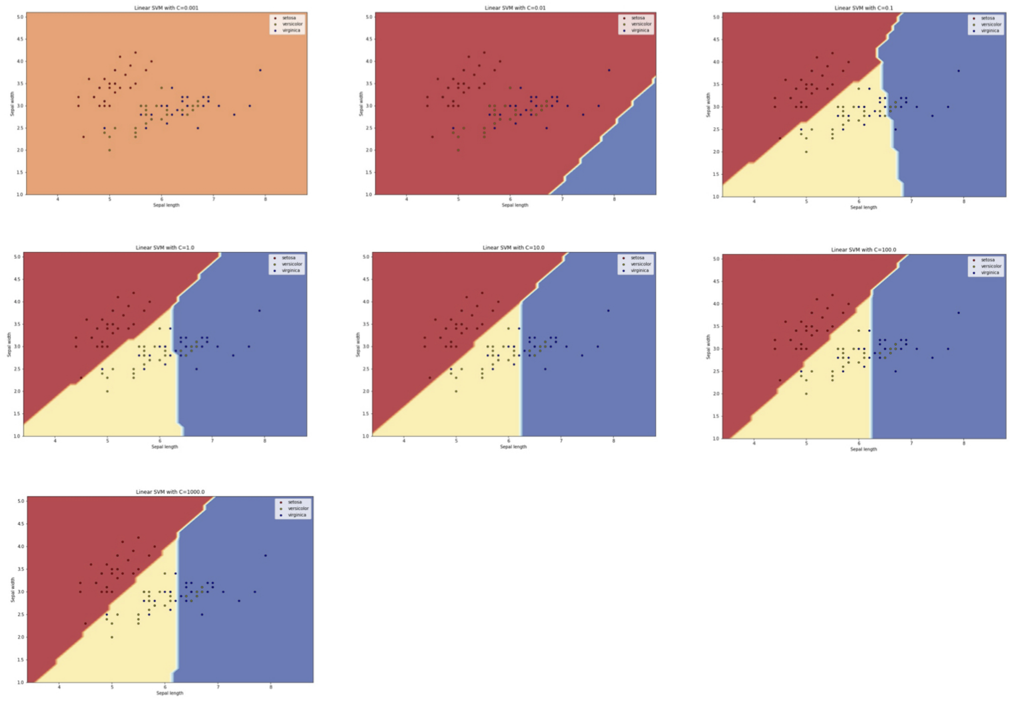

Figure 1: This figure represent the data and the boundaries obtained after setting and training the SVM with different values of C ($10^{-3}$, $10^{-2}$, $10^{-1}$, 1, $10^{1}$, $10^{2}$, $10^{3}$)

### *2.3. Analysis of Results*

Analyzing both Figure [1] and Figure [2], it is not difficult to see a correlation between C and the accuracy. As said few lines above, the more smaller is the value of C the more larger can be the margin separating hyperplane, even if that hyperplane misclassifies more points (and that is clear in the first two images with C=$10^{-3}$ and C=$10^{-2}$). This is the reason why the accuracy for those two cases is smaller rather than the accuracy we obtain when we choose higher values of C. Indeed, in this way, we obtain more smaller margin hyperplane but they are better in classifying correctly all the trained points.

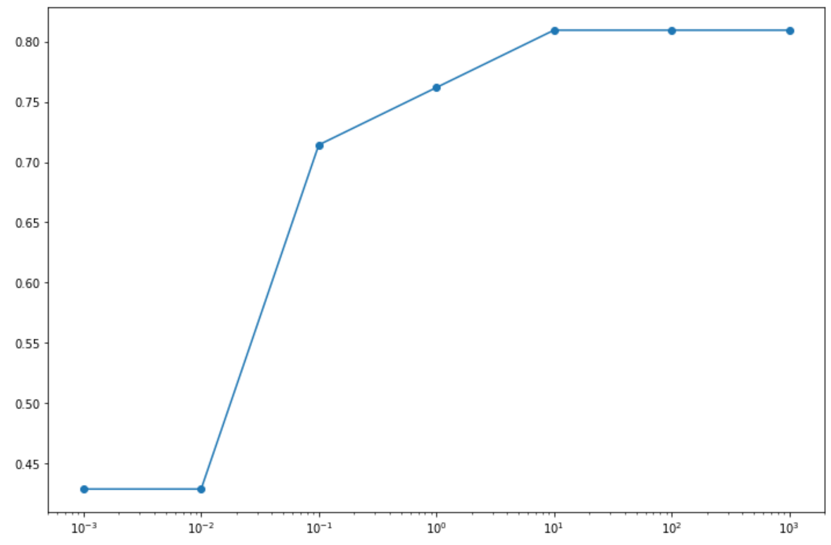

Figure 2: This figure represent the different level of accuracy that the Linear SVM obtain by testing the validation set with different values of C. In the x-axis are plotted the C values, while in y-axis the accuracies. The lower accuracy is 0.4286 with C=$10^{-3}$ and C=$10^{-2}$ while the higher is 0.8095 from C=$10^{1}$ forward.

## 3. RBF Kernel

### *3.1. About*

The Radial Basis Function kernel, a.k.a. RBF, is used to classify data that are non linearly separable in a lower dimensional space but that are in an higher one. A SVM classifier that uses a RBF kernel has two parameters: C, already explained in the section above, and gamma, which is a parameter that can be thought of as the spread of the kernel and therefore the decision region. When gamma is low, the curve of the decision boundary is very low and thus the decision region is very broad. When gamma is high, the curve of the decision boundary is high and creates islands of decision-boundaries around data points. We will see this in the results of that exercise.

### *3.2. Algorithm*

The second choice offered by the menu is to work with a RBF kernel; as the section above, if the data are not already loaded the algorithm performs what is described in Section 1 and then defines the C and gamma values. What is done for the Linear kernel is repeated here; the only differences are

the parameters used to train the model: the RBF kernel, the C values and the gamma parameter (set as 'auto'). It is possible to see the obtained decision boundaries with different values of C in Figure [3].
The second step that is done is a grid search for the best parameters; this time the model, for each loop, is set with all the possible pairs of C and gamma values from the ones that are previously defined and then is trained on the training set using a RBF kernel. It is shown in Figure [4] how these parameters score on the validation set.
The model is then re-setup and re-trained with the best found parameters (best C value: 1 and best Gamma: $10^1$) and tested with the test set. The accuracy obtained is approximately of 80%, it is possible to see the plotted decision boundaries in Figure [5].

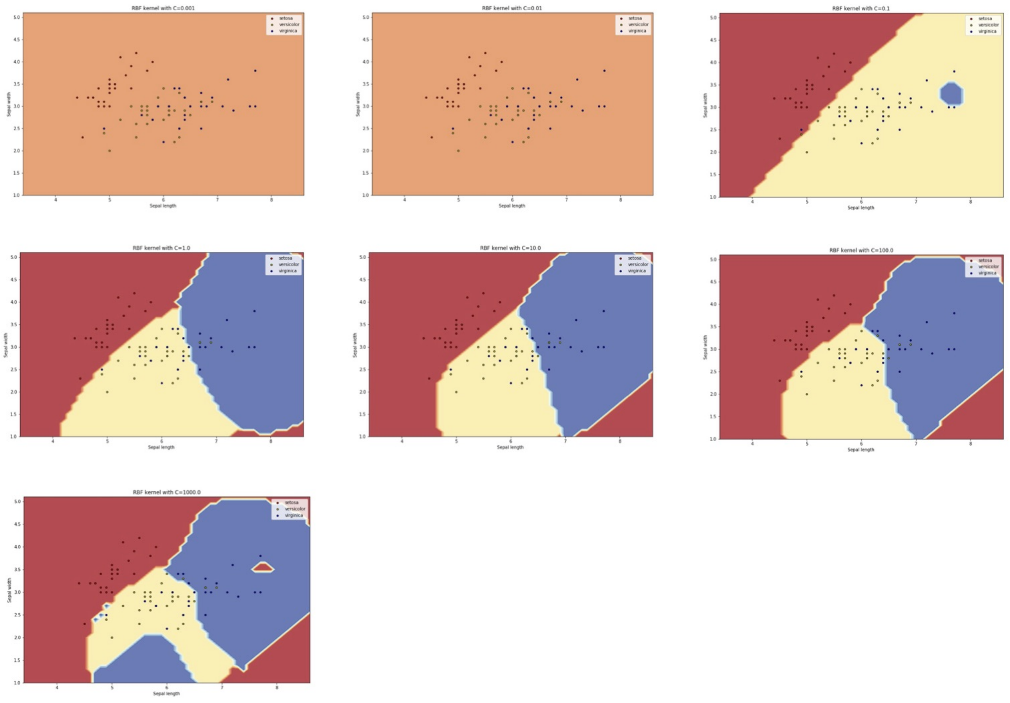

Figure 3: This figure represent the data and the boundaries obtained after setting and training the SVM with the RBF kernel, with gamma set as auto and with different values of C ($10^{-3}$, $10^{-2}$, $10^{-1}$, 1, $10^1$, $10^2$, $10^3$)

Grid Search Table

| C - GAMMA | 1E-11 | 1E-09 | 1E-07 | 1E-05 | 1E-03 | 1E-01 | 1E+01 |
|---|---|---|---|---|---|---|---|
| 1,0E-03 | 0,285 | 0,285 | 0,285 | 0,285 | 0,285 | 0,285 | 0,285 |
| 1,0E-02 | 0,285 | 0,285 | 0,285 | 0,285 | 0,285 | 0,285 | 0,285 |
| 1,0E-01 | 0,285 | 0,285 | 0,285 | 0,285 | 0,285 | 0,285 | 0,333 |
| 1,0E+00 | 0,285 | 0,285 | 0,285 | 0,285 | 0,285 | 0,666 | 0,809 |
| 1,0E+01 | 0,285 | 0,285 | 0,285 | 0,285 | 0,285 | 0,666 | 0,713 |
| 1,0E+02 | 0,285 | 0,285 | 0,285 | 0,285 | 0,666 | 0,666 | 0,619 |
| 1,0E+03 | 0,285 | 0,285 | 0,285 | 0,285 | 0,714 | 0,666 | 0,666 |

Figure 4: This figure represent the different level of accuracy that are obtained after testing the validation set with a model initialized with different values of C and gamma.

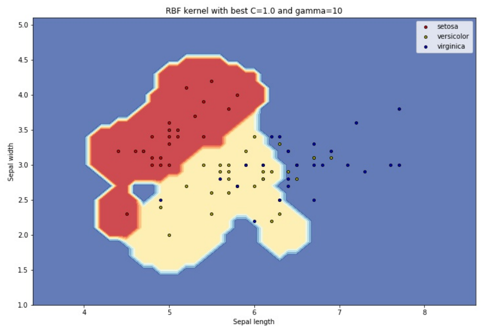

Figure 5: This figure represent the data and the boundaries of the model tested with the test set, using a model initialized with the best values of C and gamma obtained from the grid search, 1 and $10^1$ respectively, as it is possible to deduce from Figure 4.

## *3.3. Analysis of Results*

Analyzing Figure [3] there are some similarities and differences from the results obtained with a linear SVM, as we saw in Figure [1]. The first thing that jumps right out is that in the Linear model data points are separated through straight lines, while using a non linear classifier, like the RBF kernel, curves takes the place of lines. Just like happen with a Linear SVM, also with a RBF kernel the accuracy increase increasing the value of C. As said in the previous section, with low values of C the classifier is clearly tolerant of misclassified data point and so the decision boundary is more permissive compared to higher values of C; in fact this is the case when the classifier starts to become very intolerant to mismatched data points. This behaviour goes in parallel to what happen with different gamma values; with low gamma the decision boundary is not very curvy, it is more an arch, while with high gamma it creates islands.

## 4. K-Fold Cross Validation

### *4.1. About*

The goal of cross-validation is to test the models ability to predict new data that was not used in estimating it, in order to flag problems like overfitting or selection bias and to give an insight on how the model will generalize to an independent dataset. The K-Fold cross validation, a non exhaustive cross validation, randomly partitions the original sample into $k$ equal sized subsamples. Of the $k$ subsamples, a single one is retained as the validation data for testing the model, and the remaining $k-1$ are used as training data. The cross-validation process is then repeated $k$ times, with each of the $k$ subsamples used exactly once as the validation data. Summing up, the practical goal is to find the best fit model based on our sample with the lowest error on all parts of data.

### *4.2. Algorithm*

The last choice offered by the menu is to perform a K-Fold Cross Validation. The first thing that is done is to merge training and validation set together in order to obtain 70% training and 30% test data. Then, as done in the Section 2, a grid search is performed to find the best values for C and gamma but this time applying the *GridSearchCV()* function provided by the scikit-learn library. This function receives as input not only the kernel and the values of C and gamma that need to be tested, but also the *cv* parameter, that is set to 5; this last corresponds to the value of the k-fold we want to apply during the fit process. When the algorithm finds what we are looking for, it simply tests the model with the test set and print the result.

### *4.3. Analysis of Results*

The best C and gamma that comes out the grid search are $10^2$ and $10^{-2}$ respectively. The accuracy on the test set along a model trained with the rightly mentioned parameter is of approximately 87%. It is a little higher than the one performed without applying cross validation and is what we are expecting because we are considering as training model the best predictive one obtained through the 5-fold cross validation.